LSTM + Capsule Nets for Definition Extraction

Haotian Xue
Department of Computer Science
Georgetown University
Washington, DC, U.S.A.
hx82@georgetown.edu

Zirong Chen
Department of Computer Science
Georgetown University
Washington, DC, U.S.A.
zc157@georgetown.edu

I. Abstract

Capsule Network was first purposed by Geoffrey Hinton et al. [2017] and it showed extraordinary capacity in image recognition. Investigated by Wei Zhao et al. [2018], Capsule Nets also showed great potential in text classification. Combined with LSTM purposed by Hochreiter and Schmidhuber [1997], Capsule Nets achieved competitive results over the compared baseline methods in Definition Extraction.

Keywords—Capsule Network, LSTM, Definition Extraction

II. Introduction

A. Definition Extraction

Definition Extraction is a general problem in NLP (Nature Language Processing). Our work mainly focuses on one subtask: Definition Detection. The basic goal of Definition Detection is to identify whether one sentence contains definitional relations or not.

An automatic and efficient method to manage this task will bring enormous convenience to academic life. This method can be applied to understand domain-specific documents even provide clues and advice to Question Answering System.

B. Previous Work

1) Rule-based Definition Extraction

Łukasz Degórski et al. [2008] introduced several rule-based methods to deal with definition extraction. These methods are based on human-defined patterns like single-token keywords (namely, aka, equal) and trivial hand-craft grammars (be defined as, be known as).

2) Boosting

Several classifiers based on boosting algorithms like AdaBoost, GradientBoost (Jerome H. Friedman [2001]) and XGBoost (Tianqi Chen and Carlos Guestrin [2016]) are applied in this task to improve experiment results.

3) Ensembles of Classifiers

According to Łukasz Degórski et al. [2008]'s experiments, ensembles of classifiers instead of single ones used to boost results. Thus, voting system participates in this task as well.

III. Our Model

Our model, depicted in Figure 1, is a variant of the capsule networks proposed in Sabour et al. [2017]. It consists three layers: Gated-Recurrent-Unit Layer, primary capsule layer and fully connected capsule layer. In the rest of this section, detailed information of key components will be elaborated. See figure 1.

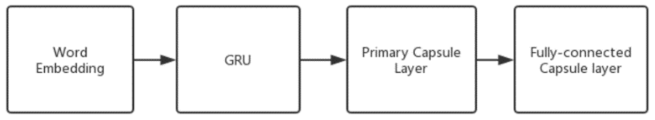


Figure 1 Model Layout

A. GRU Layer

This layer is a standard gated-recurrent-unit layer which handles sequential data and is proposed by Kyunghyun Cho et al. [2014] to address the problem of vanishing gradient.

Suppose x ∈ $R^{L\times d_w}$ denotes the input sentence representation where L is the sentence length and d_w is the dimension size of word embedding vectors. The formula of GRU layer is shown below:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\widetilde{h_t} = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t}$$

Where $x_t \in R^{d_w}$ is the V-dimensional vector responding to the $t-th$ word in original sentence and $h_t \in R^{d_h}$ is the d_h-dimensional vector responding to the output of $t-th$ hidden state in GRU layer. The output of GRU layer is $h \in R^{L\times d_h}$.

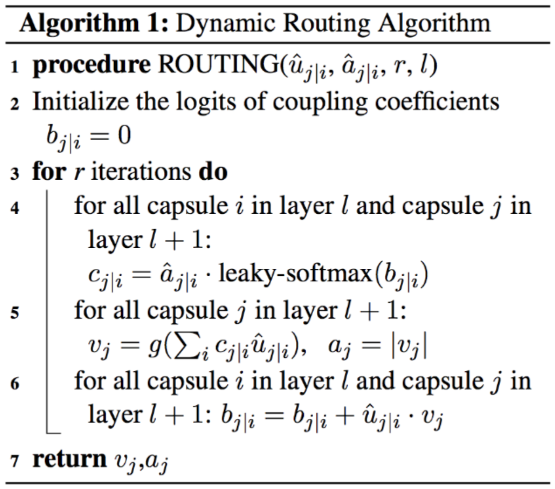
B. *Capsule Layer*

This layer replaces the scalar-in-scalar-out with vector-in-vector-out capsules to preserve the instantiated information of local word orders, syntactic and semantic word representations. The input of this layer is the output of GRU layer, $h \in R^{L \times d_h}$, where L can be seen as the initial number of capsules and d_h can be seen as the dimension of each capsule. And the output, $Cap_{out} \in R^{n_c \times d_c}$, where n_c is the output number of capsules, d_c is the output dimension size of a capsule. And both these two parameters are adjusted automatically through Dynamic Routing which proposed by Sabour et al. [2017].

1) *Dynamic Routing*

Dynamic Routing is used to iteratively learn the representation of each single capsule such that the network could automatically learn both the syntactic and semantic representation of a sentence from bottom-up(or part-whole). The input of Dynamic Routing is $\widehat{u_{j|i}}$, r and l, where $\widehat{u_{j|i}}$ represents for prediction vector from its child capsule i to the parent capsule j.

The whole computation process it shown in Figure2.



Algorithm 1: Dynamic Routing Algorithm

```
1 procedure ROUTING($\hat{u}_{j|i}$, $\hat{a}_{j|i}$, $r$, $l$)
2 Initialize the logits of coupling coefficients
    $b_{j|i} = 0$
3 for $r$ iterations do
4     for all capsule $i$ in layer $l$ and capsule $j$ in
        layer $l+1$:
        $c_{j|i} = \hat{a}_{j|i} \cdot \text{leaky-softmax}(b_{j|i})$
5     for all capsule $j$ in layer $l+1$:
        $v_j = g(\sum_i c_{j|i}\hat{u}_{j|i}), \quad a_j = |v_j|$
6     for all capsule $i$ in layer $l$ and capsule $j$ in
        layer $l+1$: $b_{j|i} = b_{j|i} + \hat{u}_{j|i} \cdot v_j$
7 return $v_j$,$a_j$
```

Figure 2 Dynamic Routing (from Zhao et al. 2018)

C. *Fully Connected Capsule Layer*

The output capsules of primary capsule layer are then flatten into a list of capsules and fed into a fully connected layer in which capsules are transformed by trainable matrix $W \in R^{(n_c \times d_c) \times n_{cat}}$, where n_{cat} is the number of categories, in this case of Definition Extraction, is two.

D. *The Architectures of Capsule Network*

The model layout is shown in Figure1. The model starts with an pre-trained word embedding layer which mappings each token in sentence to a 300-dimensional dense word vector, followed by a GRU layer where hidden dimension is 128($d_n = 128$). All other layers are capsule layers starting with a $L \times d_n$ primary layer and outputs a $n_c \times d_c$ capsules, followed by a fully-connected capsule layer.

IV. Experiment

A. *Dataset*

Dataset used in this task is released by Adobe SemEval2020 Task6 Subtask 1. This dataset consist of 16,659 training samples and 810 validation samples. After formatting these data, the general pattern of each sample is `[[sentence], label]`.

After looking into dataset several times, the distribution of training and test set is unbalanced. There are 5,569 positive and 11,090 negative samples in training set. Meanwhile, validation set has 285 positive and 525 negative samples. Thus, the ratio of positive and negative samples is ~1:2. For this reason, penalty for misclassification is utilized in serval implemented models, which boost results as well.

B. *Implement Methods*

The implementation of whole project is divided into two general sections: Statistical Machine Learning and Deep Learning.

1) *Statistical Machine Learning*

a) *Feature selection*

General features like stop words removal, stem words removal, lemmatization, punctuation removal, Bag of Words, Tf-idf are utilized in machine learning models. Due to too many distinct and domain-specific terms, n-gram cannot be extracted because of the limitation of devices.

b) *Hyper-parameters tuning*

AdaBoost: *n_estimators = 300*, *learning_rate = 1*

GradienBoost: *loss = 'deviance'*, *n_estimators = 300*

SGD: *loss = 'log'*, *penalty = 'elasticnet'*, *max_iter = 5000*, *learning_rate = 'adaptive'*,*eta0 = 0.1*, early_*stopping = True*

SVM(SVC): *kernel = 'rbf'*, *gamma = 'scale'*

XGBoost:*booster:'gbtree'*,*objective:'binary:logistic'*, *'max_depth': 20*, *lambda: 8*, *alpha: 8*, *subsample:0.75*, *colsample_bytree: 0.75*, *min_child_weight: 1.25*, *eta: 0.025*, *gamma:0.2*, *learning_rate:0.02*, *round: 1400*

2) *Deep Learning*

a) *Feature selection*

GloVe word2vector dictionaries with different sizes and dimensions are used in this task.

b) *Hyper parameters tuning*

LSTM + Capsule Nets: *num_epoch: 40*, *batch_size: 32*, *learing_rate: 3e-4*, *num_capsule: 10*, *dim_capsule: 16*, *num_routing: 5*

CNN: *num_epoch: 40*, *batch_size: 32*, *learing_rate: 3e-4*, *num_filter: 256*, *kernel_size:3*

CNN-Attention: *num_epoch: 40*, *batch_size: 8*, *learing_rate: 3e-4*, *num_filter: 256*, *kernel_size:3*

RNN: *num_epoch: 40*, *batch_size: 32*, *learing_rate: 3e-4*, *hidden_dim: 256*, *num_layers: 2*

Self-Attention: *num_epoch: 60*, *batch_size: 32*, *learing_rate: 3e-4*, *hidden_dim: 256*, *num_layers:4*, *num_heads:5*

FastText: *num_epoch: 30*, *batch_size: 32*, *learing_rate: 3e-4*, *hidden_dim: 128*

C. *Baseline Models*

In this experiment, we evaluate and compare our model with several widely used baseline methods: SVC for Statistical Machine Learning and CNN (Yoon Kim [2014]) for Deep Learning. The best results of classical models so far are XGBoost (Tianqi Chen and Carlos Guestrin [2016]) for Statistical Machine Learning and RNN-Attention (Ashish Vaswani et al. [2017]) for Deep Learning.

V. EXPERIEMNT RESULTS

A. *Evaluation Metrics*

F1_score, precision, recall, accuracy and AUC (Area Under Curve) are applied to evaluate our models. However, due to the distribution of positive and negative samples, we mainly consider F1_score and recall as performance standards.

B. *Feature Selection in Statistical Machine Learning*

Several general and commonly used features are utilized in ML Classifiers, but meanwhile, the affects brought by each feature are not clear to tell, especially when these features are applied in the same time. Due to time limit, only the comparison between BoW and Tf-idf has been done. See Figure 3.

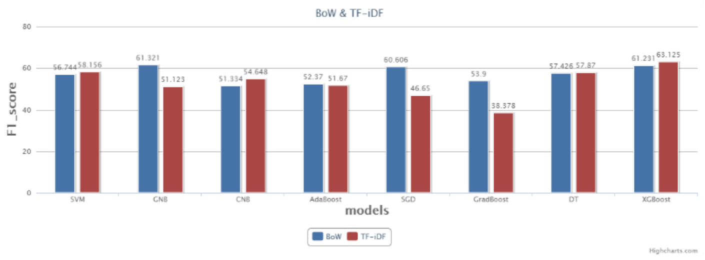


Figure 3 BoW vs Tf-idf

C. *Learning Curve*

XGBoost is an iteration-based boosting algorithm for classification and regression. Thus, it is significant to draw its learning curve. See Figure 4.

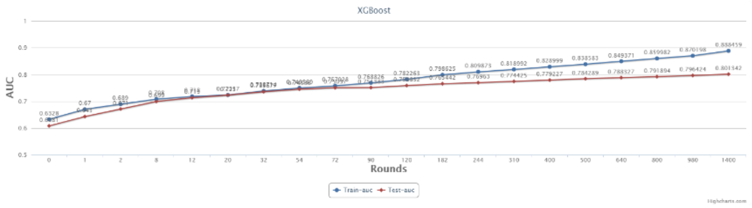
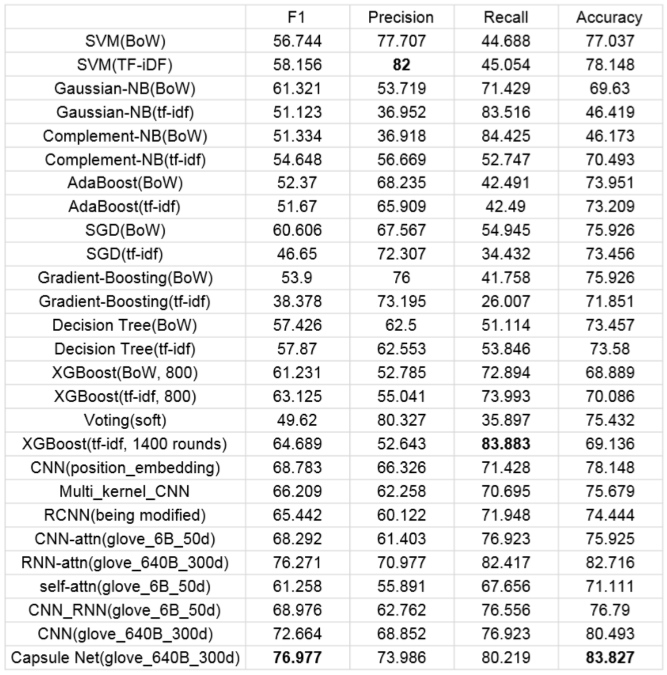


Figure 4 XGBoost Learning Curve

D. *Overall results*

Results of all models after hyper parameter tuning is generated after experiments. See Figure 5.



	F1	Precision	Recall	Accuracy
SVM(BoW)	56.744	77.707	44.688	77.037
SVM(TF-iDF)	58.156	**82**	45.054	78.148
Gaussian-NB(BoW)	61.321	53.719	71.429	69.63
Gaussian-NB(tf-idf)	51.123	36.952	83.516	46.419
Complement-NB(BoW)	51.334	36.918	84.425	46.173
Complement-NB(tf-idf)	54.648	56.669	52.747	70.493
AdaBoost(BoW)	52.37	68.235	42.491	73.951
AdaBoost(tf-idf)	51.67	65.909	42.49	73.209
SGD(BoW)	60.606	67.567	54.945	75.926
SGD(tf-idf)	46.65	72.307	34.432	73.456
Gradient-Boosting(BoW)	53.9	76	41.758	75.926
Gradient-Boosting(tf-idf)	38.378	73.195	26.007	71.851
Decision Tree(BoW)	57.426	62.5	51.114	73.457
Decision Tree(tf-idf)	57.87	62.553	53.846	73.58
XGBoost(BoW, 800)	61.231	52.785	72.894	68.889
XGBoost(tf-idf, 800)	63.125	55.041	73.993	70.086
Voting(soft)	49.62	80.327	35.897	75.432
XGBoost(tf-idf, 1400 rounds)	64.689	52.643	**83.883**	69.136
CNN(position_embedding)	68.783	66.326	71.428	78.148
Multi_kernel_CNN	66.209	62.258	70.695	75.679
RCNN(being modified)	65.442	60.122	71.948	74.444
CNN-attn(glove_6B_50d)	68.292	61.403	76.923	75.925
RNN-attn(glove_640B_300d)	76.271	70.977	82.417	82.716
self-attn(glove_6B_50d)	61.258	55.891	67.656	71.111
CNN_RNN(glove_6B_50d)	68.976	62.762	76.556	76.79
CNN(glove_640B_300d)	72.664	68.852	76.923	80.493
Capsule Net(glove_640B_300d)	**76.977**	73.986	80.219	**83.827**

Figure 5 Overall Results

VI. FUTURE WORK

What about the combination of Statistical Machine Learning models with Deep Learning models? Generally speaking, Statistical Machine Learning approaches have competitive results in terms of precision, while Deep Learning models have higher recall. So, it may be a good idea to replace last few dense layers in DL models with SVM or other Statistical Machine Learning classifiers.

VII. *RESULT ANALYSIS AND CONCLUSION*

According to the dataset itself, the definition of a definitional sentence can be highly confusing and even human can misunderstand some of these samples.

A. *Error Analysis*

Here are some mistakes our models make.

```
 ['DNA and RNA are made up of monomers known
as nucleotides', '0']

 ['Objectives arranged in this way are
 described as parfocal', '0']

 ['In doing so , monomers release water
 molecules as byproducts', '1']

 ['A flashbulb memory is an exceptionally
 clear recollection of an important event
 ( [ link ] )', '1']
```

From our point of view, external knowledge or explanation is needed for task.

B. Conclusion

The reason why self-attention do not perform well is that in this task, keywords recognition is more important than long term dependency. That is also the reason why CNN can outperform most classical DL models.

So, the conclusion is that keywords recognition is some kind of a more critical standard than other features to definition detection.

References

[1] Zhao, W., Ye, J., Yang, M., Lei, Z., Zhang, S. and Zhao, Z., 2018. Investigating capsule networks with dynamic routing for text classification. arXiv preprint arXiv:1804.00538.

[2] Sabour, S., Frosst, N. and Hinton, G.E., 2017. Dynamic routing between capsules. In Advances in neural information processing systems (pp. 3856-3866).

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[4] Kim, Y., 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

[5] Anke, L.E. and Schockaert, S., 2018, June. Syntactically aware neural architectures for definition extraction. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers) (pp. 378-385).

[6] Zhang, X., Zhao, J. and LeCun, Y., 2015. Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649-657).

[7] Lai, S., Xu, L., Liu, K. and Zhao, J., 2015, February. Recurrent convolutional neural networks for text classification. In Twenty-ninth AAAI conference on artificial intelligence.

[8] Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T., 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

[9] Zeng, D., Liu, K., Chen, Y. and Zhao, J., 2015, September. Distant supervision for relation extraction via piecewise convolutional neural networks. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 1753-1762).

[10] 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. arXiv preprint arXiv:1406.1078.

[11] Degórski, L., Marcinczuk, M. and Przepiórkowski, A., 2008, May. Definition Extraction Using a Sequential Combination of Baseline Grammars and Machine Learning Classifiers. In LREC.

[12] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[13] Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics, pp.1189-1232.

[14] Chen, T. and Guestrin, C., 2016, August. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794). ACM.