# NCODE:
# an Open Source Bilingual N-gram SMT Toolkit

**Josep M. Crego**, *François Yvon and José B. Mariño*

*jmcrego@limsi.fr*

September 5 – 10, 2011 - *FBK*, Trento (Italy)

# Table of contents

# Plan

# History

- Phrase-based approach (early 2000)
  - state-of-the-art results for many MT tasks

# History

- Phrase-based approach (early 2000)
  - state-of-the-art results for many MT tasks
- Bilingual *n*-gram approach (an alternative to PBMT)
  - Derives from the finite-state perspective introduced by (Casacuberta and Vidal, 2003)
  - First implementation dates back to 2004 (Ph.D. at UPC)
  - Extended for the last three years (Postdoc at Limsi-CNRS)

# Standard SMT mainstream

1 take a set of parallel sentences (*bitext*)
- align each pair $(\mathbf{f},\mathbf{e})$, word for word
- train translation model: the "phrase" table $\{(f,e)\}$

2 take a set of monolingual texts
- train statistical target language model

3 make sure to tune your system

4 translate $\mathbf{f} = \operatorname{argmax}_{\mathbf{e} \in E} \{\sum_{k=1}^{K} \lambda_k F_k(\mathbf{e},\mathbf{f})\}$

5 evaluate

6 not happy ? goto 1

# Underlying formal device: finite-state SMT

- phrase-table lookup [*pt*] is finite-state
- *n*-gram models [*lm*] can be implemented as weighted FSA

# Underlying formal device: finite-state SMT

- phrase-table lookup [*pt*] is finite-state
- *n*-gram models [*lm*] can be implemented as weighted FSA
- monotonic decode of $\mathbf{f}$:

$$\mathbf{e}^* = bestpath(\pi_2(\mathbf{f} \circ pt) \circ lm)$$

# Underlying formal device: finite-state SMT

- phrase-table lookup [*pt*] is finite-state
- *n*-gram models [*lm*] can be implemented as weighted FSA
- monotonic decode of **f**:

$$\mathbf{e}^* = bestpath(\pi_2(\mathbf{f} \circ pt) \circ lm)$$

- decode with reordering:

$$\mathbf{e}^* = bestpath(\pi_2(\mathbf{perm}(\mathbf{f}) \circ pt) \circ lm)$$

$perm(\mathbf{f})$ is a word lattice (FSA) containing reordering hypotheses

# Bilingual *n*-grams

- a **bilingual** *n*-gram language model as main translation model
  - Sequence of tuples (training bitexts):

| we | want | translations | perfect |
|---|---|---|---|
| nous | voulons | des traductions | parfaites |

# Bilingual *n*-grams

- a **bilingual** *n*-gram language model as main translation model
  - Sequence of tuples (training bitexts):

| we | want | translations | perfect |
| --- | --- | --- | --- |
| nous | voulons | des traductions | parfaites |

- smaller units are more **reusable** than longer ones (less sparse)

| we want | translations | perfect |
| --- | --- | --- |
| nous voulons | des traductions | parfaites |

# Bilingual *n*-grams

- a **bilingual** *n*-gram language model as main translation model
  - Sequence of tuples (training bitexts):

| we | want | translations | perfect |
|---|---|---|---|
| nous | voulons | des traductions | parfaites |

- smaller units are more **reusable** than longer ones (less sparse)

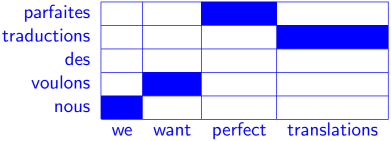
| we want | translations | perfect |
|---|---|---|
| nous voulons | des traductions | parfaites |

- translation context introduced via tuple **n**-grams

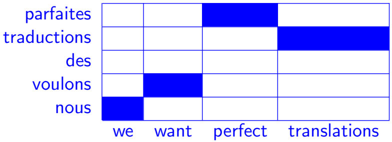
$$p((s,t)_k|(s,t)_{k-1},(s,t)_{k-2})$$

multiple back-off schemes, smoothing techniques, *etc.*

# Tuples from word alignments



# Tuples from word alignments

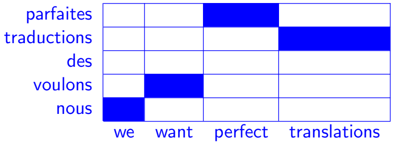



1 a **unique** segmentation of each sentence pair:

- no word in a tuple can be aligned to a word outside the tuple
- target-side words in tuples follow the original word order
- no smaller tuples can be found

| | | | | |
|---|---|---|---|---|
| we | want | NULL | translations | perfect |
| nous | voulons | des | traductions | parfaites |

# Tuples from word alignments




1 a **unique** segmentation of each sentence pair:

- no word in a tuple can be aligned to a word outside the tuple
- target-side words in tuples follow the original word order
- no smaller tuples can be found

| | | | | |
|---|---|---|---|---|
| we | want | NULL | translations | perfect |
| nous | voulons | des | traductions | parfaites |

2 source-NULLed units are not allowed (complexity issues):

- attach the target word to the **previous**/**next** tuple

| | | | |
|---|---|---|---|
| we | want | translations | perfect |
| nous | voulons | des traductions | parfaites |

## Coupling reordering and decoding

$$\mathbf{e}^* = \mathit{bestpath}(\pi_2(\mathbf{perm}(\mathbf{f}) \circ \mathit{pt}) \circ \mathit{lm})$$

- **perm** is responsible of the NP-completeness of SMT

# Coupling reordering and decoding

$$\mathbf{e}^* = \mathit{bestpath}(\pi_2(\mathbf{perm}(\mathbf{f}) \circ \mathit{pt}) \circ \mathit{lm})$$

- **perm** is responsible of the NP-completeness of SMT

Problem: Full permutations computationally too expensive (EXP search)

# Coupling reordering and decoding

$$\mathbf{e}^* = bestpath(\pi_2(\mathbf{perm}(\mathbf{f}) \circ pt) \circ lm)$$

- **perm** is responsible of the NP-completeness of SMT

Problem: Full permutations computationally too expensive (EXP search)

Sol1: Heuristic constraints (distance-based): IBM, ITG, *etc.*
POLY search, but little correlation with language

## Coupling reordering and decoding

$$\mathbf{e}^* = bestpath(\pi_2(\mathbf{perm}(\mathbf{f}) \circ pt) \circ lm)$$

- **perm** is responsible of the NP-completeness of SMT

Problem: Full permutations computationally too expensive (EXP search)

Sol1: Heuristic constraints (distance-based): IBM, ITG, *etc.*
POLY search, but little correlation with language

Sol2: Linguistically-founded rewrite rules:

- learn **reordering rules** from the bitext word alignments
  perfect translations $\rightsquigarrow$ translations perfect
- compose rules as a reordering transducer: $R = \bigcirc_i (r_i \cup Id)$
- in decoding: $perm(\mathbf{f}) = \mathbf{f} \circ R$

$perm(\mathbf{f})$ is a word lattice (FSA) with reordering hypotheses

# Plan

Bilingual *n*-gram approach to SMT

Decoding
- Search structure
- Algorithm
- Complexity and speed ups

The NCODE toolkit

Comparison: NCODE vs. MOSES

Concluding remarks

# Search structure

- Exhaustive search is *unfeasible* $\rightsquigarrow$ pruning needed!

# Search structure

- Exhaustive search is *unfeasible* $\rightsquigarrow$ pruning needed!
- Important: which hypotheses are compared to be pruned?

# Search structure

- Exhaustive search is *unfeasible* $\rightsquigarrow$ pruning needed!
- Important: which hypotheses are compared to be pruned?
- Solution: use multiple stacks

- MOSES: $[I]$ stacks (hyps. generating the **same number** of words)
  - + Problem: Search bias (translate first 'easiest' segments)
  - + Solution: Use future cost estimation ($A^*$)

# Search structure

- Exhaustive search is *unfeasible* $\rightsquigarrow$ pruning needed!
- Important: which hypotheses are compared to be pruned?
- Solution: use multiple stacks

- MOSES: $[I]$ stacks (hyps. generating the **same number** of words)
  + Problem: Search bias (translate first 'easiest' segments)
  + Solution: Use future cost estimation ($A^*$)

Feature cost estimation problem for NCODE
(multiple $n$-gram LMs without accurate estimations)

# Search structure

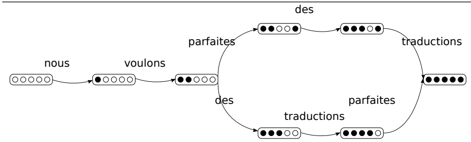
- Exhaustive search is *unfeasible* $\rightsquigarrow$ pruning needed!
- Important: which hypotheses are compared to be pruned?
- Solution: use multiple stacks

- MOSES: $[I]$ stacks (hyps. generating the **same number** of words)
  - Problem: Search bias (translate first 'easiest' segments)
  - Solution: Use future cost estimation ($A^*$)

  Feature cost estimation problem for NCODE
  (multiple *n*-gram LMs without accurate estimations)
- NCODE: $[2^J]$ stacks (hyps. translating the **same** input words)
  - Highly fair comparisons
  - Problem: efficiency problem ($2^J$)
  - Solution: limit reordering (linguistically motivated)

# Search algorithm (sketched)

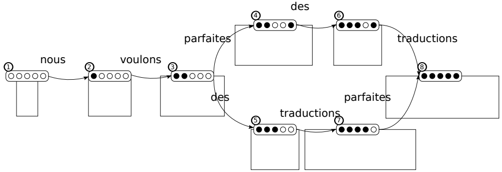
- Word lattice encoding permutations (up to $2^J$ nodes)




- word lattice $G$ as input of the search algorithm

# Search algorithm (sketched)

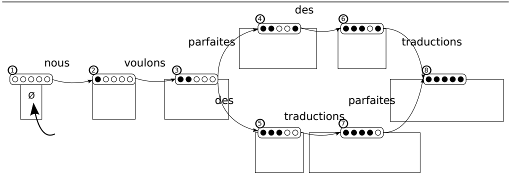
- Word lattice encoding permutations (up to $2^J$ nodes)
- Partial translation hypotheses (up to $2^J$ stacks)




- word lattice $G$ as input of the search algorithm
- nodes of the input lattice are transformed into search stacks after being topologically sorted

# Search algorithm (sketched)

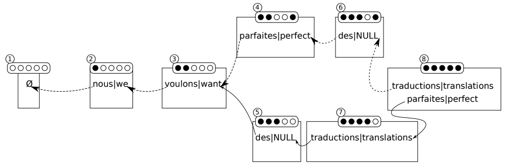
- Word lattice encoding permutations (up to $2^J$ nodes)
- Partial translation hypotheses (up to $2^J$ stacks)




- word lattice $G$ as input of the search algorithm
- nodes of the input lattice are transformed into search stacks after being topologically sorted
- search starts setting the empty hypothesis in stack ($0^J$)

# Search algorithm (sketched)

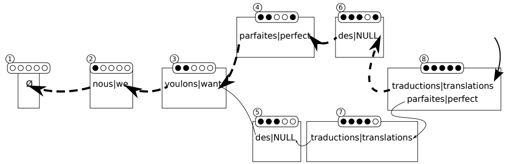
- Word lattice encoding permutations (up to $2^J$ nodes)
- Partial translation hypotheses (up to $2^J$ stacks)




- word lattice $G$ as input of the search algorithm
- nodes of the input lattice are transformed into search stacks after being topologically sorted
- search starts setting the empty hypothesis in stack $(0^J)$
- it proceeds expanding hypotheses in the stacks following the topological sort

# Search algorithm (sketched)

- Word lattice encoding permutations (up to $2^J$ nodes)
- Partial translation hypotheses (up to $2^J$ stacks)




- word lattice $G$ as input of the search algorithm
- nodes of the input lattice are transformed into search stacks after being topologically sorted
- search starts setting the empty hypothesis in stack $(0^J)$
- it proceeds expanding hypotheses in the stacks following the topological sort
- Translation output through tracing back the best hypothesis of the ending stacks

# Search complexity and speed ups

- Complexity: upper bound of the number of hypotheses valued for an exhaustive search:

$$2^J \times (|V_u|^{n_1-1} \times |V_t|^{n_2-1})$$

  - $J$ is the length of the input sentence,
  - $|V_u|$ is the size of the vocabulary of translation units,
  - $|V_t|$ is the size of the target vocabulary.
  - $n_1/n_2$ are the order of the bilingual/target $n$-gram LMs,

# Search complexity and speed ups

- Complexity: upper bound of the number of hypotheses valued for an exhaustive search:

$$2^J \times (|V_u|^{n_1-1} \times |V_t|^{n_2-1})$$

  - $J$ is the length of the input sentence,
  - $|V_u|$ is the size of the vocabulary of translation units,
  - $|V_t|$ is the size of the target vocabulary.
  - $n_1/n_2$ are the order of the bilingual/target $n$-gram LMs,

- Speed ups:
  - Recombination: exact (unless $N$-best output required)
  - $i$-best hypotheses within a stack (beam pruning)
  - $i$-best translation choices (based on uncontextualized scores)
  - prune reordering rules (reduce the size of the input lattice)
  - use several threads (when possible)

# Plan

Bilingual *n*-gram approach to SMT

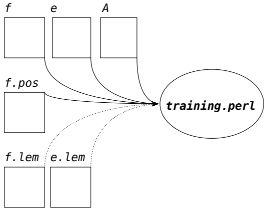
Decoding

The NCODE toolkit
- Training
- Inference
- Optimization

Comparison: NCODE vs. MOSES

Concluding remarks

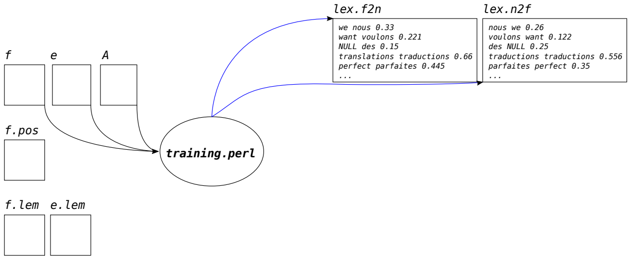
# Model estimation




`training.perl` [--first-step --last-step --output-dir]

- NCODE systems are built from a training bitext (f,e) and the corresponding word alignment (A). Part-of-speeches (f.pos) are (typically) used to learn rewrite rules
- Target n-gram LMs are **not** estimated within `training.perl`
- Training is deployed over 8 steps

# Model estimation



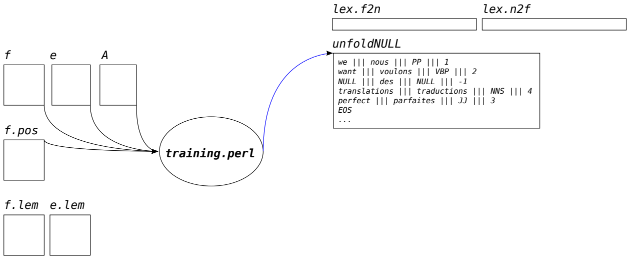

Step 0: lexicon distribution

- Distributions computed based on counts using word alignments:

$$P_{lex}(e, f) = \frac{count(f,e)}{\sum_{f'} count(f',e)} \quad ; \quad P_{lex}(f, e) = \frac{count(f,e)}{\sum_{e'} count(f,e')}$$

- **NULL** tokens are considered (to allow tuples with **NULL** target side)

# Model estimation

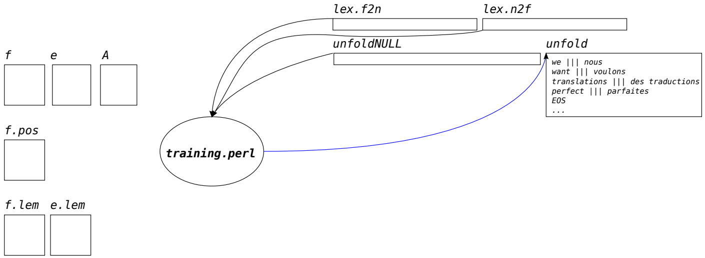



Step 1: tuple extraction

- Unfold technique previously outlined:

Minimal segmentation of source/target training sentences, following alignments and allowing source distortion

# Model estimation

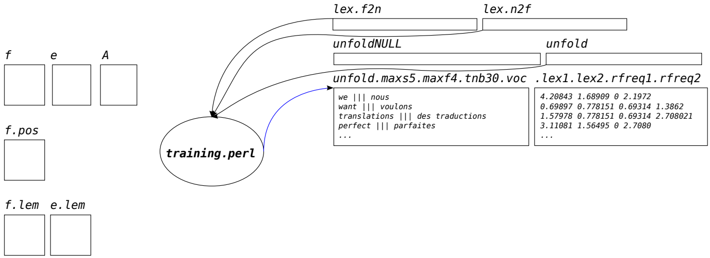



Step 2: tuple refinement (src-NULLed units)

- Source-NULLed words $(NULL|||des)$ are attached to the previous or the next unit, after evaluating the likelihood of both alternatives using the unit lexicon distribution $P_{lw}(e, f)$ (next slide):

$$max \begin{cases} P_{lw}(want|||voulons\ \mathbf{des}) \times P_{lw}(translations|||traductions) & 'attachment : previous' \\ \quad or & \\ P_{lw}(want|||voulons) \times P_{lw}(translations|||\mathbf{des}\ traductions) & 'attachment : next' \end{cases}$$

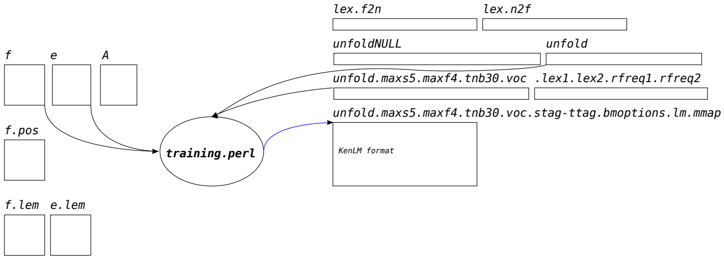
# Model estimation




Step 3: tuple pruning & uncontextualized distributions [`--max-tuple-length --max-tuple-fert --tuple-nbest`]

- Tuples filtered following several constraints (length, fertility, $n$-best translation choices per source segment)
- Conditional probability (x2): $P_{rf}(e,f) = \frac{count(f,e)}{\sum_{f'} count(f',e)}$ ; $P_{rf}(f,e) = \frac{count(f,e)}{\sum_{e'} count(f,e')}$
- Lexicon weights (x2):
  $P_{lw}(e,f) = \frac{1}{(J+1)^I} \prod_{i=1}^{I} \sum_{j=0}^{J} P_{lex}(e,f)$ ; $P_{lw}(f,e) = \frac{1}{(I+1)^J} \prod_{j=1}^{J} \sum_{i=0}^{I} P_{lex}(f,e)$

# Model estimation



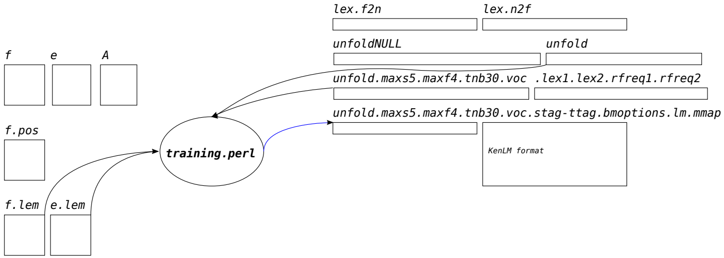

Step 4: bilingual n-gram lm [--train-src-bm --train-trg-bm --options-bm --name-src-bm --name-trg-bm]

- Standard *n*-gram LM (units built from words):

$$p(f_1^J, e_1^I) = \prod_{k=1}^{K} p((f,e)_k | (f,e)_{k-1}, \ldots, (f,e)_{k-n+1})$$

- Options passed to SRILM, Ex: –options-bm -order_3_-unk_-gt3min_1_-kndiscount_-interpolate

# Model estimation



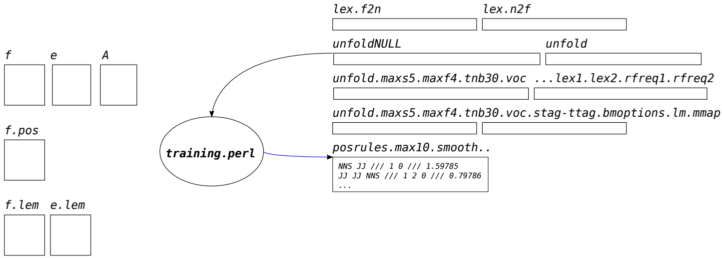

Step 4: bilingual n-gram lm [--train-src-bm --train-trg-bm --options-bm --name-src-bm --name-trg-bm]

- Bilingual units built from: POS-tags, lemmas, *etc.*, or any src/trg combination. Ex:

$$(f,e)^{wrd} : \ 'translations|||traductions'$$
$$(f,e)^{lem} : \ 'translation|||traduction'$$
$$(f,e)^{pos} : \ 'NNS|||Noun'$$
$$(f,e)^{lem:pos} : \ 'translation|||Noun'$$

- Each unit (--train-src --train-trg) is assign to **one** token (--train-src-bm --train-trg-bm)

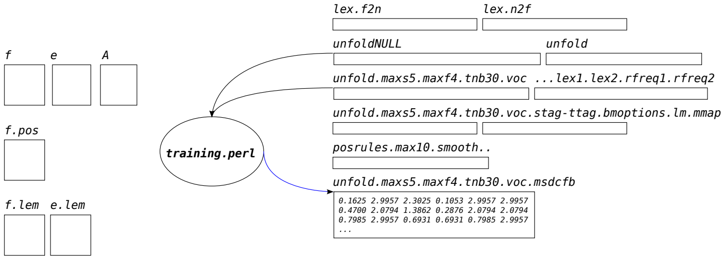
# Model estimation




Step 5: rewrite rules (POS-based) [--max-rule-length --max-rule-cost]

- Rewrite rules are automatically learned from the bitext word alignments
- POS tags are used to gain generalization power
- Rules are filtered according to: $P(f \rightsquigarrow f^r) = \frac{count(f, f^r)}{\sum_{f' \in perm(f)} count(f, f')}$

# Model estimation

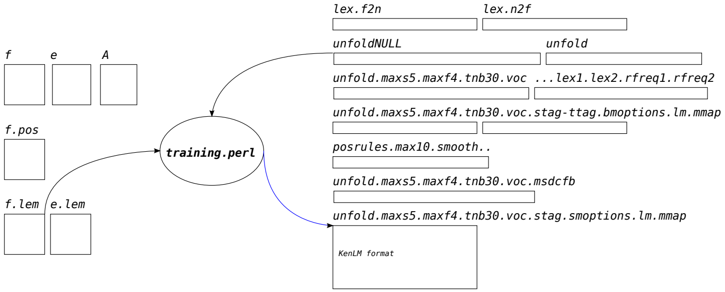



Step 6: lexicalized reordering

- **Four** orientation types: (m)onotone order; (s)wap with previous tuple; (f)orward jump; (b)ackward jump. And **two** aggregated types: (d)iscontinuous: (b) and (f); and (c)ontinuous: (m) and (s)
- Smoothed maximum likelihood estimator, $\sigma = 1/\sum_o count(o, f, e)$:

$$P(orientation|f, e) = \frac{(\sigma/4) + count(orientation, f, e)}{\sigma + \sum_o count(o, f, e)}$$

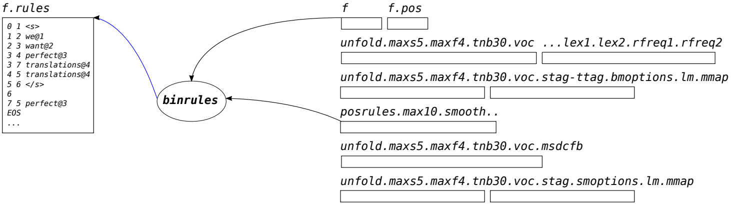
# Model estimation




Step 7: source (unfolded) n-gram lm [`--train-src-unf --options-sm --name-src-unf`]

- *n*-gram LM estimated over reordered training source words (lemmas, POS, *etc.*)
- Reordering introduced in the tuple extraction process. Ex: 'we want translations perfect'
- Options passed to SRILM, Ex: –options-sm -order_5_-unk_-kndiscount_-interpolate

# Inference



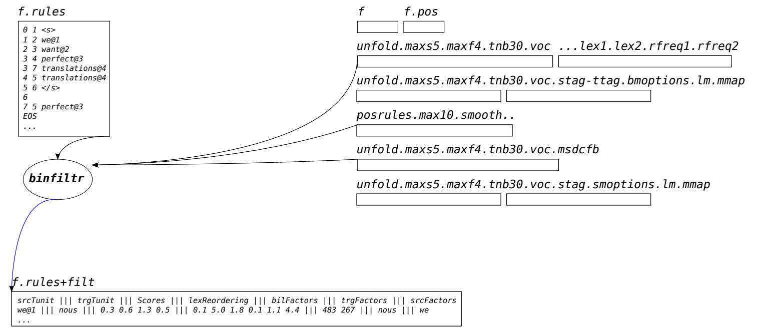

binrules `[-wrd s -tag s -rrules s -maxr i -maxc f]`

- Rules extracted from reorderings introduced in the tuple extraction

  translations perfect $\rightsquigarrow$ perfect translations

- Referred to source-side tokens (words, POS, *etc.*): NNS JJ $\rightsquigarrow$ JJ NNS
- Filter rules (discard noisy alignments) maxr=10 (size) maxc=4 (cost, -logP)

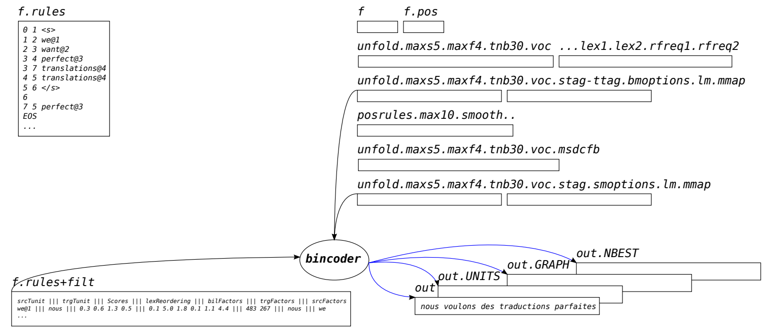
# Inference




binfiltr [`-tunits s -scores s -lexrm s -bilfactor s -srcfactor s -trgfactor s -maxs i`]

- Collect useful information for given test sentences
- Filter tuples (discard noisy alignments) maxs=6 (size)
- Bilingual/source/target factors used with bilingual/source/target *n*-gram LMs
- Multiple LM's referred to multiple factors can be used
- Sentence-based LM's also available

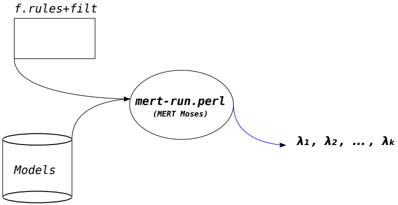
# Inference




bincoder (weights) (files) (search settings)

- Model weights
- Files: (input) language models, filtered input (output) 1-best target word/translation unit hypotheses, Search graph, *N*-best hypotheses (OPENFST)
- Search settings: beam size, translation choices, input (OOV) words strategy, threads, *etc.*

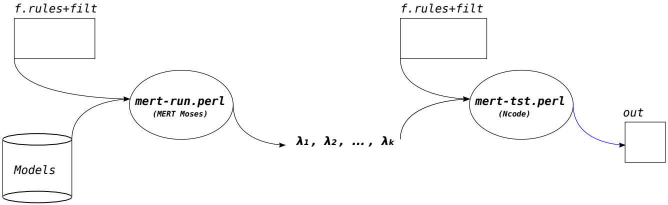
# Optimization (MERT)




`mert-run.perl`

- A wrapper for the MERT software made available in the MOSES toolkit (... soon also ZMERT)

# Optimization (MERT)




`mert-tst.perl`

- Translates a given input file using the optimized model weights

# Plan

# Experimental framework

- French-to-German (2) tasks:

  **news** : News Commentary corpus ($6^{th}$ Workshop on SMT, WMT11)
  **full** : Additional data (up to 4 million sentence pairs)

- **Tune**: newstest2010, **Test**: newstest2009, newstest2011
- Same alignment (GIZA++), target LM (SRILM)
- NCODE employs TREETAGGER POS tags (rewrite rules)
- **default** MOSES settings: 14 features
- **default** NCODE settings: 14 + 2 features:
  - Bilingual *n*-gram over tuples built from words
  - Bilingual *n*-gram over tuples built from POS tags

# Performance results

BLEU : Translation accuracy
#units : Number of phrases/tuples (millions) after training (limited to 6 tokens)
Memory : Memory (Mb) used by each decoder
Speed : Decoding speed (Words/second) (single-threaded translations)

| *System* | *Task* | *BLEU* *newstest2009* | *BLEU* *newstest2011* | *#units* | *Memory* | *Speed* |
|---|---|---|---|---|---|---|
| NCODE | *news* | 13.89 | 13.83 | 0.5 | 7.7 | 54.4 |
| | *full* | 15.09 | 15.26 | 7.5 | 9 | 33.9 |
| MOSES | *news* | 13.70 | 13.51 | 7.5 | 7.9 | 23.1 |
| | *full* | 14.66 | 14.51 | 141 | 16 | 14.7 |

# Performance results

BLEU : Translation accuracy
#units : Number of phrases/tuples (millions) after training (limited to 6 tokens)
Memory : Memory (Mb) used by each decoder
Speed : Decoding speed (Words/second) (single-threaded translations)

| *System* | *Task* | *BLEU* *newstest2009* | *BLEU* *newstest2011* | *#units* | *Memory* | *Speed* |
|---|---|---|---|---|---|---|
| NCODE | *news* | 13.89 | 13.83 | 0.5 | 7.7 | 54.4 |
| | *full* | 15.09 | 15.26 | 7.5 | 9 | 33.9 |
| MOSES | *news* | 13.70 | 13.51 | 7.5 | 7.9 | 23.1 |
| | *full* | 14.66 | 14.51 | 141 | 16 | 14.7 |

- Slightly higher accuracy results for NCODE (within the confidence margin)

# Performance results

- BLEU : Translation accuracy
- #units : Number of phrases/tuples (millions) after training (limited to 6 tokens)
- Memory : Memory (Mb) used by each decoder
- Speed : Decoding speed (Words/second) (single-threaded translations)

| *System* | *Task* | *BLEU* *newstest2009* | *BLEU* *newstest2011* | *#units* | *Memory* | *Speed* |
|---|---|---|---|---|---|---|
| NCODE | *news* | 13.89 | 13.83 | 0.5 | 7.7 | 54.4 |
| | *full* | 15.09 | 15.26 | 7.5 | 9 | 33.9 |
| MOSES | *news* | 13.70 | 13.51 | 7.5 | 7.9 | 23.1 |
| | *full* | 14.66 | 14.51 | 141 | 16 | 14.7 |

- Slightly higher accuracy results for NCODE (within the confidence margin)
- NCODE outperforms MOSES in data efficiency:
  - smaller set of tuples than phrases (full: 20 times smaller)
  - lower memory needs for NCODE (full: $\sim$ half than MOSES)

# Performance results

- **BLEU** : Translation accuracy
- **#units** : Number of phrases/tuples (millions) after training (limited to 6 tokens)
- **Memory** : Memory (Mb) used by each decoder
- **Speed** : Decoding speed (Words/second) (single-threaded translations)

| *System* | *Task* | *BLEU* *newstest2009* | *BLEU* *newstest2011* | *#units* | *Memory* | *Speed* |
|---|---|---|---|---|---|---|
| NCODE | *news* | 13.89 | 13.83 | 0.5 | 7.7 | 54.4 |
| | *full* | 15.09 | 15.26 | 7.5 | 9 | 33.9 |
| MOSES | *news* | 13.70 | 13.51 | 7.5 | 7.9 | 23.1 |
| | *full* | 14.66 | 14.51 | 141 | 16 | 14.7 |

- Slightly higher accuracy results for NCODE (within the confidence margin)
- NCODE outperforms MOSES in data efficiency:
  - smaller set of tuples than phrases (full: 20 times smaller)
  - lower memory needs for NCODE (full: ∼ half than MOSES)
- Nearly twice faster (search pruning settings are **not** tested)

# Plan

Bilingual *n*-gram approach to SMT

Decoding

The NCODE toolkit

Comparison: NCODE vs. MOSES

Concluding remarks

# Concluding remarks

- Developed to run on LINUX systems
- Written in PERL and C++
- Prerequisites
  - to compile: KENLM and OPENFST libraries
  - to run: SRILM and the MERT implementation in MOSES

# Concluding remarks

- Developed to run on LINUX systems
- Written in PERL and C++
- Prerequisites
  - to compile: KENLM and OPENFST libraries
  - to run: SRILM and the MERT implementation in MOSES
- Multithreaded
- (Multiple) src/trg/bil *n*-gram LM's handled by KENLM
- Factored src/trg/bil *n*-gram LM's

# Concluding remarks

- Developed to run on LINUX systems
- Written in PERL and C++
- Prerequisites
  - to compile: KENLM and OPENFST libraries
  - to run: SRILM and the MERT implementation in MOSES
- Multithreaded
- (Multiple) src/trg/bil *n*-gram LM's handled by KENLM
- Factored src/trg/bil *n*-gram LM's
- Under development:
  - Client/server architecture
  - Optimization by ZMERT
  - Sentence-based bonus models

# Thanks

NCODE is freely available at `http://ncode.limsi.fr/`
(`http://www.limsi.fr/Individu/jmcrego/bincoder/`)

Adrià de Gispert, Patrik Lambert, Marta Ruiz, Alexandre Allauzen, Aurélien Max, Thomas Lavergne and Artem Sokolov also contributed to create the toolkit.

`crego@systran.fr`

now at SYSTRAN Language Translation Technologies