Evaluation of Machine Learning Methods on SPiCe

Ichinari Sato ICHINARI_SATO@SHINO.ECEI.TOHOKU.AC.JP
Graduate School of Information Sciences, Tohoku University
6-6-05, Aramakiaza Aoba, Aoba-ku, Sendai City, Miyagi, Japan

Kaizaburo Chubachi KAIZABURO_CHUBACHI@SHINO.ECEI.TOHOKU.AC.JP
School of Engineering, Tohoku University
6-6-05, Aramakiaza Aoba, Aoba-ku, Sendai City, Miyagi, Japan

Diptarama DIPTARAMA@SHINO.ECEI.TOHOKU.AC.JP
Graduate School of Information Sciences, Tohoku University
6-6-05, Aramakiaza Aoba, Aoba-ku, Sendai City, Miyagi, Japan

Abstract

In this paper, we introduce methods that we used to solve problems from the sequence prediction competition called SPiCe. We train a model from sequences in train data on each problem, and then predict a next symbol following each sequence in test data. We implement several methods to solve these problems. The experiment results show that XGBoost and neural network approaches have good performance overall.

Keywords: Language model, Neural Network, Ngram, XGBoost

1. Introduction

The Sequence PredIction ChallengE (SPiCe) is a competition where each contestant should predict a next symbol that follows a given sequence. The competition has 15 problems, and each problem consists of two sets of sequences generated from a model or obtained from real data. First, a set of sequences is given as training data from which the contestants should make a model. Then, the contestants submit the top five probable symbols following each sequence in the test data. The prediction score is calculated based on the predicted symbols and the answer symbols.

In order to tackle the competition, we evaluate the performance of several existing grammar prediction methods. We then choose the best method and hyperparameter for each problem based on the experiment results.

2. Prediction Methods

We will describe the methods used to build a language model, which predicts a symbol that will come after a given sequence.

We implement following models.

1. n-gram based model
2. Combined n-gram with spectral learning model
3. Recurrent neural network language model

4. XGBoost based model
5. Combined RNN with XGBoost model
6. Neural/n-gram hybrid model

The combined methods 2 and 5 output the five symbols with the highest probabilities that we add probabilities output by a method to probabilities by another. Appendix Table 2–4 show parameters of each model.. We will explain the details of methods 1, 3, 4, and 6.

2.1. n-gram Based Model

The n-gram based model is a basic language model. This model predicts a next symbol of a given sequence by using a histogram which is calculated from substrings of length n in the training data, where n can be considered as a hyperparameter. In natural language processing, bigram ($n = 2$) and trigram ($n = 3$) are used commonly. The higher value of n will increase the number of n-grams that do not appear in the data. This problem is called the zero frequency problem. In order to solve this problem, some smoothing techniques have been proposed such as Laplace Smoothing and Kneser-Ney Smoothing. However, we did not implement these smoothing methods in our implementation.

We used n-gram model based on the 3-gram sample program by SPiCe. We extended the implementation into 3–20-gram model, and used a linear combination of distributions of each n-gram to predict next symbol. We set the upper limit to 20, because the result of prediction did not improve when the upper limit was too large.

2.2. XGBoost Based Model

XGBoost is a tree boosting system, which has been used in many machine learning and data mining competitions and achieved many successful results (Chen and Guestrin, 2016). Tree boosting is a gradient boosting method that uses a decision tree or a regression tree as a base learner (Friedman, 2001). In tree boosting, we make an ensemble model and greedily add a tree with the parameter that minimizes a loss function.

We consider the problem as the multiclass classification task that classifies input prefixes into subsequent symbols. We trained XGBoost to fit such task, then we got probabilities of all symbols and output top 5 of those in high probability order. The input features are the last 10 symbols encoded as 1-hot-vectors, that have value 1 on the symbol and 0 otherwise. To avoid overfitting, we used 3-fold cross validation and stopped training when the error rate of the validation set became a minimum. In the test phase, we averaged outputs of the classifiers trained on each fold.

2.3. Recurrent Neural Network Language Model

Bengio et al. (2003) proposed feed-forward neural network language model which is an application of deep learning for language model. Mikolov et al. (2010) showed that recurrent neural network (RNN) is more effective than a feed-forward neural network model for the language model. Moreover, an improvement of RNN called Long Short Term Memory network (LSTM) got a high score on a natural language corpus dataset called Penn Treebank (Zaremba et al., 2014).

LSTM receives one vector expressing a character or word as input and output one vector as a predictive distribution of next symbol. Past inputs remain for a long and short term as states in LSTM. While a language model using neural network has good performance, the model needs a huge amount of resources to compute. Moreover, there are lots of hyperparameters that need to be set in this model. First, we encoded a symbol in an input sequence as a 1-hot-vector. Then, we input the sequence of 1-hot-vectors into the language model. We used 3 layers that consist of 256 nodes, that are 2 layers of LSTM and 1 layer of full-linked layer. We used Back-Propagation Through Time (BPTT) as a learning method. See Table 4 in the appendix for hyperparameter.

2.4. Neural/n-gram Hybrid Model

Neubig and Dyer (2016) proposed a new model that combines neural language model and n-gram language model. They show that their model has good performance as a language model and got a higher score than existing algorithms on Penn Treebank. Moreover, this model can be implemented easily.

The language model for neural/n-gram hybrid model is represented by the following expression, where J is the number of symbols and N is the maximum n-gram length.

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_J \end{bmatrix} = \begin{bmatrix} d_{1,1} & \dots & d_{1,N} & 1 & 0 & \dots & 0 \\ d_{2,1} & \dots & d_{2,N} & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{J,1} & \dots & d_{J,N} & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{J+N} \end{bmatrix}$$

The vector $\boldsymbol{p}$ is a probabilistic distribution used to predict a next symbol for the prefix. We calculate by $\boldsymbol{p} = D\boldsymbol{\lambda}$. p_i is a probability of the i-th symbol as a next symbol, and $|\boldsymbol{p}| = 1$. The $J \times (N + J)$ matrix D consist of a $J \times N$ n-gram matrix and a $J \times J$ identity matrix. $d_{j,n}$ expresses a probability of the j-th symbol to appear after the previous $(n-1)$-gram. The vector $\boldsymbol{\lambda}$ is a coefficient learned to make $D\boldsymbol{\lambda}$ close to $\boldsymbol{p}$ by RNN.

This model has a process called *block dropout* in the training step. Since an n-gram matrix is already a sufficient approximation in an early step of the training process, RNN learns a part of $\boldsymbol{\lambda}$ which only uses the n-gram matrix. In this case a language model cannot get good performance. The hybrid model therefore uses block dropout, which drops subsets of nodes in the network while standard dropout drops single nodes or links. In this models, we replaced randomly the n-gram matrix by a zero matrix during the training step. At the beginning of the learning process, the block dropout reduces the effect of the n-gram matrix and encourages a part of $\boldsymbol{\lambda}$ using identity matrix to learn.

The hybrid model needs hyperparameters of both n-gram based model (§2.1) and RNNLM (§2.3). We used $n \in \{3, 5, 22\}$ in the n-gram part and use the same parameters as RNNLM (Table 4 in appendix) in the RNN part. Although Neubig and Dyer (2016) used modified Kneser-Ney smoothing as the best performance model in their paper, we used a simple n-gram model without smoothing in SPiCe.

3. Experiment

We evaluate the performance of above methods on solving the problems of SPiCe. The experiment results are shown in Table 5. The scores in the table are calculated by using public test score on the SPiCe system. The *rand* column shows average scores of the random baseline that randomly chooses 5 symbols from the given alphabet. We describe max score truncated beyond the third decimal point except random baseline in the table.

Since XGBoost uses much memory resource, our machine's memory overflowed on Problem 11. The same issue occured in the hybrid model because Problem 11 has large alphabet size. We adjusted a hyperparameter, BPTT length, from 35 to 15 on only Problem 11. XGBoost has good performances on solving Problems 5, 7, and 10. In Problems 4 and 15, the score differences between XGBoost and other methods are little although XGBoost has the highest scores. The RNN or the hybrid models have the highest scores on other problems.

Since combined n-gram and spectral learning method (*Comb1*) has a better total score, we combined XGBoost and RNN (*Comb2*) expected to get better score similarly. However, the total score of *Comb2* is lower than XGBoost. The hybrid models of $N \in \{3, 5\}$ have the total score over 10. The reasons of the score are that the hybrid model is good at Problems 6, 12 and 13, and it keeps the score of the problems which RNN is good at.

4. Conclusion

We used many approaches in SPiCe. XGBoost and neural/n-gram hybrid models got higher public test scores than the other models. Although the hybrid model outperformed XGBoost model about the total of public test score, there are problems in which XGBoost model outperformed hybrid model. At last, we submitted prediction results from XGBoost model and neural/n-gram hybrid model depending on the public test score.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155, 2003.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

Graham Neubig and Chris Dyer. Generalizing and hybridizing count-based and neural language models. *CoRR*, 2016. URL http://arxiv.org/abs/1606.00499.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network negularization. *CoRR*, abs/1409.2329, 2014. URL http://arxiv.org/abs/1409.2329.

Appendix

problem	N_{train}	N_{test}	$\|\Sigma\|$	$\|\Sigma_{train}\|$	avg_length	$line\_variety$
0	20000	1000	4	4	7.219	0.522
1	20000	5000	20	20	41.229	0.449
2	20000	5000	10	10	41.727	0.311
3	20000	5000	10	10	41.793	0.313
4	5987	748	33	33	7.466	0.863
5	33654	5000	49	49	119.336	0.239
6	5000	5000	60	24	24.911	0.592
7	65438	5000	20	20	59.120	0.281
8	13903	5000	48	48	30.916	0.474
9	5000	5000	11	11	25.543	0.398
10	54932	4847	20	20	35.807	0.452
11	32384	4049	6093	6093	25.543	0.914
12	200000	3000	21	21	28.739	0.632
13	26544	3318	665	665	7.659	0.894
14	10000	5000	27	27	40.846	0.417
15	50000	5000	32	32	49.648	0.537

Table 1: The properties of each dataset. N_{train} and N_{test} denote the number of sequences on the train and public test dataset respectively. $|\Sigma|$ denotes the alphabet size and $|\Sigma_{train}|$ denotes the number of symbols that appear in the train data. avg_length denotes the average length of the sequences and $line\_variety$ is the average ratio of the number of symbols that appear in a sequence with the length of the sequence.

parameter	value
rank	17
row	20
column	20
version	classic

Table 2: Spectral Learning parameters

parameter	value
step size shrinkage	0.1
subsample	0.5
colsample bytree	0.6
max depth	10
maximum delta step	10

Table 3: XGBoost parameters

parameter	value
epoch	20
unit	256
batchsize	20
BPTT length	35
grad clip	5
optimizer	Adam

Table 4: RNN parameters

	rand	Ngrams	Comb1	XGB	RNN	Comb2	Hybrid (N=3)	Hybrid (N=5)	Hybrid (N=22)
0	0.771	0.977	0.962	0.985	0.920	0.914	**0.990**	0.986	0.952
1	0.380	0.836	0.836	0.879	**0.914**	0.901	0.911	0.910	0.875
2	0.501	0.822	0.821	0.888	**0.913**	0.911	0.910	0.909	0.860
3	0.500	0.780	0.779	0.848	0.881	0.881	**0.885**	0.884	0.866
4	0.082	0.554	0.551	**0.590**	0.589	0.492	0.564	0.575	0.118
5	0.057	0.651	0.745	**0.787**	0.750	0.775	0.767	0.769	0.767
6	0.068	0.744	0.729	0.698	0.729	0.786	0.852	0.867	**0.871**
7	0.139	0.668	0.669	**0.783**	0.577	0.755	0.630	0.644	0.631
8	0.060	0.593	0.603	0.609	0.637	0.579	0.642	0.641	**0.643**
9	0.308	0.895	0.875	0.890	0.922	0.917	**0.956**	0.928	0.899
10	0.140	0.465	0.466	**0.595**	0.559	0.577	0.542	0.528	0.536
11	0.000	0.335	0.354	-	**0.509**	-	0.489	0.503	0.506
12	0.404	0.728	0.645	0.623	0.677	0.663	0.770	**0.775**	0.759
13	0.004	0.429	0.479	0.400	0.455	0.406	**0.496**	0.481	0.001
14	0.129	0.331	0.343	0.376	0.371	**0.402**	0.370	0.365	0.368
15	0.138	0.259	0.259	**0.263**	0.155	0.227	0.260	0.261	0.251
total	2.910	9.090	9.155	9.299	9.670	9.272	10.045	10.042	8.951

Table 5: Evaluation of learning methods on solving the public test datasets. *rand* is the average (20 times) of scores when we answer the problem with random symbols. *Ngrams* is a combination from 3-gram to 20-gram model. *Comb1* is a combination of n-gram based model and Spectral Learning model. *Comb2* is a combination of XGBoost model and RNN model. We add each symbol probability and take the 5 probable symbols in *Comb1* and *Comb2*.