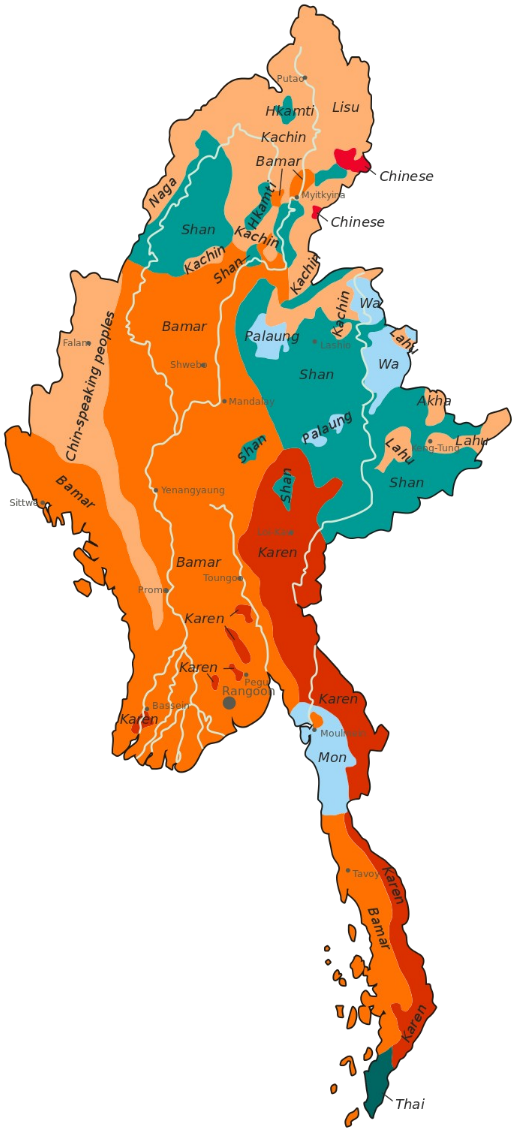
NLP R&D for Low Resources

Ye Kyaw Thu,
Visiting Professor, LST, NECTEC, Thailand

Why NLP for Ethnic Languages

- Approximately a hundred languages spoken in Myanmar
- Six language families: Sino-Tibetan, Austro-Asiatic, Tai-Kadai, Indo-European, Austronesian and Hmong-Mien (+ Myanmar Sign language and Braille)
- Aid human-human, human-machine (e.g. speech recognition, speech synthesis, question and answering) and for language understanding

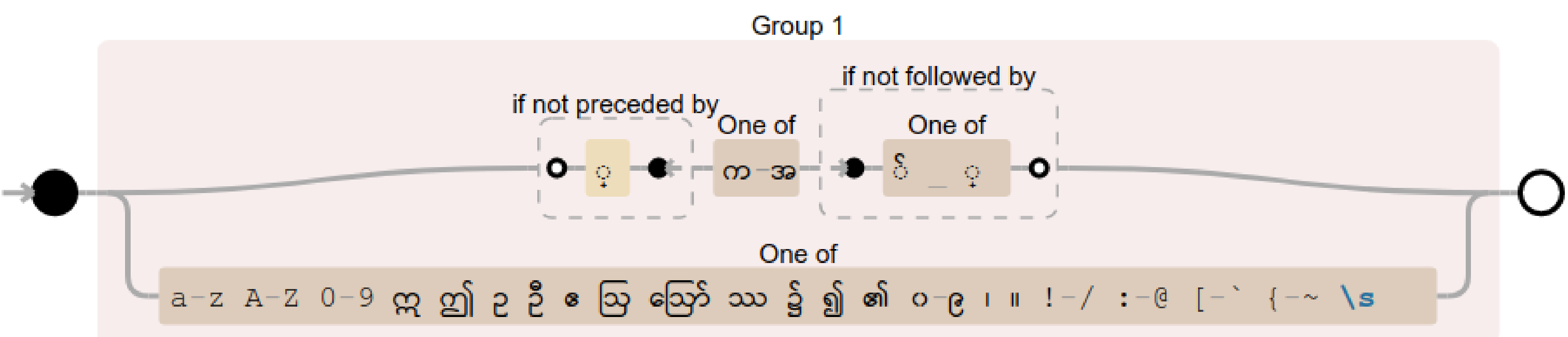
Why NLP for Ethnic Languages



- For local education
- For economic of each region
- For example, Kahin-Chinese machine translation, Rakhine-Bengali voice to voice translation, Shan-Bamar dictionary, S’gaw Kayin OCR, Pa’O text to speech, Mon ASR etc.
- Myanmar NLP researchers should do R&D on these languages

Rule Based NLP Tasks

- *sylbreak4all* (syllable breaking tool for all Major ethnic languages with their own script, e.g. Bamar, S’gaw Kayin, Pwo Kayin, Mon, Shan including some spoken dialects such as Rakhine, Dawei, Beik)



Rule Based NLP Tasks

```
myConsonant = ur"က-အ"
enChar = ur"a-zA-Z0-9"
ssSymbol = ur'္'
ngaThat = ur'င်'
aThat = ur'်
otherChar = ur"ဣဤဥဦဧဩဪဿ၌၍၏၀-၉၊။!-/:-@[-`{-~\s" #other characters for Burmese, Dawei, Beik, Rakhine
otherSkChar = ur"ၢၣၤၥၦ၀-၉၊။!-\/:-\@\[(-`{-~\\s"  #other characters for Sgaw Kayin
otherPkChar = ur"ၠၡၢၣၤ၀-၉၊။!-\/:-\@\[(-`{-~\\s"#other characters for Pwo Kayin
shConsonant = ur"ၵၶငၸသၺတထၼပၽၾမယရလဝႁဢၷၹၻၿ"; #shan consonants
otherShChar = ur"႟႐-႙၊။!-\/:-\@\[-`{-~\\s"; #other characters for Shan
otherMoChar = ur"ဣဤဥဦဧဩဪဿ၌၍၏၀-၉၊။!-/:-@[-`{-~\s"; #other characters for Mon
skConsonant = ur"ကခဂဃငစဆၡညၦၥၧၨတထဒနပဖဘမယရလဝသဟၩအ"; #Sgaw kayin consonants
```

- Added some more variable declaration for ethnic languages (variables of *sylbreak4all* test version)

Rule Based NLP Tasks

- Some Example of RE (Regular Expressions) of *sylbreak4all* (test version)

```
#Regular expression pattern for Sgaw Kayin syllable breaking
BreakSkPattern = re.compile(ur"([" + skConsonant + ur"]"+ ur"|[" + enChar + otherSkChar + ur"])", re.UNICODE)

#Regular expression pattern for Pwo Kayin syllable breaking
BreakPkPattern = re.compile(ur"([" + myConsonant + ur"]"+ ur"|[" + enChar + otherPkChar + ur"])", re.UNICODE)

#Regular expression pattern for Shan syllable breaking
BreakShPattern = re.compile(ur"([" + shConsonant + ur"](?![" + aThat + ur"])" + ur"|[" + enChar + otherShChar + ur"])",
re.UNICODE)
```

Rule Based NLP Tasks

```
$ python sylbreak4all.py –i ../input/input.pok –lang "pk" > ../output/python/output.pok
```

|ဆၧ|အ|နီၢ|န|ထိၣ်|ဘူၣ်|ထဲၩ်|လၧ|ဆၧ|အ|ဂူ့ၢ|ဂၫ|က|မံၩ်|အ့ၣ်|ၜေ|.

|အ|ဝ့ၢ|ထီး|န့ၪ|ပှၡၫ|ဘၪ|နး|ဂၫ|လၧၩ်|အ့ၣ်|.

|ဆၧ|အ|နီၢ|မွဲ|ဆၧ|အ|ကၪ|လၧ|ပ|ဂး|လီၢ|.

|ယ|ဂဲၢ|ထဲၩ်|လီၫ|ပျၩ်|ထၣ်|ကဲၪ|ခိၣ်|န|လီၫ|ထၣ်|ဆ့|လီၢ|.

ပှ|လၧ|ဖီၪ့|ဂုး|ထၣ်|အ|ဝ့ၢ|က|နၩ့|နီၢ|မွဲ|ဒၪ|နၧၫ|လီၢ|.

|ယ|ယဲး|ထဲး|ဘၪ|ဆၧ|အ|နီၢ|ၜေ|.

|ဆီၢ့|မီၪ့|ဆၧ|ကဲၪ|ခိၣ်|ယ|ဆီၢ့|မီၪ့|ဘီၪ|.

|ယ|မ့ၣ်|လဲၫ|ချဲၩ်ပှ|ဂူ့ၢ|ဂၫ|အ|လၫ|.

|ယ|အဲၪ|အ|ဝ့ၢ|နီၢ|လ|ခဲၢ့|ထုၣ်|က|ဘျၩ်|မၣ်|ယ|လီၫ|ဘၪဝ့ၪ|လၧၩ်|ထၧၩ်|ယၢ|အ့ၣ်|.

|နၢ|ဆၢ|အ|ဆၧ|ယူၫ|ဖျိၣ်|ထၪ့|ကူၧၢ့|,|အ|ဝ့ၢ|ကွ့ၬ|နဲၢ့|ဆၧ|ကူၧၢ့|လ့ၣ်|လီၢ|.

- *sylbreak4all* demo for Pwo Kayin

Rule Based NLP Tasks

```
$ python sylbreak4all.py –i ../input/input.sgk –lang "sk" > ../output/python/output.sgk
|တၢ်|ဝဲ|န့ၣ်|န|တ|ဘျုး|စဲ|ဒီး|အ|ဂၤ|တ|ခါ|ဧၣ်|.
|ပိာ်|မုၣ်|န့ၣ်|တ|တိၢ်|နီၣ်|ပှၤ|နီ|တ|ဂၤ|လၢၤ|ဘၣ်|.
|တၢ်|ဝဲ|န့ၣ်|လၢ|ပ|ဂီၢ်|ကီ|ခဲ|ဝဲ|ဒၣ်|လီၤ|.
ဒ်|န|တဲ|တ့ၢ်|အ|သိး|ယ|တဲ|နၢ်|ပၢၢ်|တ့ၢ်|လီၤ|.
|က|ကွၢ်|ထွဲ|အီၤ|အ|ဂီၢ်|က|နၢၤ|ဒၣ်|နၤ|လီၤ|.
|တၢ်|ဝဲ|န့ၣ်|န့ၣ်|မ့ၢ်|ယ|ထီၣ်|ယီၢ်|ဘၣ်|ဧါ|.
ဒ်|ယ|ဆိ|က|မိၣ်|အ|သိး|ဆိ|က|မိၣ်|တ|က့ၢ်|.
|ဘၣ်|တဲ|ပှၤ|အ|ဂ့ၢ်|န့ၣ်|သး|ဟ့|လီၤ|.
|လၢ|ခံ|က|တၢၢ်|တ|ဘျီ|က|တဲ|အီၤ|လၢ|ယ|အဲၣ်|အီၤ|န့ၣ်|အ|ခွဲး|တ|န့ၢ်|လၢၤ|ဘၣ်|.
|ပျဲ|တၢ်|မၤ|စၢၤ|တ|က့ၢ်|.
```

- *sylbreak4all* demo for S’gaw Kayin

Rule Based NLP Tasks

```
$ python sylbreak4all.py –i ../input/input.po –lang "po" > ../output/python/output.po
|နဝ်ႇ|နဝ်ႇ| |နာႇ| |တ| |အွဥ်ႉ|ဖွိုႇ| |တဝ်း|ပေါင်း| |တွမ်ႉ| |အ|လင်| |တ|ဗာႉ
|ငွေႇ|မူႉ| |တ| |တောင်|ချာ|တဝ်း|ဒွူမ်| |ပါႇ|မုဲင်ႇ|မုဲင်ႇ
|နဝ်ႇ|နဝ်ႇ| |နီ| |အ|တာႉ| |ယပ်|ခုဲင်ႉ|ငါႉ
|နာႇ| |က|ဒေါ့ႇ| |အ|တွိုင်ႇ| |ခွေ| |သျင်ႇ|ပျ| |ဗာႉ|ဒျာႉ
|က|ထိန့်ႇ|နွောင်ႇ| |ငွေႇ|နဝ်ႇ| |အဝ်ႉ|ဒျာႉ| |နာႇ| |လွူမ်
|နဝ်ႇ|နဝ်ႇ| |ခွေ| |ယမ်း| |မာႉ|ဗာႉ|ပေါင်း
|ခွေ| |စ|ဥႇ|စာႇ| |အ|တွိုင်ႇ| |စ|ဥႇ|စာႇ|ဟုဲင်း
|ဒေါ့ႇ|ဝင်ႇ|မဥ်ႇ| |နဝ်ႇ| |လွ|ထီႉ|ငါႉ
|ဆုဲင်ႇ|သွတ်| |တ| |လဲင်ႉ| |ရက်|ဒျာႉ| |ငွေႇ|နဝ်ႇ| |တဲ့| |ဒေါ့ႇ|ခွင်ႇ| |တ| |လ| |တဝ်း|ဒွူမ်
|တယ်ႉ| |နာ|ဆာ| |ဒုံး|ပျို| |ထင်ႉ|စ|ခိန်ႉ| |နဝ်ႇ| |ငွေႇ| |တဲမ်း| |ဗာႉ|ဒျာႉ| |မတ်|တန့်ႇ
```

- *sylbreak4all* demo for P’ao

Rule Based NLP Tasks

```
$ python sylbreak4all.py –i ../input/input.sh –lang "sh" > ../output/python/output.sh
|မွၼ်း| |လွင်ႈၼႆႉ| |လၢတ်ႈ|မႃး| ႁိုဝ်| ဢမ်ႇ|လၢတ်ႈ|မႃး| ႁူႉ| |။
|တႃႇ| |လုၵ်ႈႁဵၼ်းၶဝ်| |တေ|လႆႈ| ဢဝ်| |ပပ်ႉ|လိူဝ်| |။
|တွင်း|ပၢၼ်ႇၵဝ်| ဢမ်ႇ| |တွင်း|ပၢၼ်ႇ| ၵဝ်| |။
ဢ|ရ|သႃႇ| |မၼ်း|ၸတႉ| |မိူၼ်ၼမ်ႉ| |လၢင်ႉ|ဝၢၼ်ႇ| |ဝႆႉ| |။
|မိူဝ်ႈ|ပူၼ်ႉ| |မႃး| |ဝၼ်း|သုၵ်း| ၵၢင်ၼႂ်| ႑႑| |မွင်း| ၼၼ်ႉ| |သူ| | |မီး|ယူႇ| |တီႈ|လႂ်| |။
ဢၼ်ၼႆႉ| |တႃႇ|မၼ်း| |ယၢပ်ႇ| ၵိူဝ်ႈ| |။
ဢမ်ႇ|မီး| ၶပ်း|မႆ| |တႃႇၵွၵ်ႇ|ပႆႈ| ႁူႉ| |။
|တႃႇ|မၼ်းၼၢင်း| ႁဝ်း|တေ| |ထၢမ်| ဢမ်ႇ| လႆႈ| |။
ႁဝ်း| |မိူဝ်ႈၽုၵ်ႈ| ၵၢင်ၼႂ်| |တေၵွၵ်ႇ|ပႆ|တၢင်း| ၵိူဝ်ႈ| |။
ဢမ်ႇ| |မူတ်း|သႂ်| ႁူႉ| |။
```

- *sylbreak4all* demo for Shan

Rule Based NLP Tasks

- Epitran (a library and tool for transliterating orthographic text as IPA)

```
In [4]: from epitran.backoff import Backoff
        backoff = Backoff(['tha-Thai', 'kir-Arab', 'mya-Mymr'],
                          cedict_file='cedict_1_0_ts_utf-8_mdbg.txt')
```

```
In [5]: backoff.transliterate('การวิจัย')
Out[5]: 'kaːnwit͡ɕaj'
```

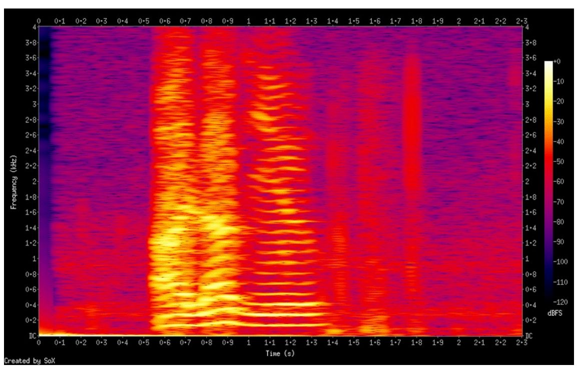
```
In [6]: backoff.transliterate('ابحاث')
Out[6]: 'ɑbkɑث'
```

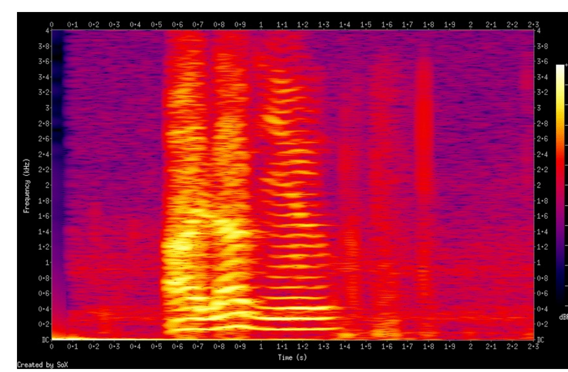
```
In [7]: backoff.transliterate('သုတေသန')
Out[7]: 'θuteθən'
```

Machine Learning in NLP

- Train a model to map to map an input ***X*** into an output ***Y***

おはようございます → **MT Model** → Buongiorno

 → **ASR Model** → I love you

I love you → **TTS Model** → 

Neural Machine Translation

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

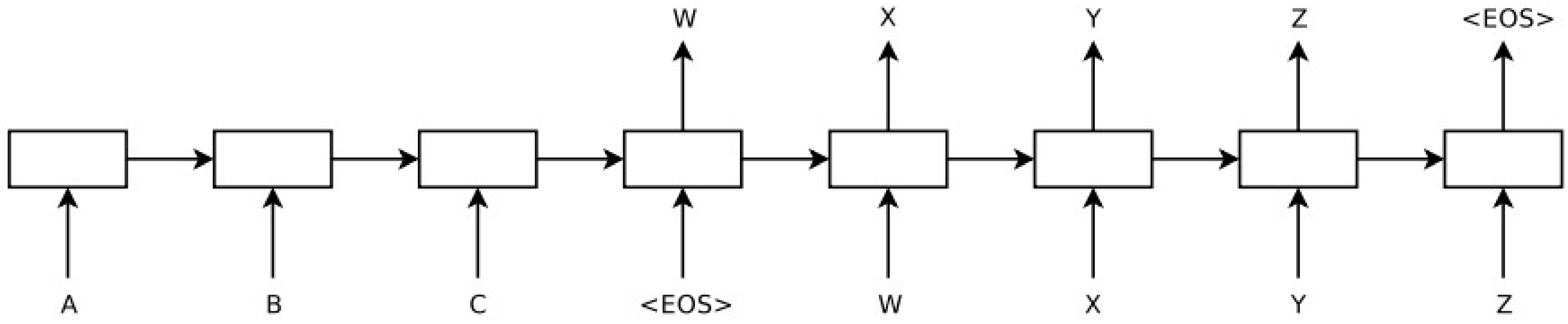
Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

- Read this paper

Neural Machine Translation



(Figure: taken from sequence to sequence learning with Neural Networks, 2014)

- Sequence to Sequence Learning

Neural Machine Translation

Neural Machine Translation by Jointly Learning to Align and Translate

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Abstract

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

Neural Machine Translation

- The most important distinguishing feature of this approach from the basic encoder–decoder is that it does not attempt to encode a whole input sentence into a single fixed-length vector. Instead, it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation. This frees a neural translation model from having to squash all the information of a source sentence, regardless of its length, into a fixed-length vector.

Neural Machine Translation

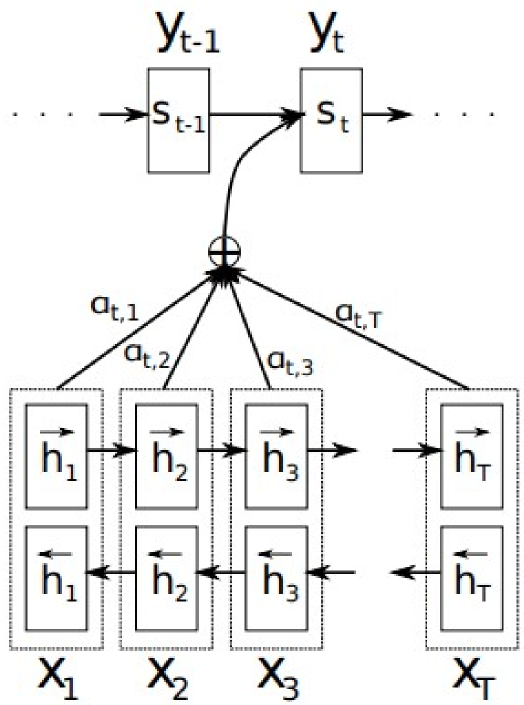
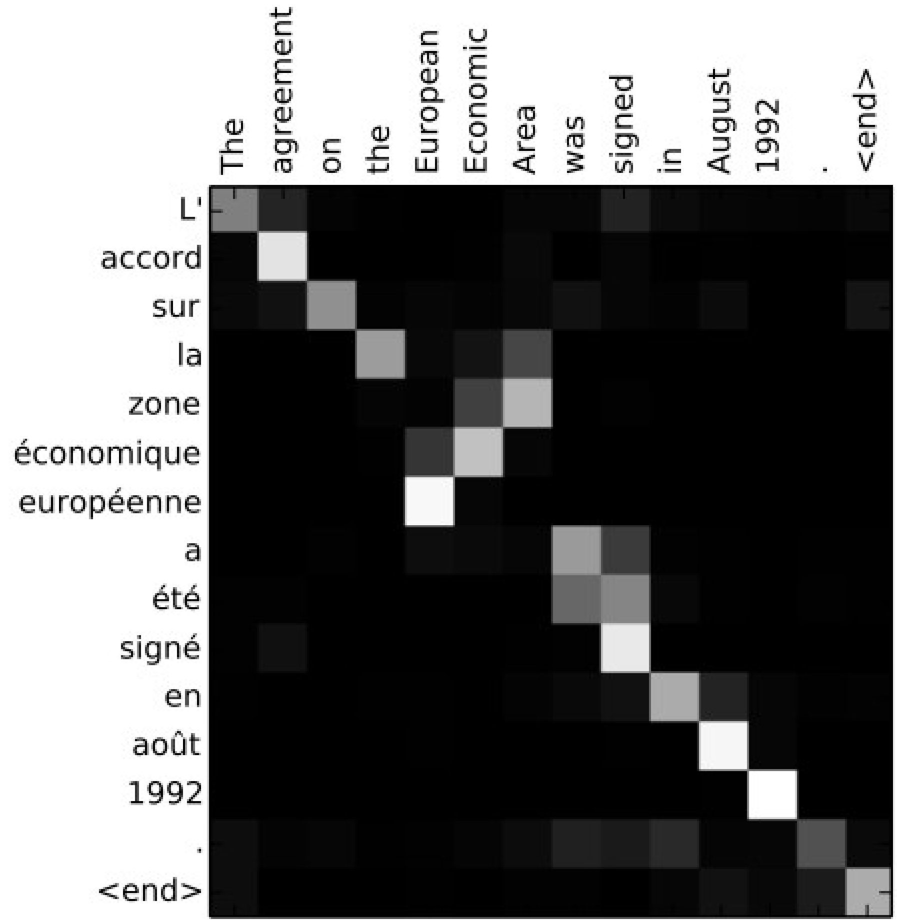


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.



(Figure: taken from NMT by Jointly Learning to Align and Translate, 2015)

Neural Machine Translation

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*, †]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

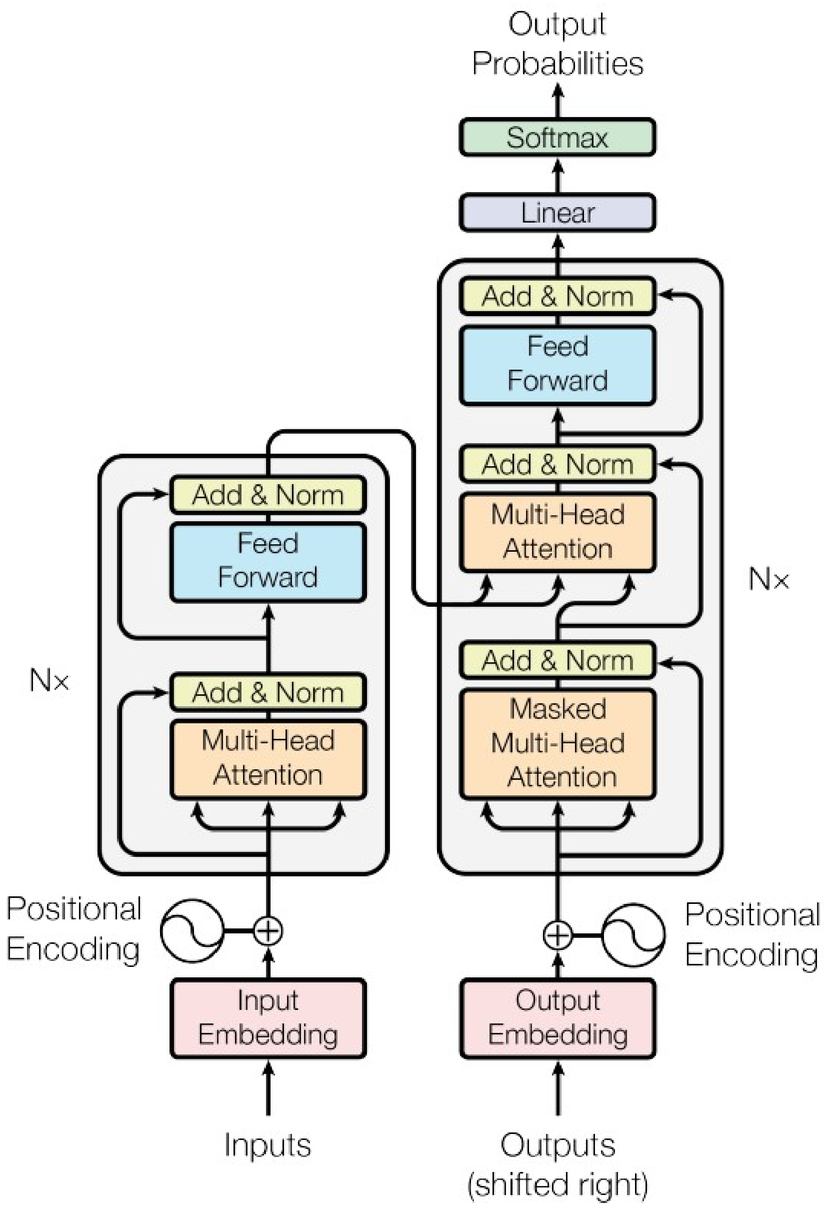
Illia Polosukhin[*, ‡]
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

- Read this paper
- Yes, “Transformer”

Neural Machine Translation



(Figure, Taken from, Attention is All You Need, 2017)

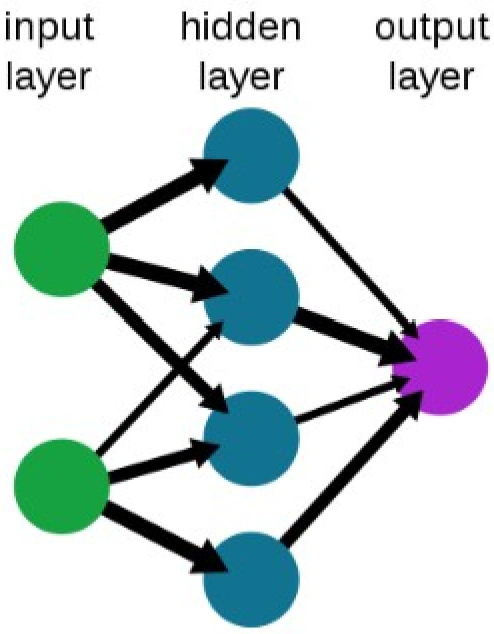
- the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention

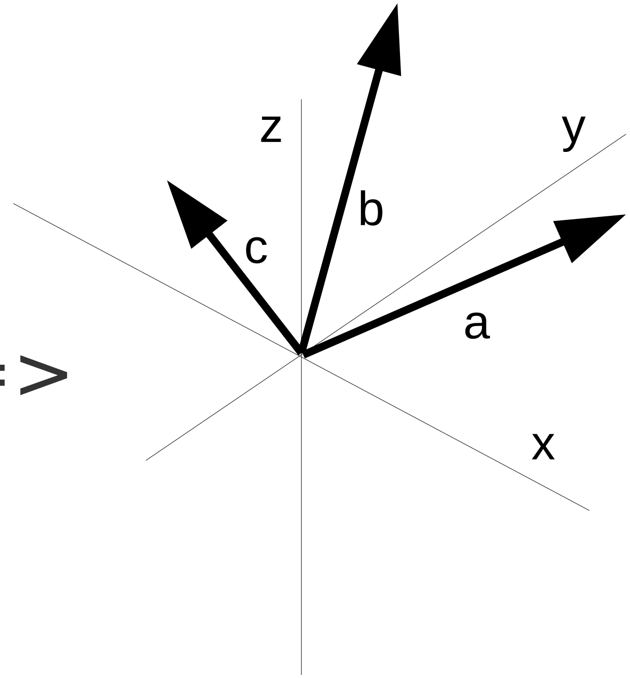
Neural Machine Translation

- Transformers: the model of choice for NLP problems
- Replacing older RNN (e.g. LSTM)
- More parallelization during training, and thus suitable to training on larger datasets
- Transformer ===> led to the development of pretrained systems such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer)
- Fine-tuned to specific language tasks

(Reference: the paper of “BERT”, Wiki also)

Neural Representation Learning for NLP Tasks

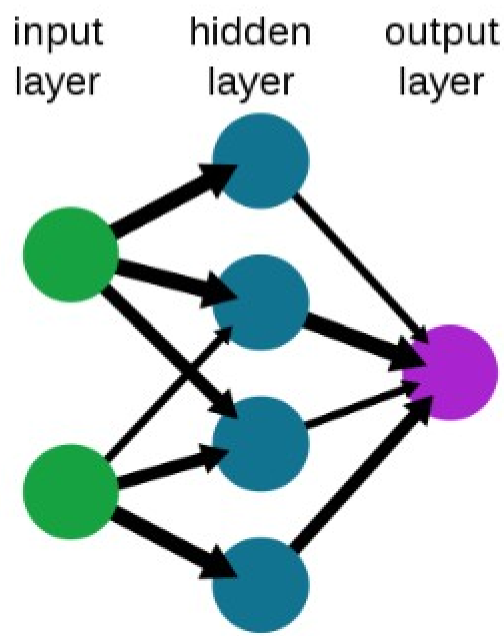
- Text ==> 

===> 

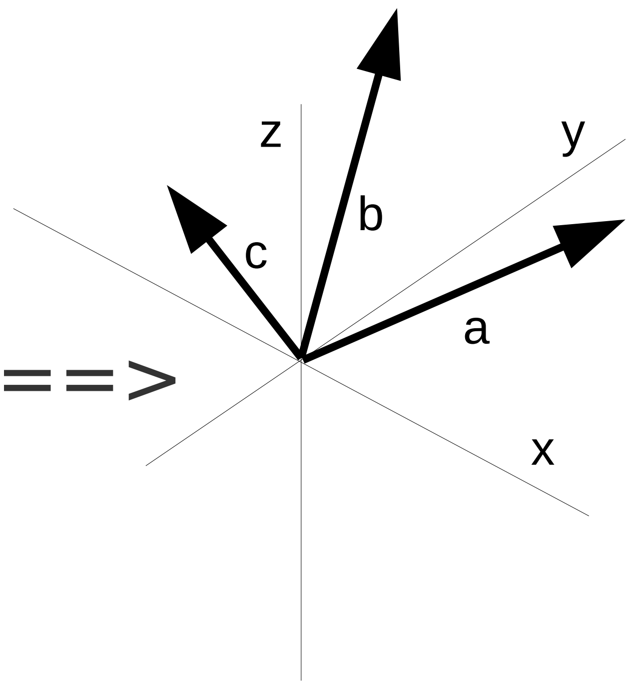
Neural Representation Learning for NLP Tasks

<u>Word</u>
or
<u>Phrase</u>
or
<u>Sentence</u>
or
<u>Sentence Pair</u>

==>



===>



Neural Representation Learning for NLP Tasks

- Vector Calculation

  (e.g. Adding)

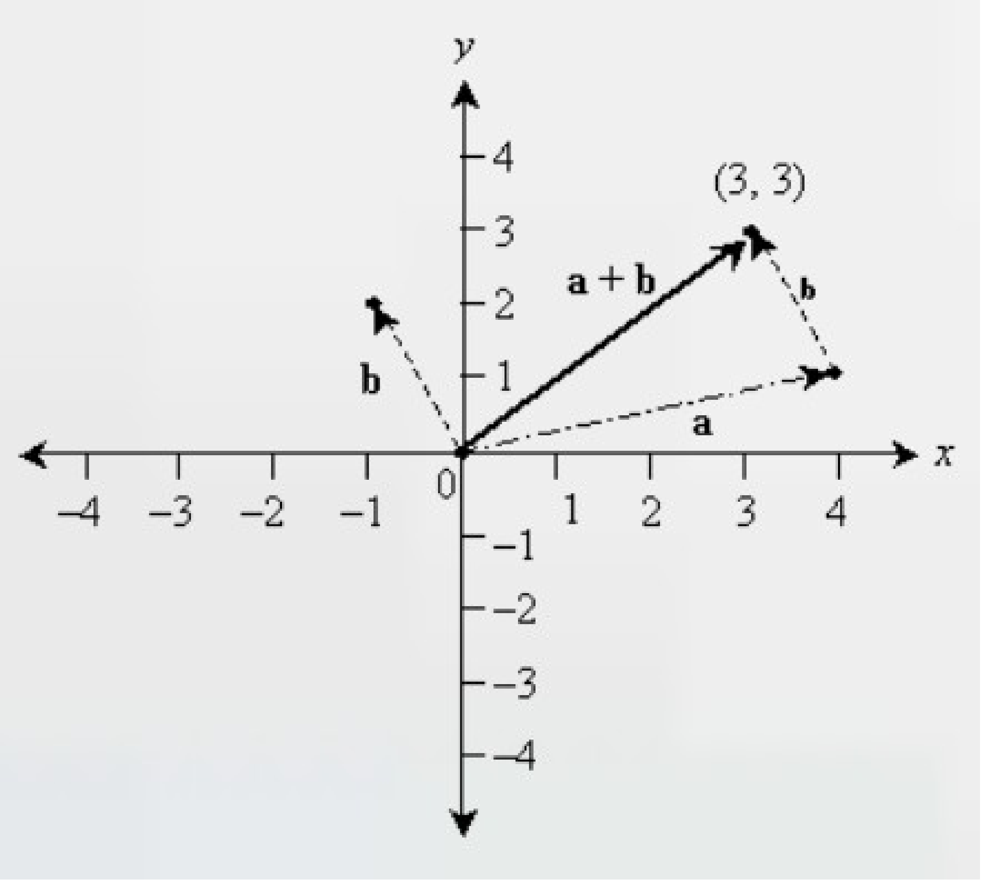


Figure taken from universalclass.com

Neural Representation Learning for NLP Tasks

- Vector Calculation

  (e.g. Adding)

$$\begin{array}{ccc} \text{Happy Face} & & \text{Happy} \quad \text{Face} \\ \begin{bmatrix}0.5\\0.2\\0.8\end{bmatrix} & = & f\left\{\begin{bmatrix}0.2\\0.1\\0.2\end{bmatrix}, \begin{bmatrix}0.3\\0.1\\0.6\end{bmatrix}\right\} \end{array}$$

Neural Representation Learning for NLP Tasks

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

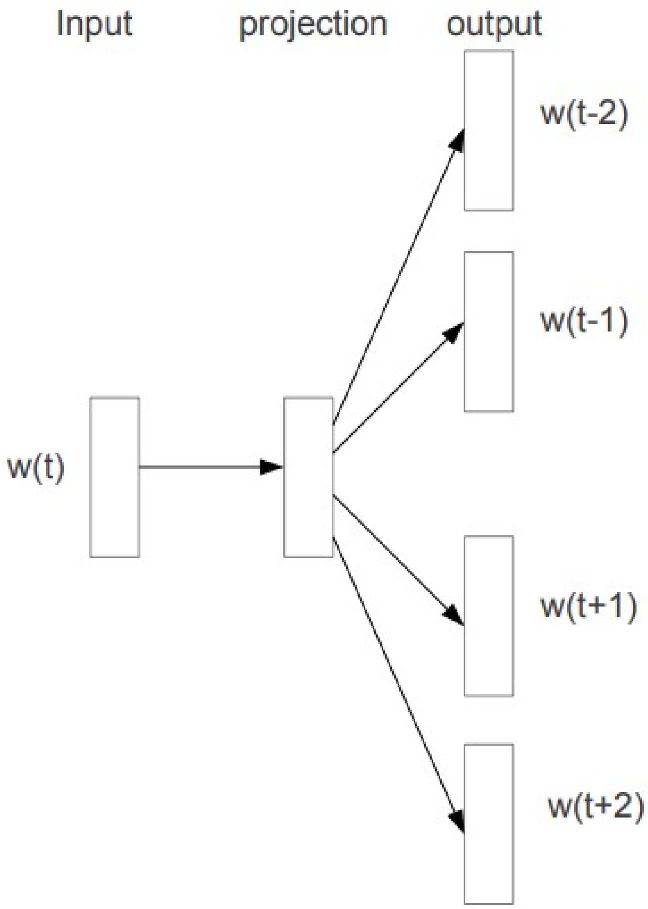
Abstract

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.

An inherent limitation of word representations is their indifference to word order and their inability to represent idiomatic phrases. For example, the meanings of "Canada" and "Air" cannot be easily combined to obtain "Air Canada". Motivated by this example, we present a simple method for finding phrases in text, and show that learning good vector representations for millions of phrases is possible.

- Yes, very famous paper!
- word2vec

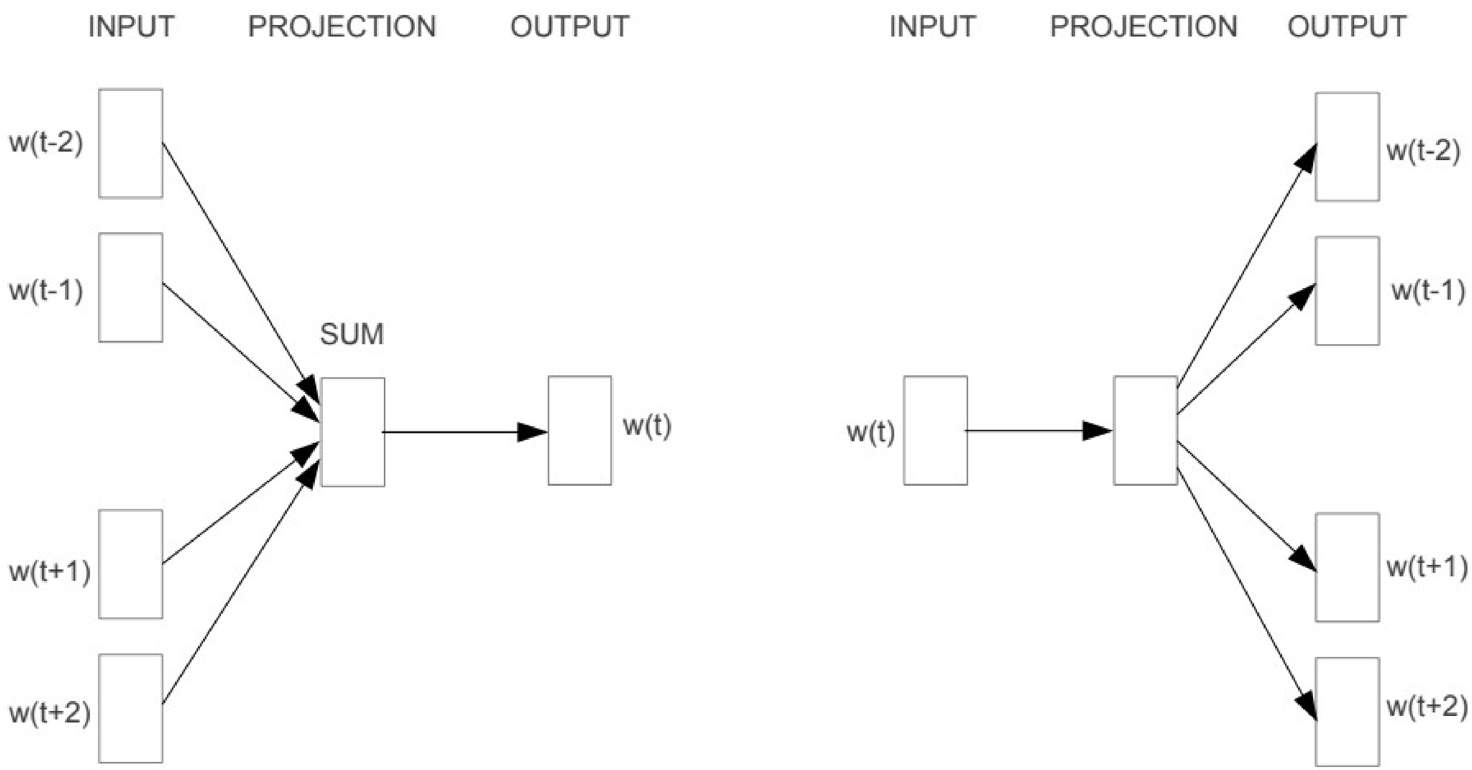
Neural Representation Learning for NLP Tasks



- Skip-gram Model Architecture

(Figure: taken from Distributed Representations of Words and Phrases and their Compositionality, 2013)

Neural Representation Learning for NLP Tasks



Left: CBOW (Continuous Bag-of-Words)   Right: Skip-gram

(Figure: taken from “Exploiting Similarities among Languages for Machine Translation”, 2013)

Neural Representation Learning for NLP Tasks

- From Mikolov paper, Skip-gram works well with small amount of data and is found to represent rare words well
- On the other hand, CBOW is faster and has better representations for more requent words
- We can do a lot with word vectors ...

Neural Representation Learning for NLP Tasks

- Oh! So many approaches .... ?!

SentiEMB

GloVe

One-Hot

ELMO

MT-emb

Word2Vec

LSA

BART

NNLM

BERT

PLSA

C&W

Neural Representation Learning for NLP Tasks

- Symbolic

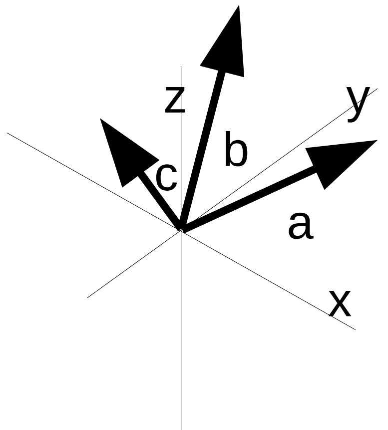
0
1
0

- One-hot Vector
- Explainable

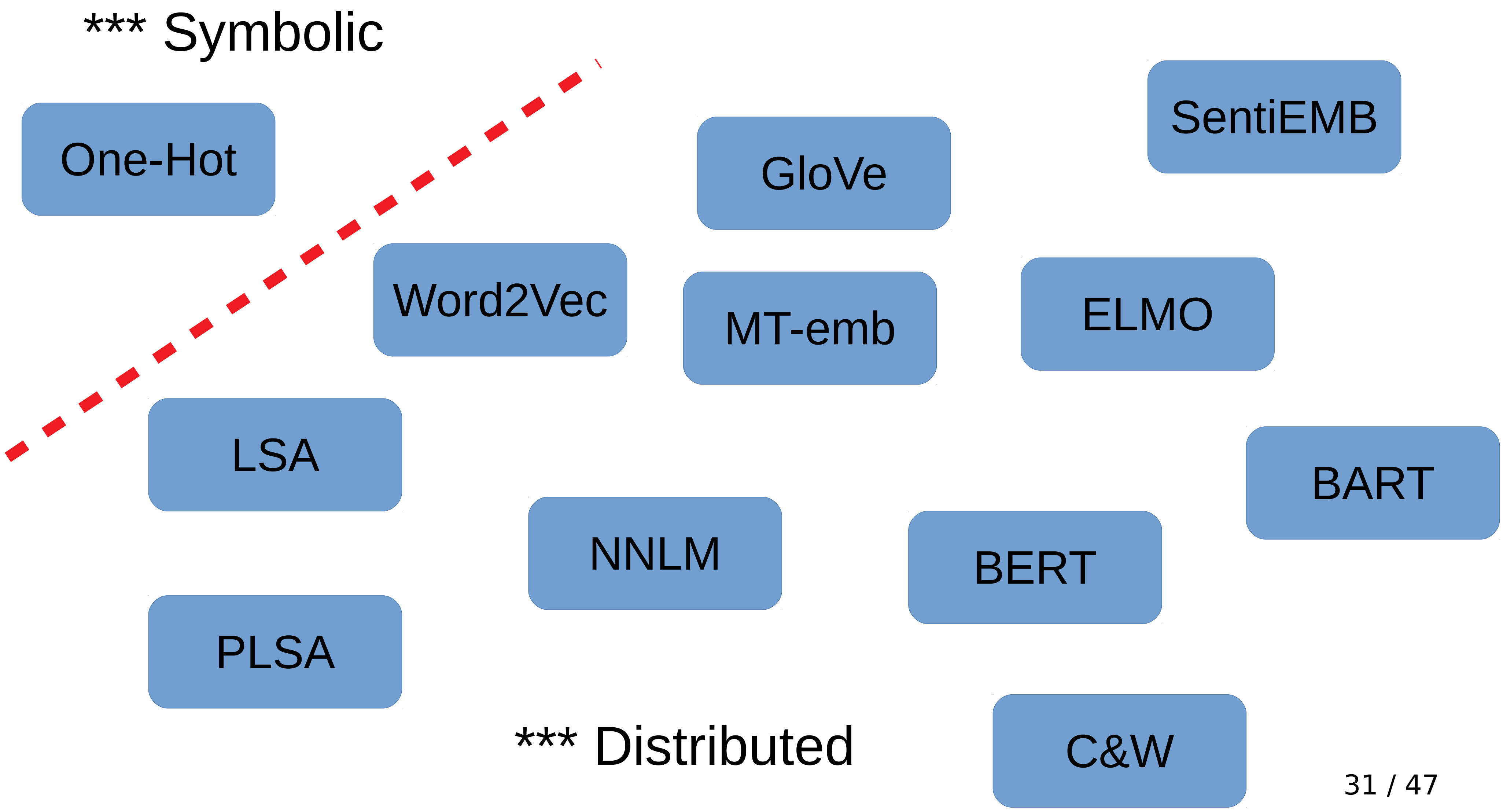
- Distributed

0.2
0.1
0.2

- Real-valued Vector
- More Explainable



Neural Representation Learning for NLP Tasks



Neural Representation Learning for NLP Tasks

*** Supervised?!

*** Semi-Supervised ?!

*** Unsupervised?!

*** FREE STUDY BY YOURSELF

Word2Vec ?

LSA ?

GloVe ?

ELMO ?

? MT-emb

One-Hot ?

BART ?

NNLM

BERT ?

PLSA ?

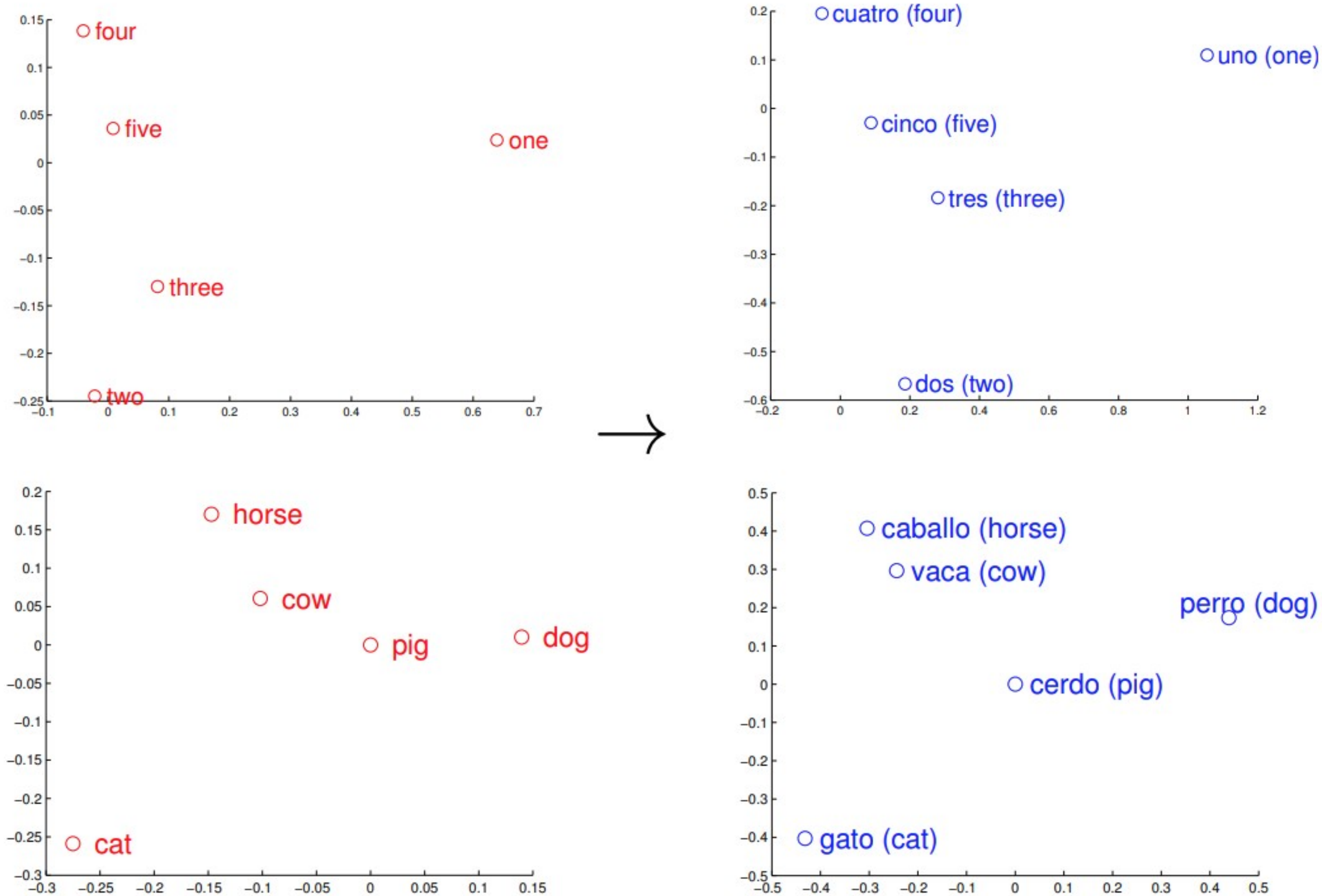
SentiEMB ?

C&W ?

Neural Representation Learning for NLP Tasks

- Non-contextualized
  - Context independent vector
- Contextualized
  - Context dependent vector

Neural Representation Learning for NLP Tasks



(Figure: taken from “Exploiting Similarities among Languages for Machine Translation”, 2013)

Neural Representation Learning for NLP Tasks

- Demo Running with FastText
- Of course, with our manually segmented Myanmar corpus

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ fasttext
usage: fasttext <command> <args>

The commands supported by fasttext are:

  supervised              train a supervised classifier
  quantize                quantize a model to reduce the memory usage
  test                    evaluate a supervised classifier
  test-label              print labels with precision and recall scores
  predict                 predict most likely labels
  predict-prob            predict most likely labels with probabilities
  skipgram                train a skipgram model
  cbow                    train a cbow model
  print-word-vectors      print word vectors given a trained model
  print-sentence-vectors  print sentence vectors given a trained model
  print-ngrams            print ngrams given a trained model and word
  nn                      query for nearest neighbors
  analogies               query for analogies
  dump                    dump arguments,dictionary,input/output vectors
```

Neural Representation Learning for NLP Tasks

- import testing fasttext with interactive Python

```
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import fasttext as ft
>>> dir(ft)
['BOW', 'EOS', 'EOW', 'FastText', '__builtins__', '__cached__', '__doc__',
'__file__', '__loader__', '__name__', '__package__', '__path__', '__spec__'
, 'absolute_import', 'cbow', 'division', 'load_model', 'print_function', 's
kipgram', 'supervised', 'tokenize', 'train_supervised', 'train_unsupervised
', 'unicode_literals']
>>> 
```

Neural Representation Learning for NLP Tasks

- Example training

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ time fasttext skipgram -input myword
.clean -output skipgram.model -minCount 1 -minn 3 -maxn 6 -lr 0.01 -dim 100 -ws 3 -
epoch 10 -neg 20
Read 5M words
Number of words:  103224
Number of labels: 0
Progress:   0.0% words/sec/thread:   15631 lr:  0.009997 avg.loss: 14.507567 ETA:
Progress:   0.1% words/sec/thread:   15695 lr:  0.009993 avg.loss: 14.479765 ETA:
Progress:   0.1% words/sec/thread:   16138 lr:  0.009990 avg.loss: 14.449356 ETA:
Progress:   0.1% words/sec/thread:   16359 lr:  0.009986 avg.loss: 14.426417 ETA:
Progress:   0.2% words/sec/thread:   16357 lr:  0.009982 avg.loss: 14.334553 ETA:
Progress:   0.2% words/sec/thread:   16275 lr:  0.009979 avg.loss: 14.171081 ETA:
Progress:   0.2% words/sec/thread:   16409 lr:  0.009975 avg.loss: 13.698857 ETA:
Progress:   0.3% words/sec/thread:   16506 lr:  0.009971 avg.loss: 13.111738 ETA:
Progress:   0.3% words/sec/thread:   16617 lr:  0.009968 avg.loss: 12.484881 ETA:
Progress:   0.4% words/sec/thread:   16685 lr:  0.009964 avg.loss: 11.999210 ETA:
Progress:   0.4% words/sec/thread:   16799 lr:  0.009960 avg.loss: 11.656706 ETA:
Progress:   0.4% words/sec/thread:   16821 lr:  0.009956 avg.loss: 11.176975 ETA:
```

Neural Representation Learning for NLP Tasks

- An example code for testing...

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ cat ./test-skipgram-model.py
#from gensim.fasttext import FastText
from gensim.models.fasttext import FastText


model = FastText.load_fasttext_format('skipgram.model.bin')


print(model.wv.most_similar('မြန်မာ', topn=5))
print(model.wv.most_similar('ဗမာ', topn=5))
print(model.wv.most_similar('ခေါက်ဆွဲ', topn=5))
print(model.wv.most_similar('ဆရာ', topn=5))
print(model.wv.most_similar('အောင်ဆန်းစုကြည်', topn=5))
```

Neural Representation Learning for NLP Tasks

- Test result with 10 epoch model

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ python ./test-skipgram-model.py
./test-skipgram-model.py:4: DeprecationWarning: Call to deprecated `load_fasttext_format` (use
load_facebook_vectors (to use pretrained embeddings) or load_facebook_model (to continue training with the
loaded full model, more RAM) instead).
  model = FastText.load_fasttext_format('skipgram.model.bin')
[('/မြန်မာ', 0.9879409074783325), ('ာမြန်မာ', 0.9874909520149231), ('(မြန်မာ', 0.9869102835655212), ('.မြန်မာ',
0.9862627983093262), ('\u200bမြန်မာ', 0.9843737483024597)]
[('..ဗမာ', 0.9498854875564575), ('ဗမာမ', 0.949866533279419), ('ဗမ', 0.9371455907821655), ('မွတ်ကုလား',
0.9333832263946533), ('မွတ်ဆလင်', 0.9323370456695557)]
[('ပဲခေါက်ဆွဲ', 0.9891856908798218), ('ခေါက်ဆွဲပြုတ်', 0.9756088256835938), ('ရှမ်းခေါက်ဆွဲ', 0.9705666303634644),
('ခေါက်ဆွဲကြော်', 0.9698923230171204), ('ခေါက်မုန့်', 0.9696319103240967)]
[('ဆရာမ၊', 0.9499595165252686), ('ဆရာဘူ', 0.9489681720733643), ('ဆရာ့', 0.9485983848571777), ('ဆရာဖေ',
0.9454444050788879), ('ဆရာမ', 0.9452099800109863)]
[('အောင်ဆန်းဆုကြည်', 0.9777907133102417), ('အောင်ဆန်းစုရှည်', 0.9766486883163452), ('ဒေါ်\u200cအောင်ဆန်းစုကြည်',
0.9694842100143433), ('အောင်ဆန်းဇာနည်', 0.9652061462402344), ('ဒေါ်အောင်ဆန်းစုကြည်', 0.963191568851471)]
```

Neural Representation Learning for NLP Tasks

- How about training with 100 epoch?

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ time fasttext skipgram -input myword.clean -output
skipgram.epoch100.model -minCount 1 -minn 3 -maxn 6 -lr 0.01 -dim 100 -ws 3 -epoch 100 -neg 20
Read 5M words
Number of words:  103224
Number of labels: 0
Progress: 100.0% words/sec/thread:   16785 lr:  0.000000 avg.loss:  1.532995 ETA:   0h 0m 0s
... ... ...
real  46m6.519s
user  178m7.088s
sys   0m24.418s
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$
```

Neural Representation Learning for NLP Tasks

- Test output with 100 epoch skipgram model

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ python ./test-skipgram-model.py
./test-skipgram-model.py:5: DeprecationWarning: Call to deprecated `load_fasttext_format` (use
load_facebook_vectors (to use pretrained embeddings) or load_facebook_model (to continue training with the
loaded full model, more RAM) instead).
  model = FastText.load_fasttext_format('skipgram.epoch100.model.bin')
[('ပါမြန်မာ', 0.9266026616096497), ('(မြန်မာ', 0.9183459877967834), ('.မြန်မာ', 0.9178306460380554), ('အဖမြန်မာ',
0.9166991710662842), ('ာမြန်မာ', 0.9056539535522461)]
[('..ဗမာ', 0.7396675944328308), ('ဗမာမ', 0.732035219669342), ('ဗမာ့သား', 0.7267118692398071), ('海',
0.705876350402832), ('ကွစစ်သား', 0.6850018501281738)]
[('ပဲခေါက်ဆွဲ', 0.9589242339134216), ('ခေါက်ဆွဲ\u200cကြော်', 0.9438843727111816), ('ခေါက်ဆွဲကြော်',
0.9276038408279419), ('ဆန်ခေါက်ဆွဲ', 0.9079042673110962), ('ခေါက်ဆွဲပြုတ်', 0.9069240093231201)]
[('ဆရာဘူ', 0.810899019241333), ('ဆရာက', 0.810529351234436), ('ဆရာမမတာ', 0.782470262878418), ('ဆရာငဲ',
0.7820311784744263), ('ဆရာမသတိရ', 0.7747628688812256)]
[('ဒေါ်\u200cအောင်ဆန်းစုကြည်', 0.8839490413665771), ('ဒေါ်အောင်ဆန်းစုကြည်', 0.881594419479370 1),
('အောင်ဆန်းစုရှည်', 0.8748890161514282), ('ဒေါ်အောင်ဆန်းစုကြည့်', 0.8685750365257263), ('အောင်ဆန်းဆုကြည်',
0.8091812133789062)]
```

Neural Representation Learning for NLP Tasks

- Testing analogies...

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ fasttext analogies ./skipgram.epoch100.model.bin
Loading model ./skipgram.epoch100.model.bin
Query triplet (A - B + C)? ရန်ကုန် - မုန့်ဟင်းခါး + မန္တလေး
မဒဂုံမုန့်တီ 0.636652
(ရန်ကုန် 0.611318
ရေခဲမုန့် 0.611283
ဟင်းပွဲ 0.598536
ဟင်းခါး 0.597337
မုန့်တီ 0.588623
ရန်ကုန်သား 0.58693
ရန်ကုန်မှာ 0.586863
ဘူဖေး 0.586214
ရန်ကုန်သူ 0.583179
```

Neural Representation Learning for NLP Tasks

- Printing word vectors...

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ fasttext print-word-vectors ./skipgram.epoch100.model.bin
ရခိုင်မုန့်တီ
ရခိုင်မုန့်တီ 0.38708 -0.41596 1.1877 -0.49699 0.13632 0.11323 0.71918 -0.30596 0.57912 -0.21668 -0.29691 -0.38338
0.28088 -0.21166 0.77997 -0.33342 0.3818 0.22022 0.13113 0.40029 0.048083 0.25381 -0.44163 -0.95053 -0.34749
0.73397 0.6415 0.41203 0.67167 -0.013714 0.40989 0.55134 0.011515 0.20331 0.52595 0.98226 0.72689 -0.1971
0.33249 -0.8207 -0.092261 -0.65617 -0.7021 -0.86417 0.7593 0.06691 -0.22292 0.29491 -0.96045 -0.60103 -0.75033
0.017968 -0.53229 -0.43376 -0.012416 0.76405 -0.13249 -0.26408 0.44248 -0.26516 1.0946 -0.25444 -0.89828
-0.14435 0.54537 0.87375 0.61472 1.3658 -0.85873 0.024179 -0.33234 -0.62328 1.1671 -1.1303 -0.55311 0.16951
0.41302 0.26516 -0.064007 -0.071959 -0.011454 -0.61469 -0.60155 0.6122 0.62414 -0.30109 0.73081 0.391 -0.48771
-0.45631 -0.32664 0.76813 0.76551 -0.26287 0.76581 -0.47029 -0.40281 1.1 -0.54765 -0.58092
မေမြို့
မေမြို့ -0.3386 0.21767 0.53761 -0.31687 0.8864 0.23766 1.0584 -0.41652 -0.10395 0.43185 -0.077917 -1.0976
-0.14907 -0.63833 0.36824 -0.17031 -0.15389 0.028631 0.18095 0.43769 0.55456 -0.78349 -0.044933 -1.1407 -0.51537
```

Neural Representation Learning for NLP Tasks

- Printing sentence vectors...

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ fasttext print-sentence-vectors ./skipgram.epoch100.model.bin
သုတေသန အလုပ် က အချိန်ပေး ရ တယ်
0.056766 -0.0090204 0.1203 -0.12249 0.021174 0.0019972 0.11291 0.058262 0.085292 -0.04926 -0.072068 -0.085208
-0.097171 -0.10251 0.12907 -0.083892 0.013129 -0.060215 0.13538 0.054718 0.019432 -0.012638 -0.056367 -0.088302
-0.075335 0.017073 0.076337 0.062211 -0.052698 0.085757 -0.054359 0.070433 -0.059417 -0.072097 -0.041723
-0.050098 0.13368 -0.021579 0.081177 -0.0046396 0.1145 0.03621 -0.030924 -0.043032 0.06388 0.017267 0.0645
-0.076853 -0.001591 0.066814 -0.097431 0.015154 -0.12528 0.045655 -0.036451 -0.012508 0.10148 0.020186
-0.014962 0.028532 0.12535 -0.0077517 0.027197 0.085593 0.08786 0.031975 0.045109 0.10923 0.027715 -0.057354
-0.14537 0.029842 0.20804 -0.10945 0.0095761 0.052942 0.066978 0.14431 -0.12915 -0.035794 -0.10376 -0.155
0.083709 0.01943 0.069052 0.092508 0.041905 0.038175 -0.034366 0.096243 -0.066582 0.08869 0.1719 -0.17195
-0.0026349 -0.11346 -0.061518 0.0928 -0.015863 0.018368
```

Neural Representation Learning for NLP Tasks

- Printing ngrams...

```
(base) ye@ykt-pro:~/tool/fastText-0.9.2/y-exp$ fasttext print-ngrams ./skipgram.epoch100.model.bin အလုပ်
အလုပ် 0.56404 2.6401 2.4382 0.79998 0.39199 0.46267 -1.8779 -0.24498 1.5072 0.28385 0.70529 -1.7516 -0.93591
-0.15229 -0.78936 -0.26373 -3.0067 -0.21905 3.6042 0.80033 1.4192 -2.1875 2.4695 0.98405 3.8094 -0.74529 -3.7497
-1.125 0.39853 -0.12399 -3.4721 0.87615 0.98688 -1.4361 0.87821 0.88224 -0.88826 1.4117 0.1046 1.6854 0.68701
2.1434 -3.1211 0.21299 -0.22395 0.35944 0.31888 1.5313 2.0806 3.1796 0.3493 1.5896 2.3567 -0.90351 -2.4818 -1.5225
2.4612 0.12279 0.41719 0.3414 -0.3878 -0.079597 0.62177 0.036651 1.0462 0.32795 -0.13061 -0.32744 -2.396 0.84532
1.3362 -0.44362 -0.18977 0.77567 -0.73559 -1.8419 2.4259 -1.9053 0.11186 -0.055772 0.79099 -0.087902 -1.8179
-2.1528 -0.83782 2.9834 -0.74282 1.8882 -0.58988 -0.38456 0.59502 -0.80597 -3.1874 -1.1762 0.31896 1.6805 2.2676
1.0498 1.3418 -1.6143
<အလ -0.22189 -1.1801 3.6275 -2.7153 -1.0705 1.9965 0.52251 3.2904 1.4133 -1.2997 -1.6002 -0.97553 -3.291 -2.7574
0.57857 -0.8405 0.16224 -1.4862 2.5817 2.2531 0.33234 -3.04 -4.859 -2.9356 -2.8257 -1.1178 1.4353 1.936 -4.2953
-2.0951 1.216 -4.8575 -0.099782 0.68939 0.26418 2.0413 5.5024 3.9845 1.2164 0.55145 2.0172 3.6092 0.6044 1.8986
-1.9278 1.7309 -0.98019 -4.5284 -1.07 -0.80443 -3.9061 3.9328 -0.49873 -0.090583 -3.1655 -2.5447 2.276 0.87676
-0.64379 1.9831 1.3341 0.6731 -4.002 3.495 2.385 2.9326 2.4258 0.77259 -1.6985 1.2077 -5.6146 0.77715 1.0069
-0.80992 0.50563 -0.59128 -0.36271 2.6366 -4.7171 -0.42545 -1.778 -4.1639 3.3349 -1.026 -3.4589 0.80159 5.6213
-0.14796 0.31708 2.6447 -2.0887 0.37429 0.42643 -3.3465 2.4121 -1.8524 -0.7817 0.2282 0.52455 -0.75269
<အလု 0.30798 -1.005 -0.98822 0.16729 0.26955 -0.058416 0.12805 -0.44793 0.20073 0.67262 -0.30991 1.1078 0.79922
-1.2401 0.79736 0.31557 1.5621 0.24718 0.93994 -1.0793 0.47533 -1.715 -0.19489 1.634 -1.6283 1.4415 0.094182 1.153
```

Reference

- Sutskever, I., Vinyals, O. & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems (p./pp. 3104–3112)
- Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate (cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation)

Reference

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (ed.), Advances in Neural Information Processing Systems 30 (pp. 5998–6008) . Curran Associates, Inc. .
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani & K. Weinberger (ed.), Advances in Neural Information Processing Systems 26 (pp. 3111--3119) .