# NAVER Machine Translation System for WAT 2015

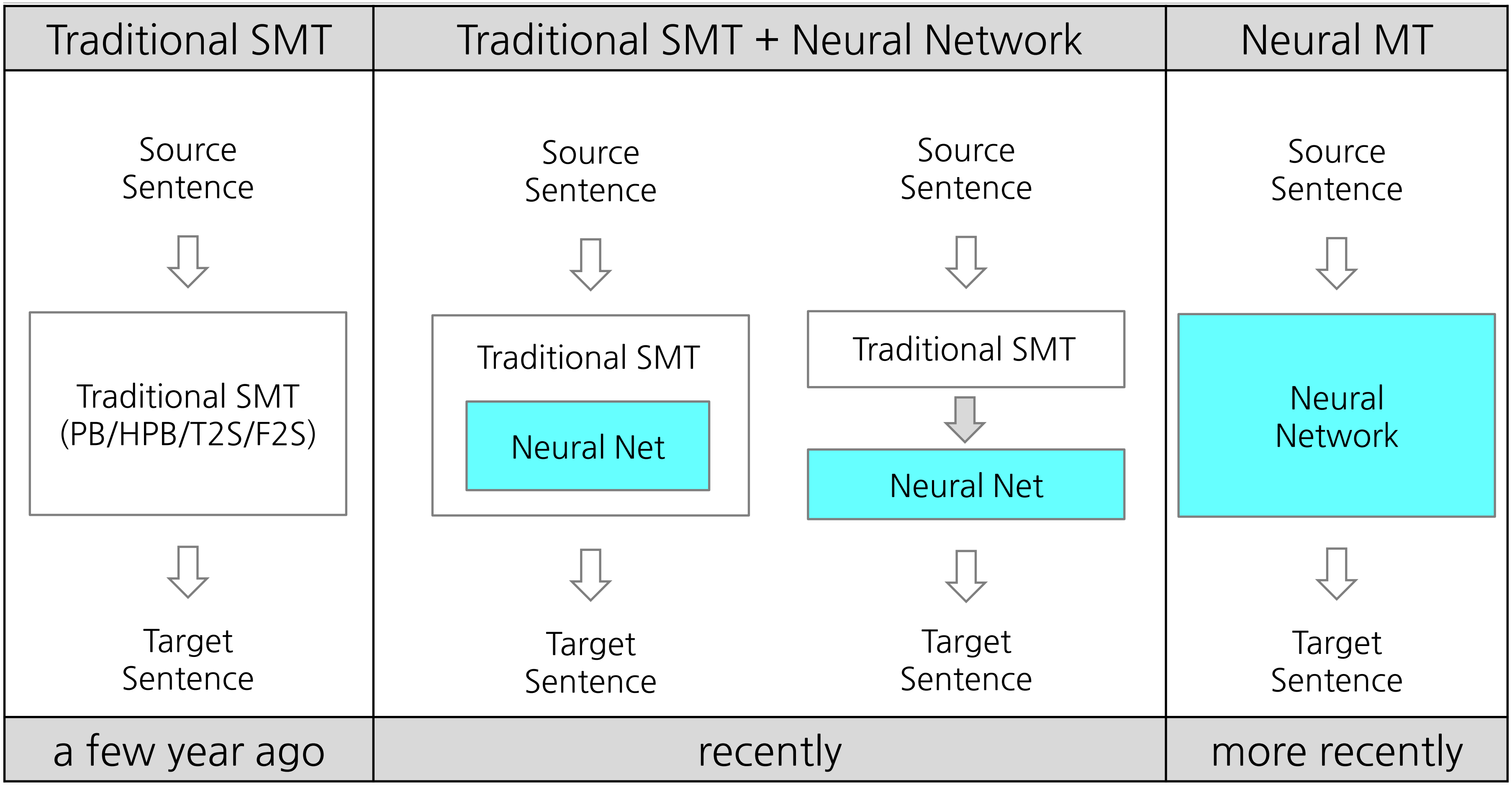
Hyoung-Gyu Lee, Jae-Song Lee, Jun-Seok Kim and Chang-Ki Lee

2015-10-16

# Contents

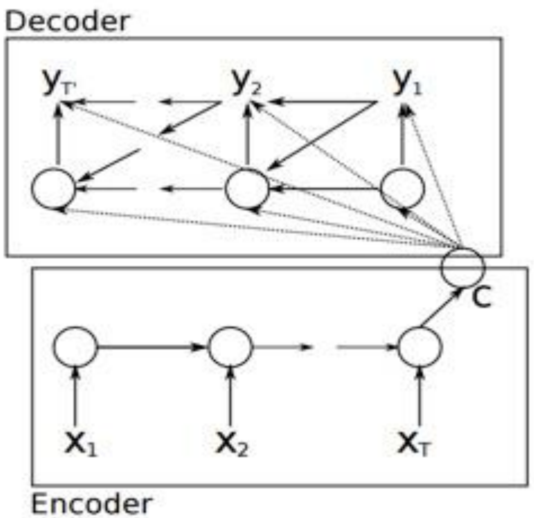
# Introduction

# Traditional SMT and Neural MT



# Neural Machine Translation

- Proposed by Google and Montreal University in 2014
- Is called
  - Sequence-to-sequence model
  - End-to-end model
- Input sentence is encoded into fix-length vector, and from the vector translated sentence is produced. That's all
- Various extensions is emerged
  - LSTM, GRU, Bidirectional Encoding, Attention Mechanism, …



# Pros and Cons of NMT

| Pros | Cons |
|---|---|
| ✓ no need domain knowledge<br>✓ no need to store explicit TM and LM<br>✓ Can jointly train multiple features<br>✓ Can implement decoder easily | ✓ Is time consuming to train NMT model<br>✓ Is slow in decoding, if target vocab. is large<br>✓ Is weak to OOV problem<br>✓ Is difficult to debug |

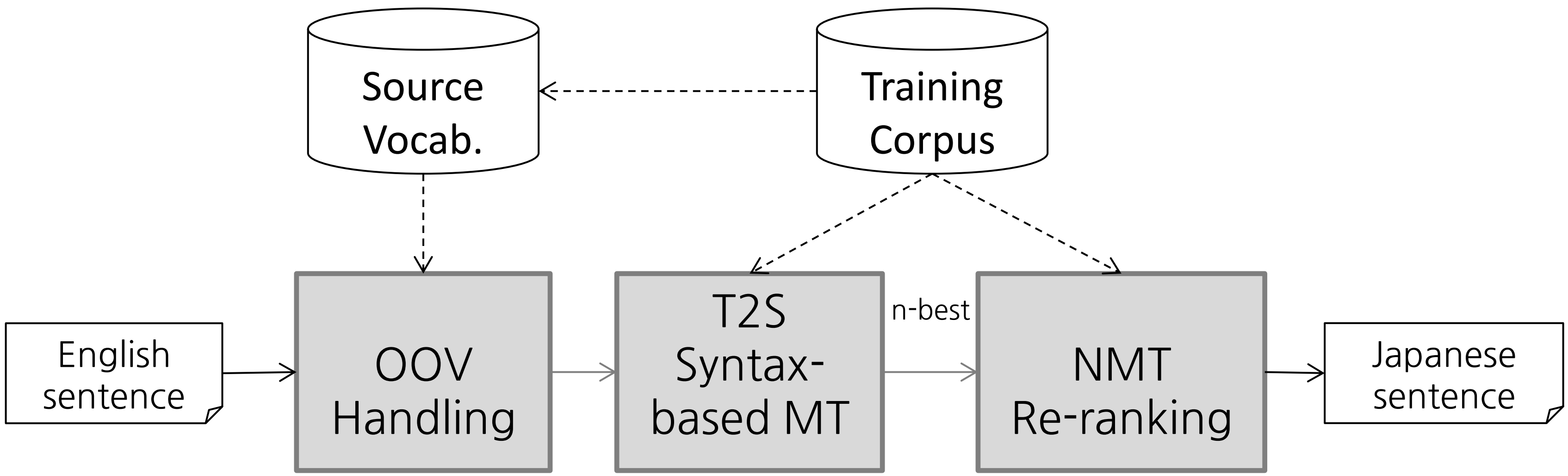
# At WAT 2015 ...

- Two tasks
  - English-Japanese MT
  - Korean-Japanese MT
- Methods of MT
  - Traditional SMT
  - Neural MT
  - Traditional SMT + Neural MT

# English-to-Japanese

# Machine Translation Task

# Outline of ENG-JPN MT Task



# Tree-to-String Syntax-based MT

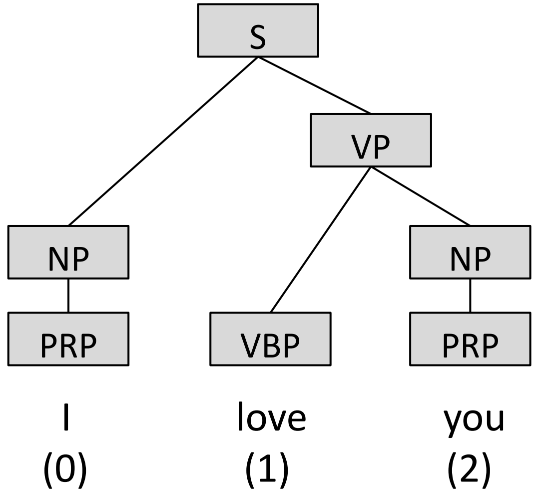
- Training Corpus
  - Translation model :
    - 1 million sentence pairs (train-1.txt)
  - Language model :
    - 3 million Japanese sentences (train-1.txt, train-2.txt)

- Tokenizer
  - English: Moses tokenizer
  - Japanese: In-house tokenizer and POS tagger

- T2S model
  - Assign linguistic syntax label to X hole of HPB model
  - Use Berkeley parser

# Tree-to-String Syntax-based MT 2/2

- Rule Augmentation
  - Proposed by CMU's venugopal and Zollmann in 2006
  - Extract more rules by modifying parse trees
  - Use relax-parser in Moses toolkit (option: SAMT 2)



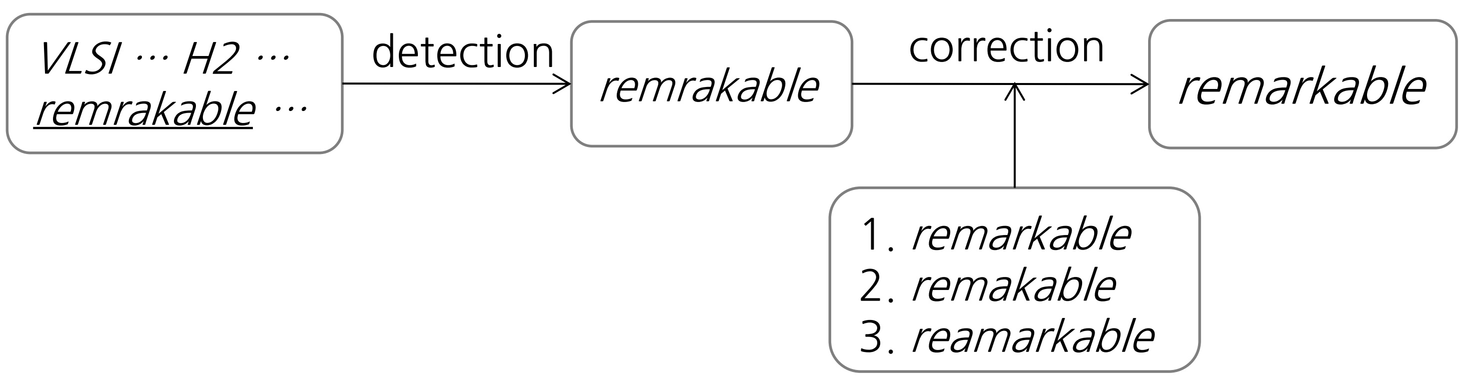

| Baseline nodes | Additional nodes |
|---|---|
| 0-0 PRP<br>0-0 NP<br>1-1 VBP<br>2-2 PRP<br>2-2 NP<br>1-2 VP<br>0-2 S | 1-2 VBP+PRP<br>0-2 PRP+VP<br>0-1 PRP++VBP |

# Handling OOV

1) Hyphen word split
- Ex.) nano-laminate -> nano laminate

2) English spell correction
- Use open source spell checker, '*Aspell*'

| | |
|---|---|
| Detection Phrase | ✓ Based on skip rules<br>✓ Skip the word containing capital, number or symbol |
| Correction Phrase | ✓ Based on edit distance<br>✓ Because large gap causes wrong correction<br>✓ Select one with shortest distance among top-3 suggestion |

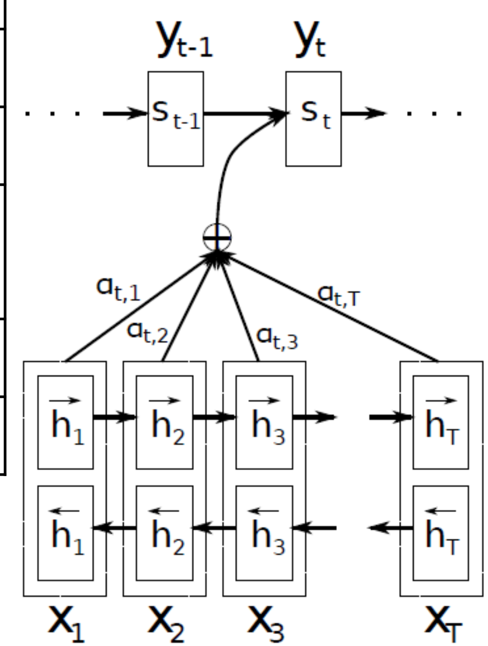



[Suggestion by Aspell]

# Neural Machine Translation (1/2)

- RNN with an attention mechanism [Bahdanau, 2015]

| | |
|---|---|
| Tokenization | English: word-level<br>Japanese: char-level |
| # of vocab. | English: 245k<br>Japanese: 6k |
| BI representation | Use<br>Ex) 大学生 => 大/B 学/I 生/I |
| Dim. of word-embedding | 200 |
| Size of recurrent unit | 1000 |
| Optimization | Stochastic gradient descent(SGD) |
| Drop-out | Don't use |
| Time of training | 10 days (4 epoch) |



# Neural Machine Translation (2/2)

$$h_t = [\overleftarrow{h}_t; \vec{h}_t]$$

$$\overleftarrow{h}_t = f_{GRU}(W_{s\_we} x_t, \overleftarrow{h}_{t+1})$$

$$\vec{h}_t = f_{GRU}(W_{s\_we} x_t, \vec{h}_{t-1})$$

$$c_t = \sum_{i=1}^{T} \alpha_{ti} h_i$$

$$\alpha_{ti} = \frac{\exp(e_{tj})}{\sum_{j=1}^{T} \exp(e_{tj})}$$

$$e_{ti} = f_{FFNN}(z_{t-1}, h_i, y_{t-1})$$

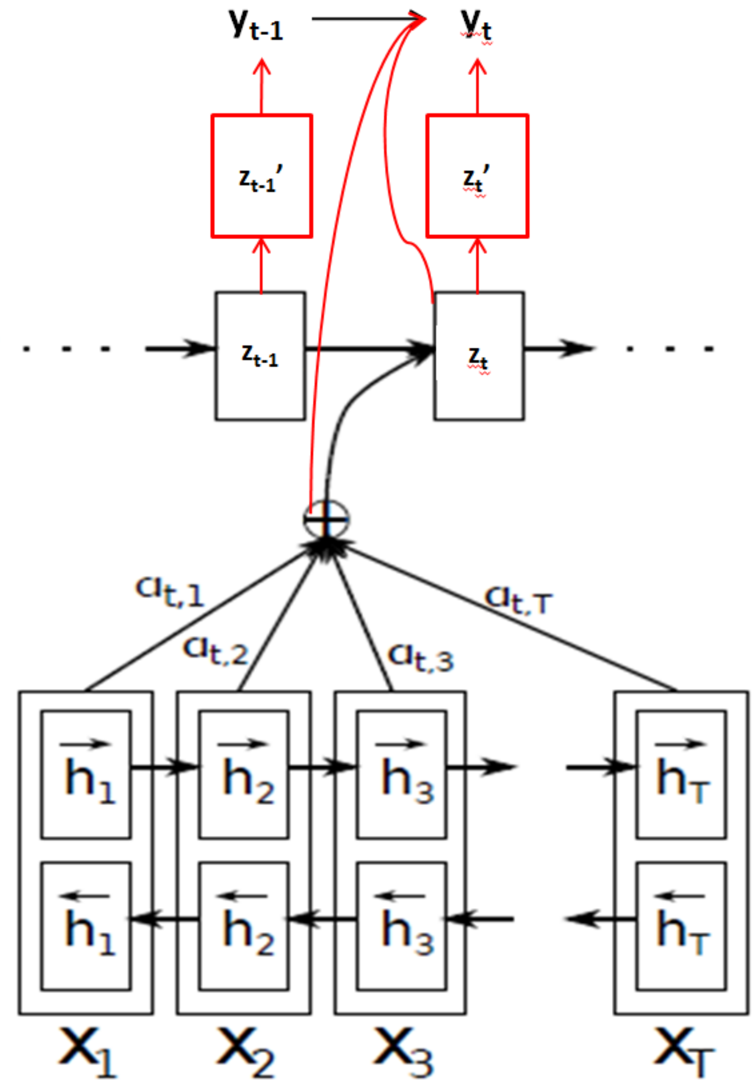
- New hidden state of the decoder

$$z_t = f_{GRU}(y_{t-1}, z_{t-1}, c_t)$$

- Prob. of the next target word

$$\begin{aligned} p(y_t|y_{<t}, x) = y_t^T f_{softmax}\{& W_{z'y} z'_t \\ &+ W_{zy} z_t + W_{cy} c_t \\ &+ W_{yy}(W_{t\_we} y_{t-1}) + b_y\} \end{aligned}$$

$$z'_t = f_{ReLU}(W_{zz'} z_t)$$




[ Modified RNN ]

# Experimental Results (T2S Syntax-based MT)

| SYS | BLEU | #Rules |
|---|---|---|
| T2S SB MT | 31.34 | 250M |
| + Rule augmentation | 32.48 | 1950M |
| + Parameter modification | 32.63 | 1950M |
| + OOV handling | 32.76 | 1950M |

- Rule augmentation increases both BLEU and #Rules
- OOV handling improves the performance

# Experimental Results (Neural MT)

| NMT Model | BLEU |
|---|---|
| RNN (target word-level) | 29.78 |
| RNN (target char-level) | 31.25 |
| RNN (target char-level with BI) | 32.05 |
| Modified RNN (target char-level with BI) | 33.14 |

- Char-level of target language is better than word-level
- BI representation is helpful
- Modified RNN is better than original RNN

# Experimental Results (/w Human evaluation)

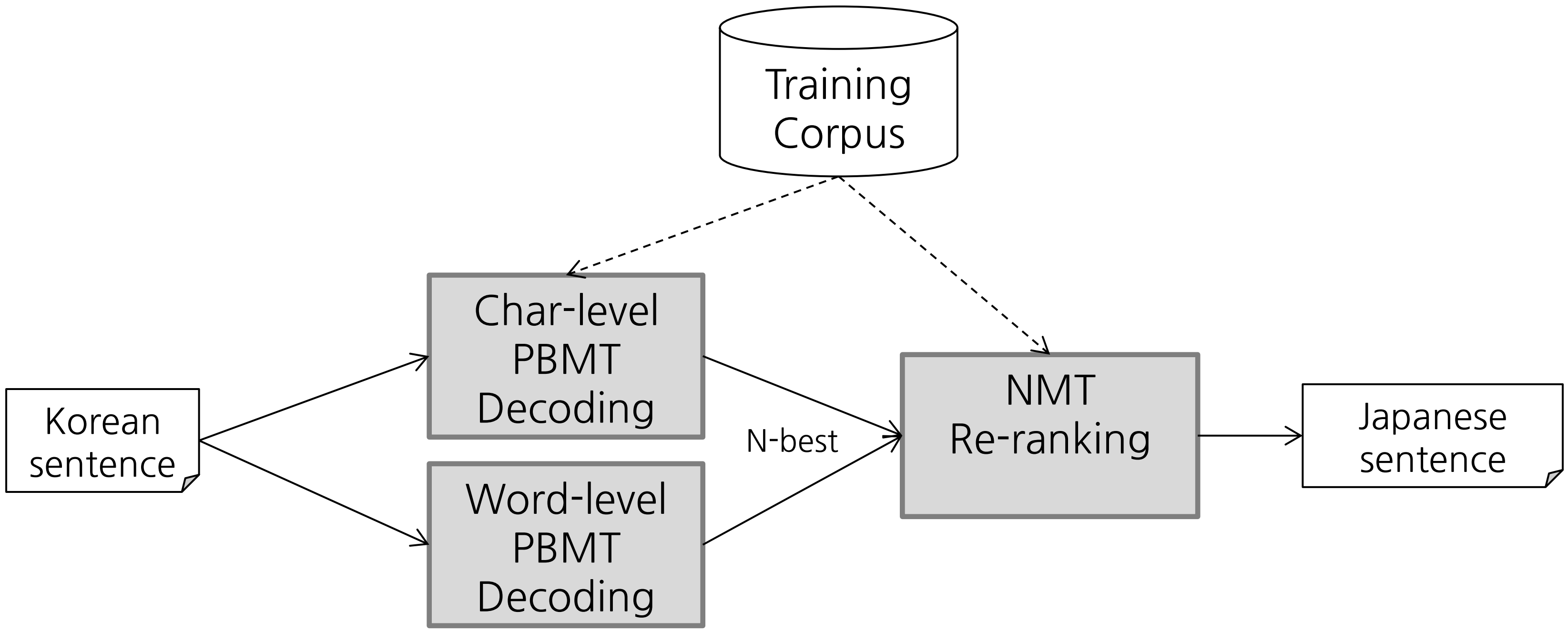
| SYS | ENG-JPN | |
|---|---|---|
| | BLEU | Human |
| T2S SB MT* only | 32.76 | - |
| NMT** only | 33.14 | 48.50 |
| T2S SB MT* + NMT** re-ranking | 34.60 | 53.25 |

- NMT only outperform T2S SB MT
- NMT re-ranking gives the best

- T2S SB MT* : Rule augmentation + Parameter modification + OOV handling
- NMT** : Modified NMT using target char. seg. with B/I

# Korean-to-Japanese Machine Translation Task

# Outline of KOR-JPN MT Task



# Phrase-based MT system

- Training Corpus
  - Translation model & Language model
    - 1 million sentence pairs (JPO corpus)

- Word-level PB MT
  - use Mecab-ko and Juman for tokenization
  - 5-gram LM

- Char-level PB MT
  - tokenize Korean and Japanese into char-level
  - 10-gram LM
  - Max-phrase length : 10

# Neural Machine Translation

- RNN using attention mechanism [Bahdanau, 2015]

| | |
|---|---|
| Tokenization | Korean: word-level<br>Japanese: char-level |
| # of vocab. | Korean: 60k<br>Japanese: 5k |
| BI representation | Use<br>Ex) 大学生 => 大/B 学/I 生/I |
| Dim. of word-embedding | 200 |
| Size of recurrent unit | 1000 |
| Optimization | Stochastic gradient descent(SGD) |
| Drop-out | Don't use |
| Time of training | 10 days (4 epoch) |

# Combination of PBMT+ NMT

- Rule-based
  - Choose the result of char-based PB  if there is OOV in word-level
  - Choose the result of word-based PB, otherwise
- NMT-based
  - Re-rank simply by NMT score

# Experimental Results

| SYS | BLEU |
| --- | --- |
| Word PB | 70.36 |
| Character PB | 70.31 |
| Word PB + Character PB | 70.91 |

- Character-level PB is comparable to Word-level PB
- Combined system has the best result

# Experimental Results (/w human evaluation)

| SYS | KOR-JPN | |
|---|---|---|
| | BLEU | Human |
| Word PB + Character PB | 70.91 | 6.75 |
| NMT only | 65.72 | - |
| Word PB + Character PB + NMT re-ranking | 71.38 | 14.75 |

- NMT only doesn't outperform PBMT
- NMT re-ranking gives the best

# Summary

- We apply different MT models for each task
- T2S/PB SMT + NMT Re-ranking is best in both tasks
- Char-level tokenization of target language is useful for NMT
  - Speed up the time of training
  - Vanish OOV problem
  - Give the better BLEU score
- BI representation of char-level tokenization is helpful also for NMT
- In the future, we will apply our method to other language-pair; CHN-JPN