Sparse Feature Learning

Philipp Koehn

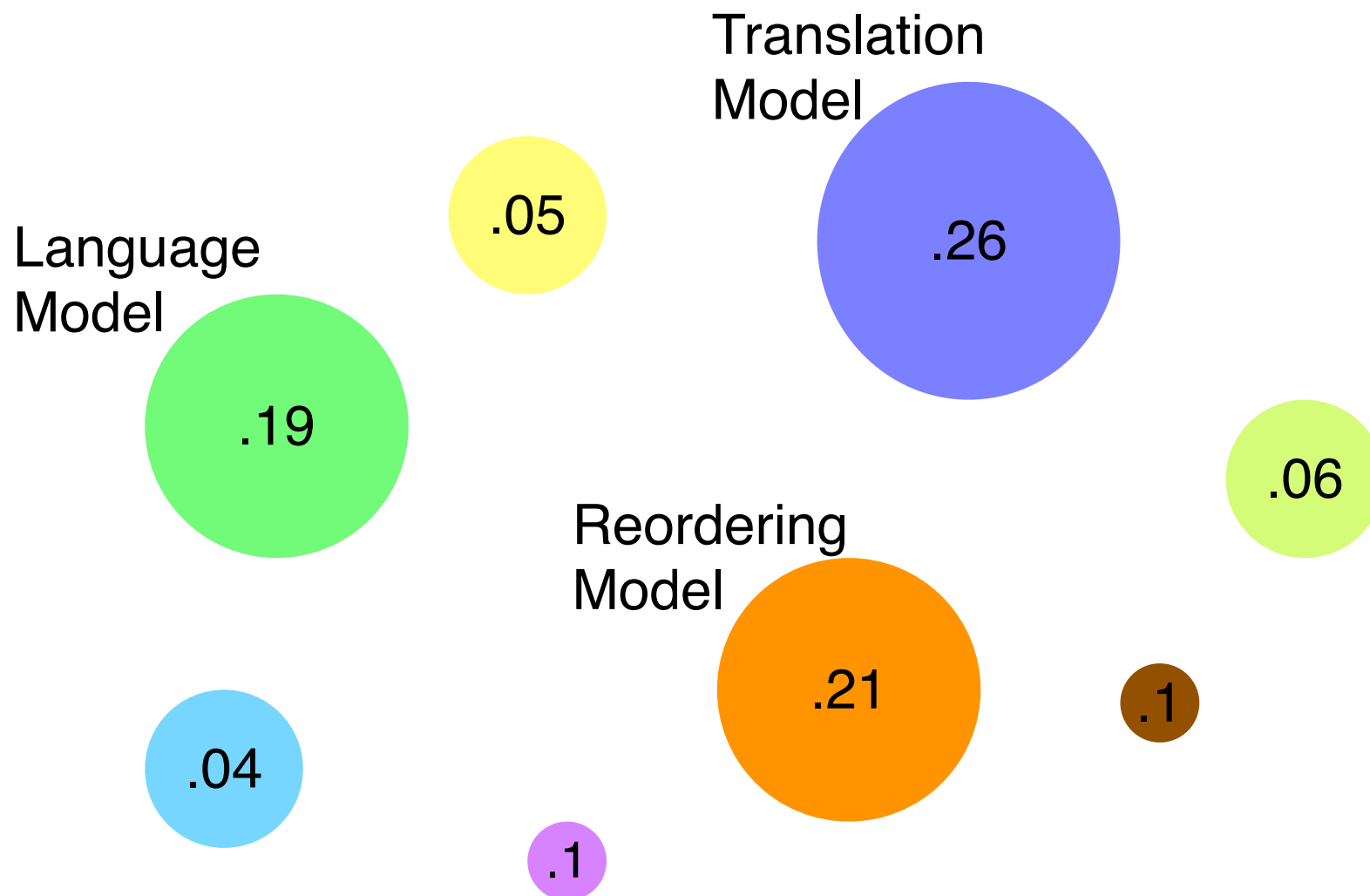3 March 2015

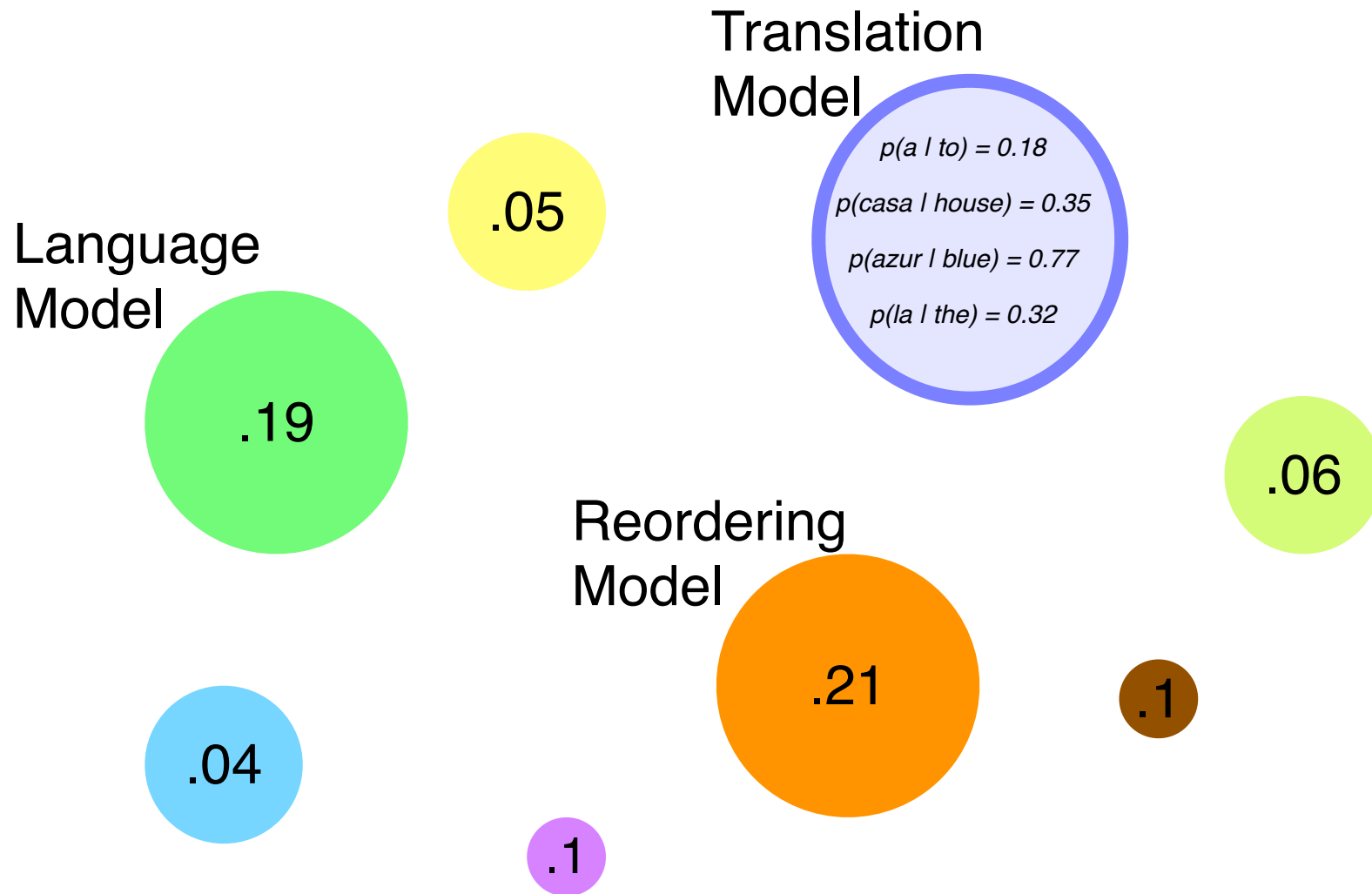# Multiple Component Models

Translation
Model

Language
Model

Reordering
Model

Component Weights

.05

Translation
Model

.26

Language
Model

.19

.06

Reordering
Model

.21

.1

.04

.1
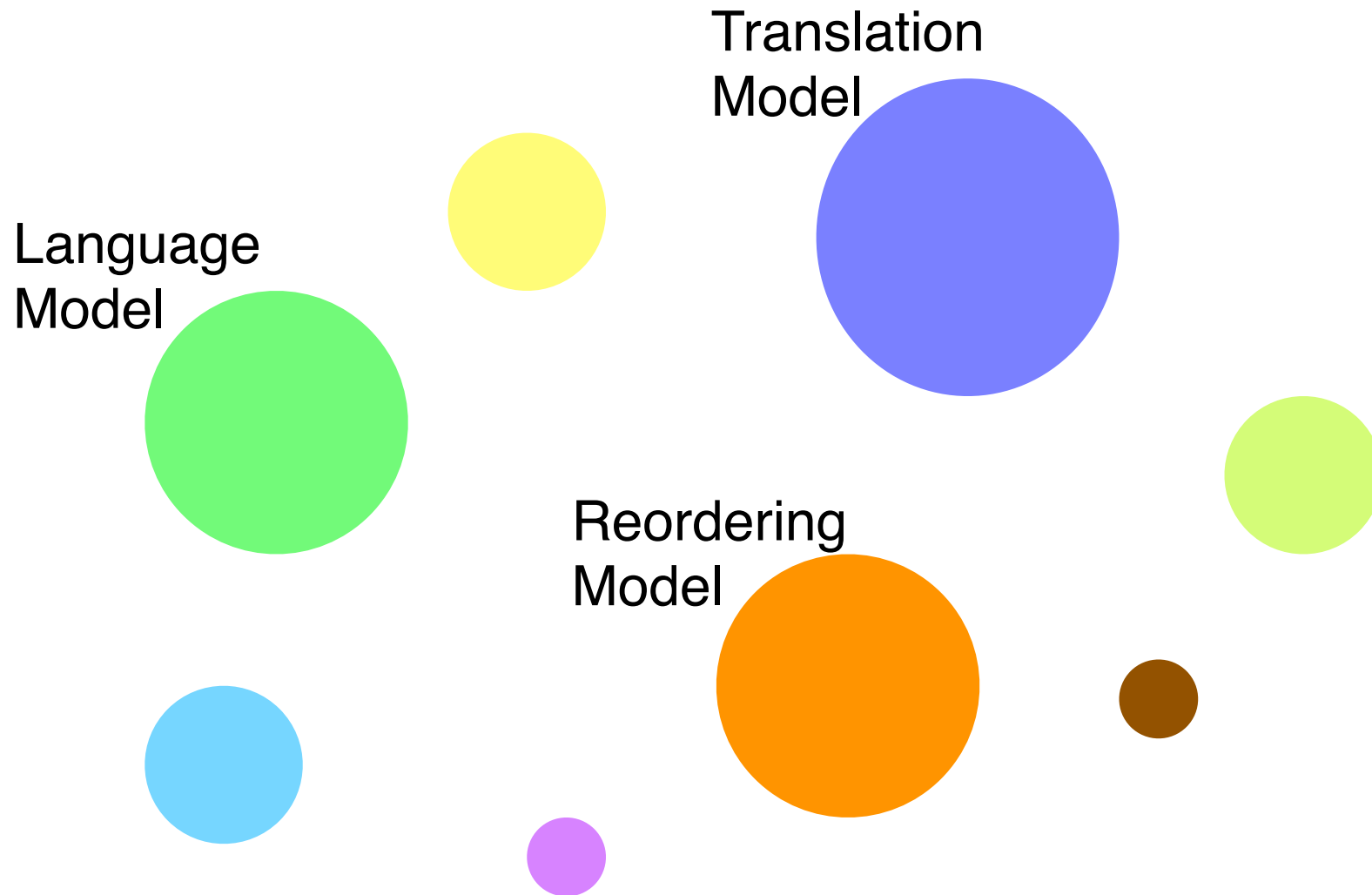
Even More Numbers Inside

Translation
Model

$p(a \mid to) = 0.18$

$p(casa \mid house) = 0.35$

$p(azur \mid blue) = 0.77$

$p(la \mid the) = 0.32$

.05

Language
Model

.19

.06

Reordering
Model

.21

.1

.04

.1
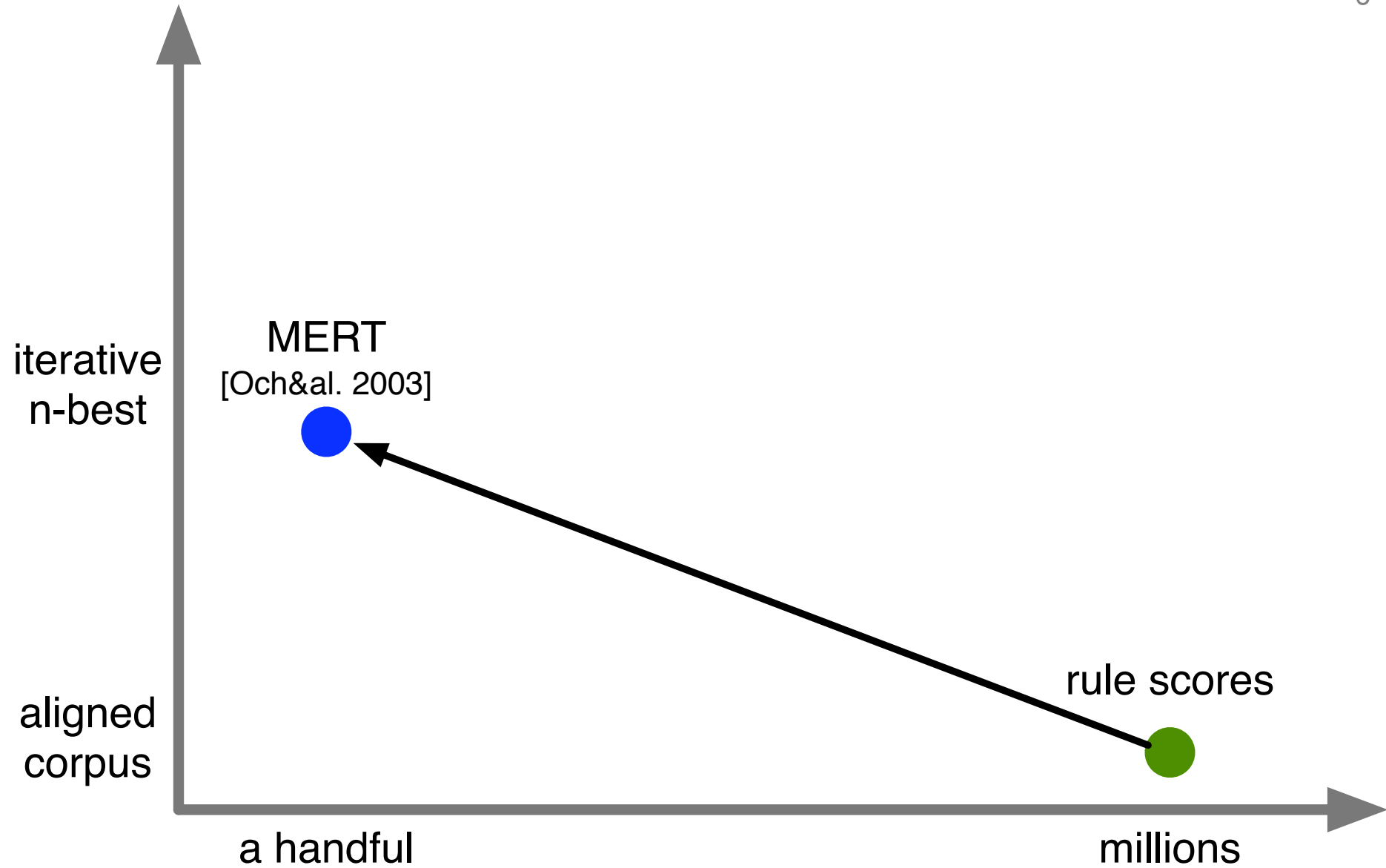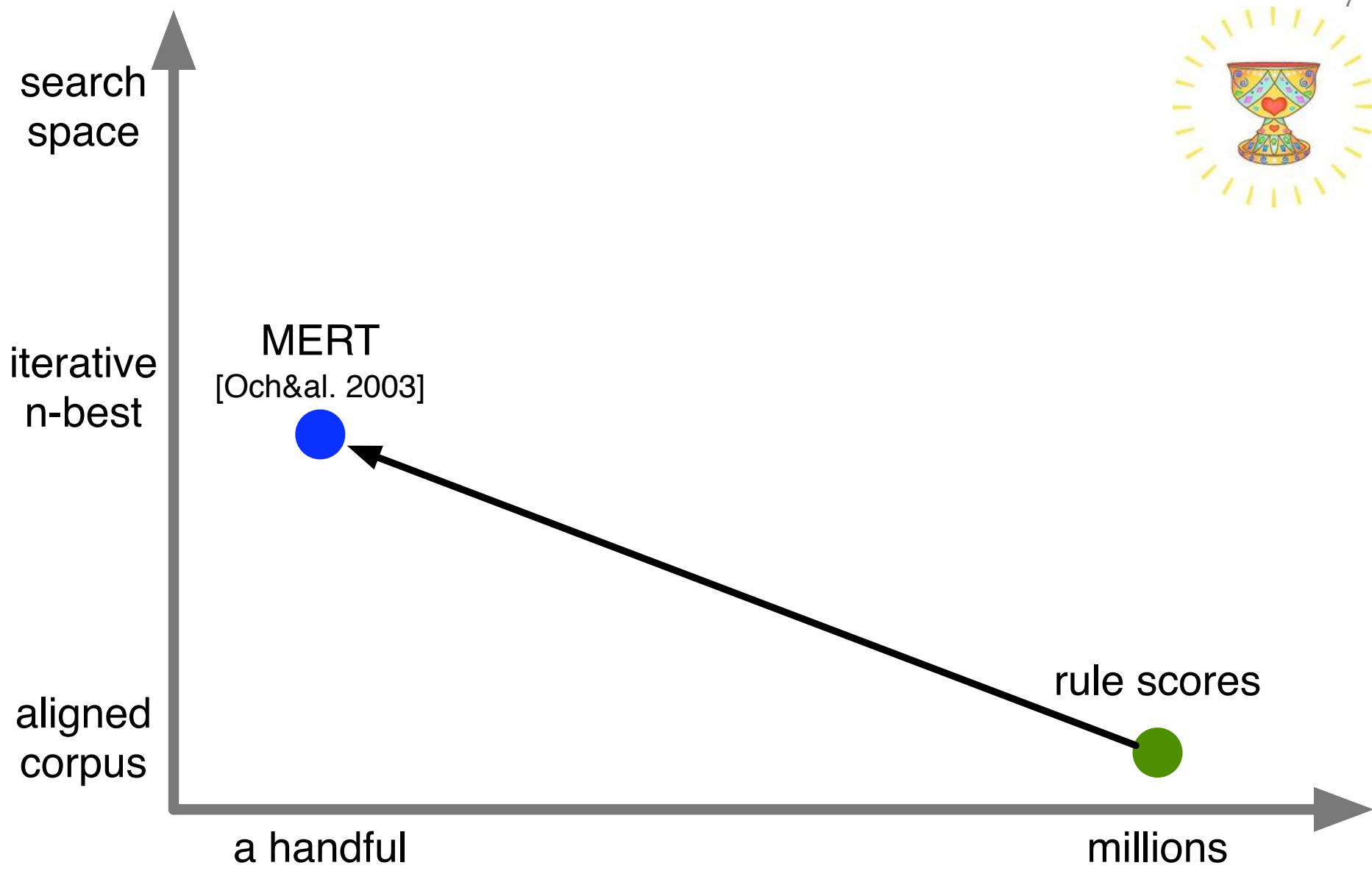
- There are millions of parameters

    – each phrase translation score
    – each language model n-gram
    – etc.

- Can we train them all discriminatively?

- This implies optimization over the entire training corpus

aligned
corpus

rule scores

millions

iterative
n-best

MERT
[Och&al. 2003]

aligned
corpus

rule scores

a handful

millions

search
space

iterative
n-best

MERT
[Och&al. 2003]

aligned
corpus

rule scores

a handful

millions

search
space

MIRA [Chiang 2007]
SampleRank [Haddow&al. 2011]

MaxViolation
[Yu et al., 2014]

iterative
n-best

MERT
[Och&al. 2003]

PRO
[Hopkins/May 2011]

Leave One Out
[Wuebker et al., 2012]

n-best

rule scores

aligned
corpus

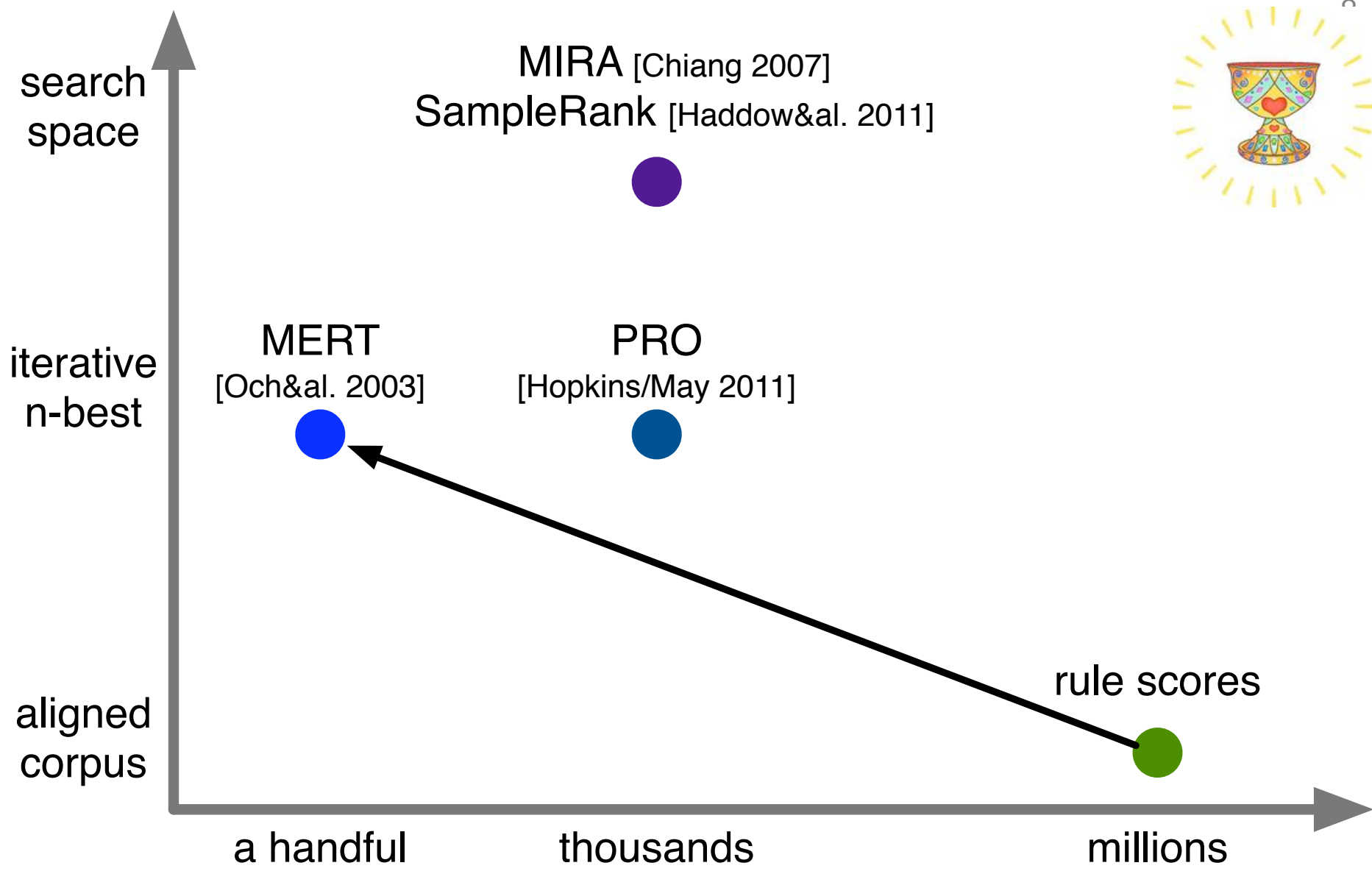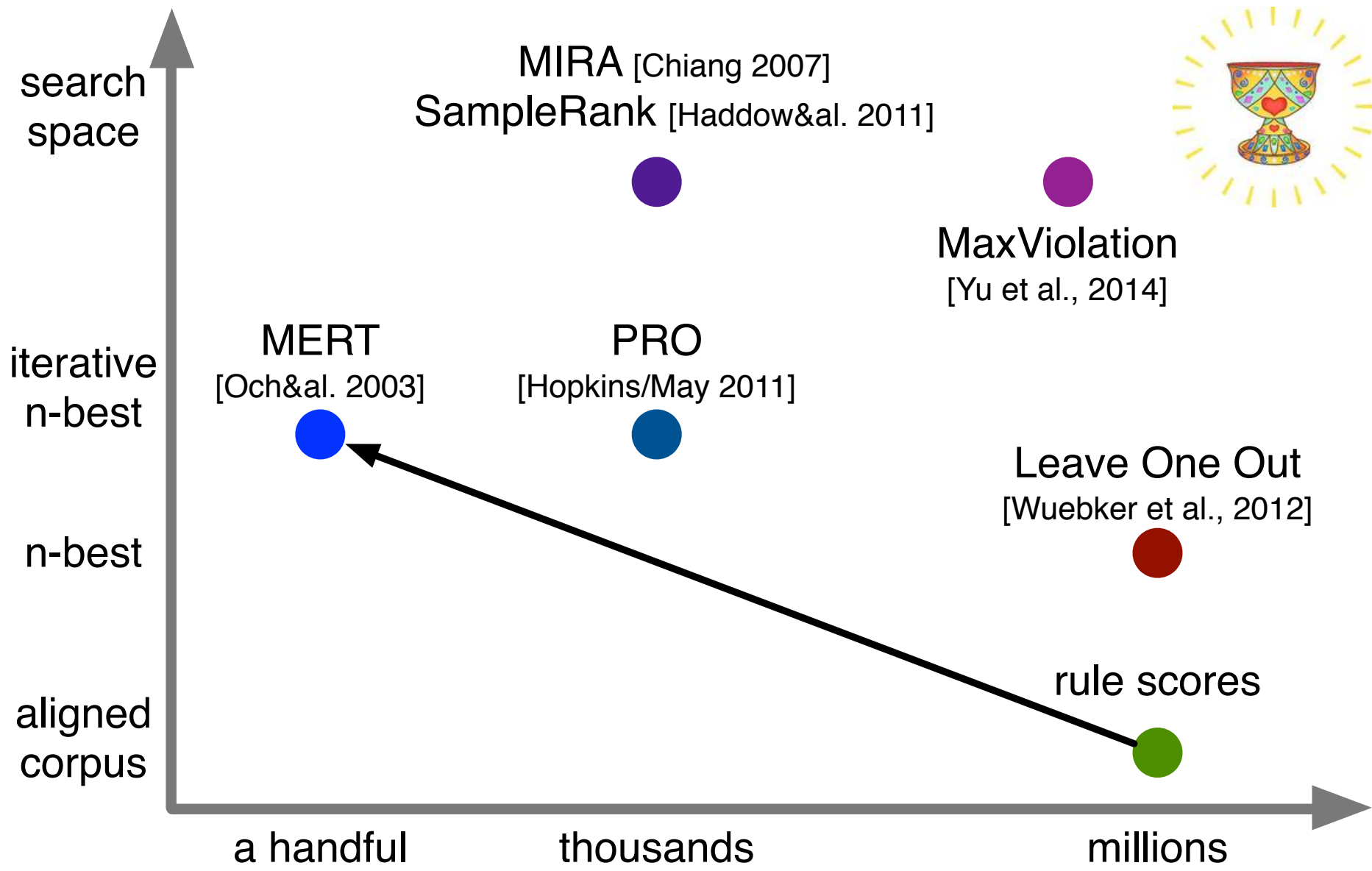a handful   thousands   millions

Strategy and Core Problems

- Process each sentence pair in the training corpus

- Optimize parameters towards producing the reference translation

- Reference translation may not be producible by model

  – optimize towards most similar translation
  – or, only process sentence pair partially

- Avoid overfitting

- Large corpora require efficient learning methods

- Optimizing BLEU requires optimizing over the entire training corpus

$$\text{BLEU}(\{\mathbf{e}_i^{\text{best}} = \text{argmax}_{\mathbf{e}_i} \sum_j h_j(\mathbf{e}_i, \mathbf{f}_i) \, \lambda_i\}, \{\mathbf{e}_i^{\text{ref}}\})$$

- Life would be easier, if we could sum over sentence level scores

$$\sum_i \text{BLEU}'(\, \text{argmax}_{\mathbf{e}_i} \sum_j h_j(\mathbf{e}_i, \mathbf{f}_i) \, \lambda_i, \, \mathbf{e}_i^{\text{ref}} \,)$$

- For instance, BLEU+1

features

Core Rule Properties

- Frequency of phrase (binned)

- Length of phrase

  - number of source words
  - number of target words
  - number of source and target words

- Unaligned / added (content) words in phrase pair

- Reordering within phrase pair

- $\mathrm{lex}(e)$ fires when an output word e is generated

- $\mathrm{lex}(f, e)$ fires when an output word e is generated aligned to a input word f

- $\mathrm{lex}(\texttt{NULL}, e)$ fires when an output word e is generated unaligned

- $\mathrm{lex}(f, \texttt{NULL})$ fires when an input word e is dropped

- Could also be defined on part of speech tags or word classes

- Replacement of lexicalized reordering model

- Features differ by

  - lexicalized by first or last word of phrase (source or target)
  - word representation replaced by word class
  - orientation type

- Indicator feature that the rule occurs in one specific domain

- Probability that the rule belongs to one specific domain

- Domain-specific lexical translation probabilities

- If we have syntactic parse trees, many more features

  - number of nodes of a particular kind
  - matching of source and target constituents
  - reordering within syntactic constituents

- Parse trees are a by-product of syntax-based models

- More on that in future lectures

Every Number in Model

- Phrase pair indicator feature

- Target n-gram feature
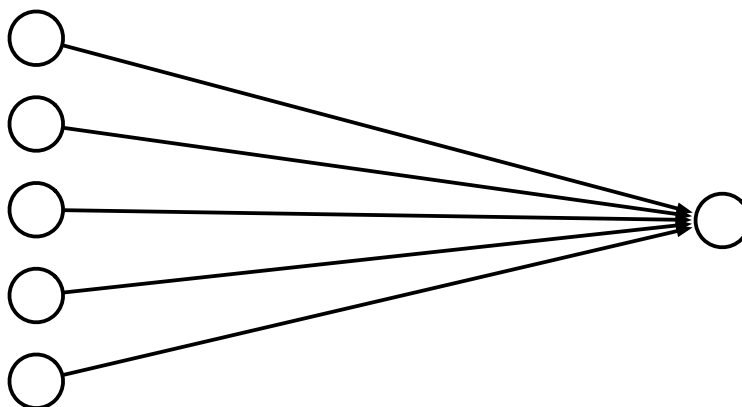
- Phrase pair orientation feature

perceptron algorithm

- We consider each sentence pair $(\mathbf{e}_i, \mathbf{f}_i)$ and its alignment $\mathbf{a}_i$

- To simplify notation, we define derivation $\mathbf{d}_i = (\mathbf{e}_i, \mathbf{f}_i, \mathbf{a}_i)$

- Model score is weighted linear combination of feature values h_j and weights λ_j

$$\text{score}(\lambda, \mathbf{d}_i) = \sum_j \lambda_j \ h_j(\mathbf{d}_i)$$

- Such models are also known as single layer perceptrons

Reference and Model Best

- Besides the reference derivation $\mathbf{d}_i^{\text{ref}}$ for sentence pair i and its score

$$\text{score}(\lambda, \mathbf{d}_i^{\text{ref}}) = \sum_j \lambda_j \; h_j(\mathbf{d}_i^{\text{ref}})$$

- We also have the model best translation

$$\mathbf{d}_i^{\text{best}} = \text{argmax}_{\mathbf{d}} \; \text{score}(\lambda_i, \mathbf{d}_i) = \text{argmax}_{\mathbf{d}} \; \sum_j \lambda_j \; h_j(\mathbf{d}_i)$$
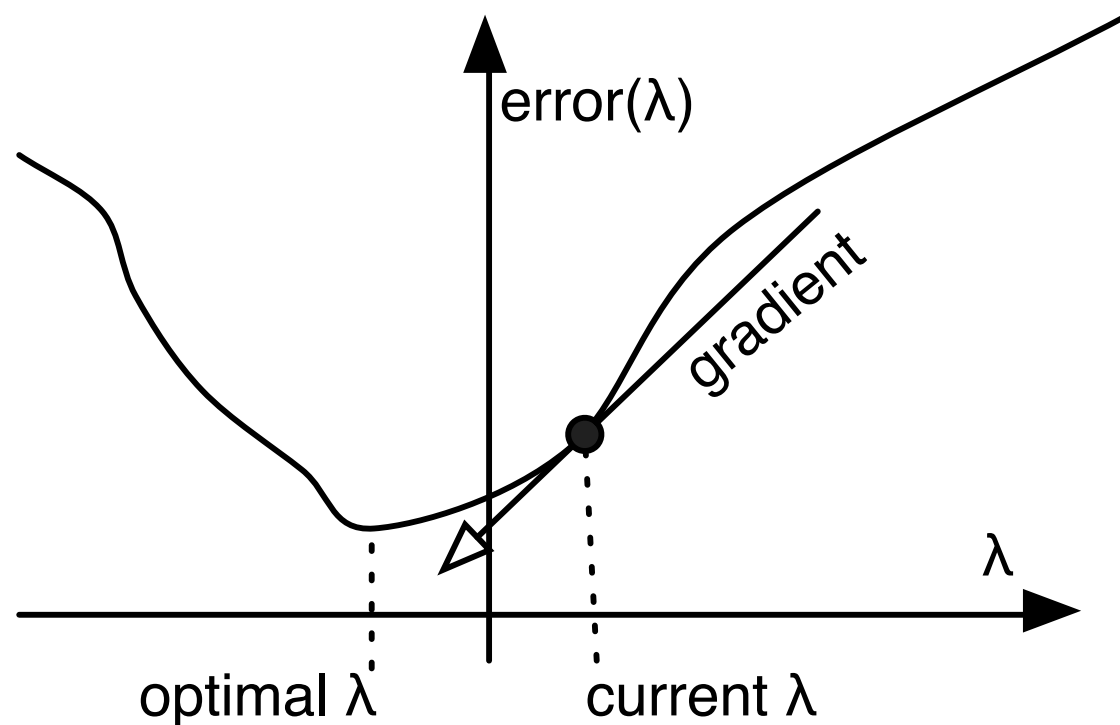
- ... and its model score

$$\text{score}(\lambda, \mathbf{d}_i^{\text{best}}) = \sum_j \lambda_j \; h_j(\mathbf{d}_i^{\text{best}})$$

- We can view the error in our model as a function of its parameters λ

$$\text{error}(\lambda, \mathbf{d}_i^{\text{best}}, \mathbf{d}_i^{\text{ref}}) = \text{score}(\lambda, \mathbf{d}_i^{\text{best}}) - \text{score}(\lambda, \mathbf{d}_i^{\text{ref}})$$

- We cannot analytically find the optimum of the curve error(λ)
- We can compute the gradient $\frac{d}{d\lambda}$error(λ) at any point
- We want to follow the gradient towards the optimal λ value

- We want to minimize the error

$$\mathrm{error}(\lambda, \mathbf{d}_i^{\mathrm{best}}, \mathbf{d}_i^{\mathrm{ref}}) = \mathrm{score}(\lambda, \mathbf{d}_i^{\mathrm{best}}) - \mathrm{score}(\lambda, \mathbf{d}_i^{\mathrm{ref}})$$

- In stochastic gradient descent, we follow direction of gradient

$$\frac{d}{d\,\lambda}\,\mathrm{error}(\lambda, \mathbf{d}_i^{\mathrm{best}}, \mathbf{d}_i^{\mathrm{ref}})$$

- For each λ_j, we compute the gradient pointwise

$$\frac{d}{d\,\lambda_j}\,\mathrm{error}(\lambda_j, \mathbf{d}_i^{\mathrm{best}}, \mathbf{d}_i^{\mathrm{ref}}) = \frac{d}{d\,\lambda_j}\,\mathrm{score}(\lambda, \mathbf{d}_i^{\mathrm{best}}) - \mathrm{score}(\lambda, \mathbf{d}_i^{\mathrm{ref}})$$

- Gradient with respect to λ_j

$$\frac{d}{d\,\lambda_j}\,\text{error}(\lambda_j, \mathbf{d}_i^{\text{best}}, \mathbf{d}_i^{\text{ref}}) = \frac{d}{d\,\lambda_j}\,\sum_{j'} \lambda_{j'}\,h_{j'}(\mathbf{d}_i^{\text{best}}) - \sum_{j'} \lambda_{j'}\,h_{j'}(\mathbf{d}_i^{\text{ref}})$$

- For $\lambda_j' \neq \lambda_j$, the terms $\lambda_{j'}\,h_{j'}(\mathbf{d}_i)$ are constant, so they disappear

$$\frac{d}{d\,\lambda_j}\,\text{error}(\lambda_j, \mathbf{d}_i^{\text{best}}, \mathbf{d}_i^{\text{ref}}) = \frac{d}{d\,\lambda_j}\,\lambda_j\,h_j(\mathbf{d}_i^{\text{best}}) - \lambda_j\,h_j(\mathbf{d}_i^{\text{ref}})$$

- The derivative of a linear function is its factor

$$\frac{d}{d\,\lambda_j}\,\text{error}(\lambda_j, \mathbf{d}_i^{\text{best}}, \mathbf{d}_i^{\text{ref}}) = h_j(\mathbf{d}_i^{\text{best}}) - h_j(\mathbf{d}_i^{\text{ref}})$$

$\Rightarrow$ Our model update is $\lambda_j^{\text{new}} = \lambda_j - (h_j(\mathbf{d}_i^{\text{best}}) - h_j(\mathbf{d}_i^{\text{ref}}))$

- Feature values in model best translation

- Feature values in reference translation

- Intuition:

  - promote features whose value is bigger in reference
  - demote features whose value is bigger in model best

Algorithm

26

Input: set of sentence pairs (**e**,**f**), set of features
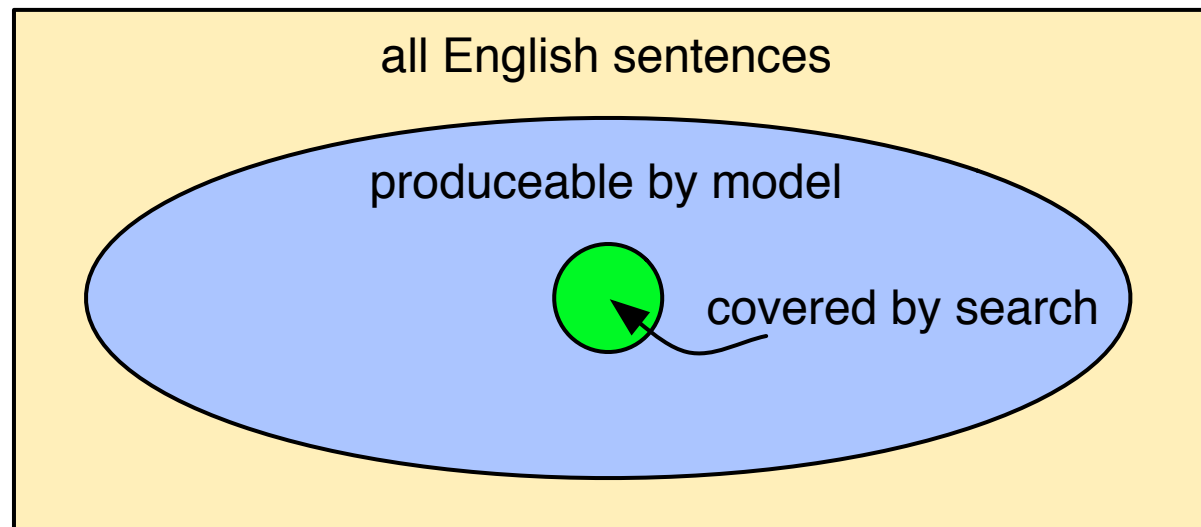**Output:** set of weights λ for each feature

1: $\lambda_i = 0$ **for all** i

2: **while** not converged **do**

3:     **for all** foreign sentences **f do**

4:        $\mathbf{d}_{\text{best}}$ = best derivation according to model

5:        $\mathbf{d}_{\text{ref}}$ = reference derivation

6:        **if** $\mathbf{d}_{\text{best}} \neq \mathbf{d}_{\text{ref}}$ **then**

7:           **for all** features h_i **do**

8:              $\lambda_i \mathrel{+}= h_i(\mathbf{d}_{\text{ref}}) - h_i(\mathbf{d}_{\text{best}})$

9:           **end for**

10:        **end if**

11:     **end for**

12: **end while**

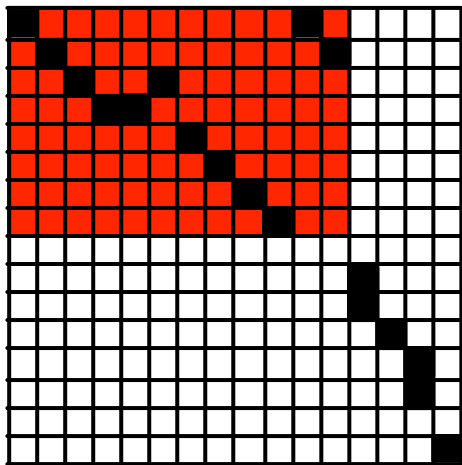# generating the reference

- Reference translation may be anywhere in this box

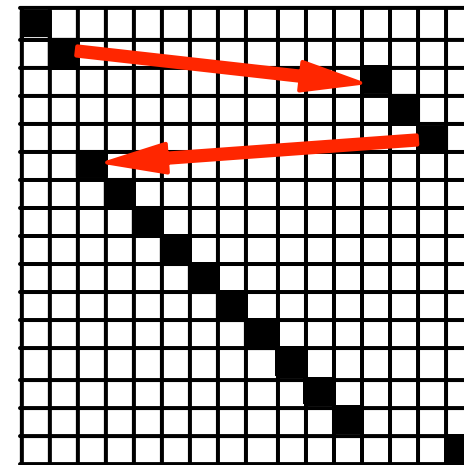- If produceable by model $\rightarrow$ we can compute feature scores

- If not $\rightarrow$ we can not

Causes

- Reference translation in tuning set not literal

- Failure even if phrase pairs are extracted from same sentence pair

- Examples

alignment points too distant
→ phrase pair too big to extract

required reordering distance too large
→ exceeds distortion limit of decoder

- BLEU+1
  - add one free n-gram count to statistics $\rightarrow$ avoids BLEU score of 0
  - however: wrong balance between 1-4 grams, too drastic brevity penalty

- BLEU impact
  - leave all other sentence translations fixed
  - collect n-gram matches and totals from them
  - add n-gram matches and total from current candidate
  $\rightarrow$ consider impact on overall BLEU score

- Incremental BLEU impact
  - maintain decaying statistics for n-gram matches, total n-grams

$$\text{count}_t = \frac{9}{10}\,\text{count}_{t-1} + \text{current-count}_t$$

- Consider the following Arabic sentence (written left-to-right in Buckwalter romanization) with English glosses:

sd	qTEp	mn	AlkEk	AlmmlH	″	brytzl	″	Hlqh	.
blocked	piece	of	biscuit	salted	″	pretzel	″	his-throat	

- Very literal translation might be

  A piece of a salted biscuit, a ″pretzel,″ blocked his throat.

- But reference translation is

  A pretzel, a salted biscuit, became lodged in his throat.

- Reference accurate, but major transformations

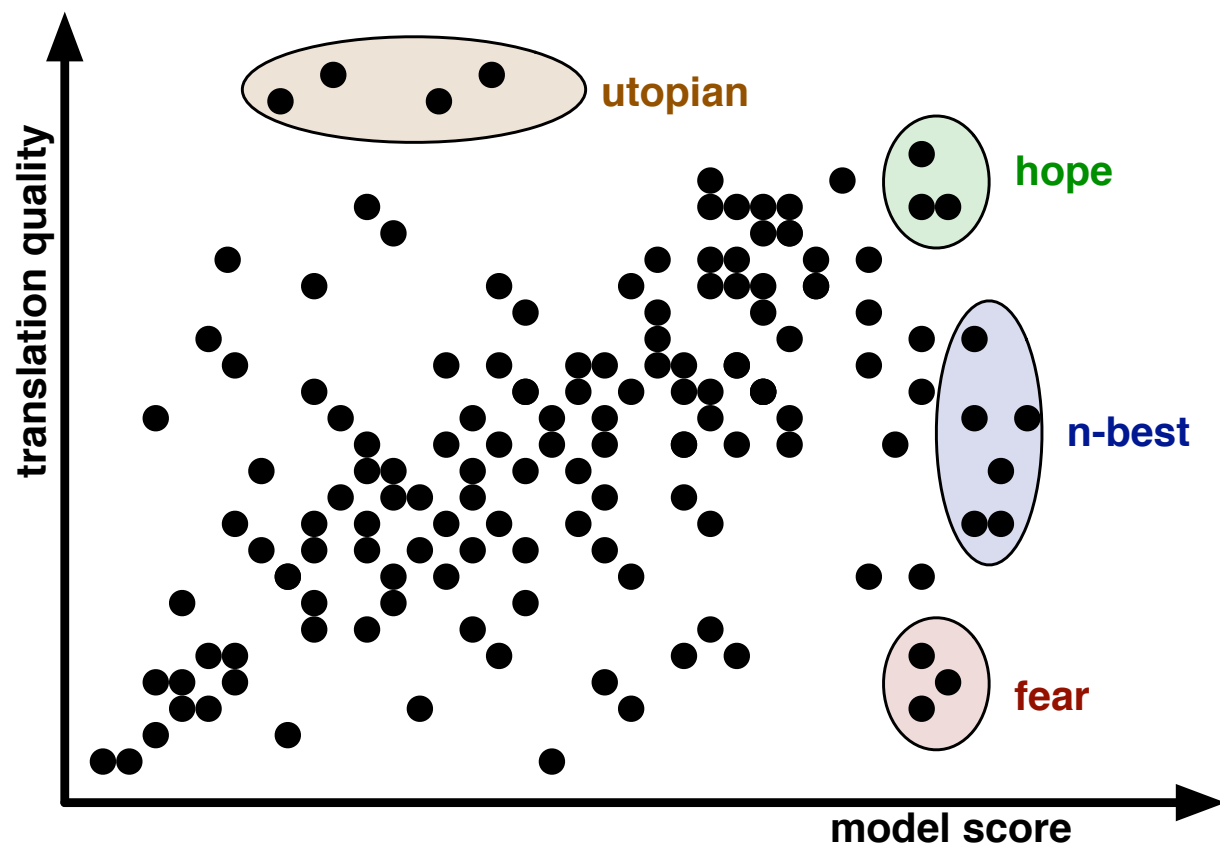- Trying to approximate reference translation may lead to bad rules

note: example from Chiang (2012)

mira

- Bad: optimize towards utopian, away from n-best

- Good: optimize towards hope, away from fear

- Hope translation

$$\mathbf{d}^{\mathrm{hope}} = \mathrm{argmax}_{\mathbf{d}} \ \mathsf{BLEU}(\mathbf{d}) + \mathrm{score}(\mathbf{d})$$

- Finding the fear translation

  – Metric difference (should be big)

  $$\Delta\mathsf{BLEU}(\mathbf{d}^{\mathrm{hope}}, \mathbf{d}) = \mathsf{BLEU}(\mathbf{d}^{\mathrm{hope}}) - \mathsf{BLEU}(\mathbf{d})$$

  – Score difference (should be small or negative)

  $$\Delta\mathrm{score}(\lambda, \mathbf{d}^{\mathrm{hope}}, \mathbf{d}) = \mathrm{score}(\lambda, \mathbf{d}^{\mathrm{hope}}) - \mathrm{score}(\lambda, \mathbf{d})$$

  – Margin

  $$v(\lambda, \mathbf{d}^{\mathrm{hope}}, \mathbf{d}) = \Delta\mathsf{BLEU}(\mathbf{d}^{\mathrm{hope}}, \mathbf{d}) - \Delta\mathrm{score}(\lambda, \mathbf{d}^{\mathrm{hope}}, \mathbf{d})$$

  – Fear translation
  $$\mathbf{d}^{\mathrm{fear}} = \mathrm{argmax}_{\mathbf{d}} \ v(\lambda, \mathbf{d}^{\mathrm{hope}}, \mathbf{d})$$

- Stochastic gradient descent update with learning weight δ_i

$$\lambda_j^{\text{new}} = \lambda_j - \delta_i \left(h_j(\mathbf{d}_i^{\text{fear}}) - h_j(\mathbf{d}_i^{\text{hope}}) \right)$$

- Updates should depend on margin

$$\delta_i = \min \left(C, \frac{\Delta\mathsf{BLEU}(\mathbf{d}_i^{\text{hope}}, \mathbf{d}_i^{\text{fear}}) - \Delta\mathrm{score}(\mathbf{d}_i^{\text{hope}}, \mathbf{d}_i^{\text{fear}})}{||\Delta h||^2} \right)$$

- The math behind this is a bit complicated

- For some features, we have a lot of evidence (coarse features)

- Others occur only rarely (sparse features)

- After a while, we do not want to change coarse features too much

$\Rightarrow$ Adaptive Regularization of Weights (AROW)

  – record confidence in weights over time
  – include this in the learning rate for each feature

Parallelization

- Training is computationally expensive

$\Rightarrow$ Break up training data into batches

- After processing all the batches, average the weights

- Not only a speed-up, also seems to improve quality

- Allows parallel processing, but requires inter-process communication

Sample Rank

- Generating hope and fear translations is expensive

- Sample good/bad by random walk through alignment space

    - use operations as in Gibbs samples
    - vary one translation option choice
    - vary one reordering decision
    - vary one phrase segmentation decision
    - adopt new translation based on relative score

- Compare current translation against its neighbors

$\rightarrow$ apply MIRA update if more costly translation has higher BLEU

- MIRA requires translation of each sentence on demand

  – repeated decoding needed
  – computationally very expensive

- Batch MIRA

  – n-best list or search graph (lattice)
  – straightforward parallelization
  – does not seem to harm performance

pro

Scored N-Best List

- Reference translation: he does not go home

- N-best list

Translation	Feature values						BLEU+1
it is not under house	-32.22	-9.93	-19.00	-5.08	-8.22	-5	27.3%
he is not under house	-34.50	-7.40	-16.33	-5.01	-8.15	-5	30.2%
it is not a home	-28.49	-12.74	-19.29	-3.74	-8.42	-5	30.2%
it is not to go home	-32.53	-10.34	-20.87	-4.38	-13.11	-6	31.2%
it is not for house	-31.75	-17.25	-20.43	-4.90	-6.90	-5	27.3%
he is not to go home	-35.79	-10.95	-18.20	-4.85	-13.04	-6	31.2%
he does not home	-32.64	-11.84	-16.98	-3.67	-8.76	-4	36.2%
it is not packing	-32.26	-10.63	-17.65	-5.08	-9.89	-4	21.8%
he is not packing	-34.55	-8.10	-14.98	-5.01	-9.82	-4	24.2%
he is not for home	-36.70	-13.52	-17.09	-6.22	-7.82	-5	32.5%

- Higher quality translation (BLEU+1) should rank higher

Pick 2 Translations at Random

- Reference translation: he does not go home

- N-best list

Translation	Feature values						BLEU+1
it is not under house	-32.22	-9.93	-19.00	-5.08	-8.22	-5	27.3%
he is not under house	-34.50	-7.40	-16.33	-5.01	-8.15	-5	30.2%
it is not a home	-28.49	-12.74	-19.29	-3.74	-8.42	-5	30.2%
it is not to go home	-32.53	-10.34	-20.87	-4.38	-13.11	-6	31.2%
it is not for house	-31.75	-17.25	-20.43	-4.90	-6.90	-5	**27.3%**
he is not to go home	-35.79	-10.95	-18.20	-4.85	-13.04	-6	31.2%
he does not home	-32.64	-11.84	-16.98	-3.67	-8.76	-4	36.2%
it is not packing	-32.26	-10.63	-17.65	-5.08	-9.89	-4	21.8%
he is not packing	-34.55	-8.10	-14.98	-5.01	-9.82	-4	24.2%
he is not for home	-36.70	-13.52	-17.09	-6.22	-7.82	-5	**32.5%**

- Higher quality translation (BLEU+1) should rank higher

One is Better than the Other

- Reference translation: he does not go home

- N-best list

Translation	Feature values						BLEU+1
it is not under house	-32.22	-9.93	-19.00	-5.08	-8.22	-5	27.3%
he is not under house	-34.50	-7.40	-16.33	-5.01	-8.15	-5	30.2%
it is not a home	-28.49	-12.74	-19.29	-3.74	-8.42	-5	30.2%
it is not to go home	-32.53	-10.34	-20.87	-4.38	-13.11	-6	31.2%
it is not for house	**-31.75**	**-17.25**	**-20.43**	**-4.90**	**-6.90**	**-5**	**27.3%**
he is not to go home	-35.79	-10.95	-18.20	-4.85	-13.04	-6	31.2%
he does not home	-32.64	-11.84	-16.98	-3.67	-8.76	-4	36.2%
it is not packing	-32.26	-10.63	-17.65	-5.08	-9.89	-4	21.8%
he is not packing	-34.55	-8.10	-14.98	-5.01	-9.82	-4	24.2%
he is not for home	**-36.70**	**-13.52**	**-17.09**	**-6.22**	**-7.82**	**-5**	**32.5%**

- Higher quality translation (BLEU+1) should rank higher

- Pairwise sample

  - $\overrightarrow{\textbf{bad}} = (-31.75, -17.25, -20.43, -4.90, -6.90, -5)$
  - $\overrightarrow{\textbf{good}} = (-36.70, -13.52, -17.09, -6.22, -7.82, -5)$

- Learn a classifier

  - $\overrightarrow{\textbf{bad}} - \overrightarrow{\textbf{good}} \rightarrow$ ☹
  - $\overrightarrow{\textbf{good}} - \overrightarrow{\textbf{bad}} \rightarrow$ ☺

- Use off the shelf maximum entropy classifier to learn weights for each feature
  e.g., MegaM (http://www.umiacs.umd.edu/~hal/megam/)

Sampling

- Collect samples for each sentence pair in tuning set

- For each sentence, sample 1000-best list for 50 pairwise samples

- Reject samples if difference in BLEU+1 score is too small (≤ 0.05)

- Iterate process

  1. set default weights
  2. generate n-best list
  3. build classifier
  4. adopt classifier weights
  5. go to 2, unless converged

leave one out

Leave One Out Training

- Train initial baseline model

- Force translate the training data:
  require decoder to match the reference translation

- Collect statistics over translation rules used

- Leave one out:
  do not use translation rules originally collected from current sentence pair

- Related to jackknife

  – 90% of training data used for rule collection
  – 10% to validate rules
  – rotate

- Relaxed leave-one-out

  - allow rules originally collected from current sentence pair
  - very costly $\rightarrow$ only used, if everything else fails

- Allow single word translations (avoid OOV)

- Larger distortion limit

- Word deletion and insertion (very costly)

- Generate 100-best list

- Collect fractional counts from derivations

$\Rightarrow$ Much smaller model

$\Rightarrow$ Sometimes better model

max-violation perceptron

and

forced decoding

Machine Translation: Sparse Feature Learning

Perceptron over Full Training Corpus

- Early work on stochastic gradient descent over full training corpus unsuccessful

- One reason: Search errors break theoretical properties of convergence

- Are unreachable reference translations a problem?

  - yes: ignoring them leaves out large amounts of training data
  - no: data selection, non-literal translations are lower quality

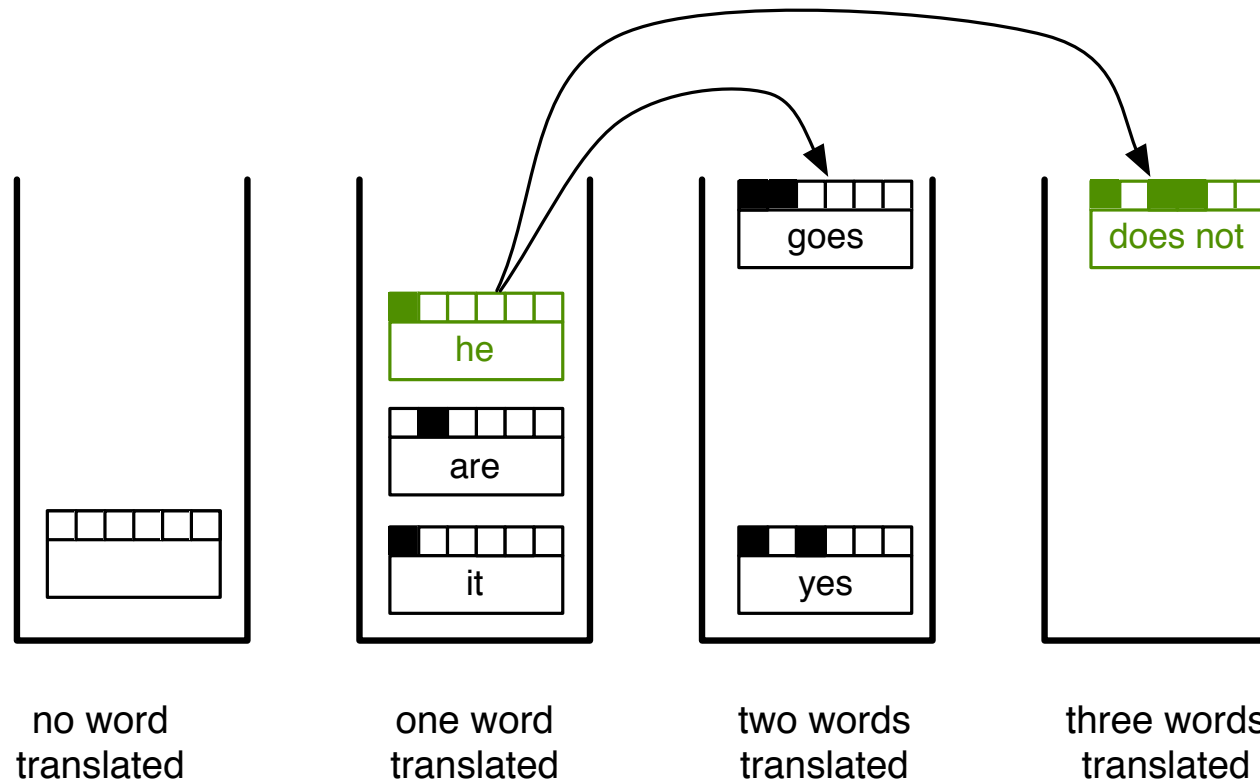- Idea: update when partial reference derivation falls out of beam
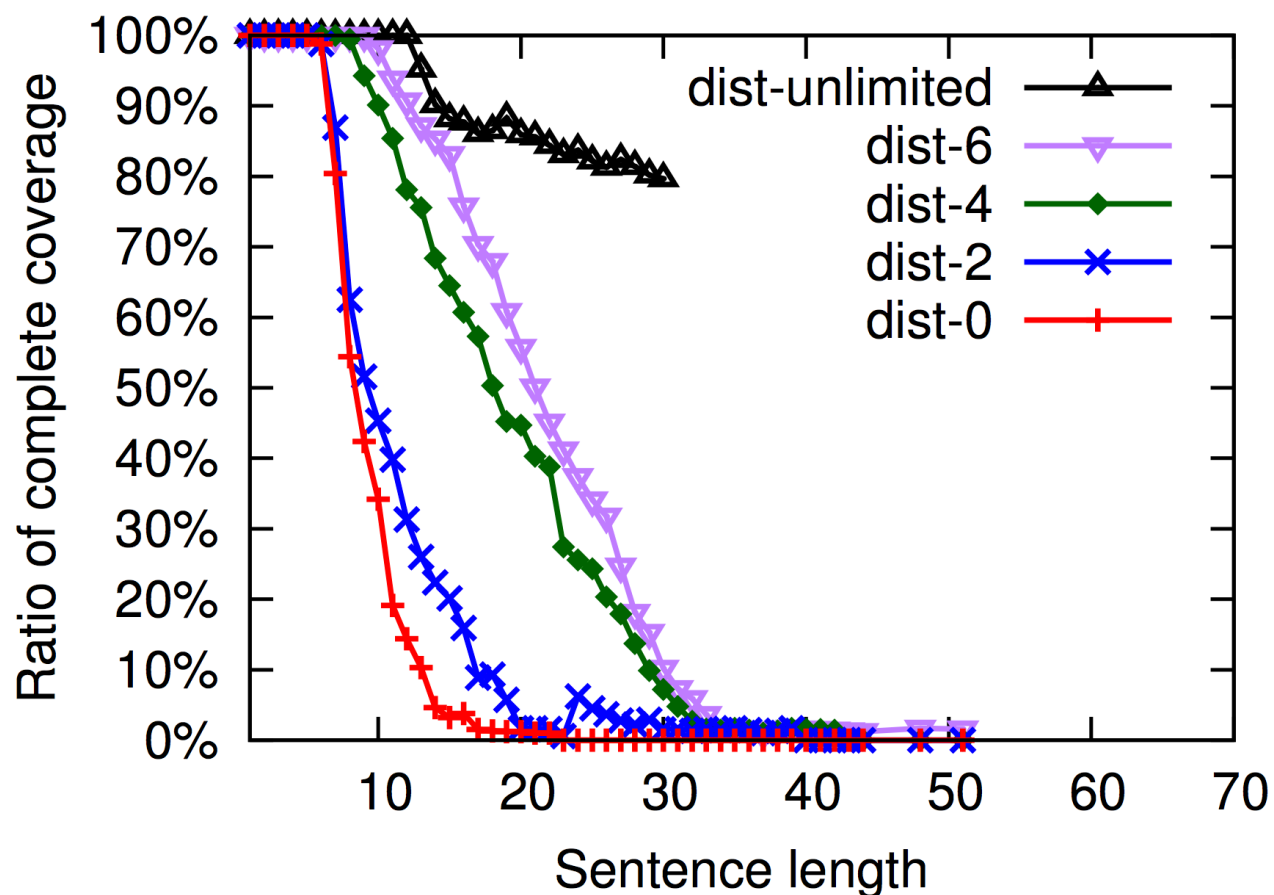
Reachabillity by distortion limit and sentence length

Chinese–English NIST [Yu et al., 2013]
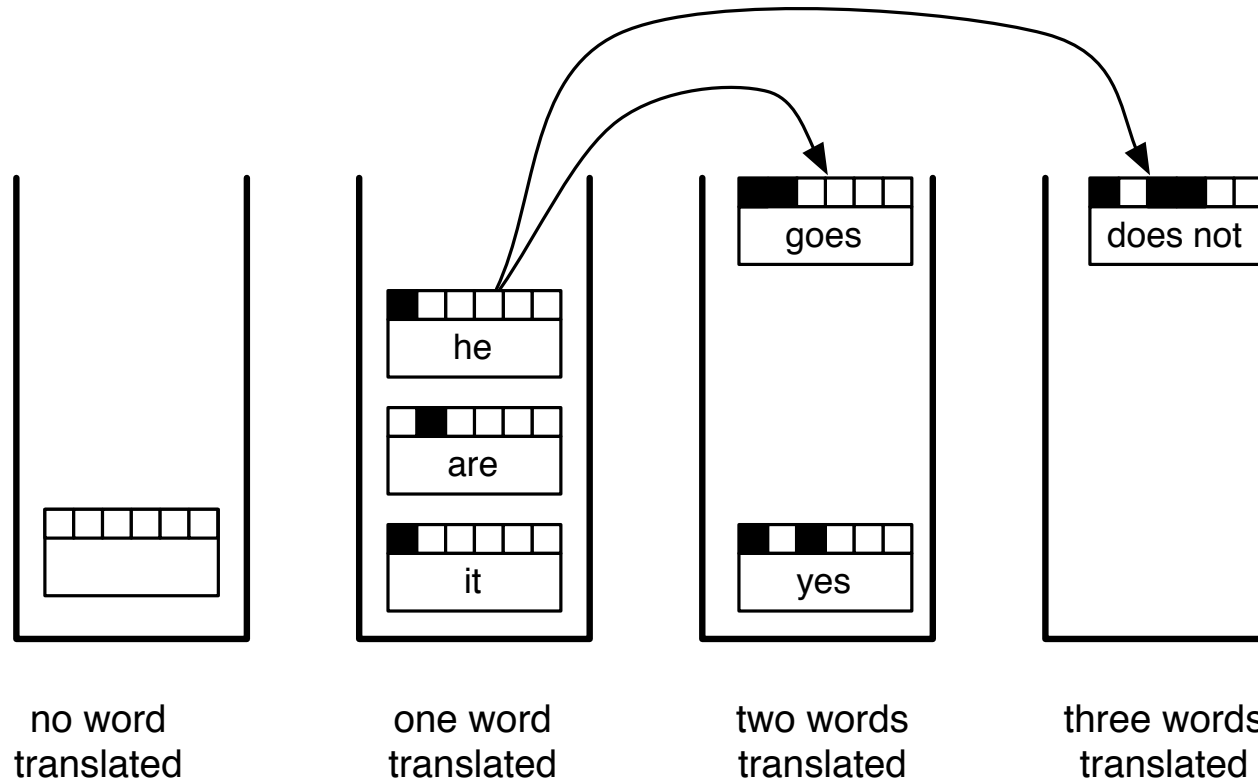
- Extend partial translations (=hypotheses) by adding translation options

- Organize hypotheses in stacks, prune out bad ones

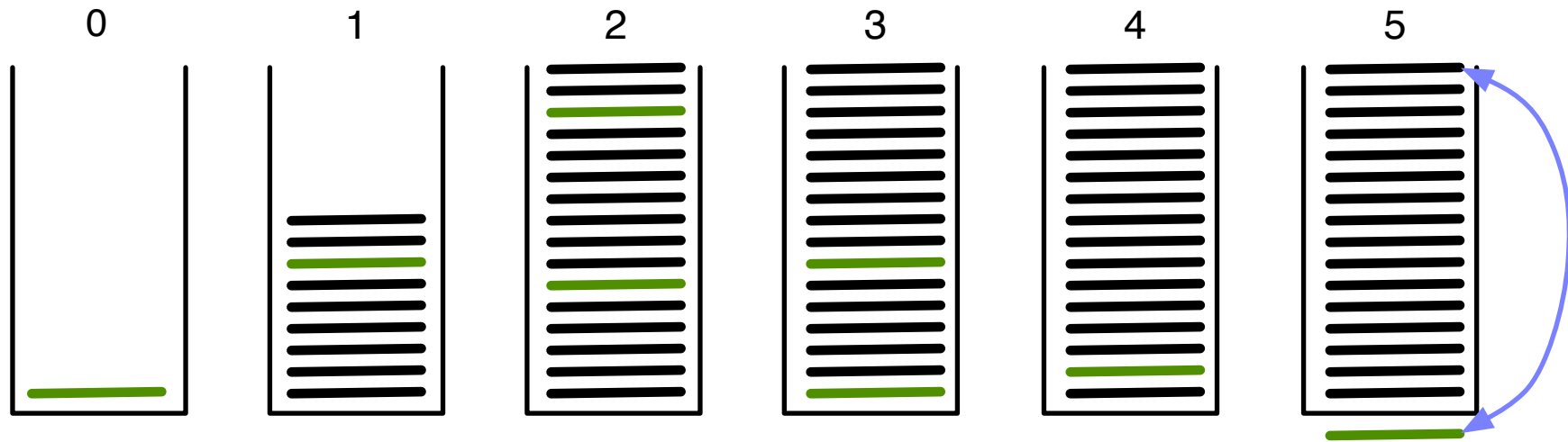- Some hypotheses match the reference translation

he does not go home

Early Updating

- At some point the best reference derivation may fall outside the beam

- Early updating

  – perceptron update between partial derivations
  – best derivation vs. best reference derivation outside beam

- Note: a reference derivation may skip a bin (multi-word phrase translation)

  → only stop when no hope that reference derivation will be in a future stack
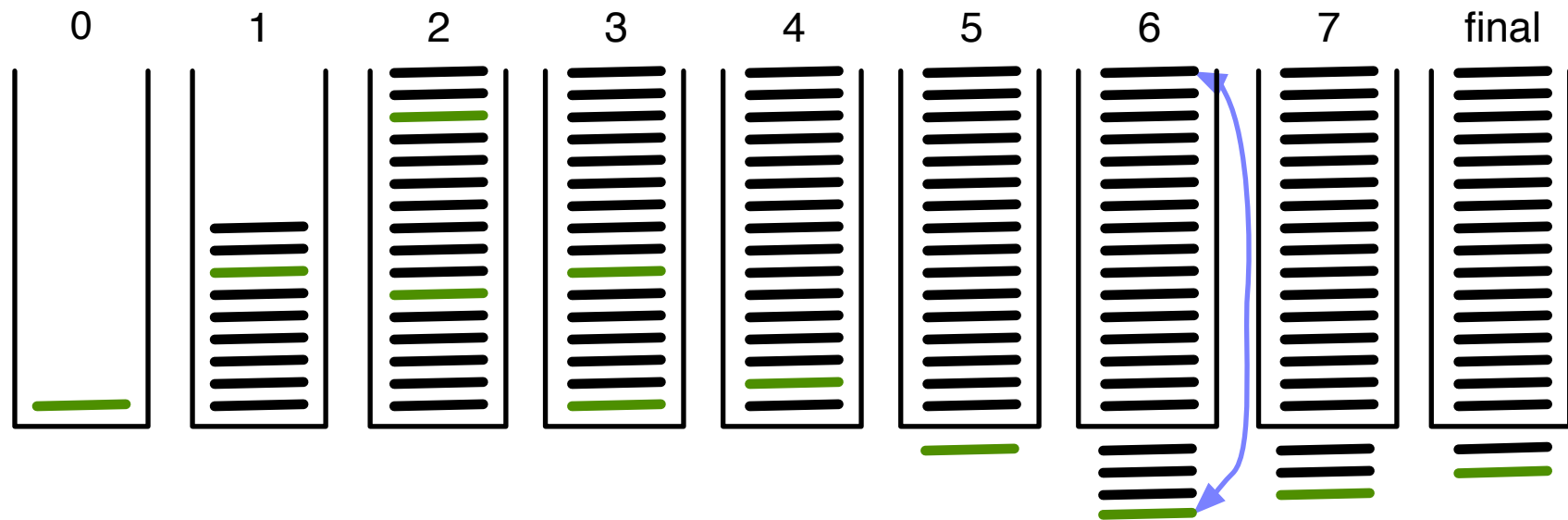
- Complete search process

- Keep best reference derivations

- Maximum violation update

  - find stack where maximal model score difference between
    * best derivation
    * best reference derivation
  - update between those two derivations

- Shown to be successful [Yu et al., 2013]

  – optimization over full training corpus
  – over 20 million features
  – relatively small data conditions (5-9 millions words)
  – gain: +2 BLEU points

- Features

  – rule id
  – word edge features (first and last word of phrase),
    defined over words, word clusters, or POS tags
  – combinations of word edge features
  – non-local features: ids of consecutive rules, rule id + last two English words

- Address overfitting: leave-one-out or singleton pruning

Summary

MIRA [Chiang 2007]
SampleRank [Haddow&al. 2011]

MaxViolation
[Yu et al., 2014]

search
space

MERT
[Och&al. 2003]

PRO
[Hopkins/May 2011]

iterative
n-best

Leave One Out
[Wuebker et al., 2012]

n-best

rule scores

aligned
corpus

a handful   thousands   millions