Evaluation of Machine Learning Methods on SPiCe

Ichinari Sato[1], Kaizaburo Chubachi[2], Diptarama[1]

[1]Graduate School of Information Sciences, Tohoku University, Japan
[2]School of Engineering, Tohoku University, Japan

Team: ushitora

Agenda

- Used methods
  - XGBoost
  - LSTM
- Mixture of Distributions Language Model [Neubig, Dyer 2016]
  - Neural/n-gram Hybrid Language Model [Neubig, Dyer 2016]

At The Beginning Of SPiCe

Used Methods

- n-gram based
- n-gram & spectral learning combined[Balle, 2013]
- XGBoost based [Chen& Guestrin, 2016]
- Long Short Term Memory [Zaremba et al., 2014]
- XGBoost & LSTM combined
- Neural/n-gram hybrid [Neubig & Dyer, 2016]

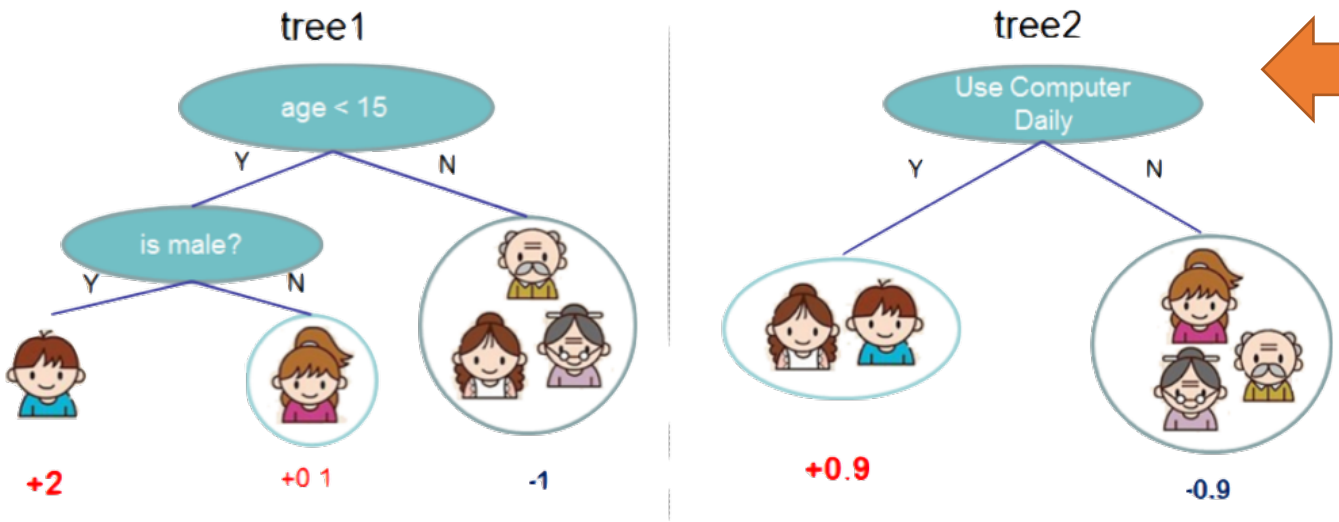
eXtreme Gradient Boosting (XGBoost)

[Chen & Guestrin, 2016]

- XGBoost is a tree boosting system.

Tree boosting

Tree Ensemble Model



The output is the sum of predictions from each tree

$$f(\ \ ) = 2 + 0.9 = 2.9$$

Training Phase

Add a tree

that minimize a **loss function**

Loss function:

- Log loss
- Mean squared error

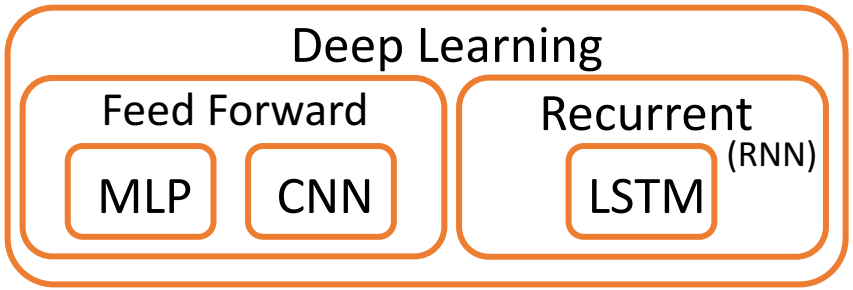
etc.

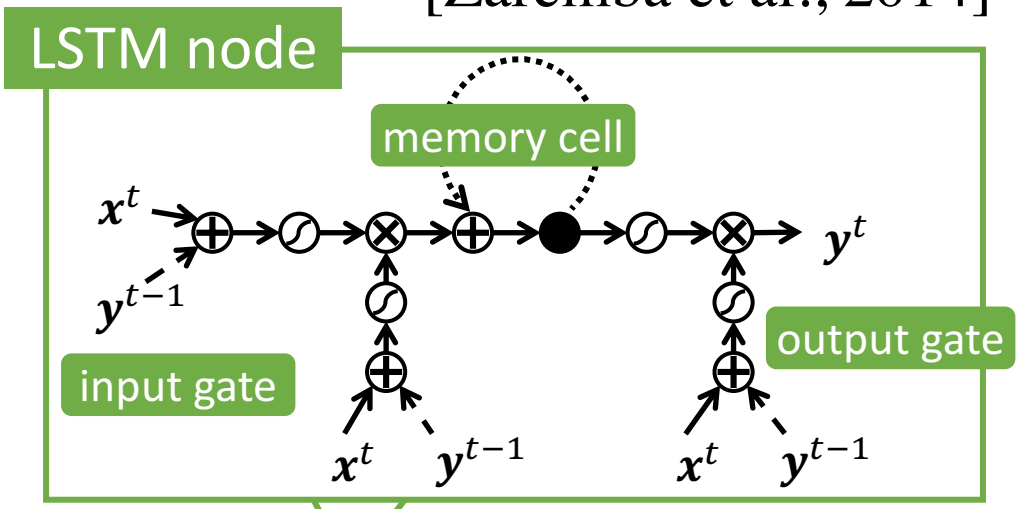
XGBoost for Language Model

- The input is the last 10 symbols encoded as 1-hot-vector.

Long Short Term Memory (LSTM) for LM

[Zaremba et al., 2014]

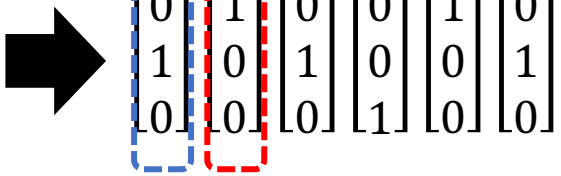




train

0 1 2 3 4 5

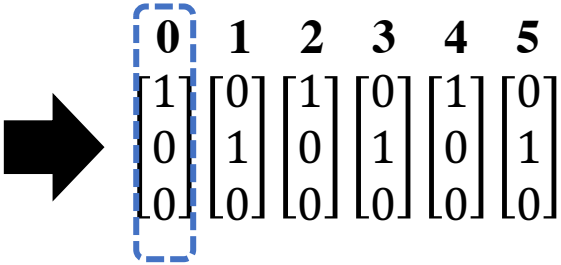
212312

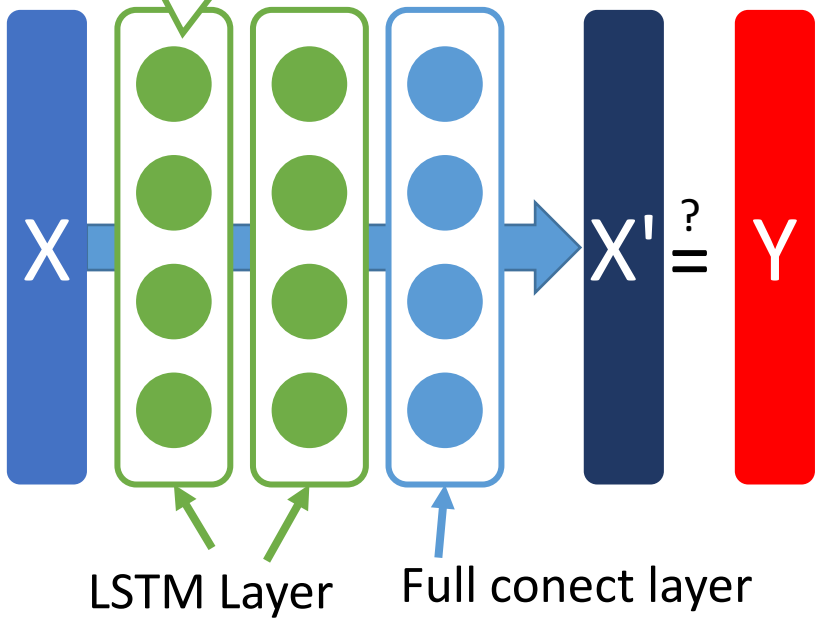


predict

0 1 2 3 4 5

121212 ?





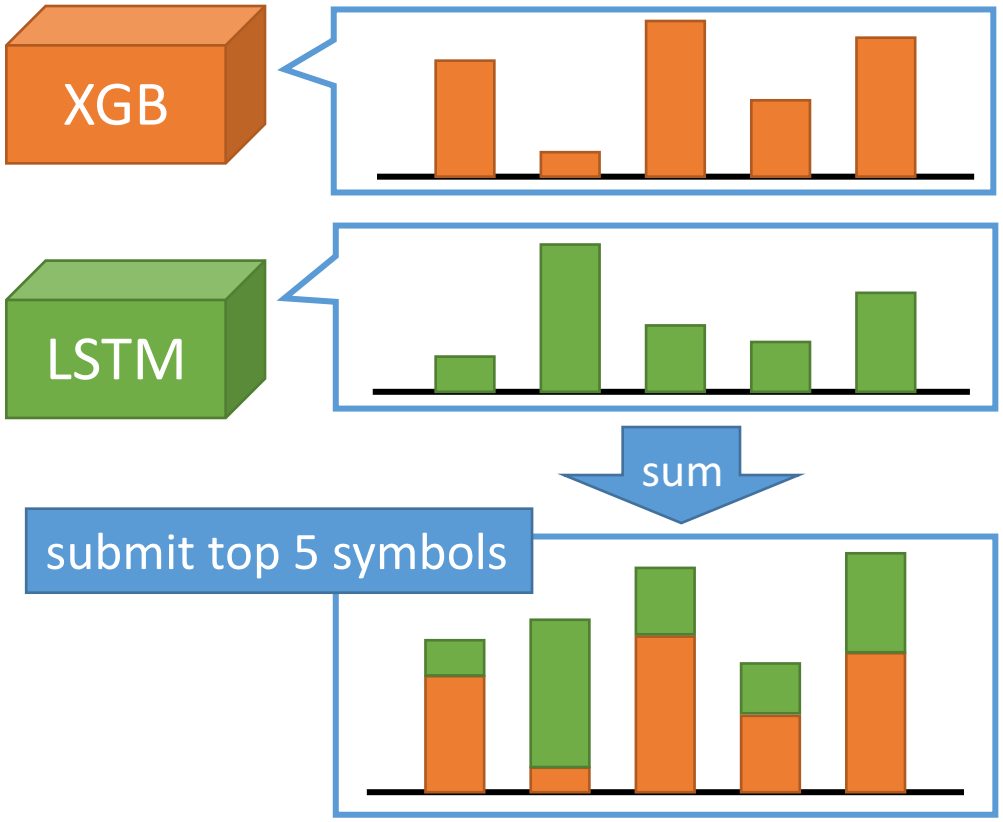
Public Test Score (1)

P	random	Ngrams	XGB	LSTM
0	0.771	0.969	**0.985**	0.920
1	0.380	0.836	0.879	**0.914**
2	0.501	0.822	0.888	**0.913**
3	0.500	0.780	0.848	**0.882**
4	0.082	0.554	**0.590**	0.589
5	0.057	0.651	**0.787**	0.751
6	0.068	**0.744**	0.698	0.729
7	0.139	0.668	**0.783**	0.589
8	0.060	0.593	0.609	**0.637**
9	0.308	0.895	0.890	**0.922**
10	0.140	0.465	**0.595**	0.559
11	0.000	0.335	---	**0.509**
12	0.404	**0.728**	0.623	0.677
13	0.004	0.429	0.400	**0.473**
14	0.129	0.331	**0.376**	0.371
15	0.138	0.259	**0.263**	0.155
total	2.910	9.090	9.229	**9.670**

more difficult

memory overflow

How To Combine



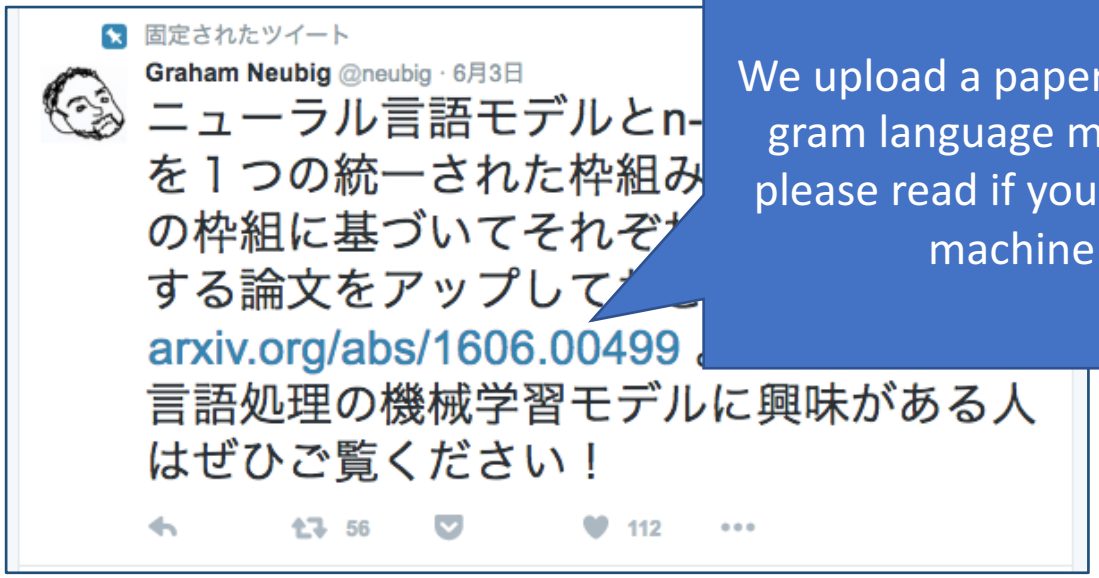
- Simple Linear Sum
  - n-gram & spectral learning is good.
  - XGB & LSTM is not good.
- We must find a better ensemble method.
- What can we do?

P	XGB	LSTM	XGB+LSTM
0	**0.985**	0.920	0.914
1	0.879	**0.914**	0.901
2	0.888	**0.913**	0.911
3	0.848	**0.882**	0.881
4	**0.590**	0.589	0.492
5	**0.787**	0.751	0.775
6	0.698	0.729	**0.786**
7	**0.783**	0.589	0.755
8	0.609	**0.637**	0.579
9	0.890	**0.922**	0.917
10	**0.595**	0.559	0.577
11	---	**0.509**	---
12	0.623	**0.677**	0.663
13	0.400	**0.473**	0.406
14	0.376	0.371	**0.402**
15	**0.263**	0.155	0.227
total	9.229	**9.670**	9.272

We Got A Chance

At June 3rd

Graham Neubig tweet "we published a paper of new language model."



We upload a paper which formularize neural and n-gram language model to one general framework. please read if you interested on language model or machine learning model for NLP.

Generalizing and Hybridizing Count-based and Neural Language Models [EMNLP 2016]

Mixture of Distribution LM (MODLM)

[Neubig & Dyer, 2016]

$$P(\tilde{w}|\boldsymbol{c}) = \sum_{k=1}^{K} \underbrace{\lambda_k(\boldsymbol{c})}_{\text{weight}} \underbrace{P_k(\tilde{w}|\boldsymbol{c})}_{\text{prediction distribution}}$$

($\boldsymbol{c}$ is context, $\boldsymbol{c} = w_1, w_2, \cdots, w_m$)

n-gram LM

$|\Sigma| = 3, n = 4$

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} d_{1,1} & d_{1,2} & d_{1,3} & d_{1,4} \\ d_{2,1} & d_{2,2} & d_{2,3} & d_{2,4} \\ d_{3,1} & d_{3,2} & d_{3,3} & d_{3,4} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix}$$

1-gram 2-gram 3-gram 4-gram heuristic

Neural Net LM

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

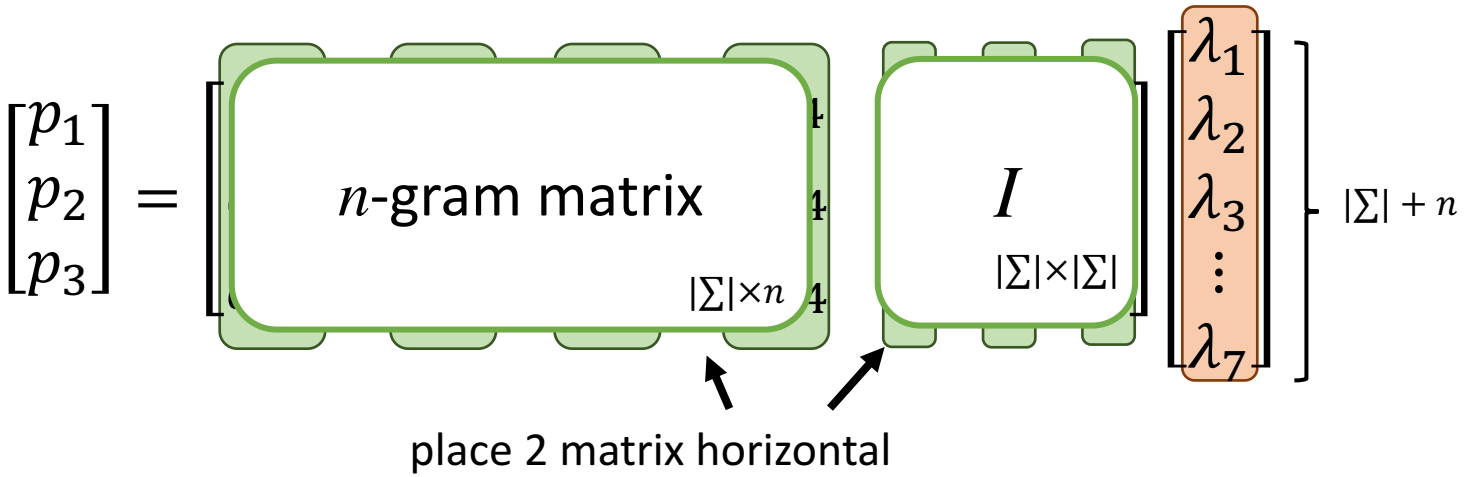
Kronecker δ distributions for each symbol

learning target

Let's combine!

Neural/*n*-gram Hybrid LM

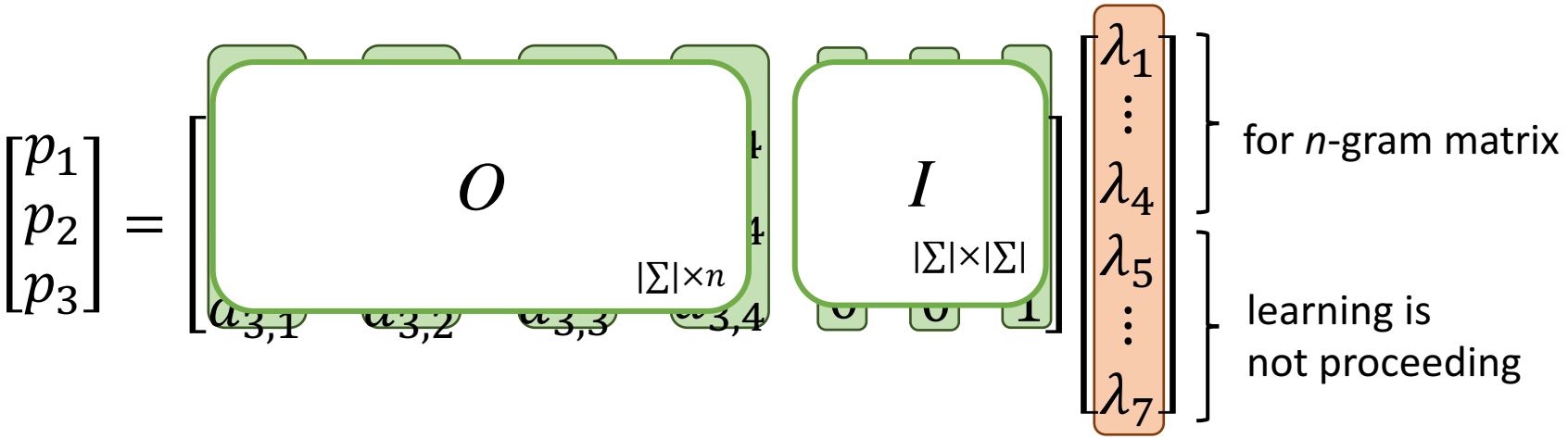
[Neubig, Dyer 2016]



block dropout

When this model learns λ, a part of λ cannot proceed to learn.

randomly drop out n-gram matrix (50%)



Public Test Score (2)

- total score (exclude Problem 11)

  LSTM < XGBoost < Hybrid
  9.161 9.229 9.556

- When we submitted to private test,

  we chose XGB or Hybrid by problem.

Is final result also higher?

P	XGB	LSTM	Hybrid	
1	0.879	**0.914**	0.911	Hybrid
2	0.888	**0.913**	0.910	Hybrid
3	0.848	0.882	**0.885**	Hybrid
4	**0.590**	0.589	0.564	XGB
5	**0.787**	0.751	0.767	XGB
6	0.698	0.729	**0.852**	Hybrid
7	**0.783**	0.589	0.630	XGB
8	0.609	0.637	**0.642**	Hybrid
9	0.890	0.922	**0.956**	Hybrid
10	**0.595**	0.559	0.542	XGB
11	---	**0.509**	0.489	Hybrid
12	0.623	0.677	**0.770**	Hybrid
13	0.400	0.473	**0.496**	Hybrid
14	**0.376**	0.371	0.370	XGB
15	**0.263**	0.155	0.260	XGB
total	9.229	9.670	**10.045**	

Final Result

Almost all scores is decrease from public to private.

We think that our model was over-fitting.

Finally, our public test rank is 2nd
and private tests rank is 3rd.

Position	Team Name
1	shib
2	ushitora
3	ToBeWhatYouWhatToBe
4	Markov_s_Principle
5	vha

Position	Team Name
1	shib
2	ToBeWhatYouWhatToBe
3	ushitora
4	Markov_s_Principle
5	zzzzzzzz

P	public	private	model
1	0.9146	0.9135	Hybrid
2	0.9137	0.9083	Hybrid
3	0.8853	0.8862	Hybrid
4	0.6060	0.5514	XGB
5	0.7873	0.5514	XGB
6	0.8719	0.8364	Hybrid
7	0.7832	0.7846	XGB
8	0.6431	0.5890	Hybrid
9	0.9563	0.9353	Hybrid
10	0.5960	0.5519	XGB
11	0.5096	0.4265	Hybrid
12	0.7751	0.7629	Hybrid
13	0.4959	0.3834	Hybrid
14	0.4024	0.3681	XGB
15	0.2765	0.2609	XGB
total	10.4169	9.7098	

Conclusion

- We tried some methods including a new method.
- The hybrid model got the best score in public test.
- Our models were over-fitting.