Action Recognition and State Change Prediction in a Recipe Understanding Task Using a Lightweight Neural Network Model (Student Abstract)

Qing Wan, Yoonsuck Choe
Department of Computer Science and Engineering, Texas A&M University
College Station, TX, 77843, USA
frankwan@email.tamu.edu, choe@tamu.edu

Abstract

Consider a natural language sentence describing a specific step in a food recipe. In such instructions, recognizing actions (such as press, bake, etc.) and the resulting changes in the state of the ingredients (shape molded, custard cooked, temperature hot, etc.) is a challenging task. One way to cope with this challenge is to explicitly model a simulator module that applies actions to entities and predicts the resulting outcome (Bosselut et al. 2018). However, such a model can be unnecessarily complex. In this paper, we propose a simplified neural network model that separates action recognition and state change prediction, while coupling the two through a novel loss function. This allows learning to indirectly influence each other. Our model, although simpler, achieves higher state change prediction performance (67% average accuracy for ours vs. 55% in (Bosselut et al. 2018)) and takes fewer samples to train (10K ours vs. 65K+ by (Bosselut et al. 2018)).

Introduction

Understanding actions in sentences and the resulting changes in the state of the entities is still a challenging task in NLP. This is due to the various causal factors involved and necessary common-sense knowledge for correct inference of the actions and states. Recently, (Bosselut et al. 2018) introduced a novel neural process network, specifically designed to understand procedural language with the goal of modeling action dynamics and predicting their effects on entities. However, they had to train their model on a set of over 65K recipes selected from their whole 120K dataset under certain rules. Their proposed model achieved 55% accuracy on the test set in predicting the state change in the recipe sentences. This is somewhat limited performance, and furthermore, their model was fairly complex, with encoding, action selection, entity selection, state prediction, and action simulation.

To tackle these problems, we introduce a new deep neural network model with a much more concise architecture, and aim at solving a main task that is to correctly recognize actions and predict state changes in a sentence of a recipe. Also, we contribute a novel cost function. Training on any small randomly selected subset (10K samples) of the 65K, our model achieves an average state change prediction accuracy of 67% on the test set of (Bosselut et al. 2018), which is an improvement of 12% over their results.

Methodology

Our model is built on Recurrent Neural Networks (RNN, Graves et al. 2006) with Gated Recurrent Units (GRU, Cho et al. 2014). In our design, three different types of neural network layers are used: Dynamic RNN (encoder), MLP (decoder), and predictor. We find that this simple architecture is sufficient for action language understanding.

We encode each word in a sentence with a one-hot vector/code (Hinton et al. 1984) and count a whole sequence of vectors of words that are generated by encoding to produce the input representation. That is, in our model, we do not use any word embedding.

We use two different metrics, loss function and error function, for training control. We train our model by loss function and validate the model by error function.

We design the loss function under the scope of tangent. Let $\mathbf{Y}, \mathbf{P} \in [0, 1]^m$ (finite dimensional space) and $\mathbf{Y}$ is a label, $\mathbf{P}$ is a prediction for $\mathbf{Y}$, then the loss function is defined as

$$l(\mathbf{Y}, \mathbf{P}) = \sum_{i=1}^{i=m} 10 \times \tan(\frac{\pi}{2}|\mathbf{Y}(i) - \mathbf{P}(i)|). \qquad (1)$$

Then, $l(\mathbf{Y}, \mathbf{P}) = 0$ iff $\mathbf{Y} = \mathbf{P}$ due to the nonnegativity of $\tan((\frac{\pi}{2}|\mathbf{X}|)$. With the help of the following property

$$\tan(|x|) \geq |x|, x \in (-\frac{\pi}{2}, \frac{\pi}{2}), \qquad (2)$$

preservation of triangle inequality and convexity can be verified.

We define an error function with the help of cross entropy and the error is expected to serve as an accuracy indicator that will decide when to save our model during training. Cross entropy for discrete topology on $\mathcal{X}$ is well-defined as:

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log_2 q(x). \qquad (3)$$

Our error function $\epsilon(\mathbf{Y}, \mathbf{P})$ is defined by

$$\epsilon(\mathbf{Y}, \mathbf{P}) = |H(P_{\mathbf{P}}, Q_{\mathbf{Y}}) - H(Q_{\mathbf{Y}}, Q_{\mathbf{Y}})|, \quad (4)$$

where $P_{\mathbf{P}}$ and $Q_{\mathbf{Y}}$ denotes probability mass function (pmf) generated by the predicted vector $\mathbf{P}$ and the label $\mathbf{Y}$, respectively; and a Softmax function can implement the generation of $P_{\mathbf{P}}$ and $Q_{\mathbf{Y}}$. Qualitatively, the more accurately a neural network predicted, a lower ϵ should be achieved. This is guaranteed by

$$\begin{aligned} &|H(P_{\mathbf{P}}, Q_{\mathbf{Y}}) - H(Q_{\mathbf{Y}}, Q_{\mathbf{Y}})| \\ &\leq \sum_{x \in \mathcal{X}} |(Q_{\mathbf{Y}}(x) - P_{\mathbf{P}}(x)) \log_2 Q_{\mathbf{Y}}(x)| \\ &\leq \max_{x \in \mathcal{X}} \{ |\log_2 Q_{\mathbf{Y}}| \} \sum_{x \in \mathcal{X}} |Q_{\mathbf{Y}} - P_{\mathbf{P}}|, \end{aligned}$$

where the last $\leq$ is due to bounded $|\log_2 Q_{\mathbf{Y}}|$ for finite dimensions. So a continuity will follow. This continuity warrants the use of it as an accuracy indicator for model saving.

Experiments

This section describes how the experiments are conducted.

Data Source & Data Selection

We used the dataset and corpora from (Bosselut et al. 2018). It has a total of 120K samples. Bosselut et al. used carefully filtered 65,815 samples from the whole 120k for training, and 693 samples for testing. Besides, they provided corpora (token libraries) for vocabularies of sentences(text), verbs and state changes.

In our experiment, we randomly selected a 10K subset (15%) from their 65,815 samples and separate this 10k into 9k for training loss function and 1k for validating error function. We use their corpora for text, verb and state change. We add “UNK” (unknown) type to each corpus for vocabulary out of their corpora.

Encoding/Decoding

We directly feed a one-hot word vector (no embedding) into a GRU cell each time. We pad each sentence to the maximum length of sentences in the recipe. Our model has two encoding layers each with 1600 units and 800 units, respectively. Both verb decoding and state change decoding are analyzed by an independent MLP with 500 hidden units. Each MLP is connected to the final state of the GRU encoding layer.

Loss Function

To ovecome finite word-length effect, a modified loss,

$$l(\mathbf{Y}, \mathbf{P}) = \sum_{i=1}^{i=m} 10 \times \tan(0.499\pi \times |\mathbf{Y}(i) - \mathbf{P}(i)|), \quad (5)$$

is applied in practice and turn $l(\mathbf{Y}, \mathbf{P})$ into a bounded function.

Two loss functions, action loss and state loss, are defined and are summed to produce total loss, based on which learning is conducted.

Training, Validating, Testing

Training is conducted on a GPU with the RMSProp optimizer (Tieleman and Hinton 2012) under a learning rate of 0.0001. We train total loss on 9K samples for 201 epochs within which we validate error function on 1K samples every two epochs. If a lower error monitored during training, the model would be stored.

After training finished, the trained model will be tested on (Bosselut et al. 2018)’s 693 test set. One missing tolerance, compatible with their benchmark, is applied. It allows a prediction to have one missing item at most when compared with the label. Accuracy is measured as percentage of sentence predictions satisfying the rule.

Results

We independently repeat our experiments for 4 times and report accuracies on both action recognition and state change predition (Table 1). Note that (Bosselut et al. 2018) does not report the action accuracy.

Table 1: Test our model on Bosselut’s 693 test set

Our experiment No.		1	2	3	4
Action acc (%)		81.2	80.9	80.9	80.9
State acc (%)		**66.6**	**66.6**	**67.0**	**67.2**
Bosselut’s Model	Action acc (%)	N/A			
	State acc (%)	55			

Our results significantly outperform those in (Bosselut et al. 2018) under the same benchmark while using only a 15% subset of data for training.

Conclusion

We contribute a lightweight neural network with a novel cost function. With an improvement of 12% and significantly smaller training data used, our model outperforms a previous language understanding model where the goal is to recognize verbs and predict state changes.

References

Bosselut, A.; Levy, O.; Holtzman, A.; Ennis, C.; Fox, D.; and Choi, Y. 2018. Simulating action dynamics with neural process networks. In *The International Conference on Learning Representations*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *The Conference on Empirical Methods in Natural Language Processing*.

Graves, A.; Fernández, S.; Gomez, F.; and Schmidhuber, J. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, 369–376. ACM.

Hinton, G. E.; McClelland, J. L.; Rumelhart, D. E.; et al. 1984. *Distributed representations*. Carnegie-Mellon University Pittsburgh, PA.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.