Explainable Neural Approaches to Question Classification

Group 7:
Kai Liang, Xiaoxiao Wen, Weitao Luo, Yijie Zhang

Introduction

- Text classification is a classical task for natural language processing (NLP)
- We revisit the task of question classification, which is important for question answering
- Our study focuses on two parts
  - Implement neural classifiers (LSTM and TextCNN) to compare with non-neural baseline (FastText)
  - Add an unsupervised component to increase interpretability
    - A layer of binary latent variables that extract rationales (short and meaningful tokens) from input to use for classification
    - Train the model via variational inference using REINFORCE

Related Work

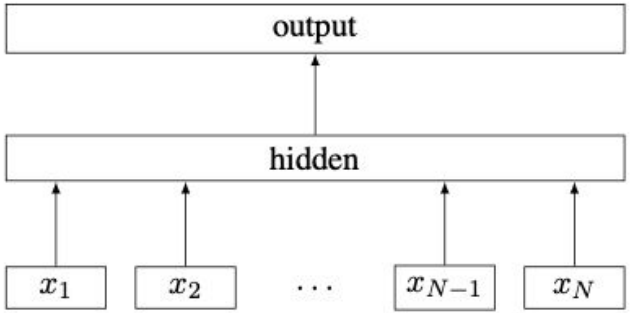
- Traditional text classifiers
  - Nearest neighbours, Naive Bayes, decision trees and SVMs with hand-crafted features
- Recently, the task is dominated by complex neural models
  - LSTM, TextCNN, RCNN and BERT
- Limitation of neural models
  - Processing speed
    - Previous work tackle by proposing simple but powerful non-neural models (e.g. FastText)
  - Interpretability
    - Deterministic scope: attention mechanism
    - Generative scope: add latent variables that extract rationales from input sequence to feed into the classifiers

Methods

1. FastText
2. LSTM
3. TextCNN
4. Rationale Extraction

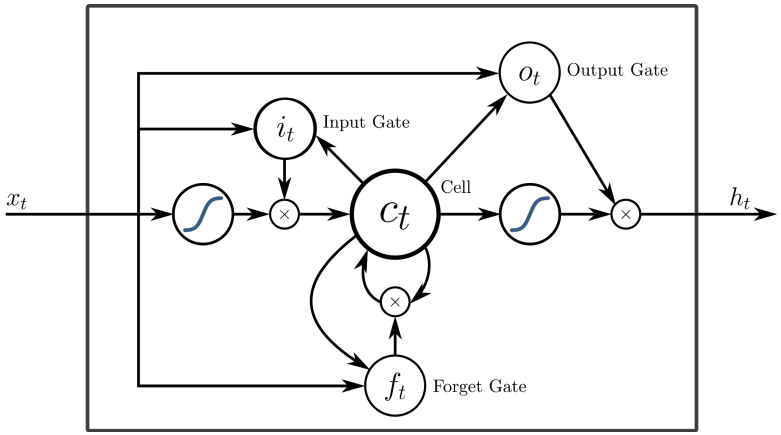
FastText

- A simple and efficient model for text classification
- It is many orders of magnitude faster than neural models for both training and evaluation
- Comparable to deep learning classifiers in terms of accuracy
- Model architecture similar to CBOW
  - Except that the goal is changed from predicting the middle word to the label of the text sequence
- Average n-gram features of input to obtain the hidden vector, and project to output space using a linear layer



LSTM

- LSTM is proposed to overcome the gradient vanishing problem of vanilla RNN
- **Adaptive gating mechanism**
- Components of a LSTM unit: A cell, an input gate, an output gate and a forget gate



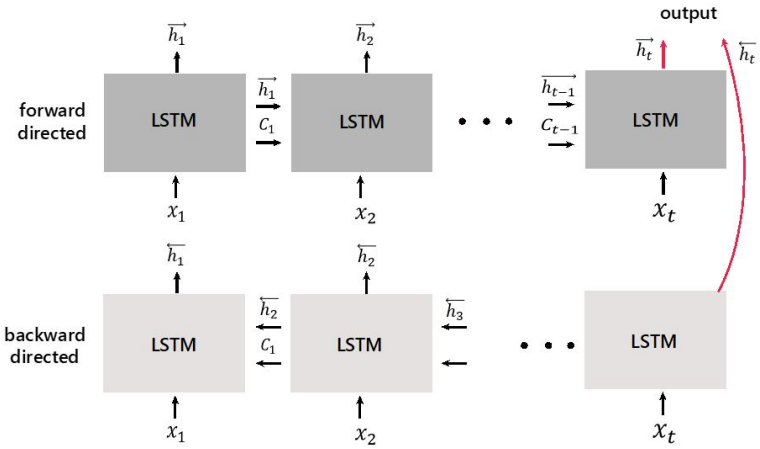
LSTM

- Bidirectional LSTM: capture both the past and future information at the same time
- Two sub-networks for the forward and backward pass respectively

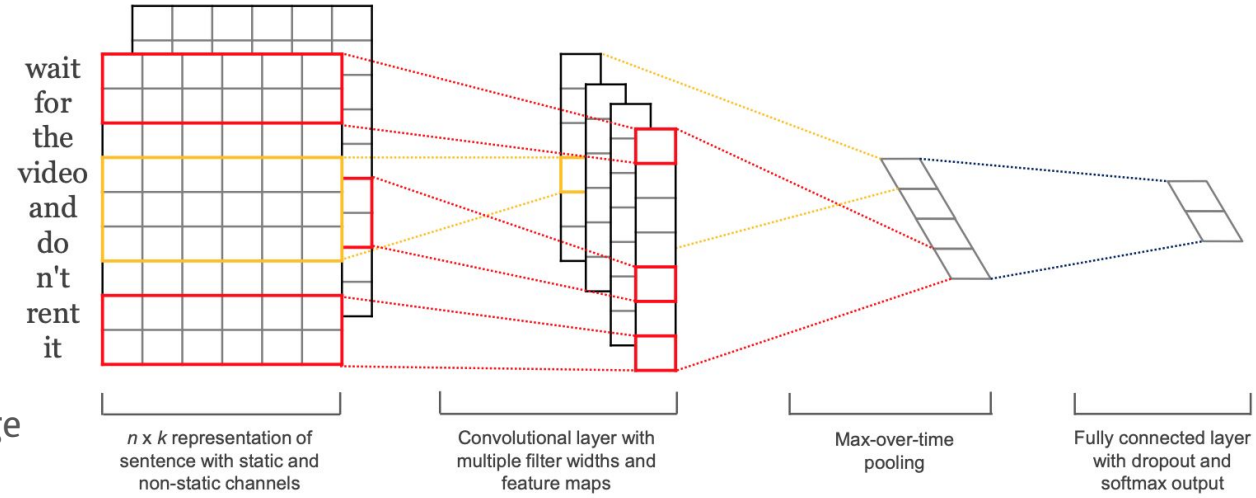
$$\overrightarrow{\mathbf{h}_t} = \overrightarrow{f}\left(\mathbf{x}_t, \overrightarrow{\mathbf{h}_{t-1}}\right)$$

$$\overleftarrow{\mathbf{h}_t} = \overleftarrow{f}\left(\mathbf{x}_t, \overleftarrow{\mathbf{h}_{t+1}}\right)$$

$$\mathbf{h}_t = \left[\overrightarrow{\mathbf{h}_t} \oplus \overleftarrow{\mathbf{h}_t}\right]$$



TextCNN



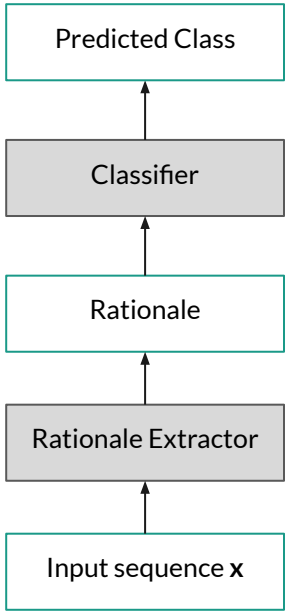
- CV vs NLP
- Image vs sentence
- Local features in image
- Model architecture
  - Embedding layer (pre-trained word vectors, e.g. word2vec or Glove)
  - Convolution layer (capture key local features)
  - Max-Pooling layer
  - Full-Connection layer
  - Softmax layer
- replicate CNN's success in the context of text classification

Rationale Extraction

Task: Extract a subset of tokens (rationales) from the input sequence **x**

Requirements: **Concise**

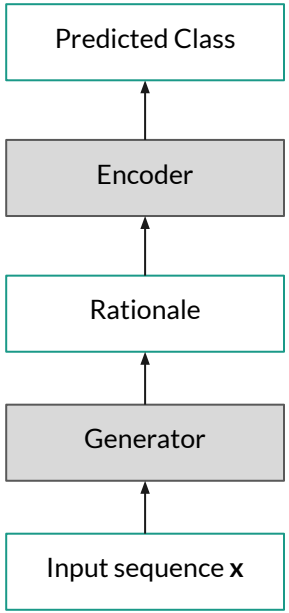
1. Short
2. Meaningful



Network Architecture

Components

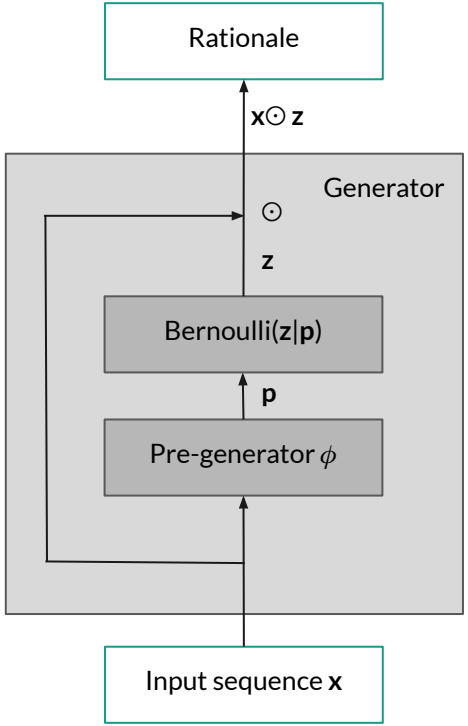
1. Generator (Rationale Extractor)
2. Encoder (Classifier)



Generator

Pre-generator **pregen**$(\cdot|\phi)$

- 2-layer bidirectional LSTM connected to
  - a fully-connected layer or average-pooling layer
  - with Sigmoid activation
- Input: **x**
- Output: **p** (probability parameters for each token in **x**)



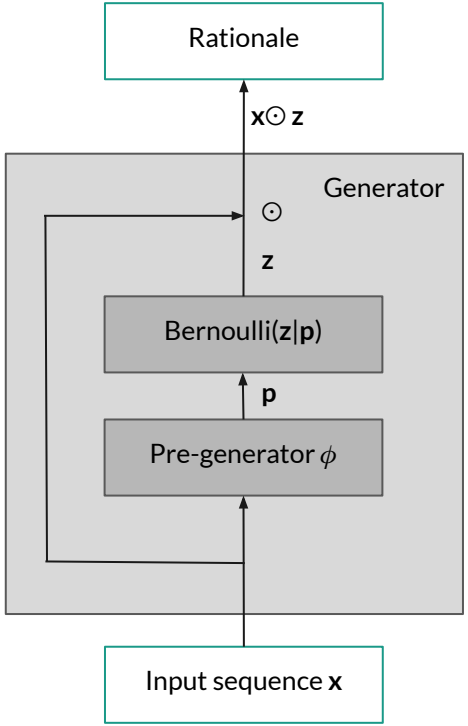
Generator

Bernoulli distribution (**z**|**p**)

- Sample binary latent variable **z** using learned **p**
- Obtain rationales by computing element-wise product of **x** and **z**

Given **x**, z_t for all t in the input sequence is independent

$$p(\mathbf{z}|\mathbf{x}, \phi) = \prod_{t=1}^{l} p(\mathbf{z}_t|\mathbf{x}, \phi)$$

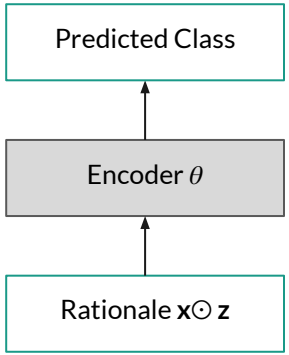


Encoder

Encoder **enc**$(\cdot|\theta)$

- Arbitrary classifier for the classification task suffices (here we use TextCNN)
- Input: **x** $\odot$ **z** (or generally any input sequence **x**)
- Output: y

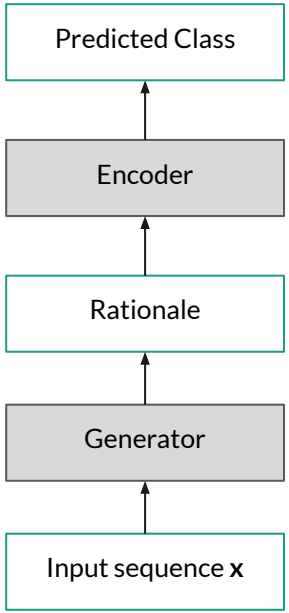
The predicted class label y follows the categorical distribution Cat(y|**enc**(**x** $\odot$ **z**|θ))



Objective

2 models -> 2 aspects

1. Generator: extract short and meaningful rationales from input **x**
2. Encoder: predict the correct label y for given input sequence



Objective

Task: Maximize the conditional likelihood of the targe $p(y|\mathbf{x}, \theta) = \mathbf{Cat}(y|\mathbf{enc}(\mathbf{x}|\theta))$

It is conditioned on $\mathbf{z}$ as $p(y|\mathbf{x}, \theta) = \int_{\mathbf{z}} p(y|\mathbf{x}, \mathbf{z}, \theta)p(\mathbf{z}|\mathbf{x}, \phi)d\mathbf{z} = \mathbb{E}_{p(\mathbf{z}|\mathbf{x}, \phi)}[p(y|\mathbf{x}, \mathbf{z}, \theta)]$

Jensen's inequality $\log p(y|\mathbf{x}, \theta) = \log \mathbb{E}_{p(\mathbf{z}|\mathbf{x}, \phi)}[p(y|\mathbf{x}, \mathbf{z}, \theta)] \geq \mathbb{E}_{p(\mathbf{z}|\mathbf{x}, \phi)}[\log p(y|\mathbf{x}, \mathbf{z}, \theta)]$

Lower Bound $\mathcal{E}(\phi, \theta) = \mathbb{E}_{p(\mathbf{z}|\mathbf{x}, \phi)}[\log p(y|\mathbf{x}, \mathbf{z}, \theta)]$

- where $p(y|\mathbf{x}, \mathbf{z}, \theta) = \mathrm{Cat}(y|\,\mathrm{enc}(\mathbf{x} \odot \mathbf{z}|\theta))$

Maximize Lower Bound $\Leftrightarrow$ Minimize Cross Entropy Loss

Objective

To extract short and coherent rationales, 2 additional regularizers

1. Penalize length of the selected tokens (or $\mathbf{z}$) $\lambda_1 \|\mathbf{z}\|$
2. Penalize discontinuity in the selected tokens $\lambda_2 \sum_{t=2}^{l} |\mathbf{z}_t - \mathbf{z}_{t-1}|]$

Final loss

$$\begin{aligned}\mathcal{L}(y, \mathbf{x}, \mathbf{z}) &= \mathbb{E}_{p(\mathbf{z}|\mathbf{x},\phi)}[L(y, \mathbf{x}, \mathbf{z})] \\ &= \mathbb{E}_{p(\mathbf{z}|\mathbf{x},\phi)}[\mathbf{CrossEnt}(y, \mathbf{enc}(\mathbf{x} \odot \mathbf{z}|\theta)) \\ &+ \lambda_1 \|\mathbf{z}\| + \lambda_2 \sum_{t=2}^{l} |\mathbf{z}_t - \mathbf{z}_{t-1}|]\end{aligned}$$

Gradient Estimation: REINFORCE

1. Gradient estimation for the generator

$$\frac{\partial \mathcal{L}(y, \mathbf{x}, \mathbf{z})}{\partial \phi} = \frac{\partial \mathbb{E}_{p(\mathbf{z}|\mathbf{x}, \phi)}[L(y, \mathbf{x}, \mathbf{z})]}{\partial \phi}$$
$$= \mathbb{E}_{p(\mathbf{z}|\mathbf{x}, \phi)}[L(y, \mathbf{x}, \mathbf{z}) \frac{\partial \log p(\mathbf{z}|\mathbf{x}, \phi)}{\partial \phi}]$$

2. Gradient estimation for the encoder

$$\frac{\partial \mathcal{L}(y, \mathbf{x}, \mathbf{z})}{\partial \theta} = \mathbb{E}_{p(\mathbf{z}|\mathbf{x}, \phi)}[\frac{\partial L(y, \mathbf{x}, \mathbf{z})}{\partial \theta}]$$

Experiments

- Dataset
  - Question Classification Dataset
  - Preprocessing
  - Pre-trained Word Embeddings
- Models settings
  - Hyperparameter settings

Dataset

- Question Classification Dataset
  - Contains 6 coarse classes: abbreviation, entity, description, human, location and numeric
  - Training set
    - The first 90% for training (9180 data)
    - The rest 10% for validation (1020 data)
  - Test set (500 data)
- Preprocessing
  - The tokens are stemmed by the Porter stemmer
  - Input sequences are padded to the length of longest input sequence in a batch
- Pre-trained Word Embeddings
  - Google's pre-trained 300d word2vec, <unk> is initialized randomly, <pad> with zeros

Model Settings

- Hyperparameter settings are obtained by performing grid search on all models
- Final settings are listed in the tables given below

Parameter	Value
lr	0.1
epoch	70
lr_update_rate	100
embed_dim	100
window_size	5
min_count	1

FastText

Parameter	Value
lr	0.001
epoch	10
batch_size	64
max_grad_norm	5.0
lstm_layer	2
lstm_direction	2

LSTM

Parameter	Value
lr	0.001
epoch	10
batch_size	64
max_grad_norm	5.0
kernel_size	(2, 3, 4)
kernel_num	32
dropout	0.5

TextCNN

Parameter	Value
lr	0.001
epoch	50
batch_size	64
max_grad_norm	5.0
lstm_layer	2
lstm_direction	2
λ_1	0.01
λ_2	0.001

Generator Network

Results

- Classification results on full input sequences

Model	Overall Accuracy	Abbreviation			Entity			Description			Human			Location			Numeric		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
FastText	0.860	1.00	0.78	**0.88**	0.83	0.62	0.71	0.79	0.98	0.87	0.87	0.94	**0.90**	0.88	0.86	0.87	0.97	0.88	**0.92**
LSTM	0.862	0.83	0.56	0.67	0.86	0.76	**0.80**	0.81	0.95	0.88	0.88	0.89	0.89	0.82	0.90	0.86	0.98	0.82	0.89
TextCNN	**0.876**	0.83	0.56	0.67	0.93	0.66	0.77	0.82	0.99	**0.90**	0.88	0.92	**0.90**	0.84	0.93	**0.88**	0.96	0.88	**0.92**

- Model performance are evaluated in terms of overall accuracy, per-category precision, recall and f1-score, which are common metrics of such tasks

Results

- Classification results on extracted rationales alone (Keep Rate: portion of selected tokens)

Model	Overall Accuracy	Keep Rate
LSTM-FC	**0.706**	0.503
LSTM-AP	0.594	**0.276**

- Examples of the extracted rationales

Model	Extracted Rationales	Class
LSTM-FC	**what is** maryland <unk> **state** bird <unk>	Entity
LSTM-AP	**who** discov <unk> <unk>	Human

Discussion

- The classification results reveal that both of the neural models are superior to the non-neural baseline (i.e. FastText) in terms of the overall accuracy
- Also, our TextCNN model achieves the best performance not only in the overall accuracy, but also in the f1-score of 4/6 classes, which proves its strong model capacity

Discussion

- The rationale extraction results show that when the aggregation of the LSTM output is learned with a fully-connected layer, the overall accuracy for classification is decreased by 19.4%, while only 50.3% of tokens are used
- On the other hand, when the aggregation is simply averaging over the LSTM output, the overall accuracy for classification is decreased by 32.2%, while only 27.6% tokens are used
- From the qualitative results, for both models, the generator network is able to extract rationales that have the largest contribution to the classification
    - 'who' for the class 'Human'
    - 'state' for the class 'Entity'

Conclusion

- In this study, we firstly implement and compare a set of models (i.e. FastText, LSTM and TextCNN) for the task of question classification
- We conclude that both the neural models (i.e. LSTM and TextCNN) outperform the non-neural model (i.e. FastText) in terms of various metrics, and TextCNN achieves the best performance in our dataset
- We then combine unsupervised learning with the classification models by adding a layer of binary latent variables to extract rationales from the input sequences for classification, which is trained via a REINFORCE-style algorithm
- Our results confirm that the extracted rationales as features are able to reach a similar-level of test performance as the original sequences, which enables better interpretability of our neural models

References

1) Bastings, J., Aziz, W., & Titov, I. (2019). Interpretable Neural Predictions with Differentiable Binary Variables. arXiv preprint arXiv:1905.08160.
2) Chen, K., Wang, J., Chen, L. C., Gao, H., Xu, W., & Nevatia, R. (2015). Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*.
3) Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
4) Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222-2232.
5) Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.
6) Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.
7) Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.
8) Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, February). Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.
9) Lei, T., Barzilay, R., & Jaakkola, T. (2016). Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*.

References

10) Li, X., & Roth, D. (2002, August). Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1* (pp. 1-7). Association for Computational Linguistics.
11) Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
12) Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).
13) Porter, M. F. (1997). Readings in information retrieval. *San Francisco, CA, USA: Morgan Kaufmann Publishers Inc*, 313-316.
14) Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*(pp. 5998-6008).
15) Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3-4), 229-256.
16) Zhang, D., & Lee, W. S. (2003, July). Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 26-32). ACM.