# KyotoEBMT System Description for the 2nd Workshop on Asian Translation

John Richardson Raj Dabre Chenhui Chu
Fabien Cromières Toshiaki Nakazawa Sadao Kurohashi

# Outline

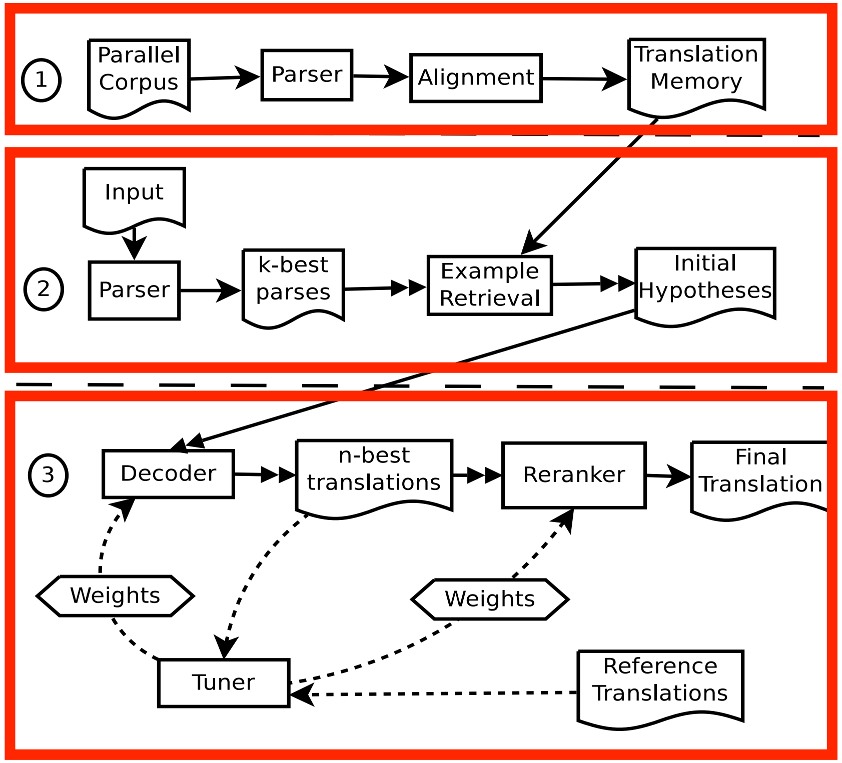
- Overview of the system
- Improvements since WAT2014
- Results for WAT2015
- Conclusion

# Overview of Kyoto-EBMT

# KyotoEBMT Overview

- Example-Based MT paradigm
  - Need parallel corpus
  - Few language-specific assumptions
    - still a few language-specific rules
- Tree-to-Tree Machine Translation
  - Maybe the least commonly used variant of x-to-x
  - Sensitive to parsing quality of both source and target languages
  - Maximize the chances of preserving information
- Dependency trees
  - Less commonly used than Constituent trees
  - Most natural for Japanese
  - Should contain all important semantic information

# KyotoEBMT pipeline

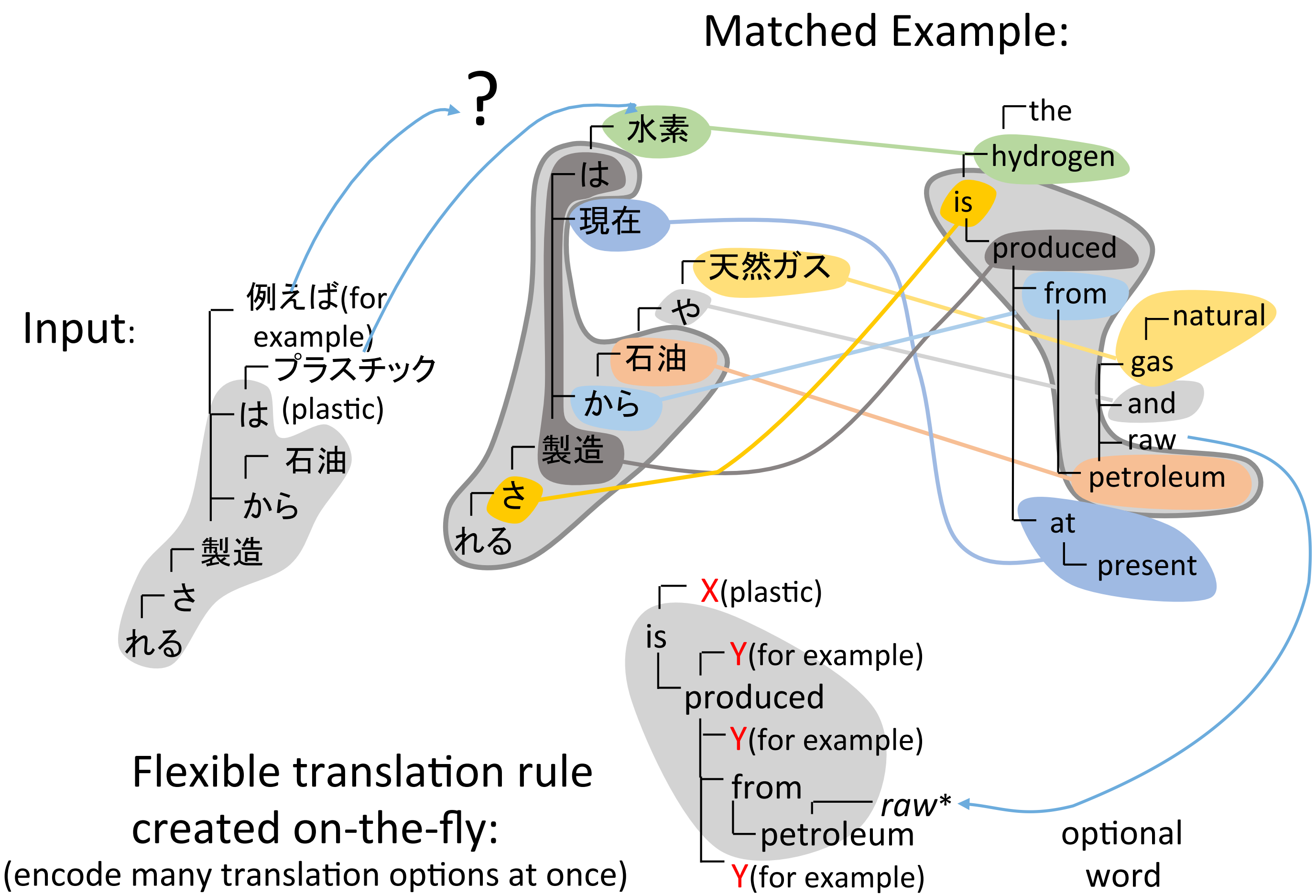



- Somehow classic pipeline
  - 1- Preprocessing of the parallel corpus
  - 2- Processing of input sentence
  - 3- Decoding/Tuning/Reranking
- Tuning and reranking done with kbMira
  - seems to work better than PRO for us

# Other specificities

- No "phrase-table"
  - all translation rules computed on-the-fly for each input
  - cons:
    - possibly slower (but not so slow)
    - computing significance/ sparse features more complicated
  - pros:
    - full-context available for computing features
    - no limit on the size of matched rules
    - possibility to output perfect translation when input is very similar to an example
- "Flexible" translation rules
  - Optional words
  - Alternative insertion positions
  - Decoder can process flexible rules more efficiently than a long list of alternative rules
    - some "flexible rules" may actually encode >millions of "standard rules"

# Flexible Rules Extracted on-the-fly




X: Simple case
(X has an equivalent in the source example)

Y: ambiguous insertion position

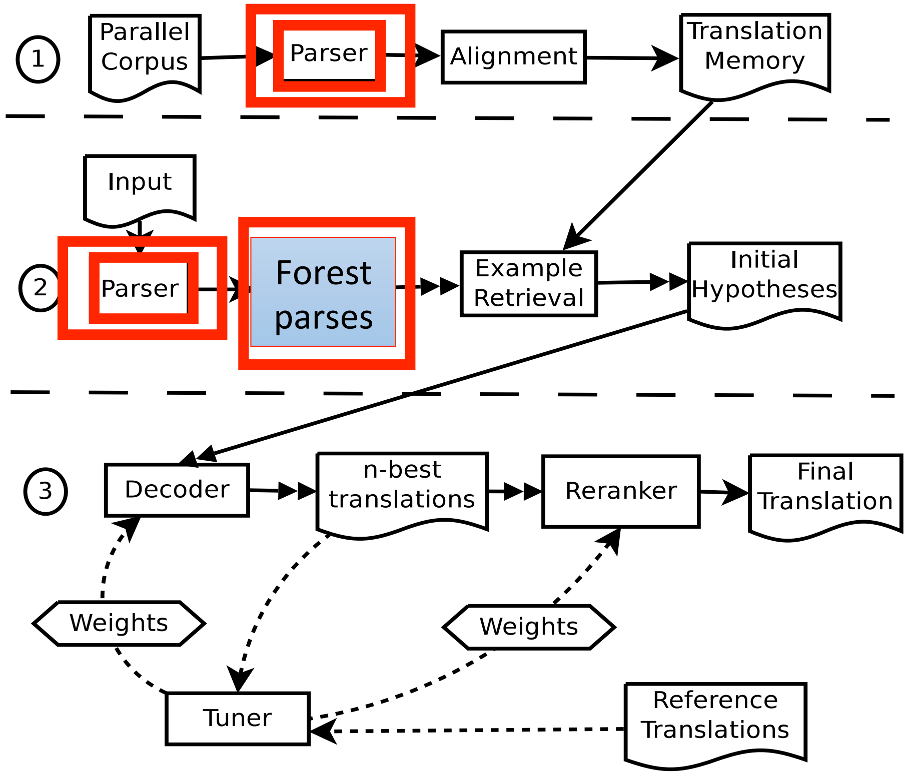
"raw": null-aligned -> optional

# Other specificities

- No "phrase-table"
  - all translation rules computed on-the-fly for each input
  - cons:
    - possibly slower (but not so slow)
    - computing significance/ sparse features more complicated
  - pros:
    - full-context available for computing features
    - no limit on the size of matched rules
    - possibility to output perfect translation when input is very similar to an example
- "Flexible" translation rules
  - Optional words
  - Alternative insertion positions
  - Decoder can process flexible rules more efficiently than a long list of alternative rules
    - some "flexible rules" may actually encode >millions of "standard rules"

# Improvements since WAT2014

# KyotoEBMT improvements

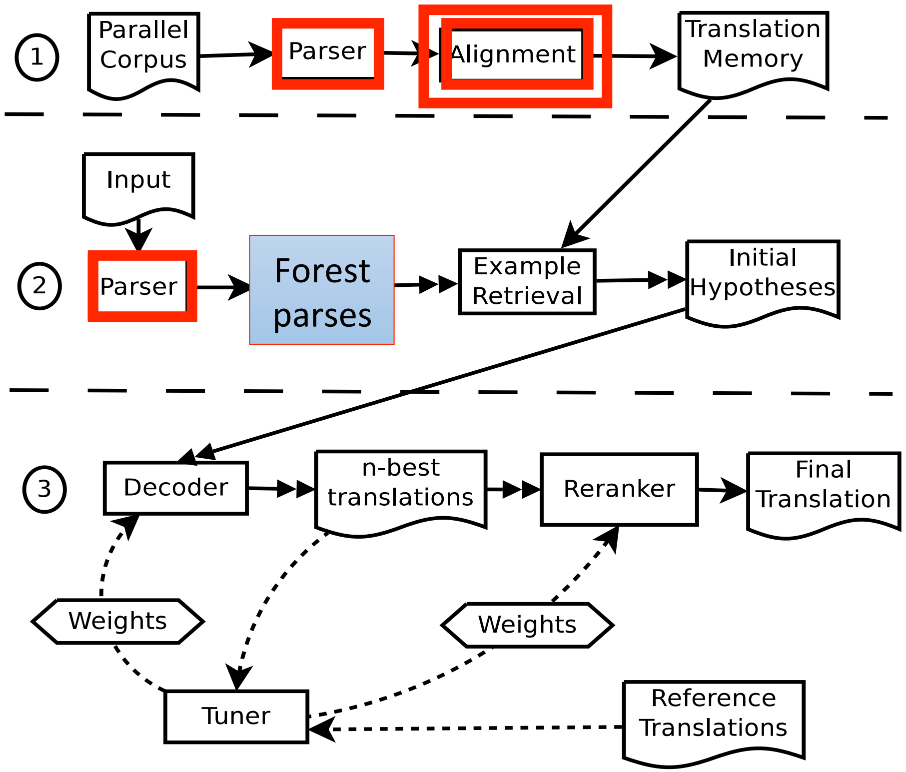



- Our system is <u>very</u> sensitive to parsing errors
- Continuous improvements to
  - Juman
  - KNP
  - SKP
- Added support for parse forests
  - (compact representations)

# Forest Input

- A <u>partial solution to the issues of Tree-to-Tree MT</u>
  - Can help with parsing errors
  - Can help with syntactic divergences
- In WAT2014,
  - we used 20-best input parses
  - n-best list of all inputs merged and reranked
- Now, with forest:
  - an exponential number of input parses can be encoded
  - the selection of parses is done during decoding

# KyotoEBMT improvements




- System is also very sensitive to alignment errors
- We used to correct alignments by using dependency trees (Nakazawa and Kurohashi, 2012)
- Now we further improve them with Nile (Riesa et al., 2011)

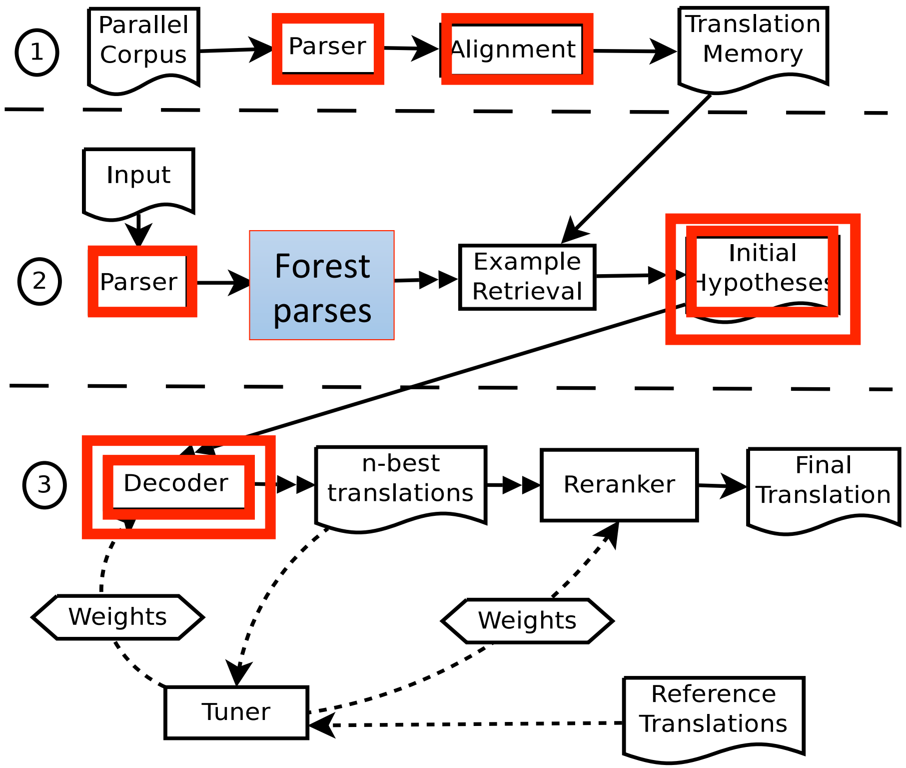
# Alignment Improvements

- Used Nile (Riesa et al., 2011) to improve the alignment
  - As suggested by (Neubig and Duh, 2014)
  - Require us to parse into constituent trees as well
    - Ckylark parser for Japanese (Oda+, 2015)
    - Berkeley Parser for Chinese/English
- Nile becomes the third element of an alignment pipeline

| | Giza++ | with dependency trees: (Nakazawa and Kurohashi, 2012) | with constituent trees: Nile |
|---|---|---|---|
| JC alignment F -> | F:0.63 | F:0.69 | F:0.75 |

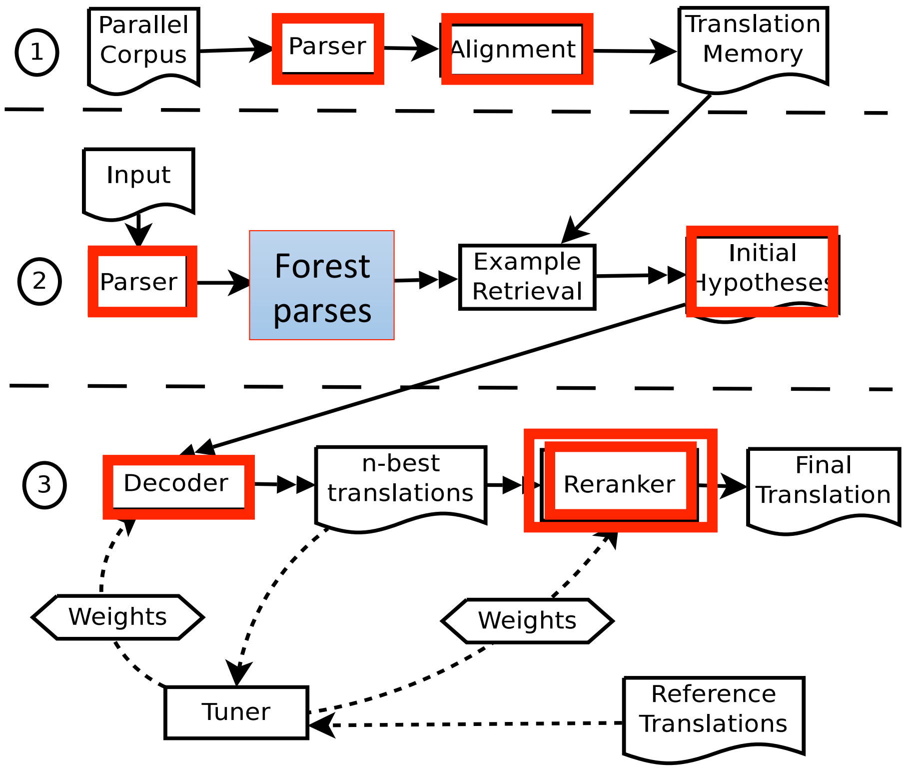
(Giza++ / Nile only -> F:0.70)

# KyotoEBMT improvements




- Many small improvements
  - Better handling of flexible rules
  - Bug fixes
- 10 new features
  - alignment score
  - context similarity score based on word2vec vectors
  - …

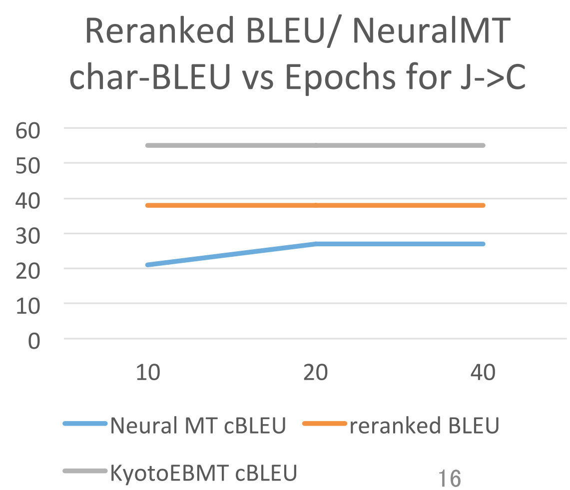
# KyotoEBMT improvements



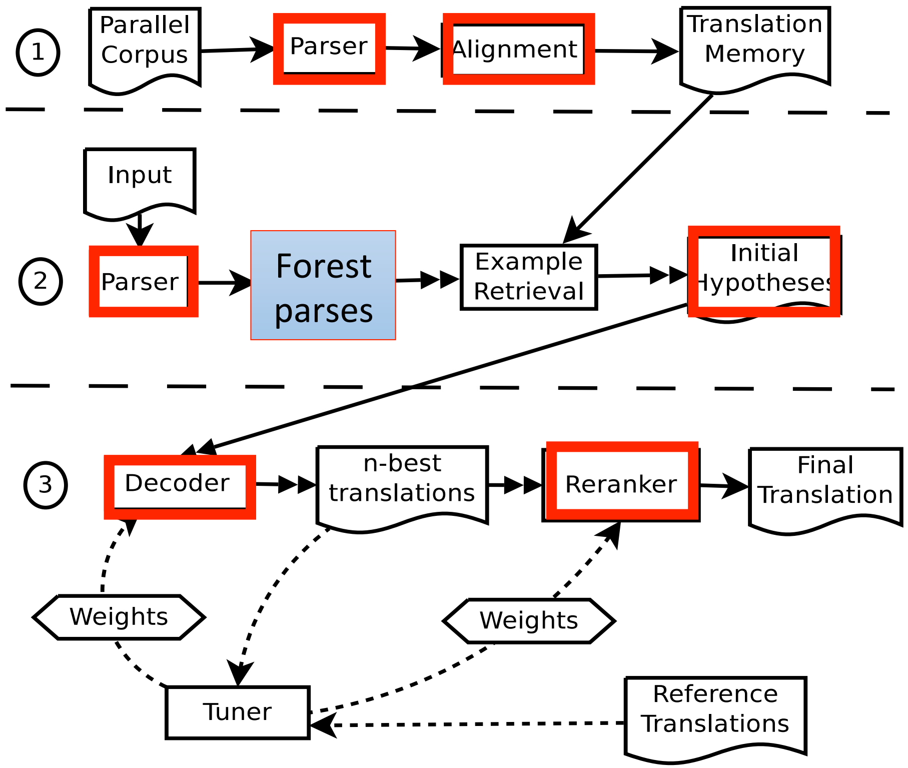

- Reranking
- Previously used features:
  - 7-gram language model
  - RNNLM language model
- Now also using a Neural MT based bilingual Language Model

# Bilingual Neural Network Language Model

- Combine Neural MT with EBMT
- We use the state-of-the-art model described by (Bahdanau et al., 2015)
  - Model seen as a Language Model conditionalized on the input
- Remarks:
  - Processing Japanese and Chinese as sequences of characters gave good results
    - No need to limit vocabulary (~4000/6000 characters for J/C)
    - Avoid segmentation issues
    - Faster training
  - Neural MT models alone produced bad translations
    - eg. Character BLEU for C->J almost half that of KyotoEBMT
  - Reranking performances saturates before MT performances



# KyotoEBMT improvements




- <u>Improved working methods</u> (that matters!)
  - automatic nightly testing for variations in BLEU/ assertion errors/ memory leaks
- Overall improvements across all the pipeline
- Estimating the global contribution of each element is tough, but here are the final results, …

# Results

# Results for WAT2015

| | Reranking | BLEU | RIBES | HUMAN |
|---|---|---|---|---|
| J->E | NO | 21.31 (+0.71) | 70.65 (+0.53) | 16.50 |
| | YES | 22.89 (+1.82) | 72.46 (+2.56) | 32.50 |
| E->J | NO | 30.69 (+0.92) | 76.78 (+1.57) | 40.50 |
| | YES | 33.06 (+1.97) | 78.95 (+2.99) | 51.00 |
| J->C | NO | 29.99 (+2.78) | 80.71 (+1.58) | 16.00 |
| | YES | 31.40 (+3.83) | 82.70 (+3.87) | 12.50 |
| C->J | NO | 36.30 (+2.73) | 81.97 (+1.87) | 16.75 |
| | YES | 38.53 (+3.78) | 84.07 (+3.81) | 18.50 |

# Results for WAT2015

| | Reranking | BLEU | RIBES | HUMAN |
|---|---|---|---|---|
| J->E | NO | **+0.71** | 70.65 [illegible] | [illegible] |
| | YES | **+1.82** | [illegible] | [illegible] |
| E->J | NO | **+0.92** | [illegible] | [illegible] |
| | YES | **+1.97** | [illegible] | [illegible] |
| J->C | NO | **+2.78** | 80.71 [illegible] | [illegible] |
| | YES | **<u>+3.83</u>** | 82.70 (+3.87) | 12.50 |
| C->J | NO | **+2.73** | 81.97 (+1.87) | 16.75 |
| | YES | **<u>+3.78</u>** | 84.07 (+3.81) | 18.50 |

The various improvements
lead to good changes in BLEU.
Almost +4 BLEU for the JC/CJ

# Results for WAT2015

| | Reranki[illegible] | [illegible] | [illegible] | HUMAN |
|---|---|---|---|---|
| J-> [illegible] | [illegible] | [illegible] | [illegible] | 16.50 |
| | [illegible] | [illegible] | [illegible] | 32.50 |
| E->J | NO | [illegible] | [illegible] | 40.50 |
| | YES | 33.06 (+1.9[illegible] | 78.95 (+2.99) | 51.00 |
| J->C | NO | 29.99 (+2.78) | 80.71 (+1.58) | 16.00 |
| | YES | 31.40 (+3.83) | 82.70 (+3.87) | 12.50 |
| C->J | NO | 36.30 (+2.73) | 81.97 (+1.87) | 16.75 |
| | YES | 38.53 (+3.78) | 84.07 (+3.81) | 18.50 |

Mystery!
Only for J->C, we find that reranking decreased
Human Evaluation score.
(While still improving BLEU/RIBES)

# Results for WAT2015

| | Reranking | BLEU | RIBES | HUMAN |
|---|---|---|---|---|
| J->E | NO | 21.31 (+0.71) | 70.65 (+0.53) | 16.50 |
| | YES | 22.89 (+1.82) | 72.46 (+2.56) | 32.50 |
| E->J | NO | 30.69 (+0.92) | 76.78 (+1.57) | 40.50 |
| | YES | 33.06 (+1.97) | 78.95 (+2.99) | 51.00 |
| J->C | NO | 29.99 (+2.78) | 80.71 (+1.58) | 16.00 |
| | YES | 31.40 (+3.83) | 82.70 (+3.87) | 12.50 |
| C->J | NO | 36.30 (+2.73) | 81.97 (+1.87) | 16.75 |
| | YES | 38.53 (+3.78) | 84.07 (+3.81) | 18.50 |

# Code is available and Open-sourced

- Version 1.0 released
  - 1 year after version 0.1
  - 2 years after development started
- Downloadable at: http://nlp.ist.i.kyoto-u.ac.jp/kyotoebmt/
- GPL Licence

# Conclusion

- KyotoEBMT is a (Dependency) Tree-to-Tree MT system with state-of-the-art results
- Open-sourced (http://nlp.ist.i.kyoto-u.ac.jp/kyotoebmt/)
- Improvements across the whole pipeline lead us to close to +4 BLEU improvements
- Some future works:
  - Make more use of the target structure
  - Use of deep learning features in the decoder
    - eg. as in (Devlin et al., 2014)
  - …

# Thank you!