OOV Handling by Learning Subword using CNN based N-grams

Yonathan Purbo Santosa

Institut für Informatik
Rheinische Friedrich-Wilhelms-Universität Bonn

November 13, 2019

Overview

Introduction

Why OOV handling is needed?

- Word embedding needs large corpus
- Language itself is creative and changing overtime [1, 2];
- Some word embedding have no OOV handling method [3, 4, 5]; and
- Improvements should be able to be achieved over using random or unknown embedding for OOV.

Introduction

Previous state-of-the-art Mimick

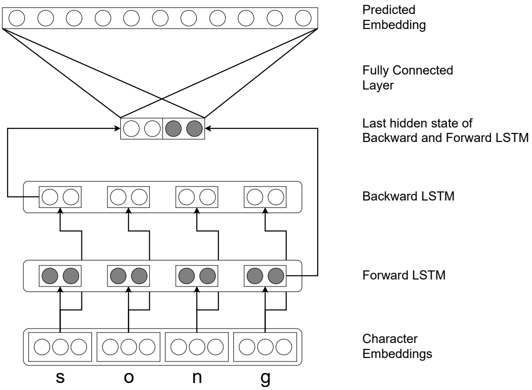


Figure: MIMICK Architecture

Introduction

What could be the problem with Mimick?

- MIMICK used the last hidden state of bi-LSTM to infer OOV embedding [6];
- By default, bi-LSTM has cell gate that could drop previous information if there is a trigger from the input or previous hidden state; and
- The last hidden state only encoded the recent sequence if there is several subsequence that is important.

Objective

- Similar with MIMICK, this problem is tackled from quasi-generative perspective
- Instead of using bi-LSTM, CNN that based on n-grams can be used to learn character sequence [7].
- Given a kernel K, an input that has similar features to the kernel can still gives a high response.
- The fully-connected can be used to learn the embedding from the max-over-time pooled features

CNN N-grams for OOV handling

- Given a word and its embedding (w_i, e_i)
- The word is transformed into sequence of character embedding $g_i \in \mathcal{G}$

$$h : w_i \rightarrow [c_1, c_2, \dots, c_n] \rightarrow [g_1, g_2, \dots, g_n]$$

- The sequence of character embedding then processed by the OOV handling model to predict the embedding $\tilde{e}_i$

$$OOV_{model}(w_i) = \tilde{e}_i$$

- The original word embedding were randomly split into 80% : 20% for train and validation split.

CNN N-grams for OOV handling

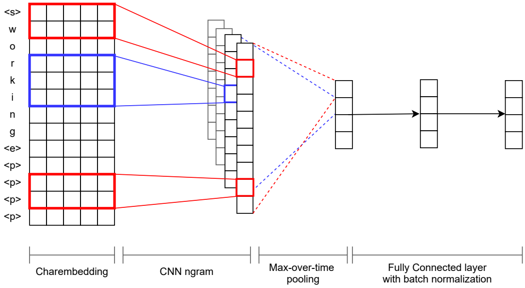


Figure: CNN N-grams for OOV Handling Architecture

- $[2, 3, 4, 5, 6, 7]$-grams were used as the feature extractor
- For each grams $\{20, 50, 100\}$ number of features were used
- ReLU activation function was used before the max-over-time pooling
- Using 50% dropout

CNN N-grams for OOV handling

- The predicted embedding was used for calculating the deviation from the original embedding e_i by using MSE

$$\mathcal{L} = \|e_i - \tilde{e}_i\|_2^2$$

- The kernel and other parameters were updated with learning rate $\eta = 0.1$

Evaluation on Downstream Tasks

Part-of-Speech Tagging

- Using brown POStagging datasets from NLTK[1]
- Following POS-tagger model from Wang et al. (2015) [8]
- Sentence length limited to 5 words
- Longer sentences are randomly sliced into 5 words and shorter sentences are padded to make it 5 words long

[1] Available at `https://www.nltk.org/`

Evaluation on Downstream Tasks

Part-of-Speech Tagging

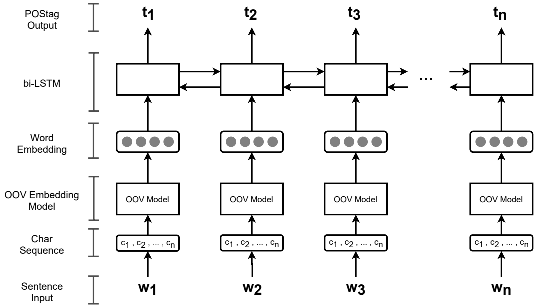


Figure: POS-tagger Model

Evaluation on Downstream Tasks

Part-of-Speech Tagging Result

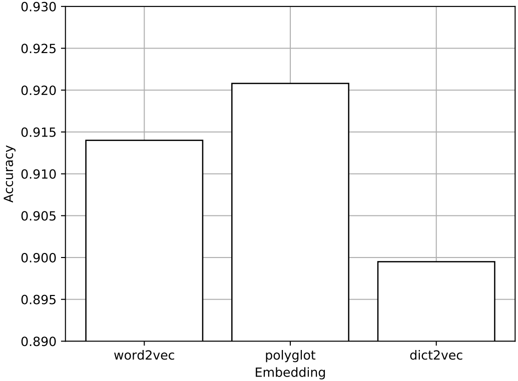
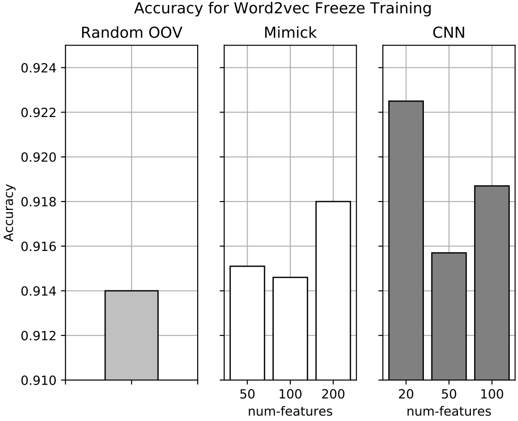


Figure: Random OOV Embedding

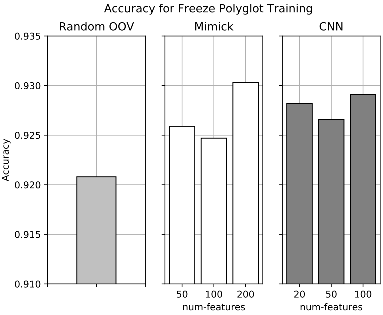
Evaluation on Downstream Tasks

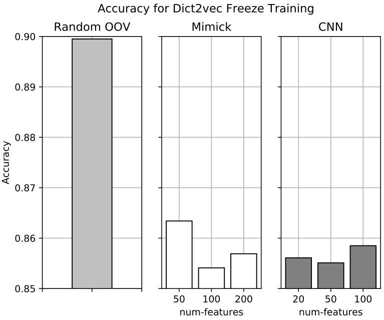
Part-of-Speech Tagging Result



Evaluation on Downstream Tasks

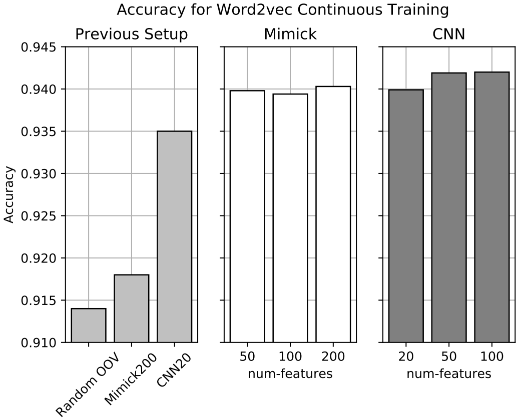
Part-of-Speech Tagging Result





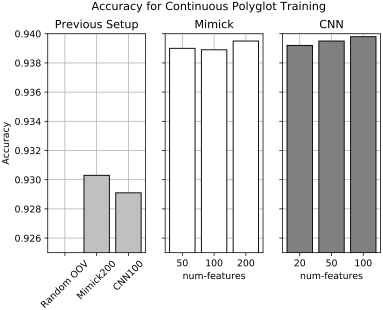
Evaluation on Downstream Tasks

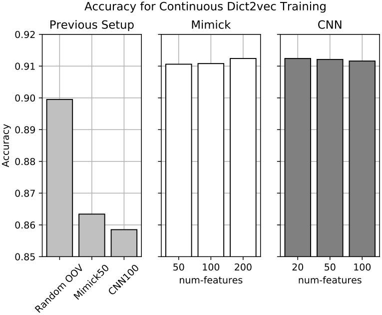
Part-of-Speech Tagging Result



Evaluation on Downstream Tasks

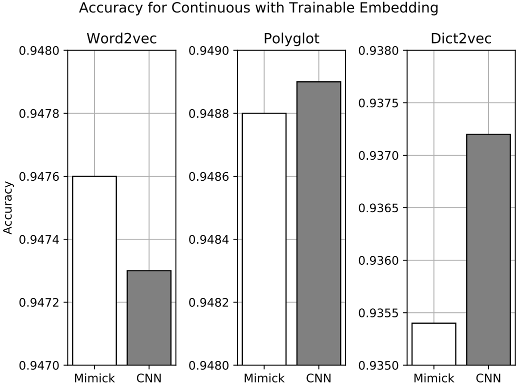
Part-of-Speech Tagging Result





Evaluation on Downstream Tasks

Part-of-Speech Tagging Result



Evaluation on Downstream Tasks

Word Similarity

- Using 12 different datasets to increase the data test size
- Only OOV embedding was inferred from the OOV handling model
- The cosine distance of a given pairs was calculated and compared with the dataset with Spearman's correlation coefficient.

Evaluation on Downstream Tasks

Word Similarity Results

Table: Word Similarity Task Results (word2vec)

Dataset	OOV	Invocab	OOV Ratio	Mimick*	CNN*
card	989	317	75.73%	**52.27**	27.25
mc30	4	35	10.26%	748.11	**758.79**
men	83	668	11.05%	**672.54**	664.64
mturk287	97	402	19.44%	518.15	**519.76**
mturk771	18	1095	1.62%	**657.93**	657.04
rg65	2	46	4.17%	709.77	**732.61**
rwstanford	1494	1457	50.63%	**233.83**	217.47
simlex	20	1008	1.95%	422.77	**425.90**
simverb	90	737	10.88%	**302.37**	292.91
verb143	4	113	3.42%	467.60	**478.25**
wordsim	15	422	3.43%	**658.92**	648.25
yp130	6	141	4.08%	**460.32**	443.22
			average	**492.05**	488.84

* multiplied by 1000

Evaluation on Downstream Tasks

Word Similarity Results

Table: Word Similarity Task Results (polyglot)

Dataset	OOV	Invocab	OOV Ratio	Mimick*	CNN*
card	864	442	66.16%	**128.93**	114.11
mc30	1	38	2.56%	605.25	605.25
men	14	737	1.86%	490.57	**492.17**
mturk287	76	423	15.23%	443.31	**458.81**
mturk771	3	1110	0.27%	**432.48**	432.20
rg65	1	47	2.08%	531.59	**524.77**
rwstanford	999	1952	33.85%	272.33	**290.78**
simlex	4	1024	0.39%	232.20	**234.16**
simverb	53	774	6.41%	**137.01**	134.42
verb143	0	117	0.00%	335.81	335.81
wordsim	0	437	0.00%	412.83	412.83
yp130	5	142	3.40%	**44.76**	44.62
			average	338.92	**339.99**

* multiplied by 1000

Evaluation on Downstream Tasks

Word Similarity Results

Table: Word Similarity Task Results (Dict2vec)

Dataset	**OOV**	**Invocab**	**OOV Ratio**	**Random***	**Mimick***	**CNN***
card	828	478	63.40%	48.07	80.89	**95.32**
mc30	0	39	0.00%	847.57	847.57	847.57
men	1	750	0.13%	713.16	723.63	**723.89**
mturk287	2	497	0.40%	652.27	**655.32**	653.13
mturk771	0	1113	0.00%	683.91	683.91	683.91
rg65	0	48	0.00%	832.86	832.86	832.86
rwstanford	619	2332	20.98%	214.60	**403.79**	400.27
simlex	3	1025	0.29%	454.80	**460.66**	459.87
simverb	24	803	2.90%	375.15	390.09	**393.39**
verb143	0	117	0.00%	187.82	187.82	187.82
wordsim	18	419	4.12%	642.71	718.72	**723.72**
yp130	2	145	1.36%	577.76	621.38	**621.75**
			average	519.22	550.55	**551.96**

* multiplied by 1000

Conclusion

- Machine learning can be used to generate OOV embedding
- Compared to random OOV embedding, machine learning method has higher performance in downstream tasks
- CNN generally performs better than bi-LSTM for POS-tagging and word similarity tasks
  - MIMICK has higher accuracy in POS-tagging: the pre-trained embedding & the OOV handling model non-trainable
  - CNN N-grams has higher accuracy in POS-tagging: both pre-trained embedding and the OOV handling model trainable
  - Despite that MIMICK preforms better *out-of-the-box*, this was not the case for word similarity task

Future Works

- A function that can generate an entire embedding
- Save space (e.g. word2vec 3 million words and phrases is around 2GB)
- Save computational needs for new entries cases

Thank you for your attention!

References

Julie C. Forrester (2008)
A Brief Overview of English as a Language in Change
Canadian Center of Science and Education 2008, Vol. 4(4)

Daniel Jurafsky and James H. Martin (2009)
Speech and Language Processing (3rd Edition)
Prentice-Hall, Inc.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena (2010)
Polyglot: Distributed Word Representations for Multilingual NLP
Proceedings of the Seventeenth Conference on Computational Natural Language Learning
Association for Computational Linguistics

Julien Tissier, Christophe Gravier, and Amaury Habrard (2017)
Dict2vec : Learning Word Embeddings using Lexical Dictionaries
Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 254–263
Association for Computational Linguistics

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean (2013)
Efficient Estimation of Word Representations in Vector Space
1st International Conference on Learning Representations, ICLR 2013

References

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein (2017)
Mimicking Word Embeddings using Subword RNNs
Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing

Yoon Kim (2014)
Convolutional Neural Networks for Sentence Classification
Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1746–1751

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis (2015)
Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation
Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing 1520–1530