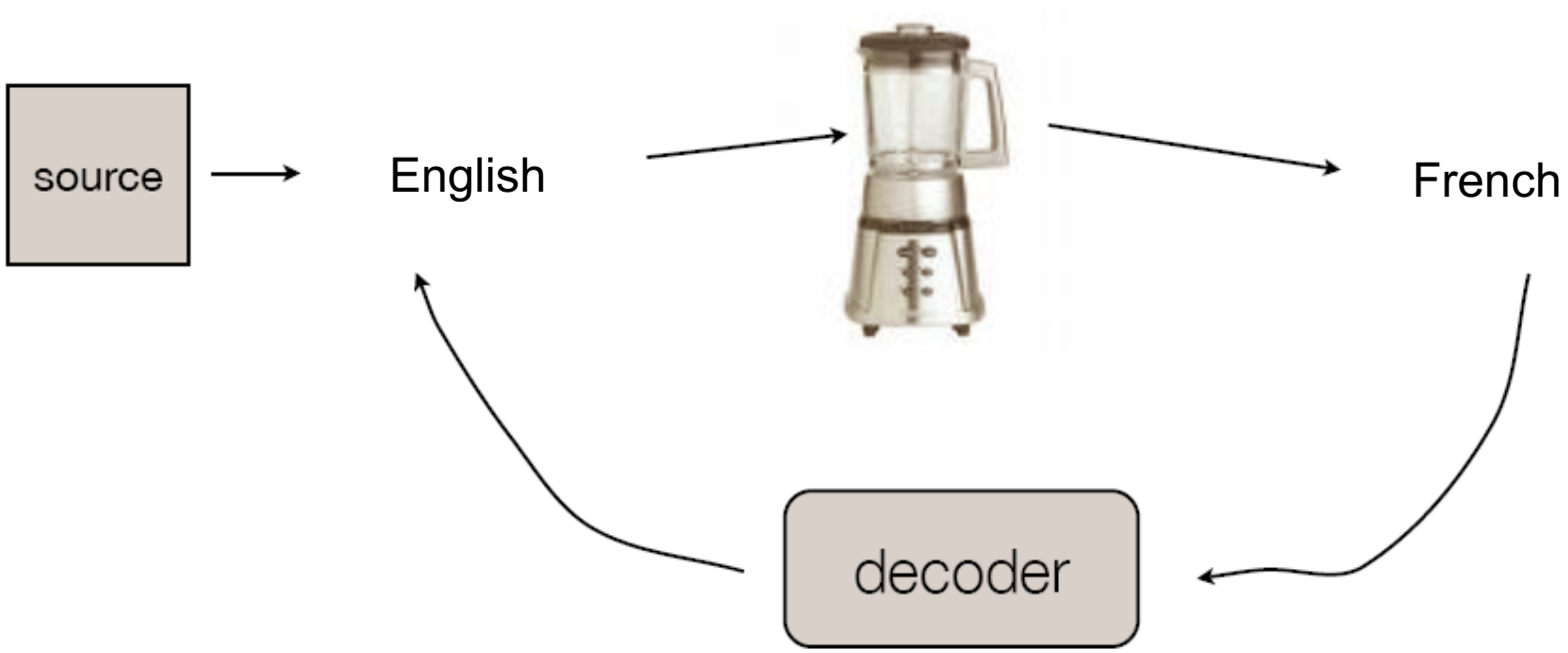
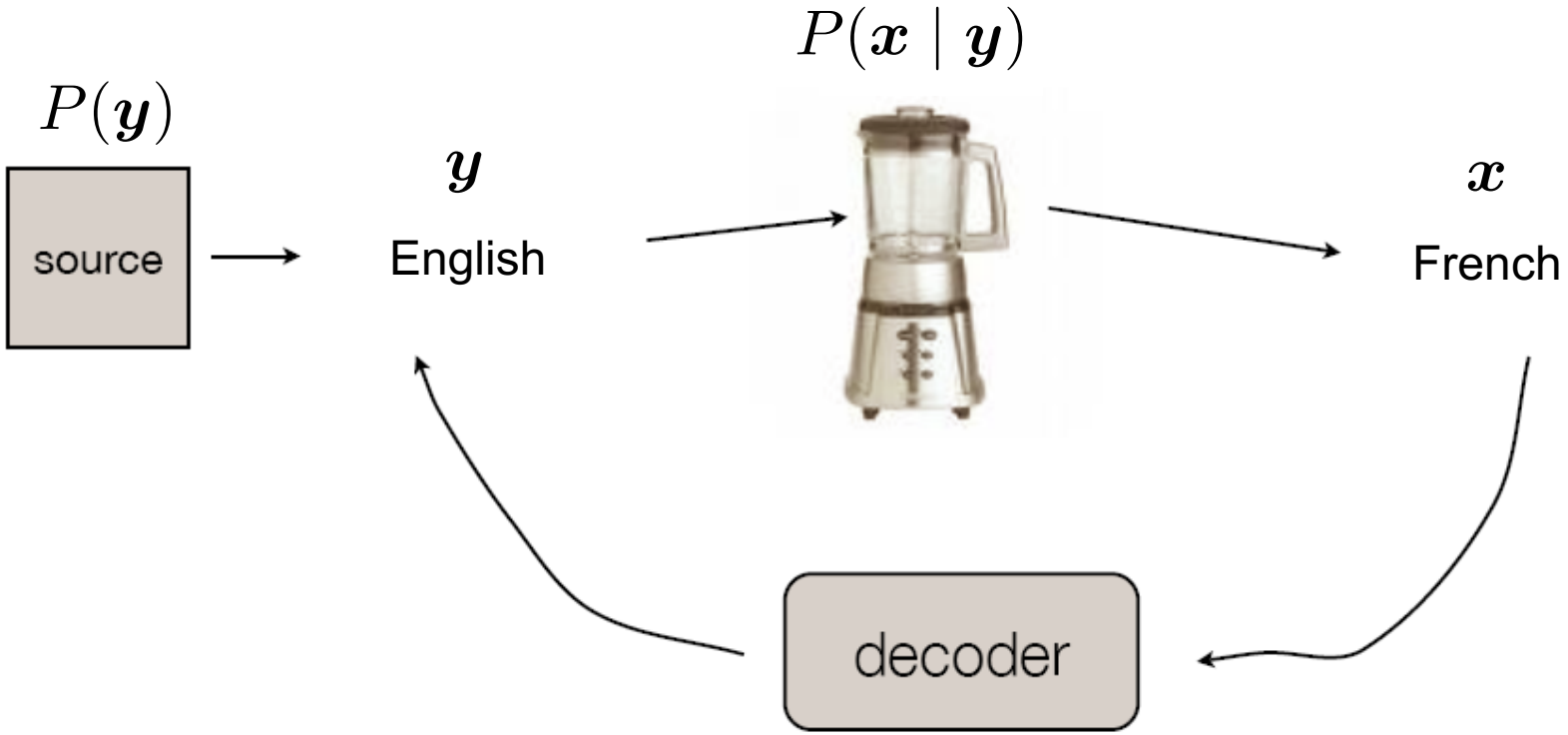
Discriminative Training

Kevin Gimpel

Noisy Channel Model



Noisy Channel Model for Translating French (x) to English (y)



$$y^* = \underset{y}{\text{argmax}}\ P(y \mid x)$$

$$= \underset{y}{\text{argmax}}\ \frac{P(x \mid y)P(y)}{P(x)}$$

$$= \underset{y}{\text{argmax}}\ P(x \mid y)P(y)$$

Noisy Channel

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}}\ P(\boldsymbol{x} \mid \boldsymbol{y})\, P(\boldsymbol{y})$$

Noisy Channel

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}}\ P(\boldsymbol{x} \mid \boldsymbol{y})\, P(\boldsymbol{y})$$

assumes we have the right model, and that we estimate it perfectly

Noisy Channel

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y}} P(\boldsymbol{x} \mid \boldsymbol{y})\, P(\boldsymbol{y})$$

assumes we have the right model, and that we estimate it perfectly

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y}} P(\boldsymbol{x} \mid \boldsymbol{y})^{\alpha}\, P(\boldsymbol{y})^{\beta}$$

Noisy Channel

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}}\ P(\boldsymbol{x} \mid \boldsymbol{y})\, P(\boldsymbol{y})$$

assumes we have the right model, and that we estimate it perfectly

$$\begin{aligned}\boldsymbol{y}^* &= \underset{\boldsymbol{y}}{\operatorname{argmax}}\ P(\boldsymbol{x} \mid \boldsymbol{y})^{\alpha}\, P(\boldsymbol{y})^{\beta} \\ &= \underset{\boldsymbol{y}}{\operatorname{argmax}}\ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y})\end{aligned}$$

extra parameters to tune, can tune to optimize BLEU (or whatever metric you want)

"tuning"

Noisy Channel → Linear Model?

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y})$$

since we're not using idealized decoding rule anymore, why not add more feature functions?

Noisy Channel → Linear Model?

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y})$$

since we're not using idealized decoding rule anymore, why not add more feature functions?

"word count feature":

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y}) + \gamma |\boldsymbol{y}|$$

Noisy Channel → Linear Model?

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y})$$

since we're not using idealized decoding rule anymore, why not add more feature functions?

"word count feature":

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y}) + \boxed{\gamma \, |\boldsymbol{y}|}$$

"reverse translation model feature":

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y}) + \gamma \, |\boldsymbol{y}| + \boxed{\delta \ \log P(\boldsymbol{y} \mid \boldsymbol{x})}$$

Noisy Channel → Linear Model?

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y})$$

since we're not using idealized decoding rule anymore, why not add more feature functions?

"word count feature":

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y}) + \gamma |\boldsymbol{y}|$$

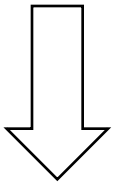
"reverse translation model feature":

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y}) + \gamma |\boldsymbol{y}| + \delta \log P(\boldsymbol{y} \mid \boldsymbol{x})$$

but if we keep adding features, tuning gets harder…

Noisy Channel → Linear Model

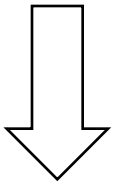
$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y})$$



generalize to a linear model

Noisy Channel → Linear Model

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y})$$



generalize to a linear model

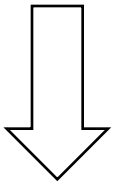
$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \sum_{i=1}^{K} \theta_i f_i(\boldsymbol{x}, \boldsymbol{y})$$

"feature weights"

"feature functions"

Noisy Channel → Linear Model

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y})$$



generalize to a linear model

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \sum_{i=1}^{K} \theta_i f_i(\boldsymbol{x}, \boldsymbol{y})$$

we will write it like this:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$$

"feature weight vector"

"feature function vector"

Log-Linear Models

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}}\ \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$$

if you want a probability distribution over translations of a given source sentence, exponentiate and normalize:

Log-Linear Models

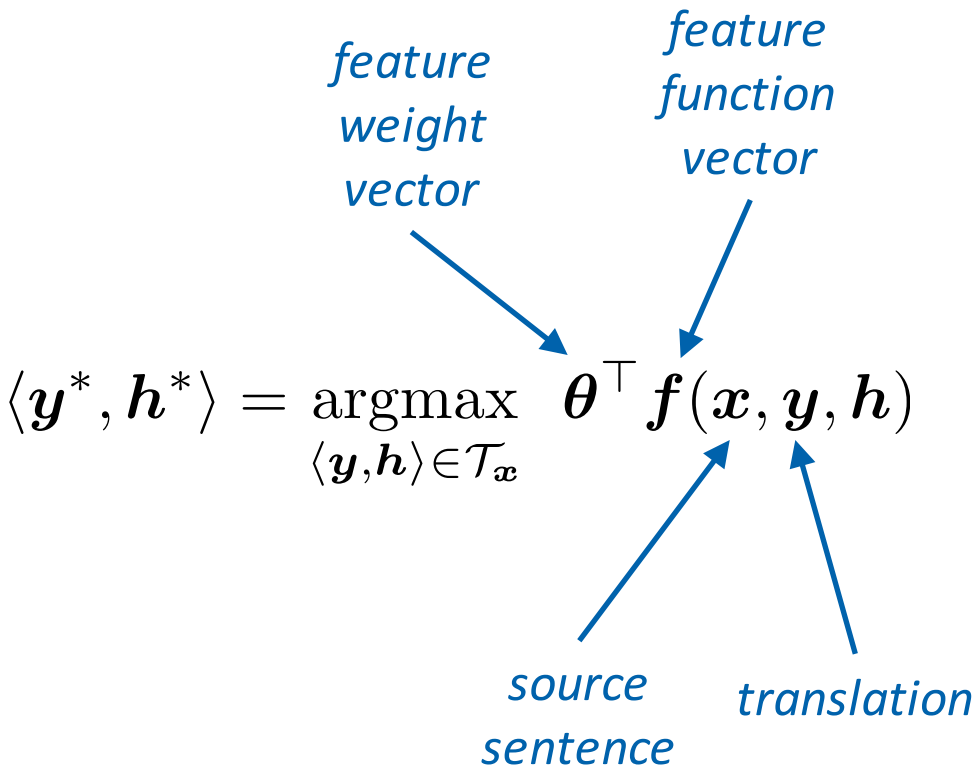
$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}}\ \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$$

if you want a probability distribution over translations of a given source sentence, exponentiate and normalize:

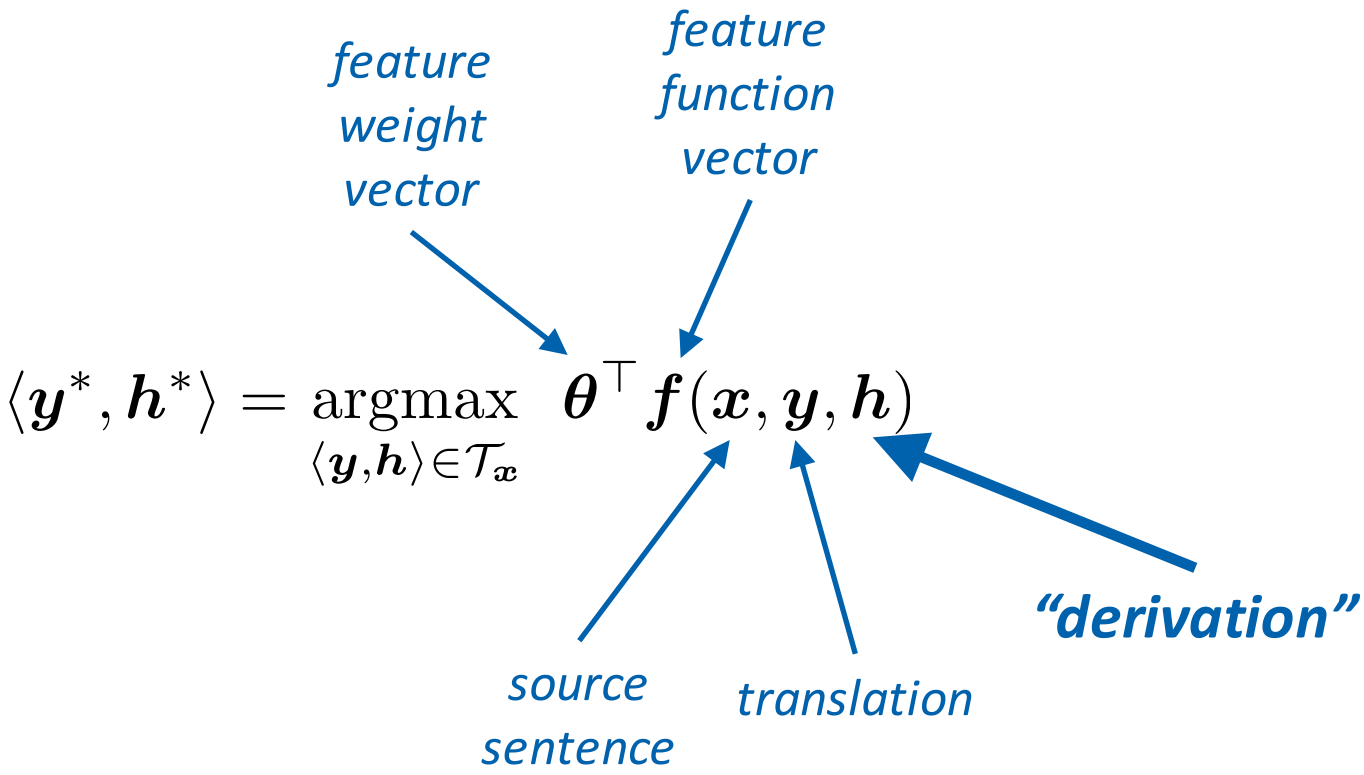
$$P(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})\}}{\sum_{\boldsymbol{y}'} \exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}')\}}$$

this is a (conditional) “log-linear” model

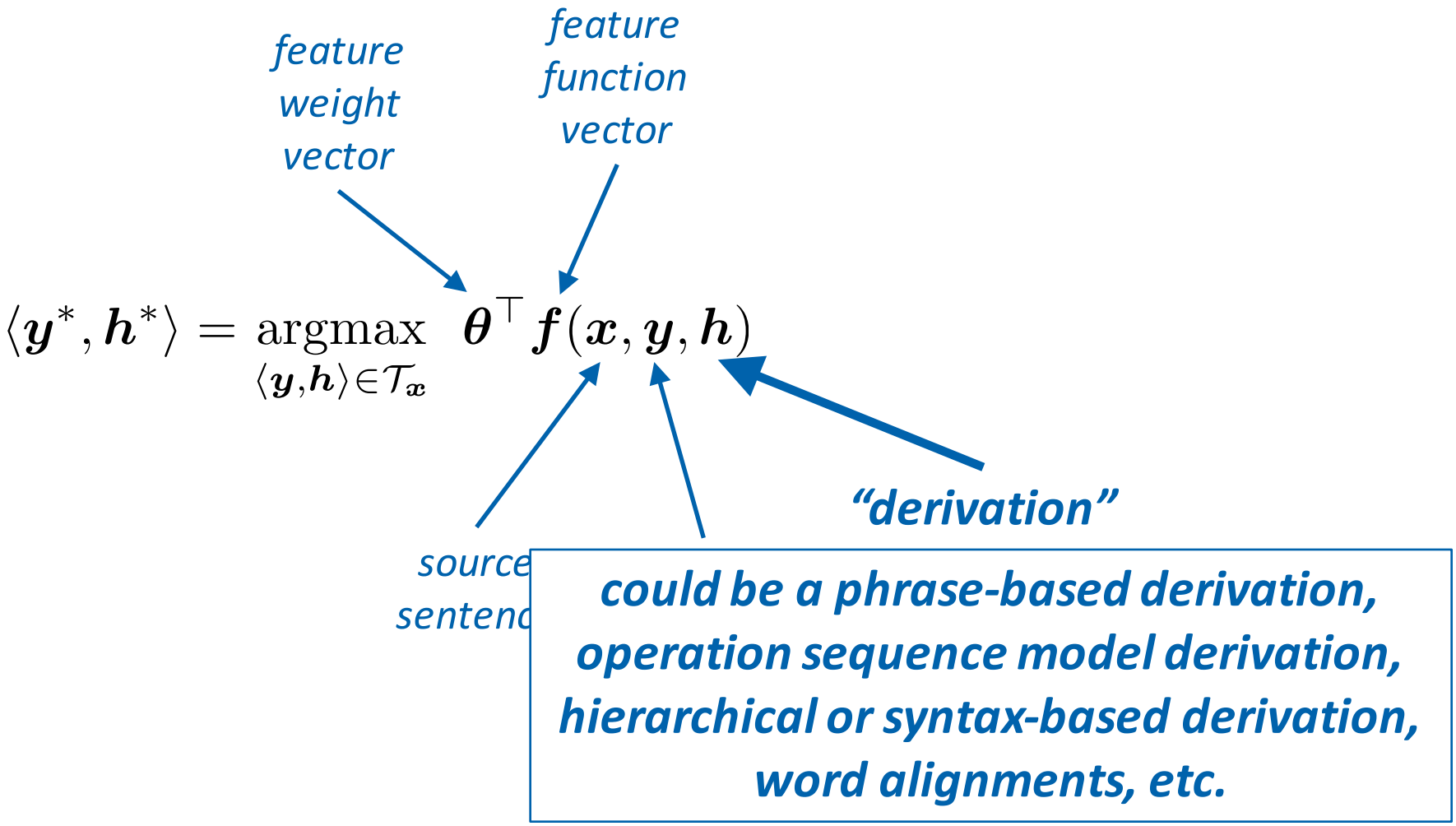
More General Formulation: Derivations



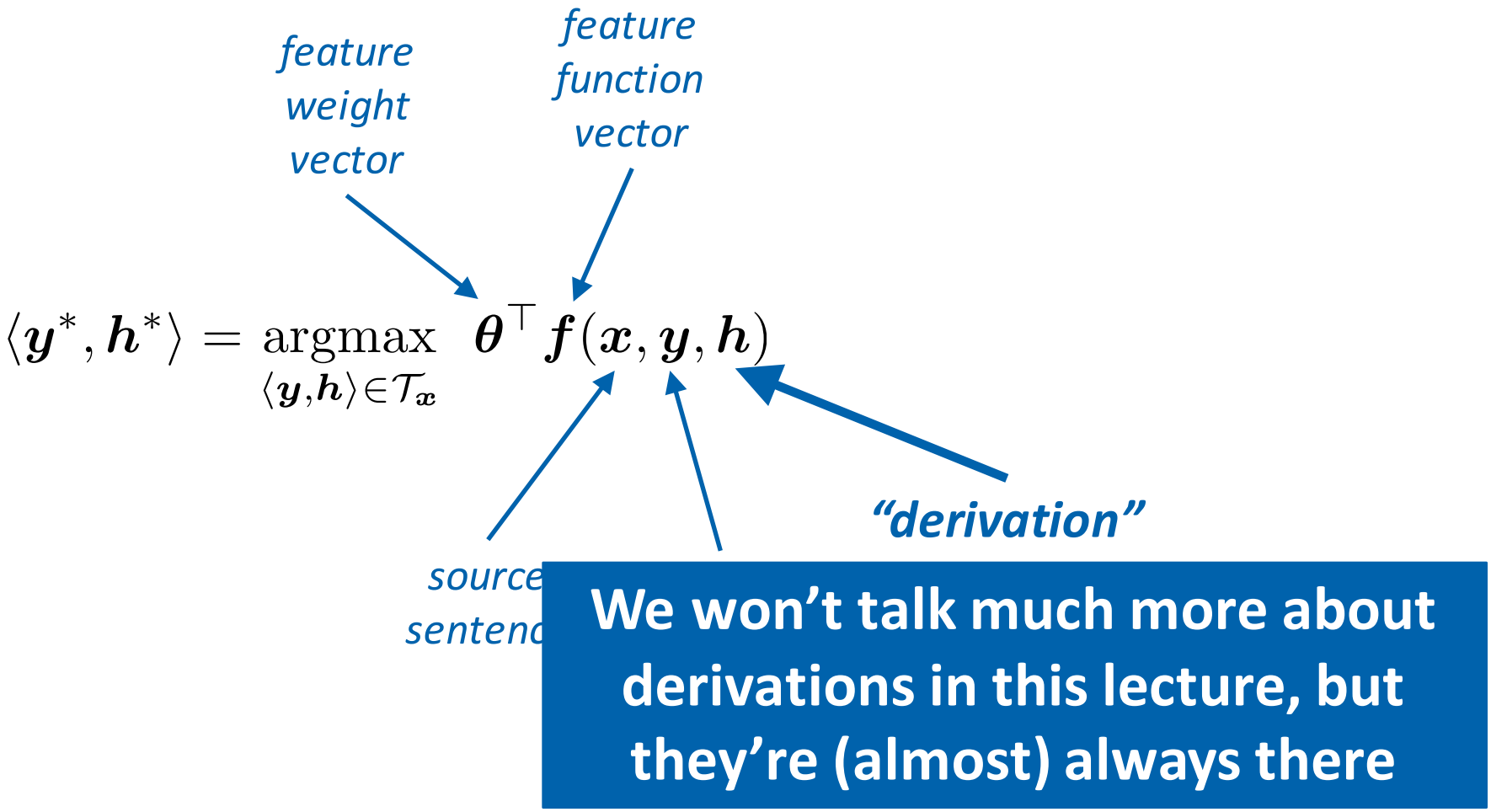
More General Formulation: Derivations



More General Formulation: Derivations



More General Formulation: Derivations



Overview

This lecture is about algorithms for choosing the feature weights θ

Since we have a linear model, we can use supervised machine learning

But MT differs from typical supervised tasks (as we'll see), so MT training procedures differ too

We'll start with a way to visualize training for machine translation

African National Congress	opposition	sanction	Zimbabwe
非国大	反对	制裁	津巴布韦

African National Congress | opposition | sanction | Zimbabwe

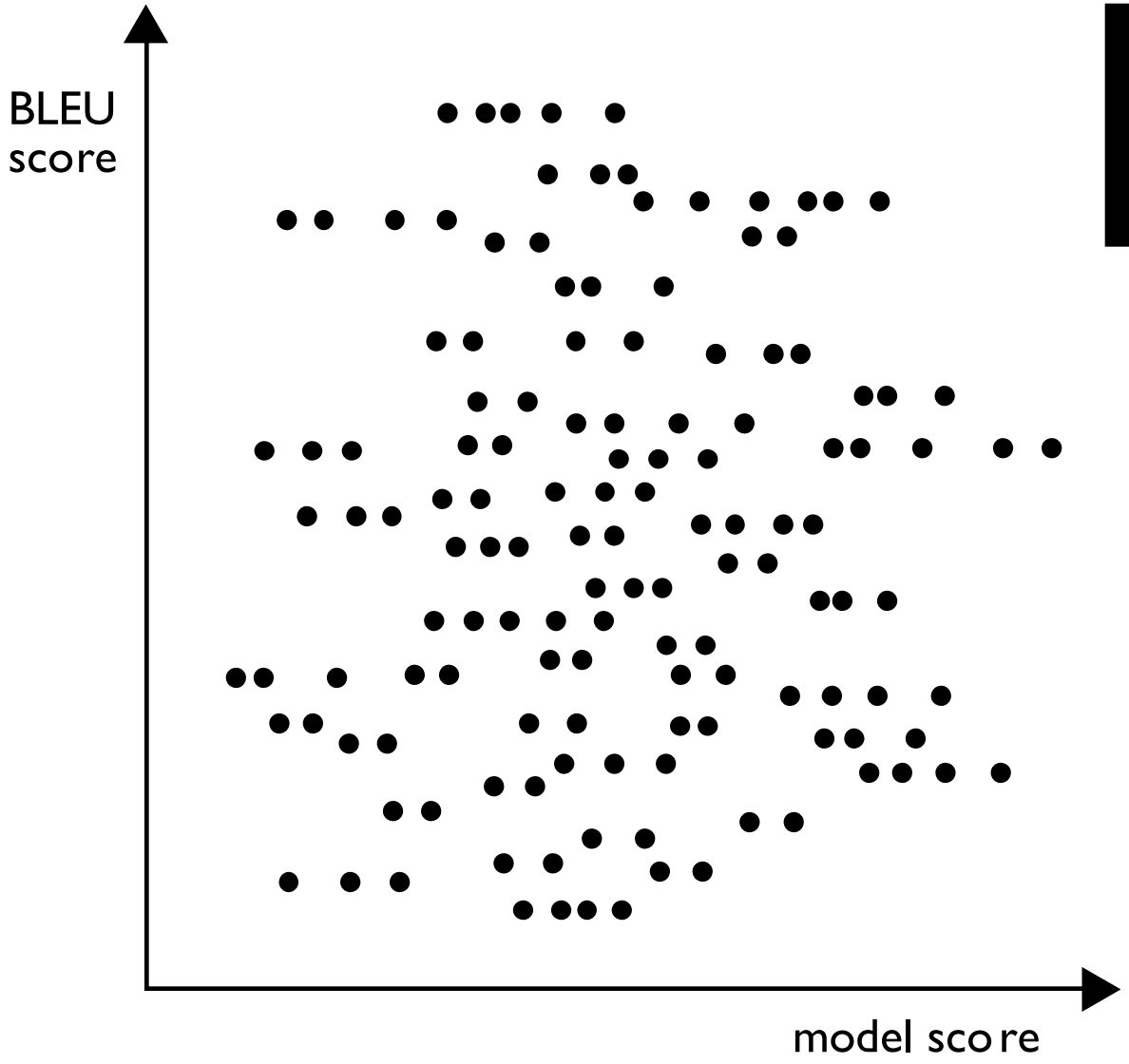
非国大 | 反对 | 制裁 | 津巴布韦

Gold standard:
African National Congress opposes sanctions against Zimbabwe

African National Congress | opposition | sanction | Zimbabwe

非国大 | 反对 | 制裁 | 津巴布韦

Gold standard:
African National Congress opposes sanctions against Zimbabwe

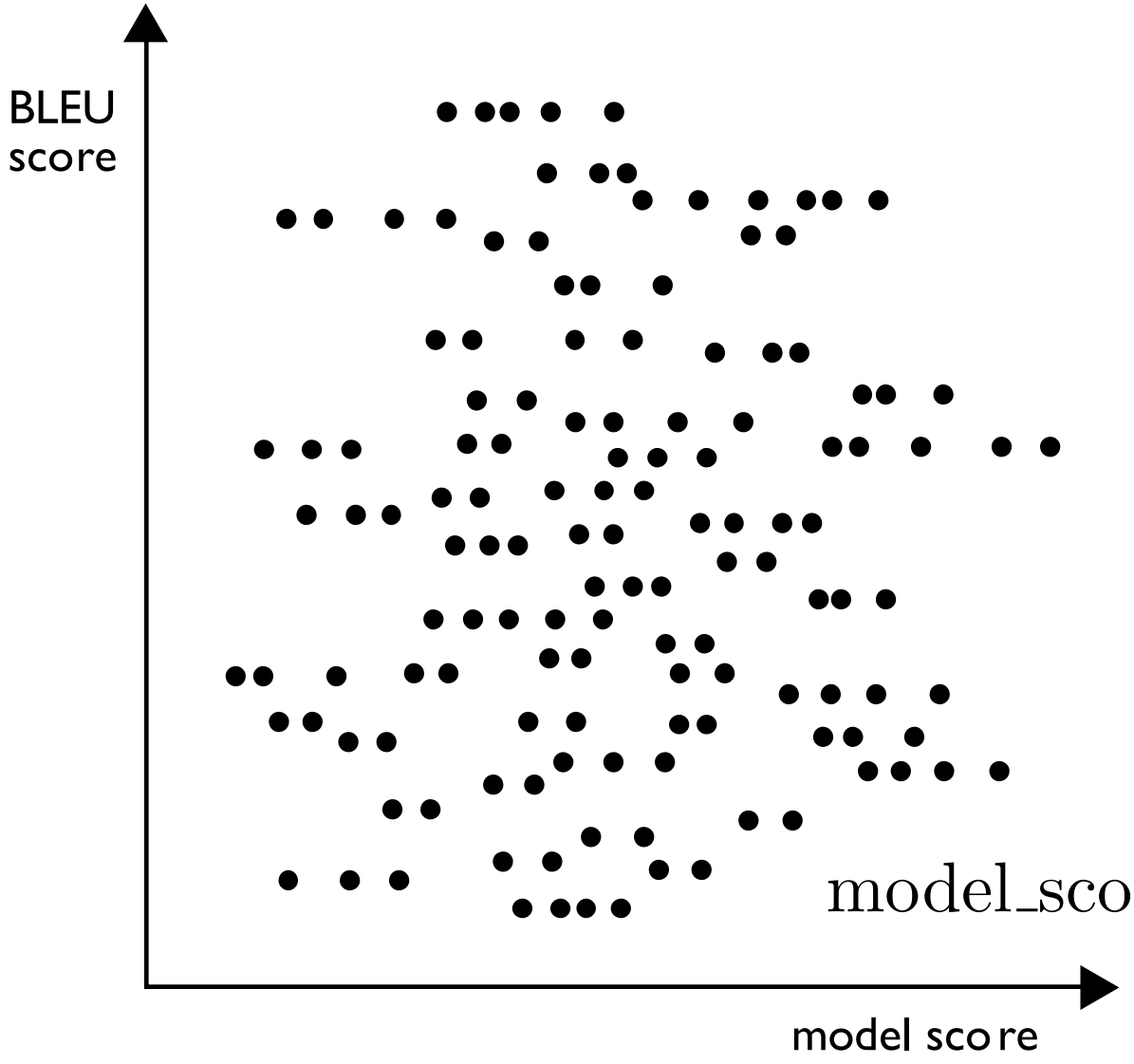


each point is a translation

African National Congress | opposition | sanction | Zimbabwe

非国大 反对 制裁 津巴布韦

Gold standard:
African National Congress opposes sanctions against Zimbabwe

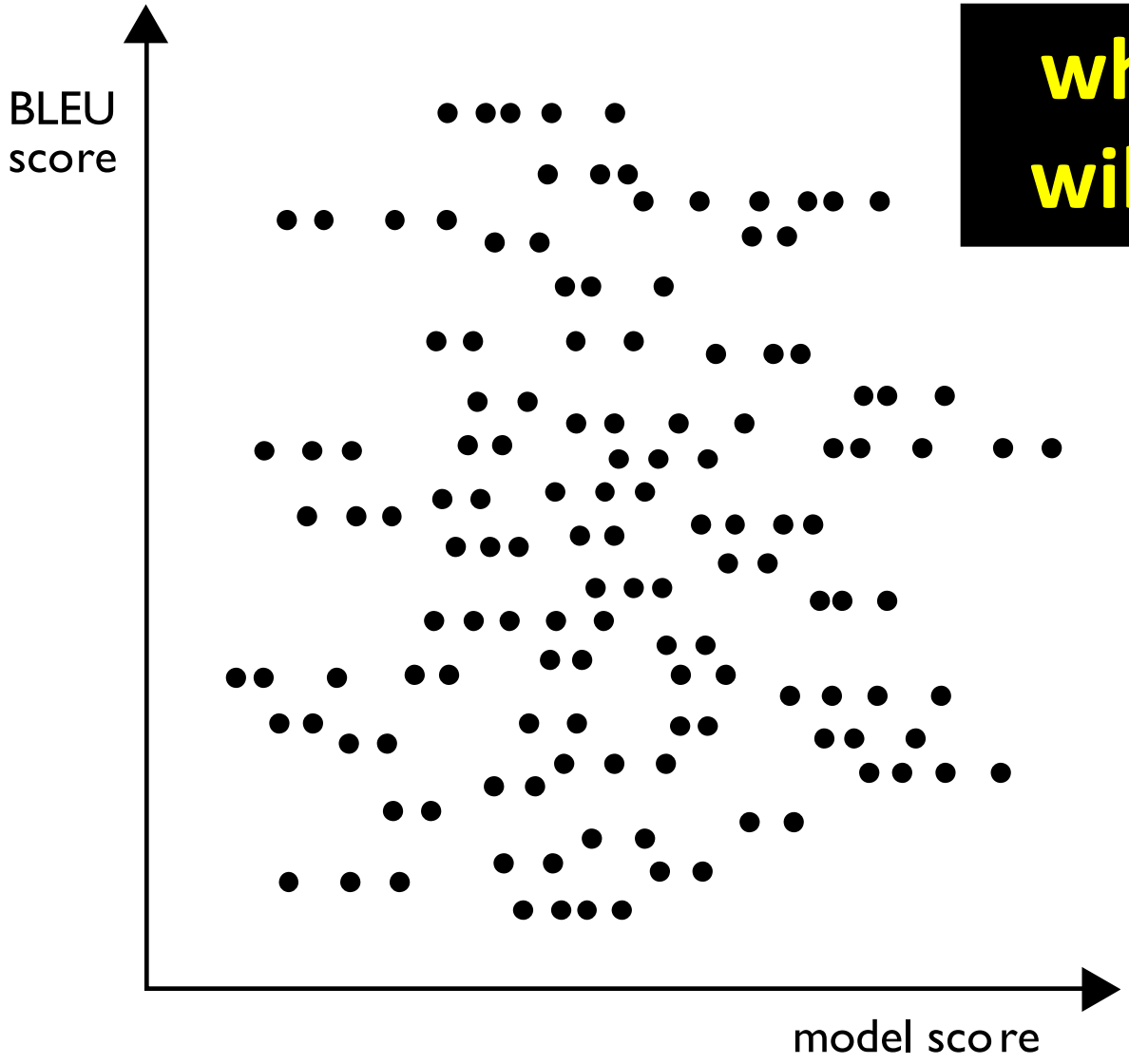


$$\text{model\_score}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$$

African National Congress | opposition | sanction | Zimbabwe

非国大 | 反对 | 制裁 | 津巴布韦

Gold standard:
African National Congress opposes sanctions against Zimbabwe

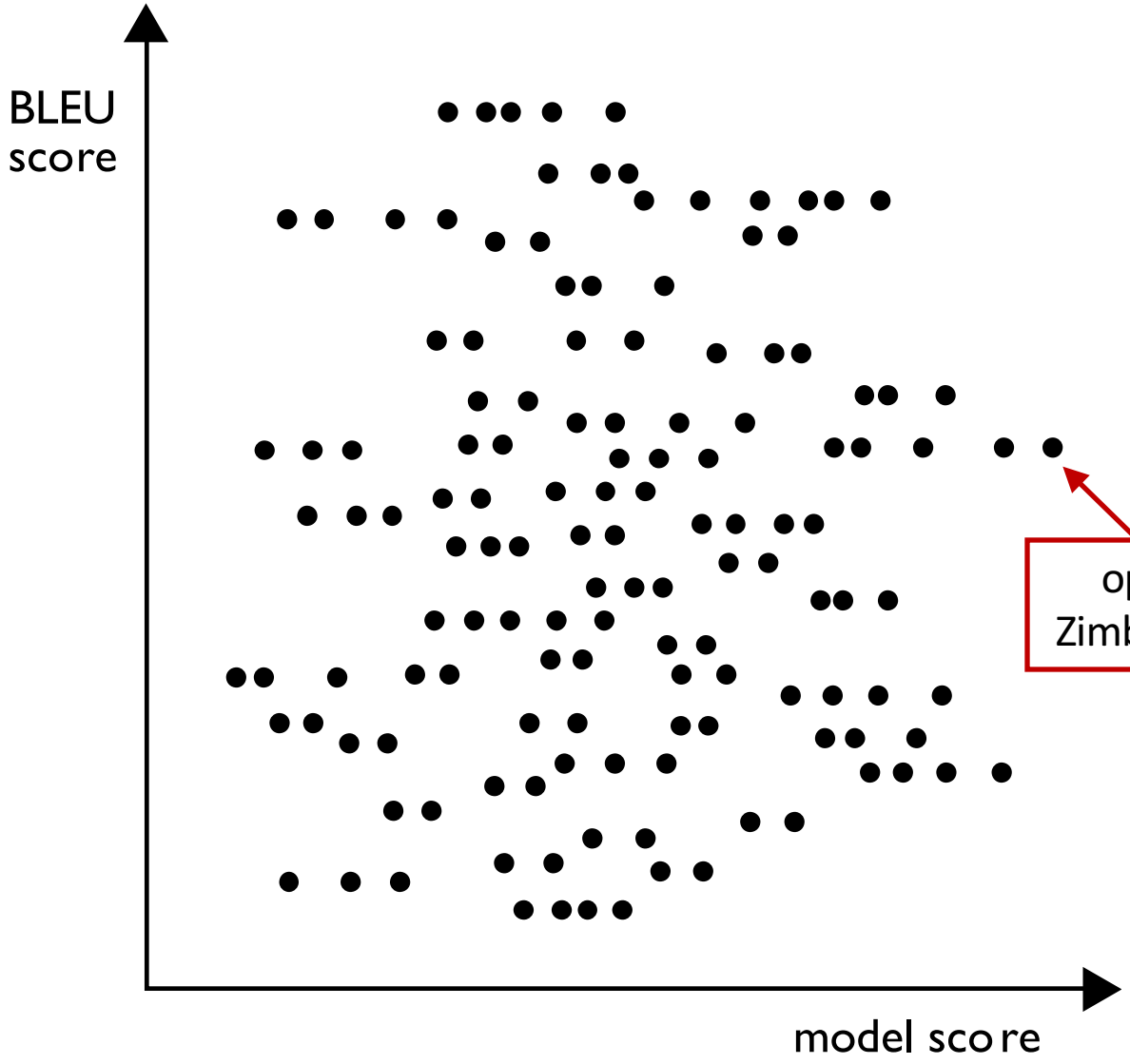


which translation will be predicted?

African National Congress | opposition | sanction | Zimbabwe

非国大 | 反对 | 制裁 | 津巴布韦

Gold standard:
African National Congress opposes sanctions against Zimbabwe



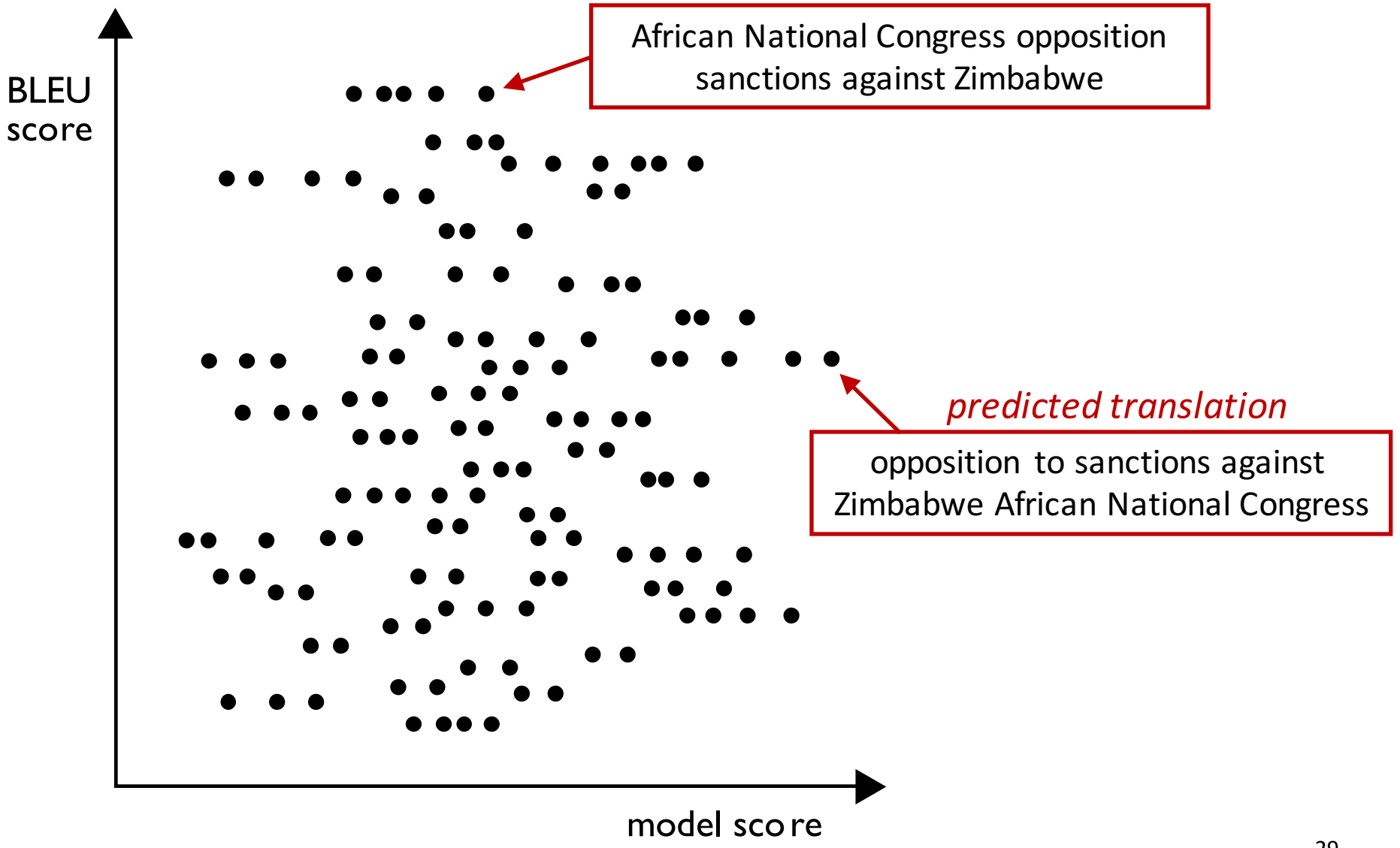
predicted translation

opposition to sanctions against Zimbabwe African National Congress

African National Congress | opposition | sanction | Zimbabwe

非国大 | 反对 | 制裁 | 津巴布韦

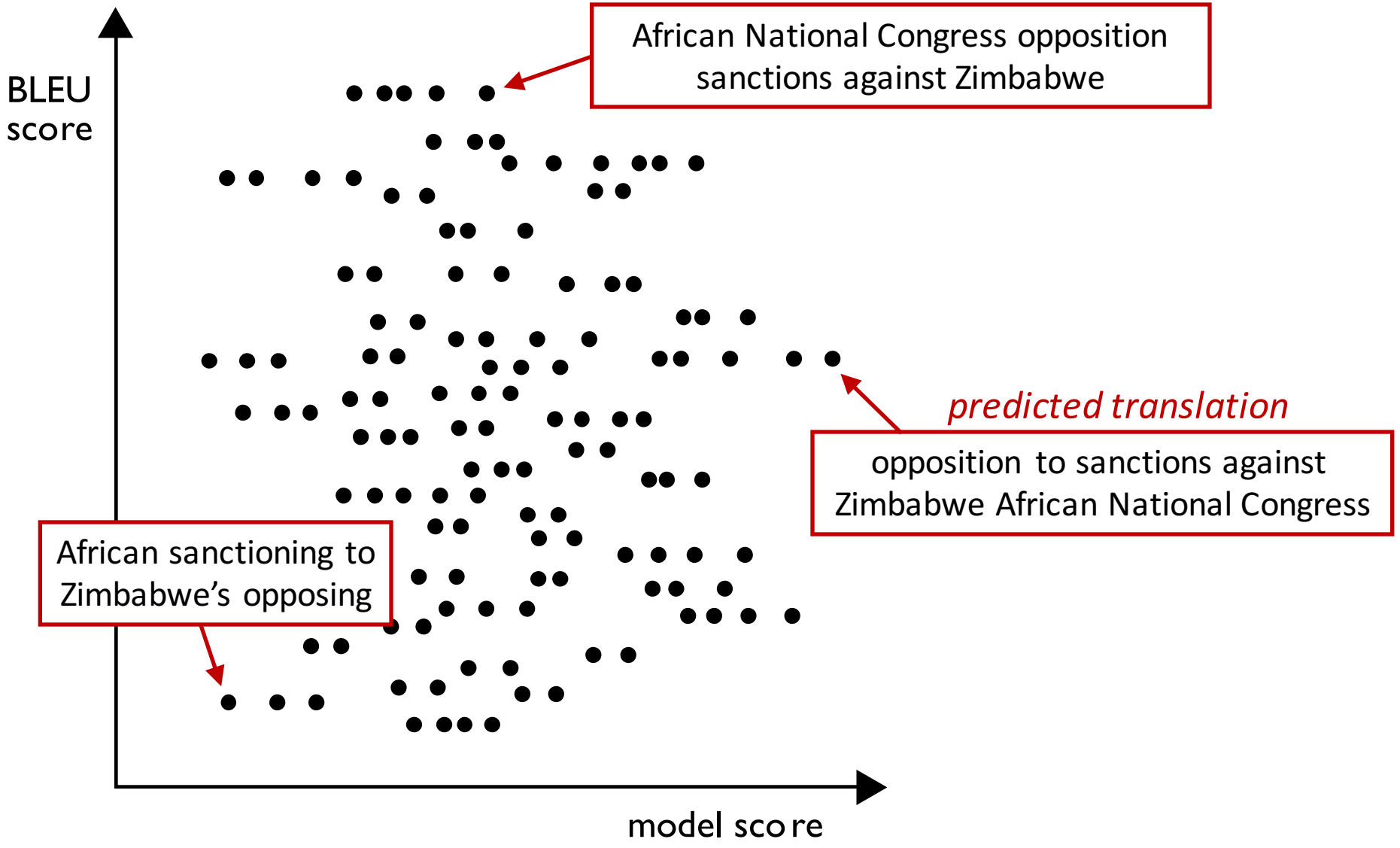
Gold standard:
African National Congress opposes sanctions against Zimbabwe



African National Congress opposition sanction Zimbabwe

非国大 反对 制裁 津巴布韦

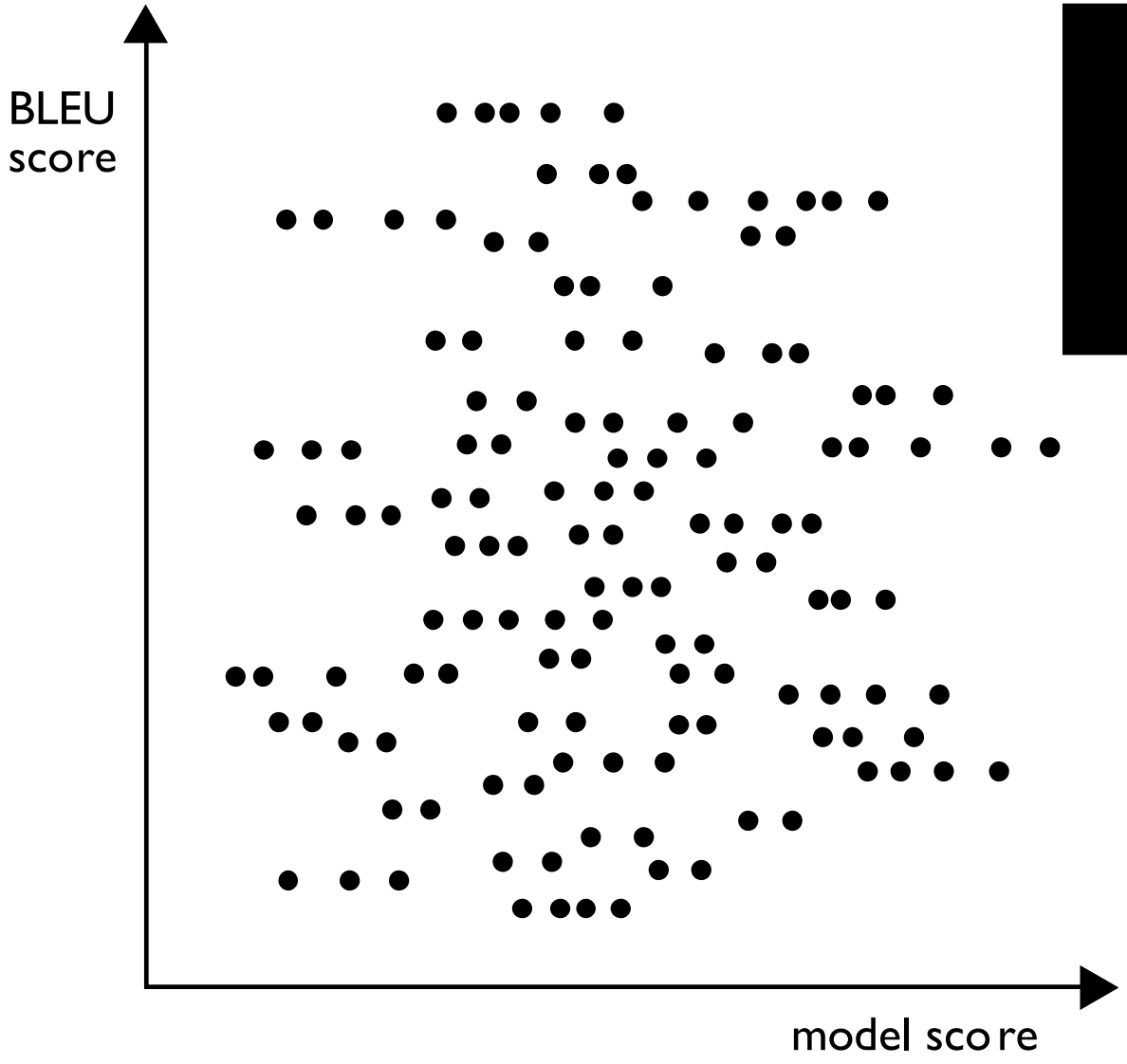
Gold standard:
African National Congress opposes sanctions against Zimbabwe



African National Congress opposition sanction Zimbabwe

非国大 反对 制裁 津巴布韦

Gold standard:
African National Congress opposes sanctions against Zimbabwe

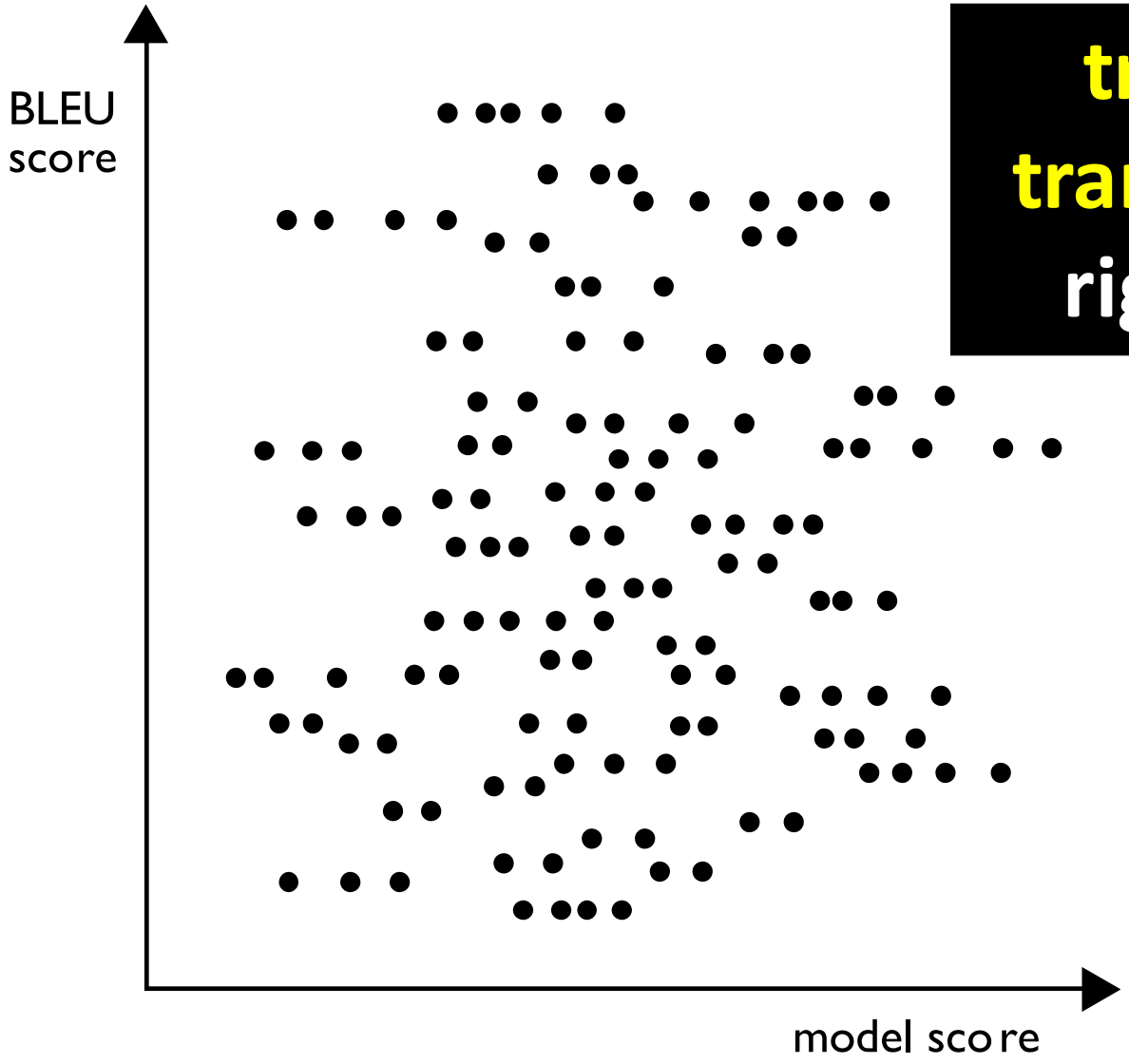


training moves translations in this plot

African National Congress | opposition | sanction | Zimbabwe

非国大 | 反对 | 制裁 | 津巴布韦

Gold standard:
African National Congress opposes sanctions against Zimbabwe

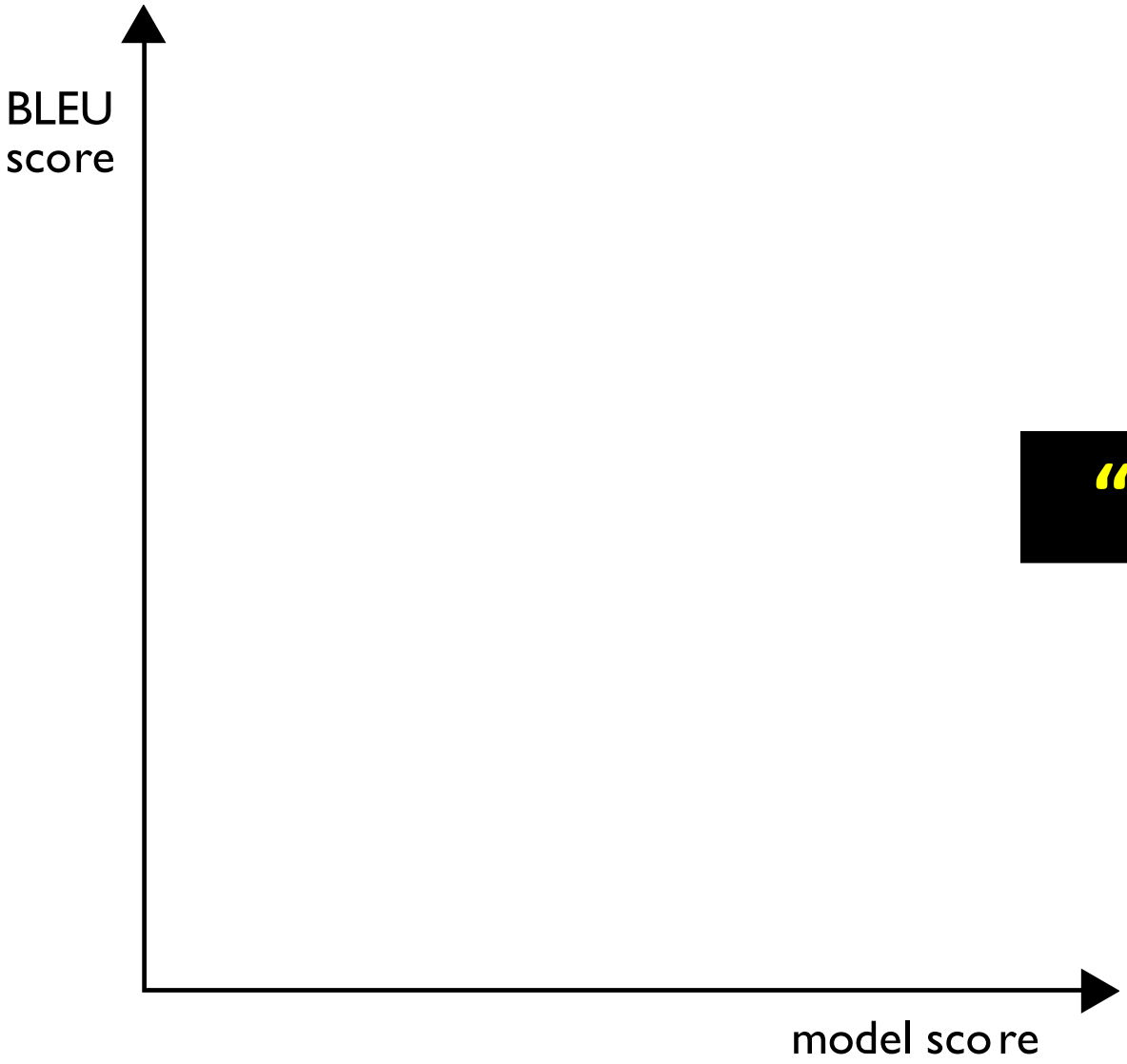


training moves translations left or right in this plot

African National Congress | opposition | sanction | Zimbabwe

非国大 | 反对 | 制裁 | 津巴布韦

Gold standard:
African National Congress opposes sanctions against Zimbabwe

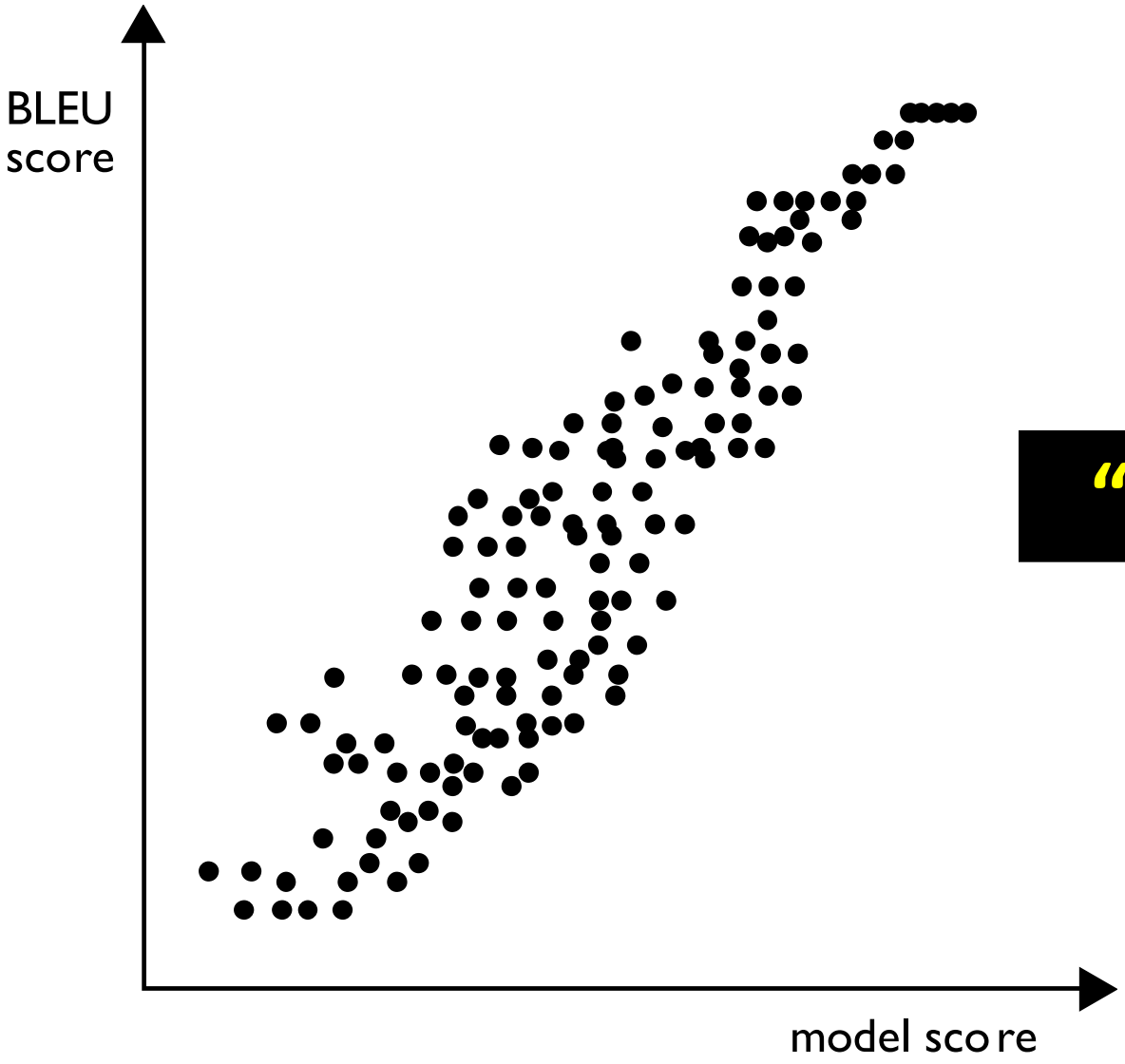


“ideal” model?

African National Congress | opposition | sanction | Zimbabwe

非国大 反对 制裁 津巴布韦

Gold standard:
African National Congress opposes sanctions against Zimbabwe

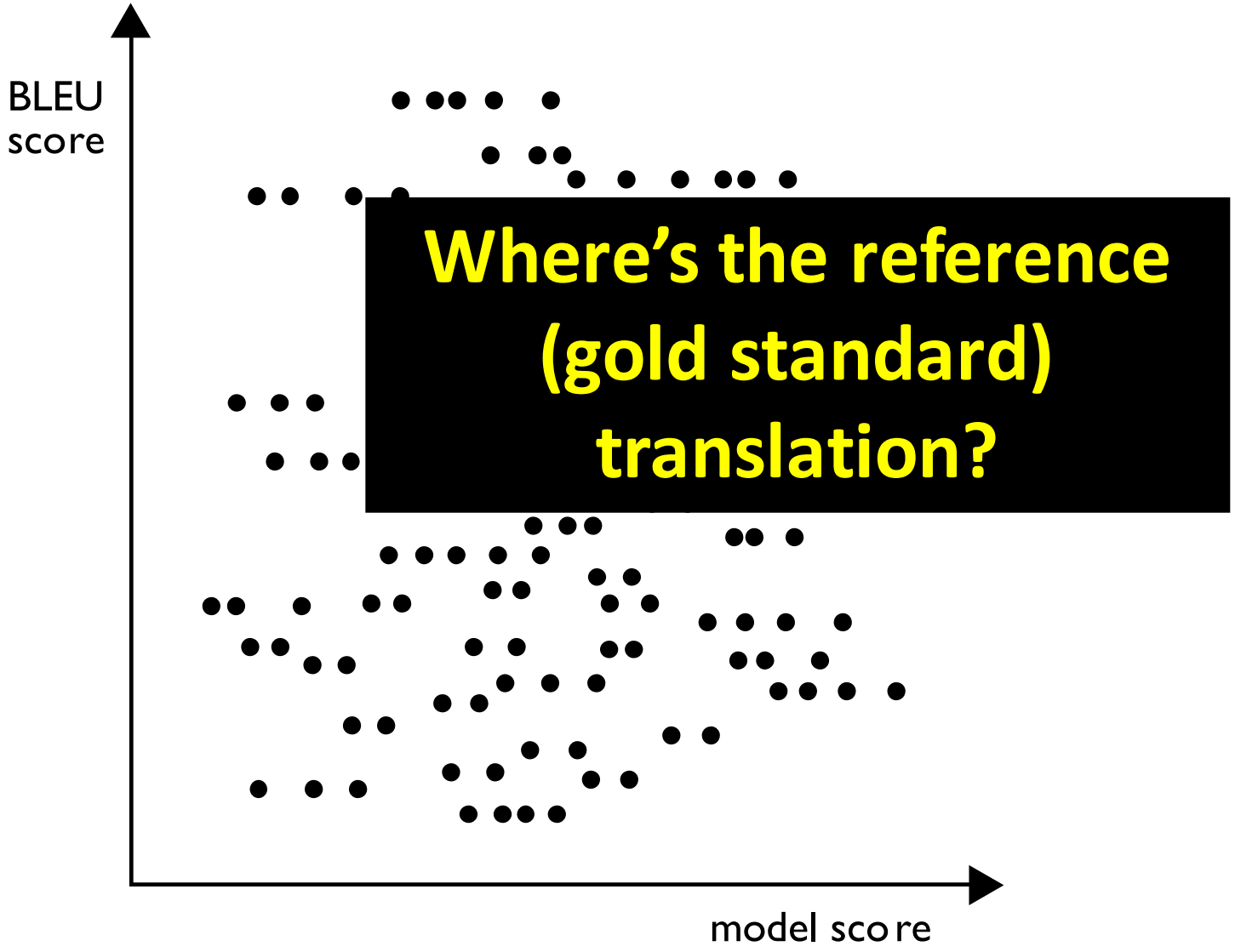


“ideal” model?

African National Congress opposition sanction Zimbabwe

非国大 反对 制裁 津巴布韦

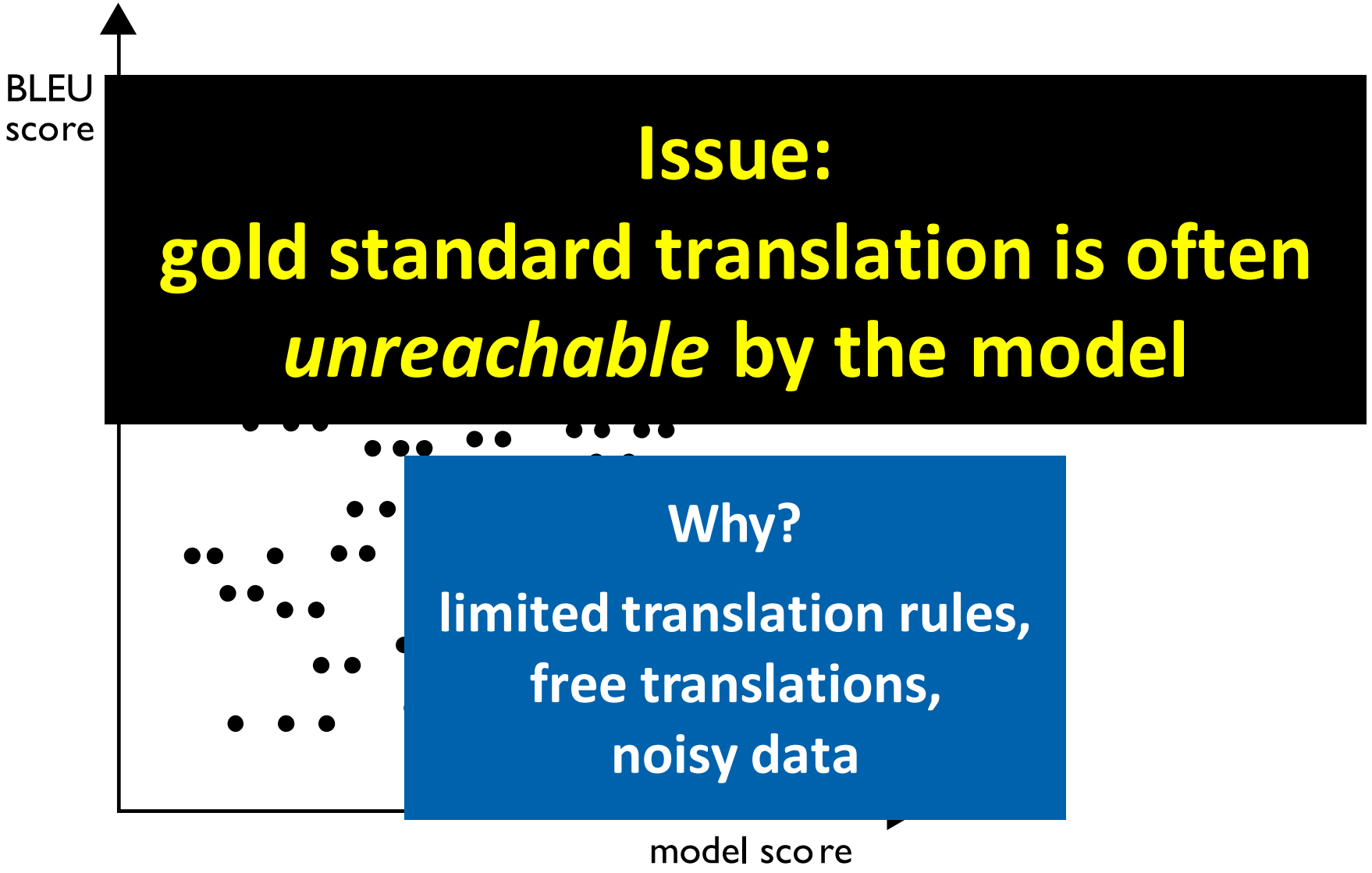
Gold standard:
African National Congress opposes sanctions against Zimbabwe



African National Congress opposition sanction Zimbabwe

非国大 反对 制裁 津巴布韦

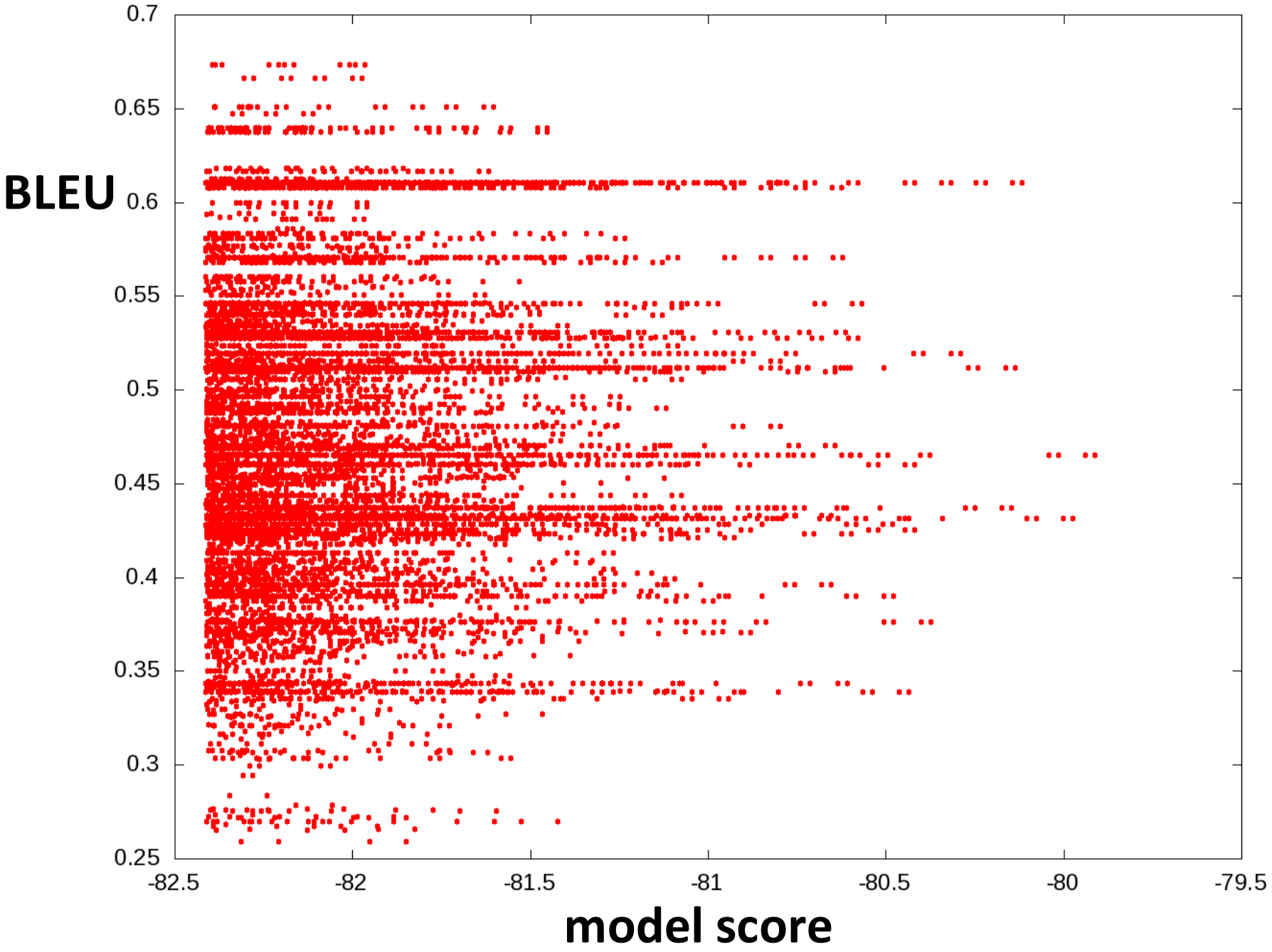
Gold standard:
African National Congress opposes sanctions against Zimbabwe

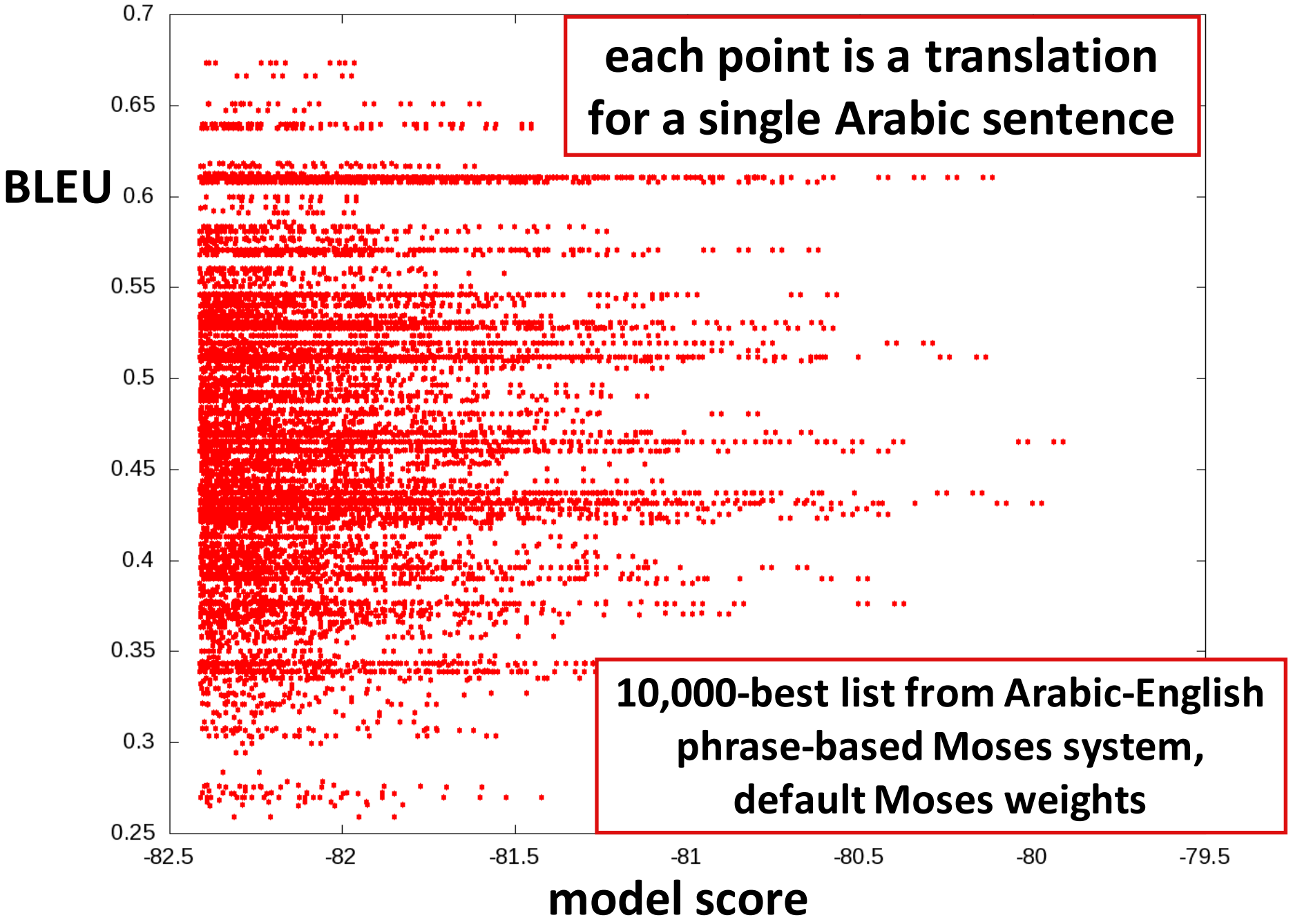


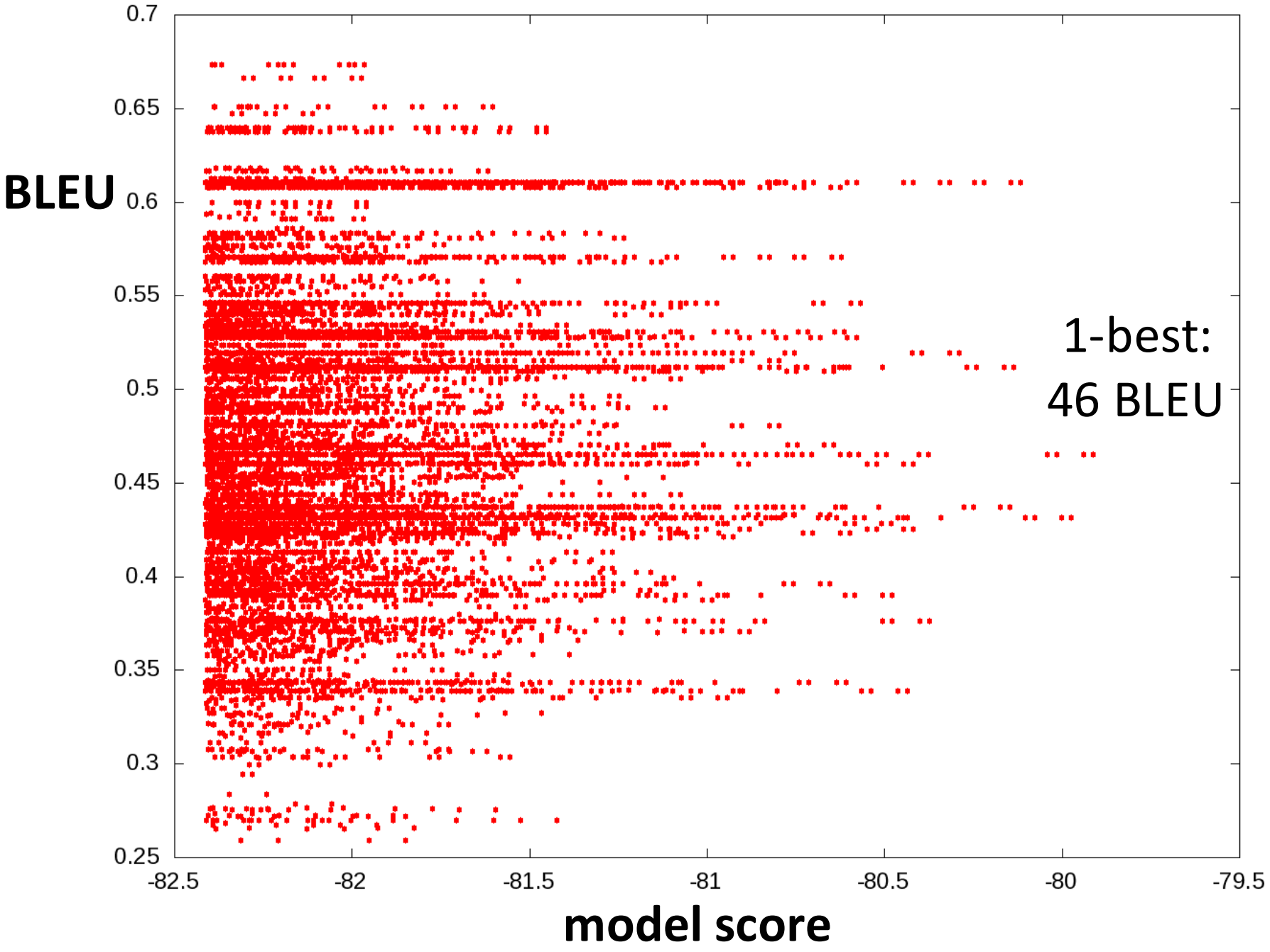
Issue:
gold standard translation is often *unreachable* by the model

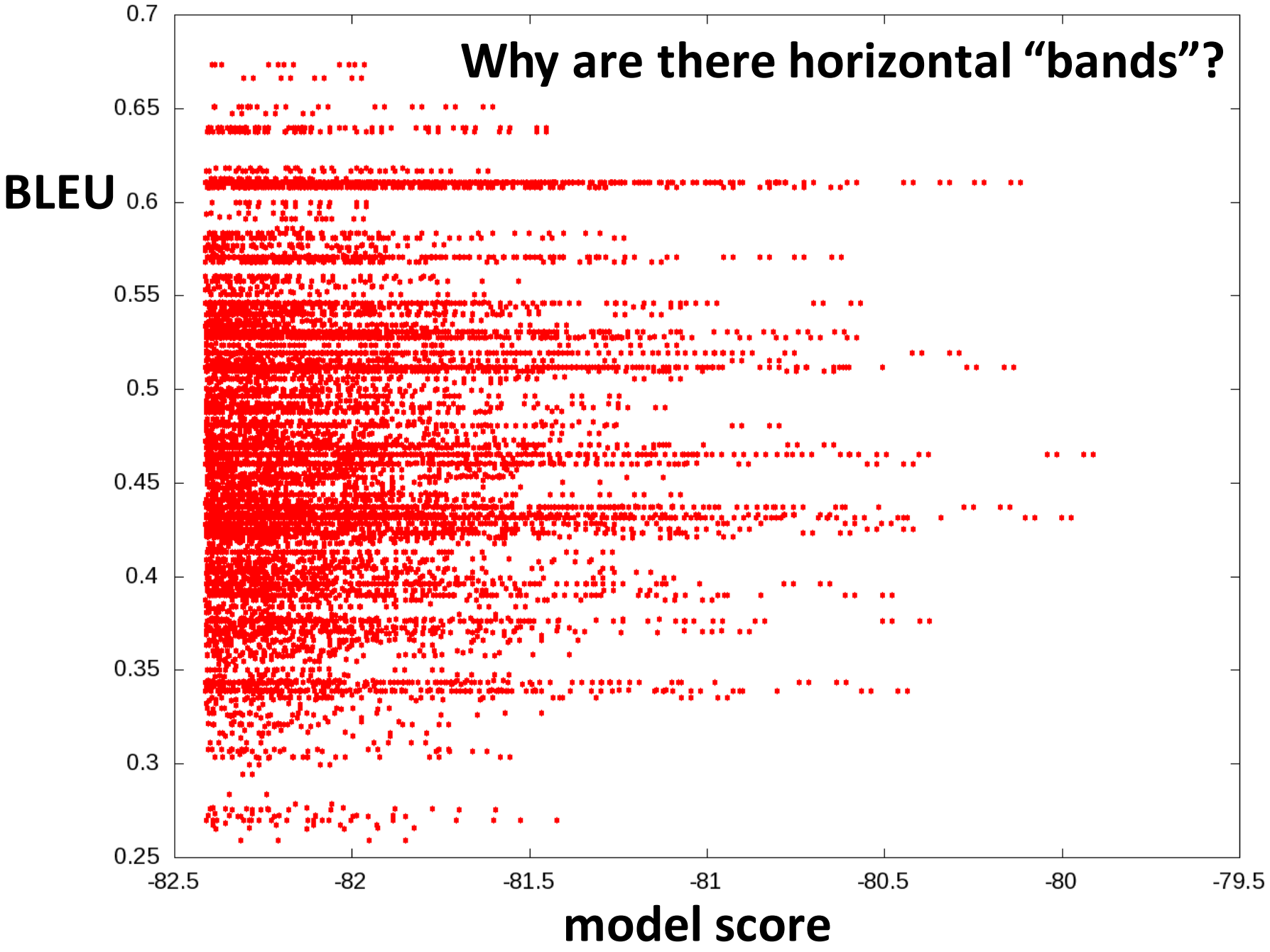
Why?

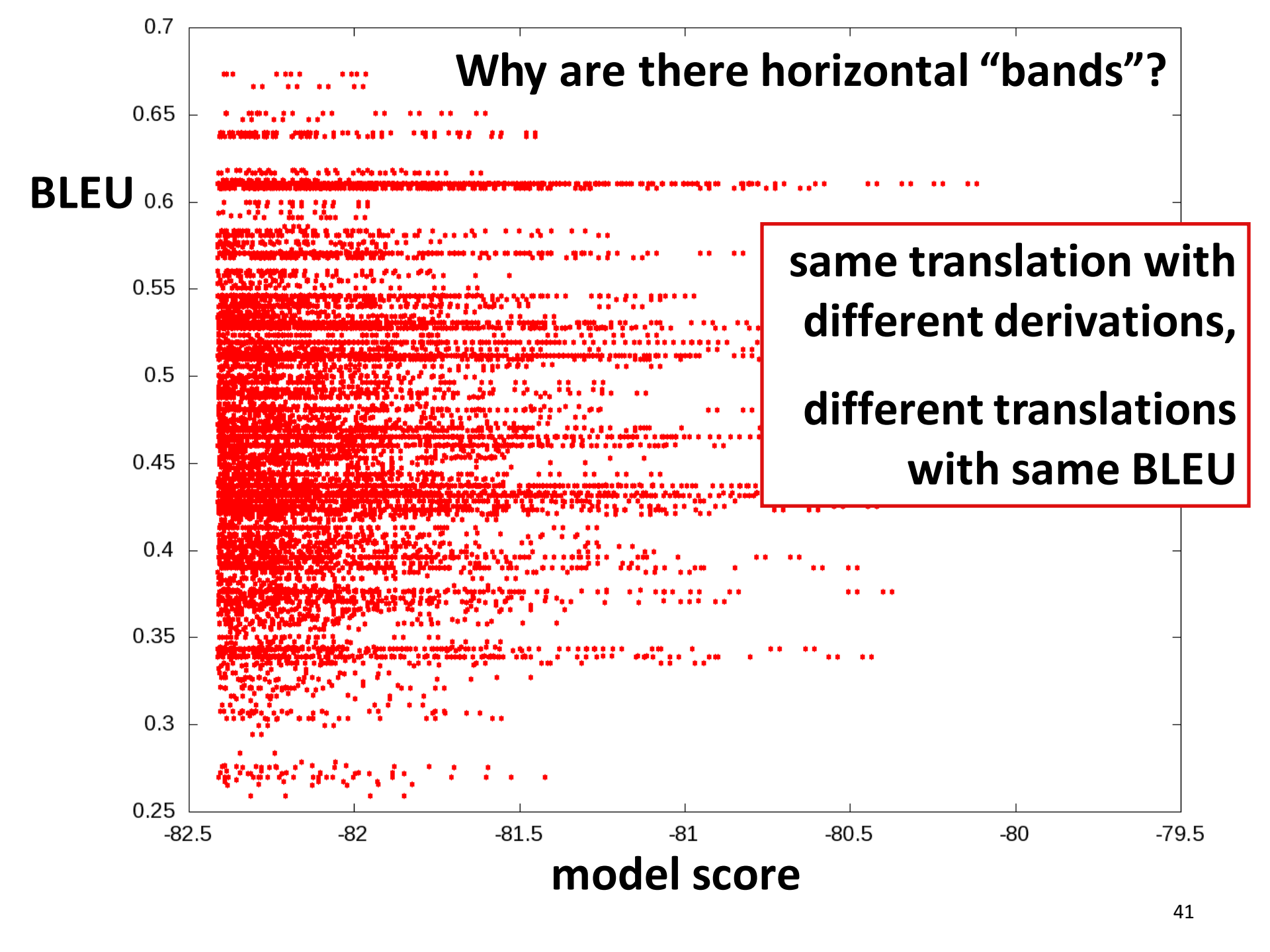
limited translation rules,
free translations,
noisy data


BLEU
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
-82.5
-82
-81.5
-81
-80.5
-80
-79.5
model score


each point is a translation for a single Arabic sentence
BLEU
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
-82.5
-82
-81.5
-81
-80.5
-80
-79.5
model score
10,000-best list from Arabic-English phrase-based Moses system, default Moses weights


BLEU
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
-82.5
-82
-81.5
-81
-80.5
-80
-79.5
model score
1-best:
46 BLEU


Why are there horizontal “bands”?
BLEU
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
-82.5
-82
-81.5
-81
-80.5
-80
-79.5
model score


Why are there horizontal “bands”?
BLEU
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
-82.5
-82
-81.5
-81
-80.5
-80
-79.5
model score
same translation with different derivations,
different translations with same BLEU

Roadmap

- We'll cover some of the most widely-used discriminative training algorithms for MT:
  - Minimum Error Rate Training*
  - Minimum Bayes Risk
  - Pairwise Ranking Optimization*
  - Batch MIRA*

* implemented in Moses!

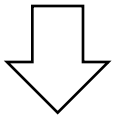
2003:
MERT

Minimum Error Rate Training in Statistical Machine Translation

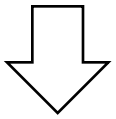
Franz Josef Och
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
och@isi.edu

Minimum Error Rate Training (MERT)

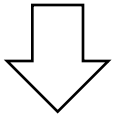
$$\min_{\boldsymbol{\theta}} \ \text{error}(\text{predicted translations})$$



$$\min_{\boldsymbol{\theta}} \ -\text{BLEU}(\text{references}, \text{predicted translations})$$



$$\max_{\boldsymbol{\theta}} \ \text{BLEU}(\text{references}, \text{predicted translations})$$



$$\max_{\boldsymbol{\theta}} \ \text{BLEU}(\text{references}, \{\text{translate}(\boldsymbol{x}_i, \boldsymbol{\theta})\}_{i=1}^{N})$$

$$\text{translate}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{\boldsymbol{y}}{\text{argmax}} \ \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$$

Minimum Error Rate Training (MERT)

$$\max_{\boldsymbol{\theta}} \ \mathrm{BLEU}(\mathrm{references}, \{\mathrm{translate}(\boldsymbol{x}_i, \boldsymbol{\theta})\}_{i=1}^{N})$$

optimizing this objective is intractable in general – how can we do it?

generate k-best lists of translations, approximately optimize on k-best lists, repeat with new parameters (pool k-best lists across iterates)

MERT

given a k-best list,

randomly choose a feature (e.g., reverse translation model) and vary a step size δ :

$$\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) + \delta \log P(\boldsymbol{y} \mid \boldsymbol{x})$$

find value of δ that maximizes BLEU

MERT

more generally,

randomly choose a "search direction" ψ and

vary a scalar multiplier δ:

$$\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) + \delta \, \boldsymbol{\psi}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$$

MERT

more generally,

randomly choose a "search direction" ψ and

vary a scalar multiplier δ:

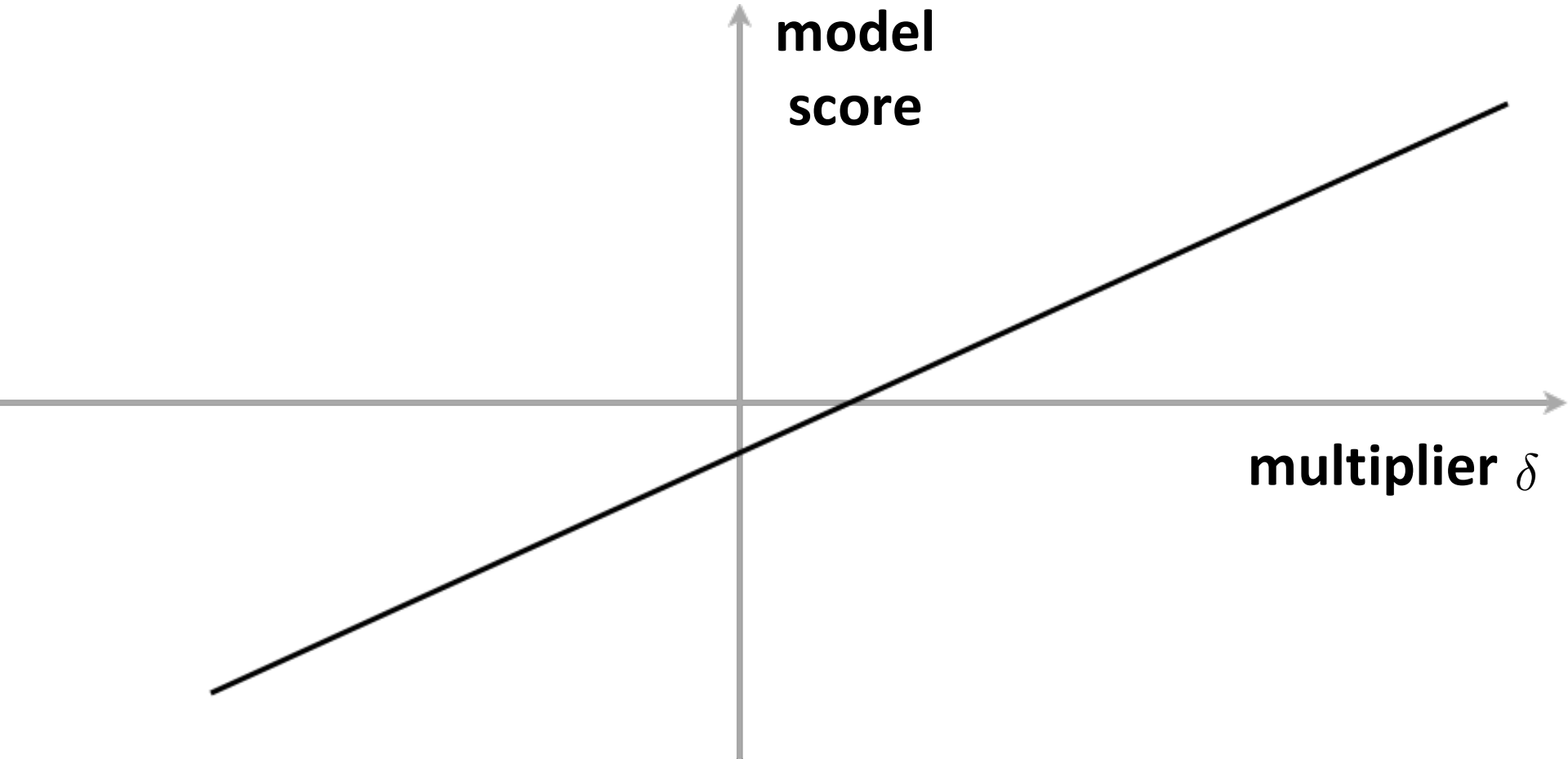
$$\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) + \delta\, \boldsymbol{\psi}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$$

single-feature special case: let ψ be a "one-hot" vector (all zeroes except a single 1)

for the reverse translation model feature one-hot vector ψ, this becomes:

$$\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) + \delta \log P(\boldsymbol{y} \mid \boldsymbol{x})$$

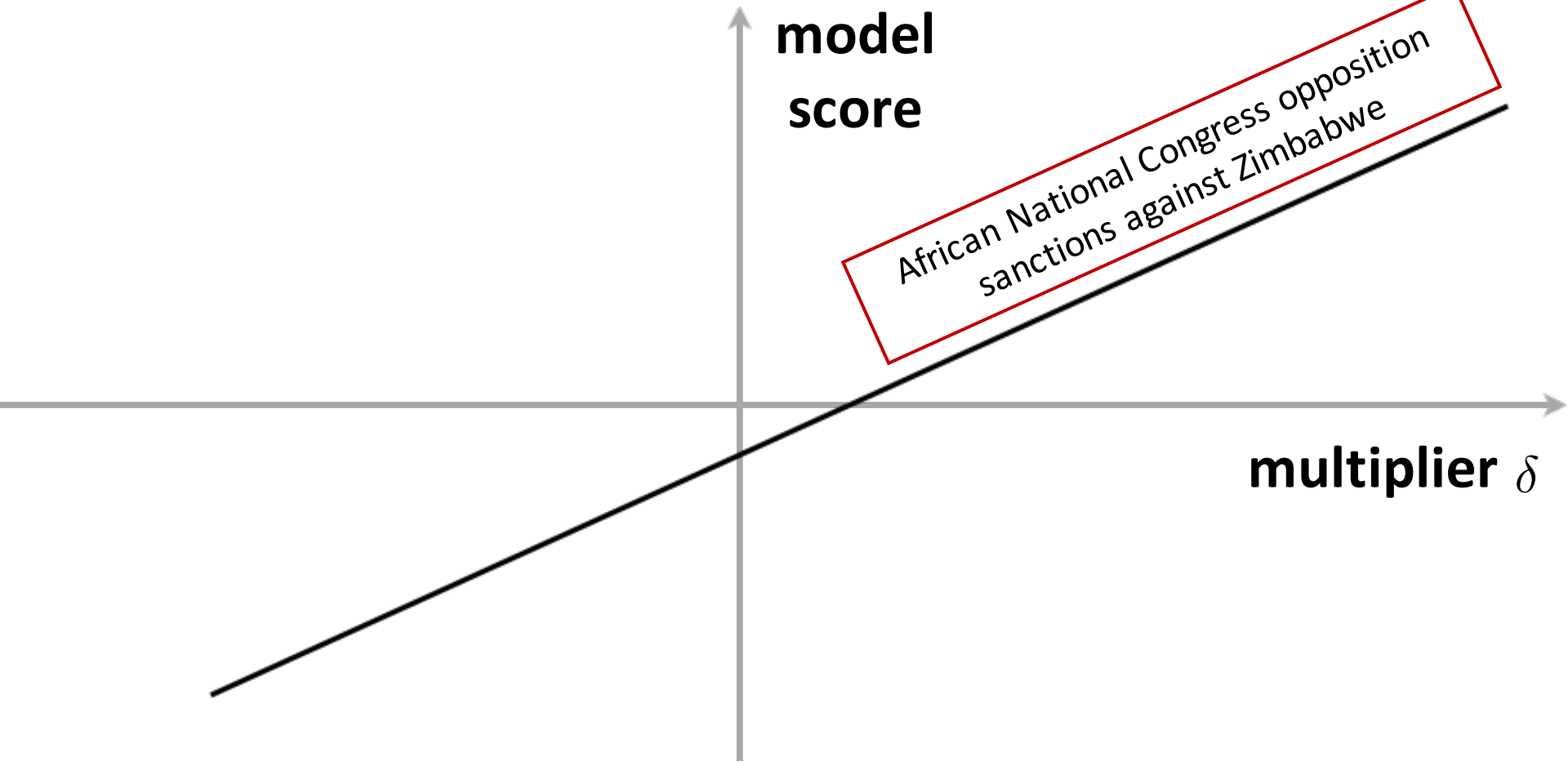
MERT



credit: Chris Dyer

MERT

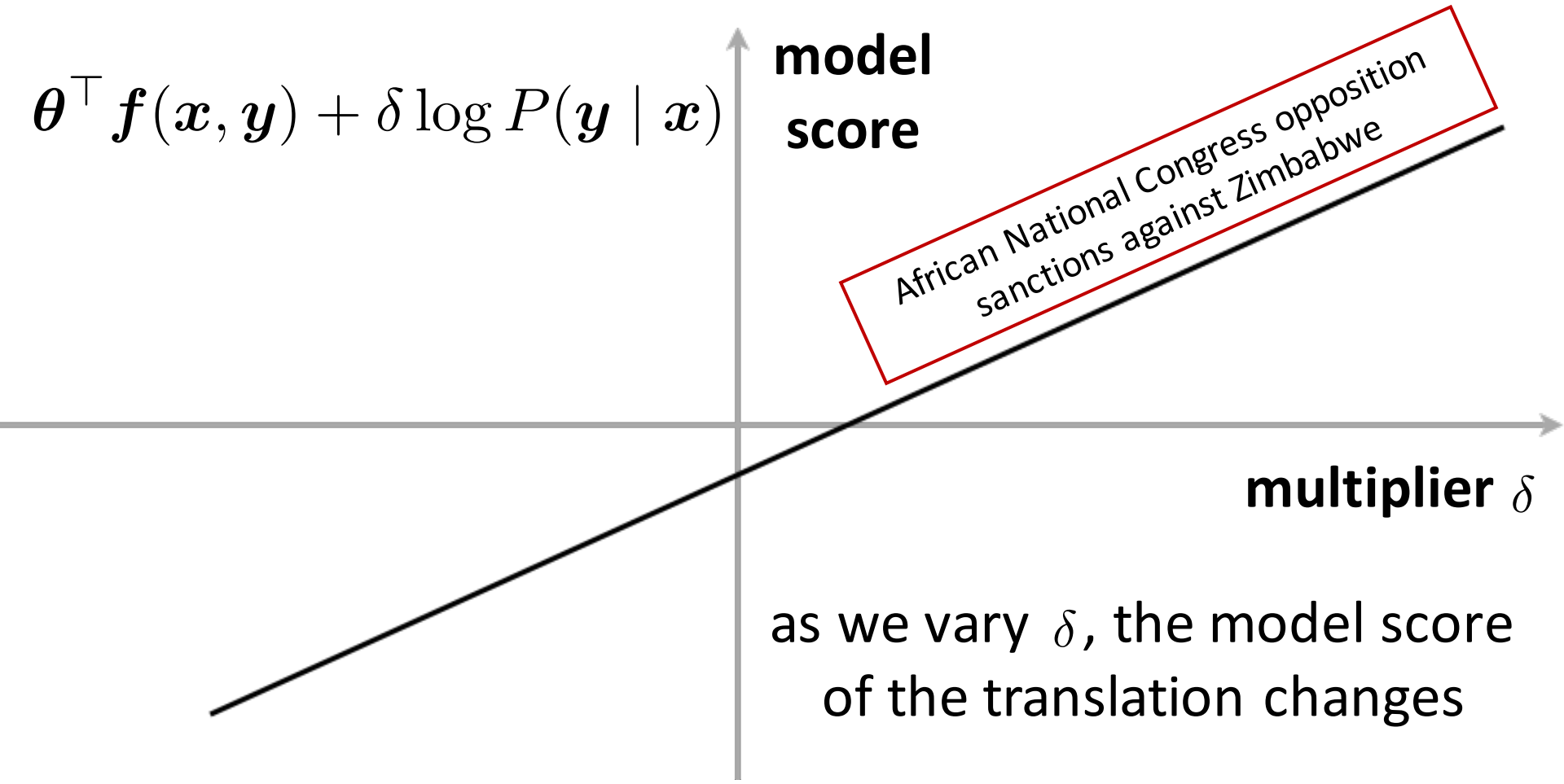
each line is a translation from the k-best list:



credit: Chris Dyer

MERT

$$\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) + \delta \log P(\boldsymbol{y} \mid \boldsymbol{x})$$

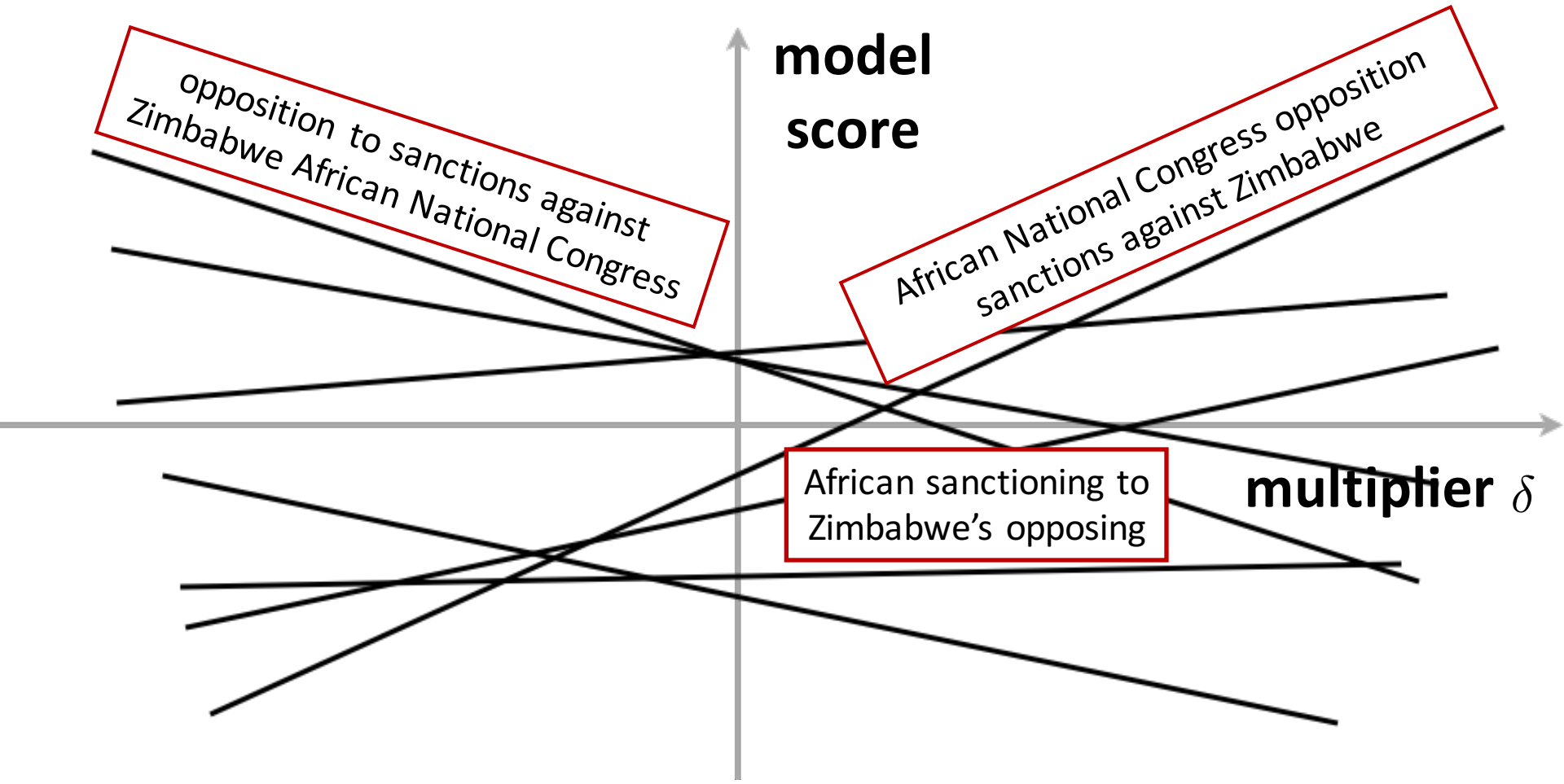


as we vary δ, the model score of the translation changes

credit: Chris Dyer

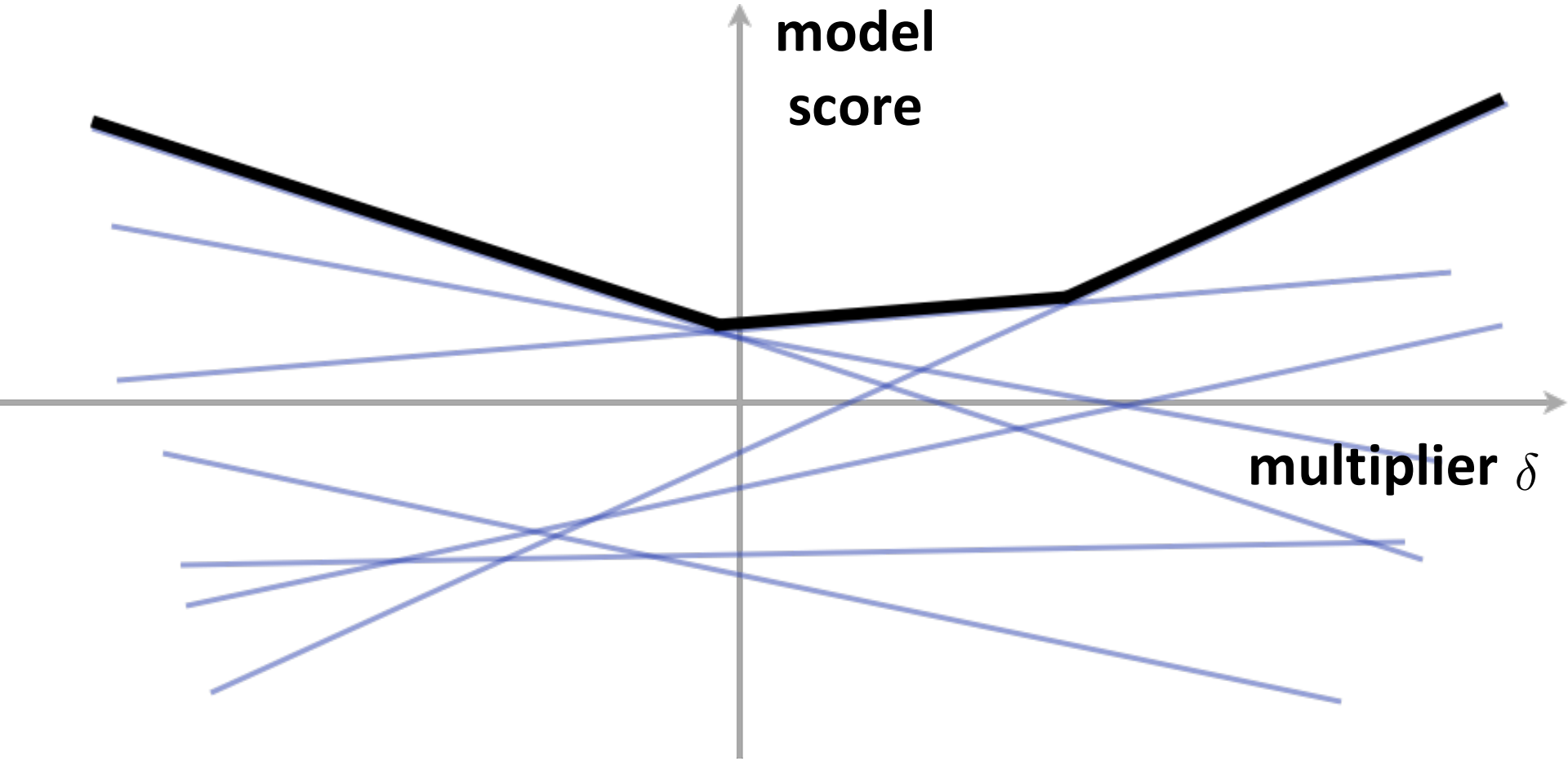
MERT

each line is a translation from the k-best list:



credit: Chris Dyer

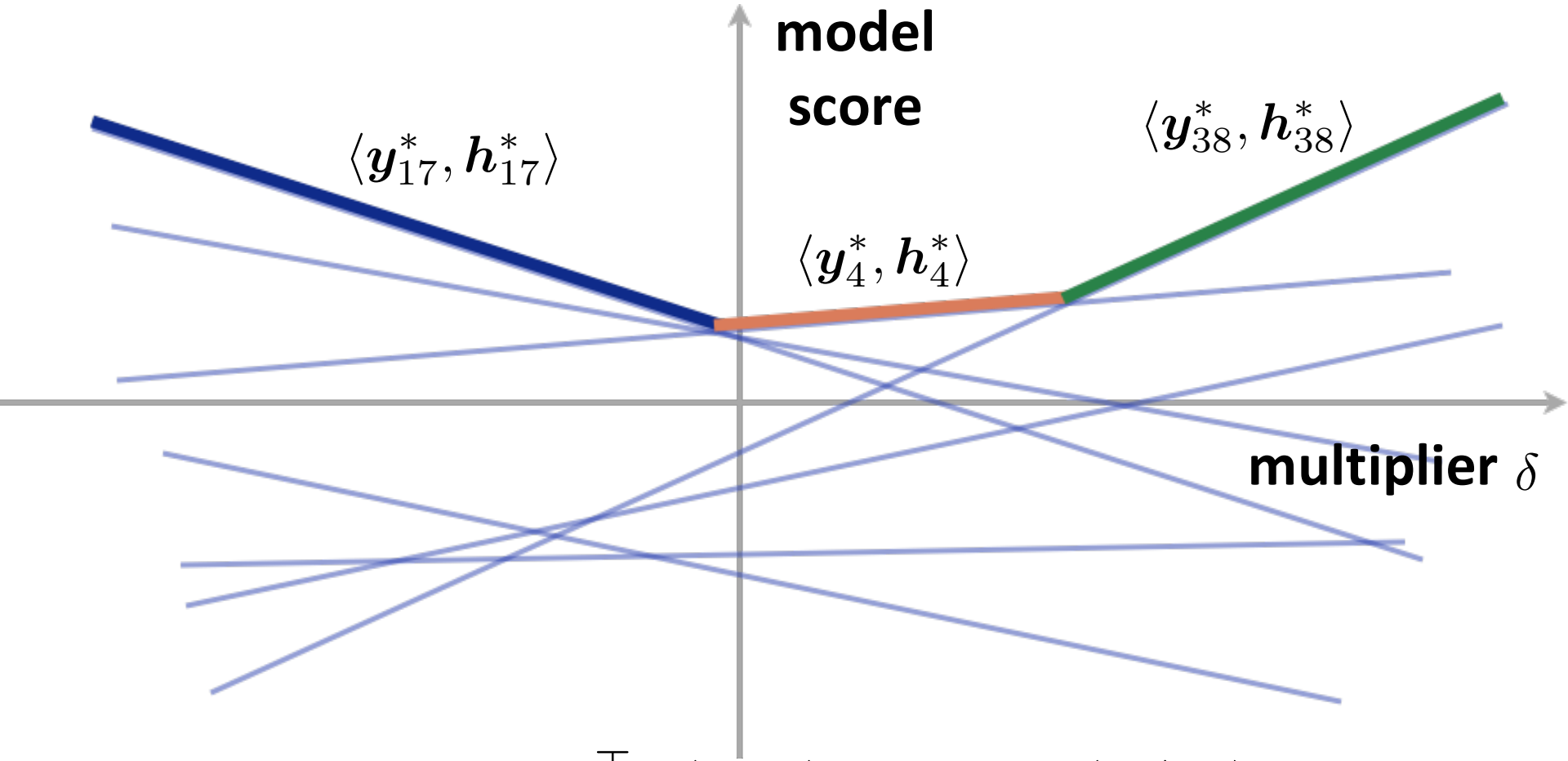
MERT



$$\max_{\boldsymbol{y}} \quad \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) + \delta \log P(\boldsymbol{y} \mid \boldsymbol{x})$$

credit: Chris Dyer

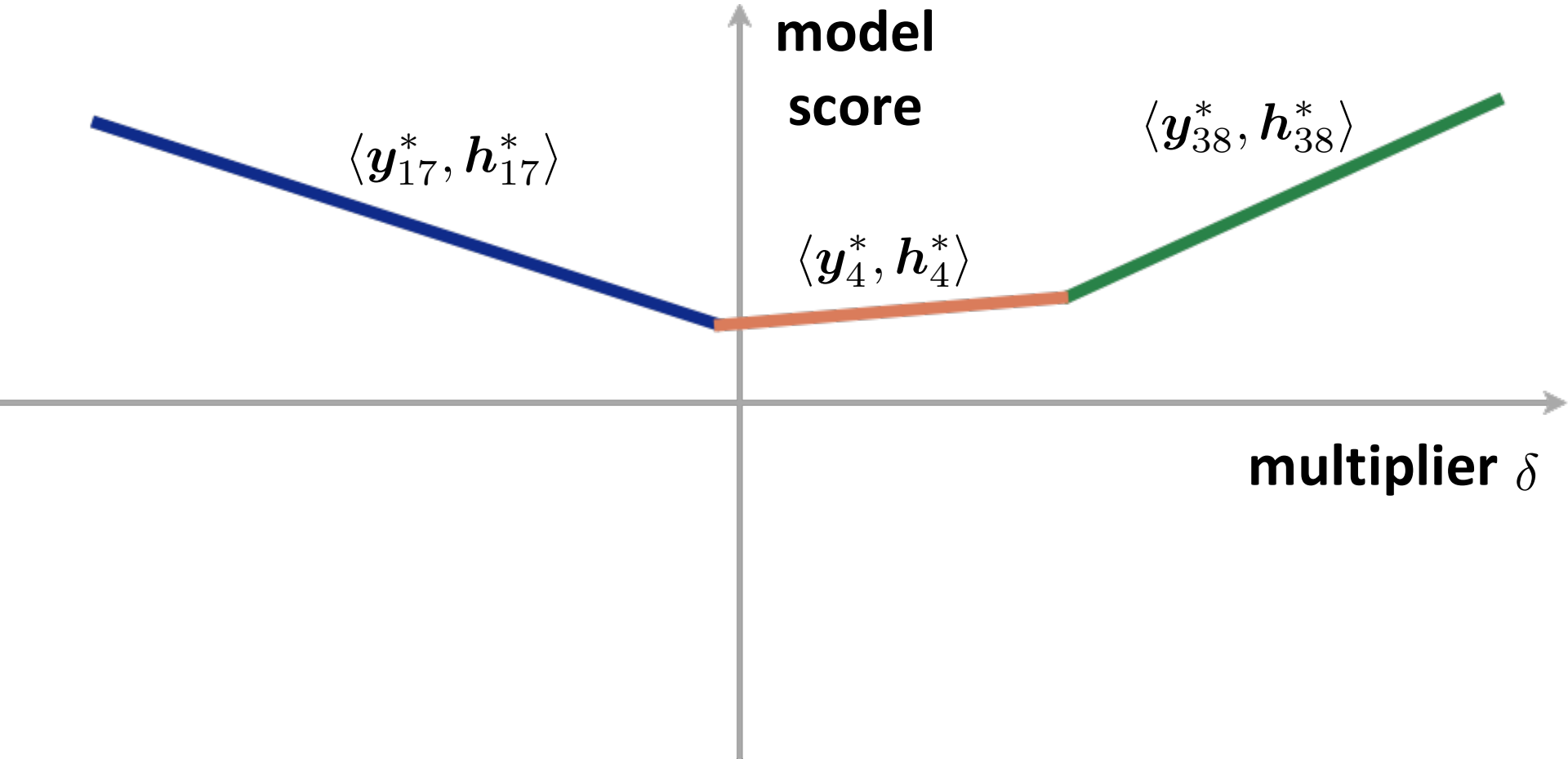
MERT



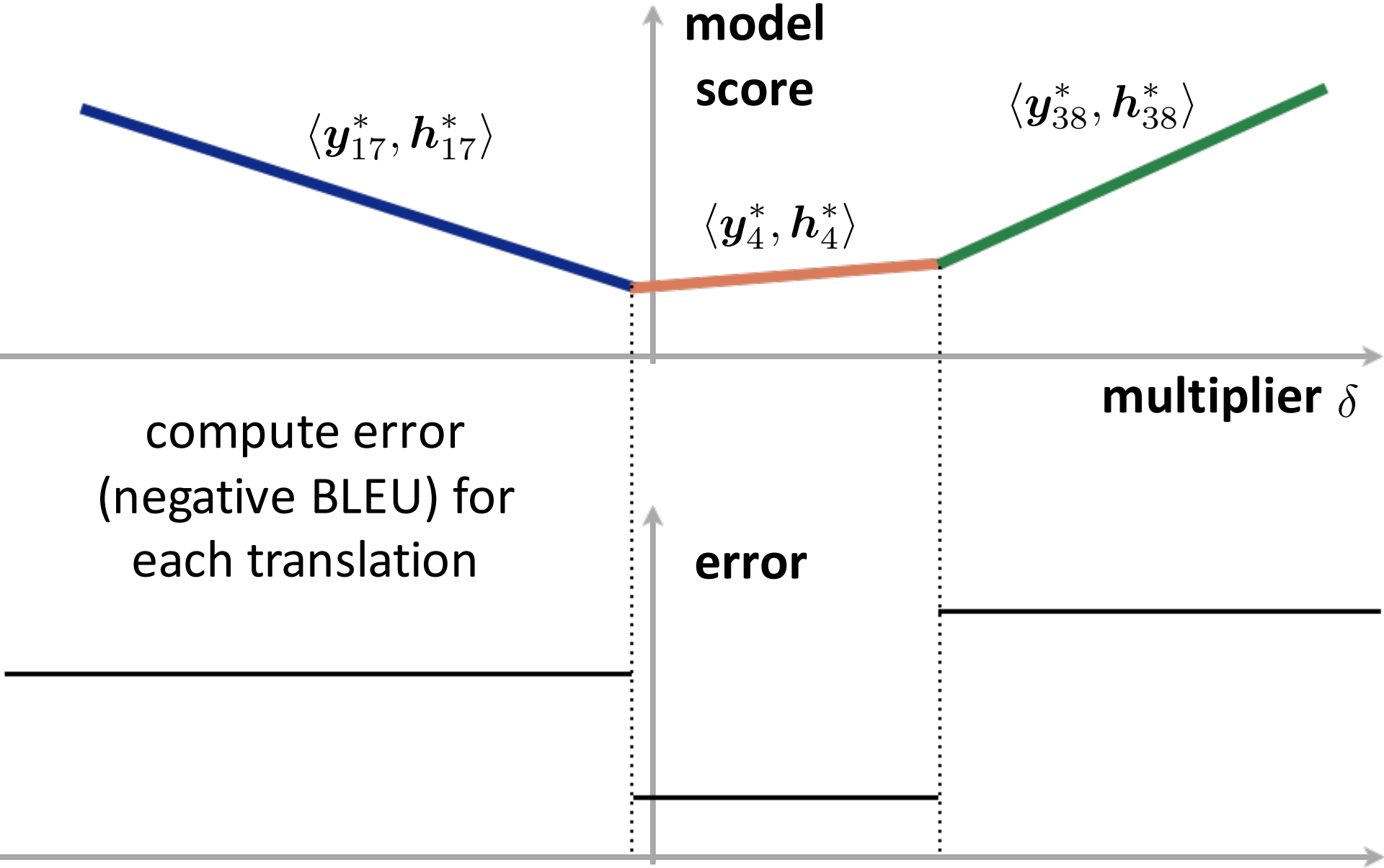
$$\underset{\boldsymbol{y}}{\operatorname{argmax}} \; \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) + \delta \log P(\boldsymbol{y} \mid \boldsymbol{x})$$

credit: Chris Dyer

MERT

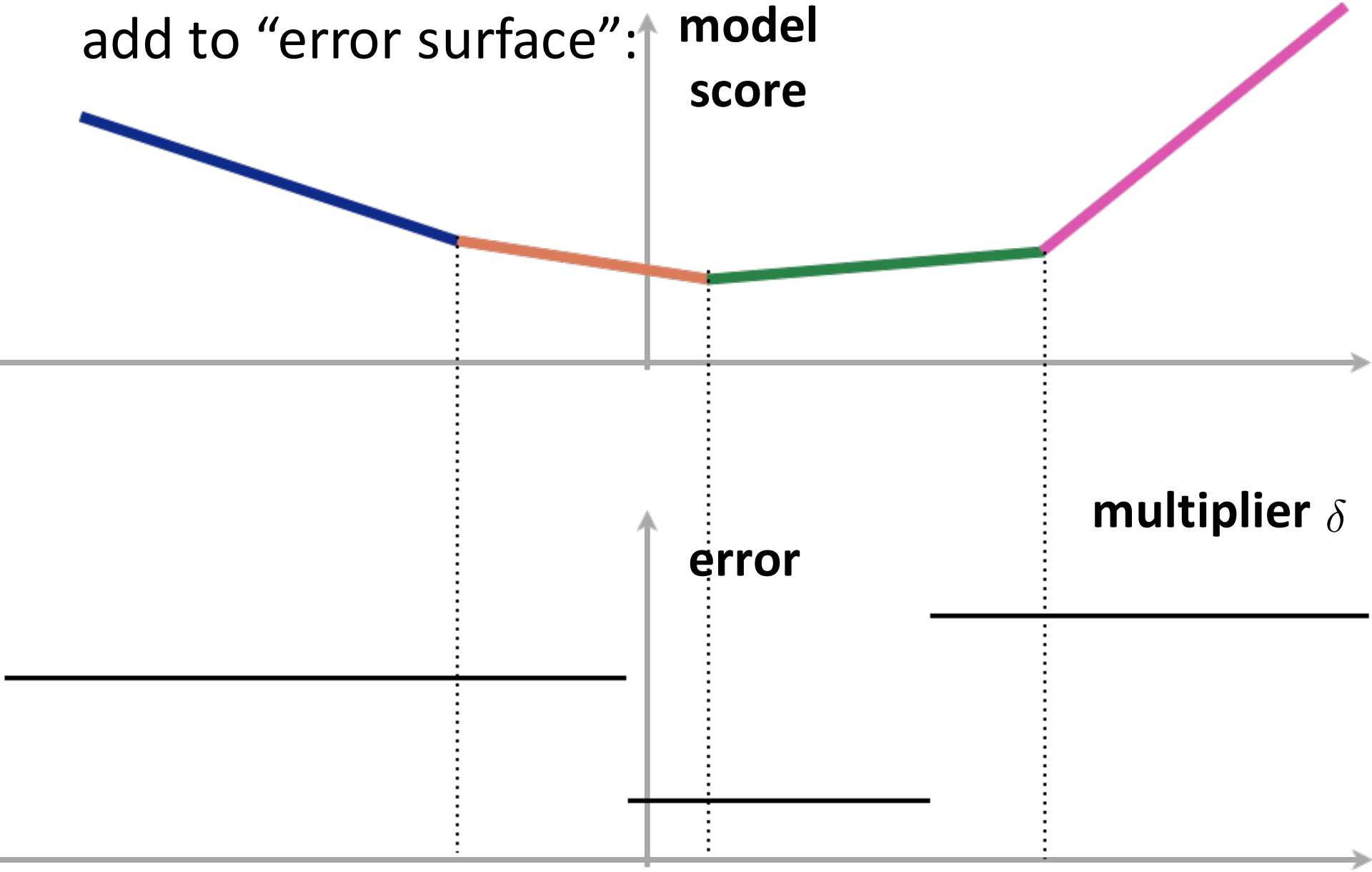


credit: Chris Dyer



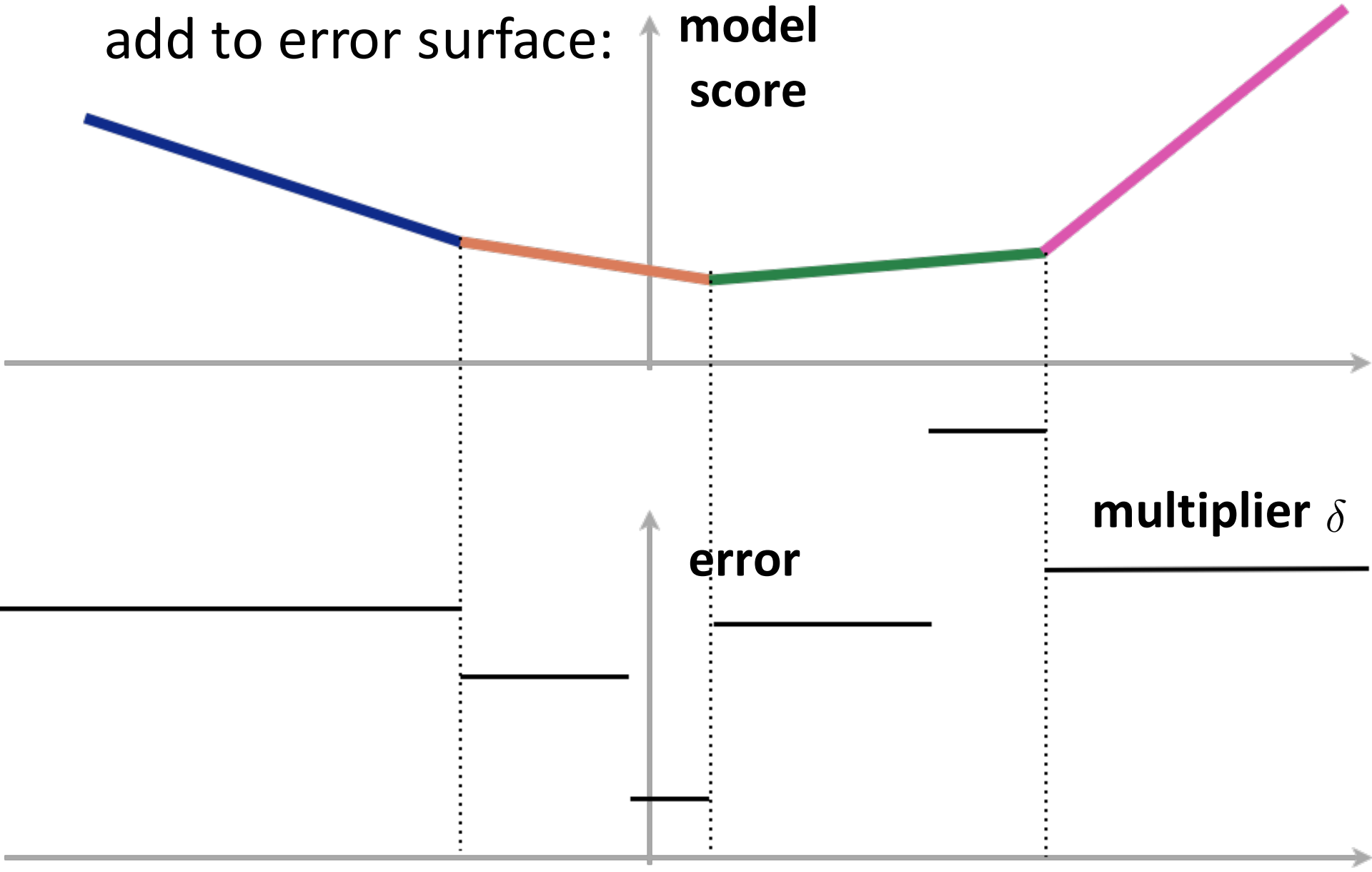
credit: Chris Dyer

repeat for each sentence,
add to “error surface”:

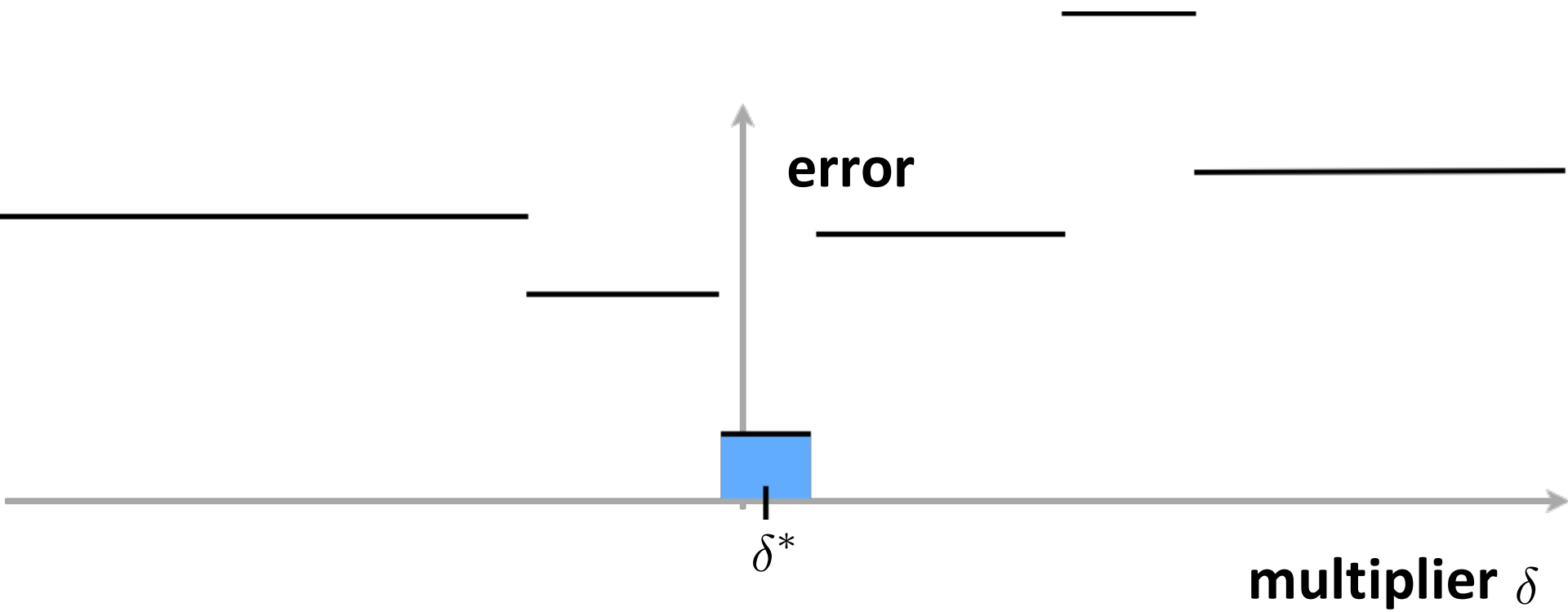


credit: Chris Dyer

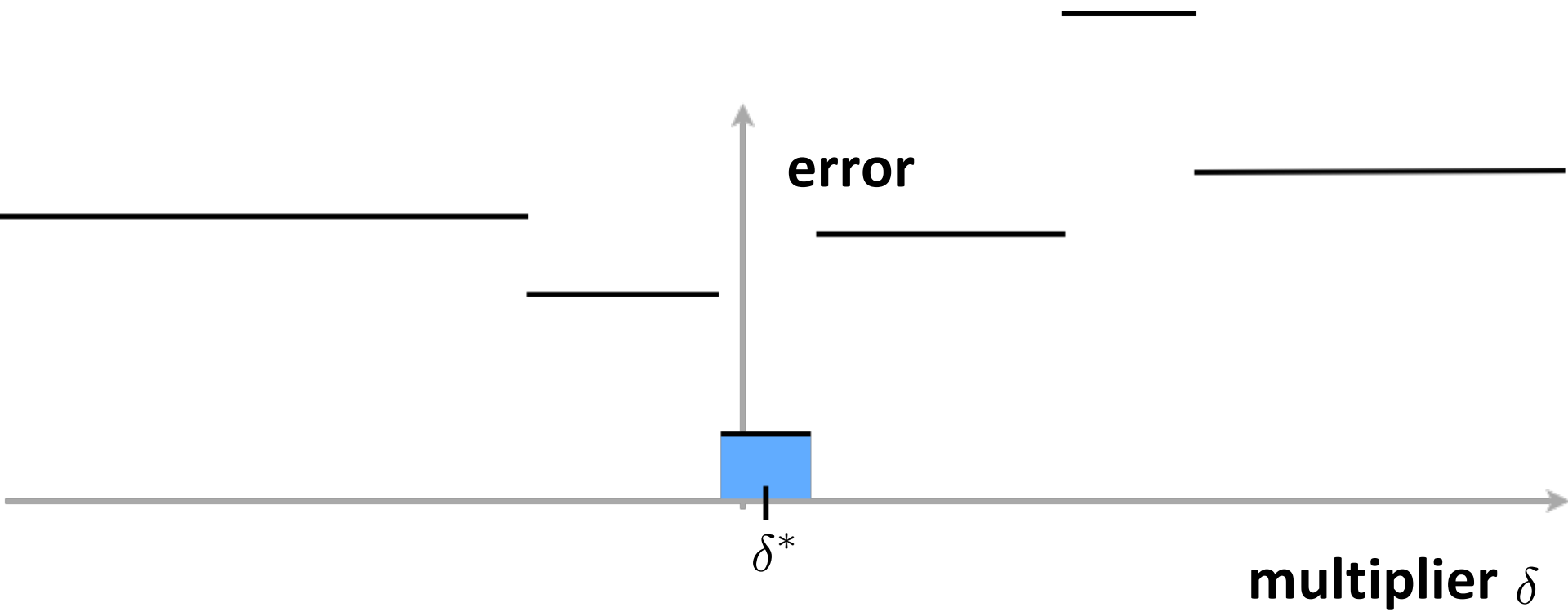
repeat for each sentence,
add to error surface:



credit: Chris Dyer



credit: Chris Dyer



$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta} + \delta^* \boldsymbol{\psi}$$

credit: Chris Dyer

Error Surface (Smith and Eisner, 2006)

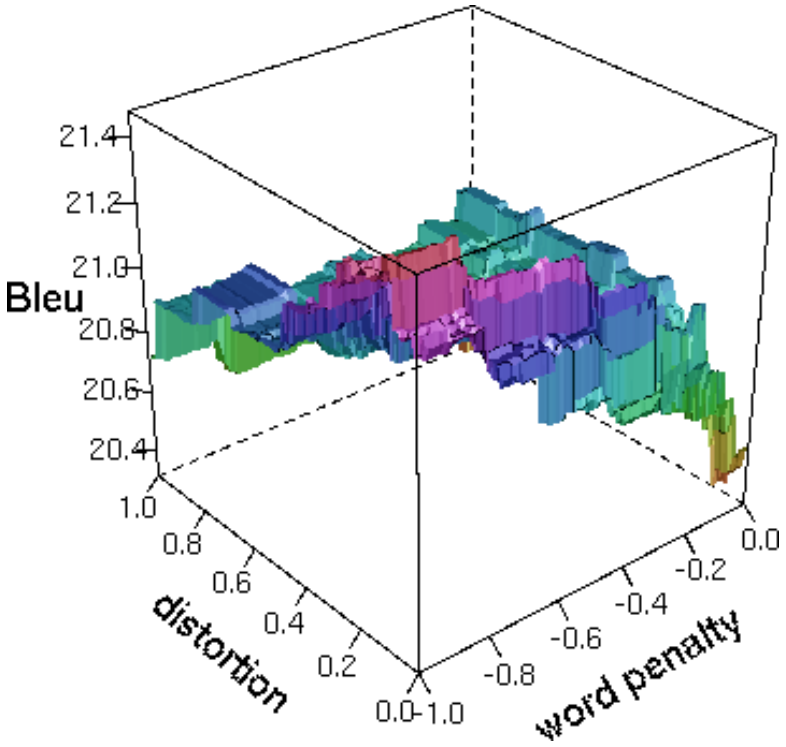
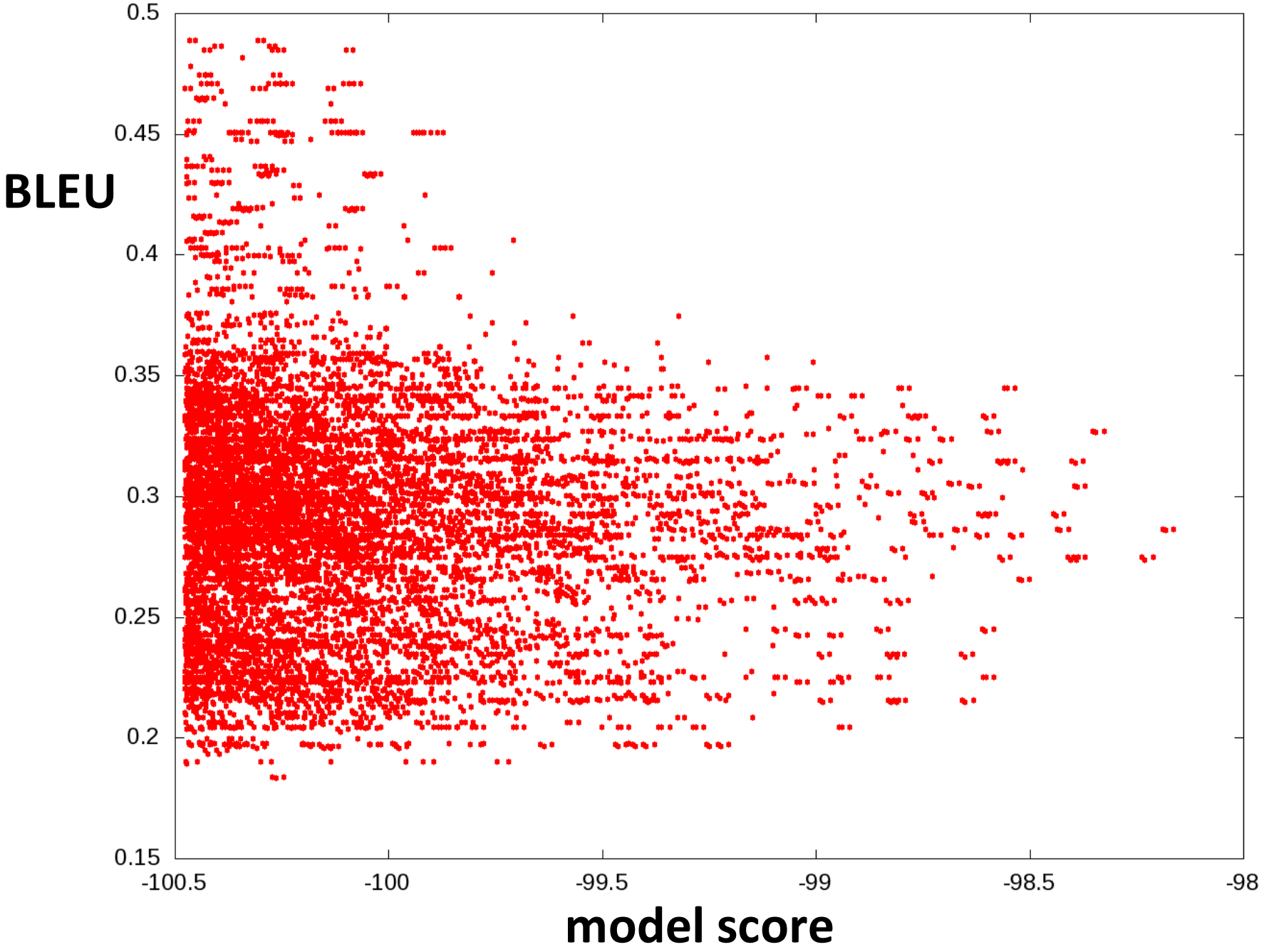
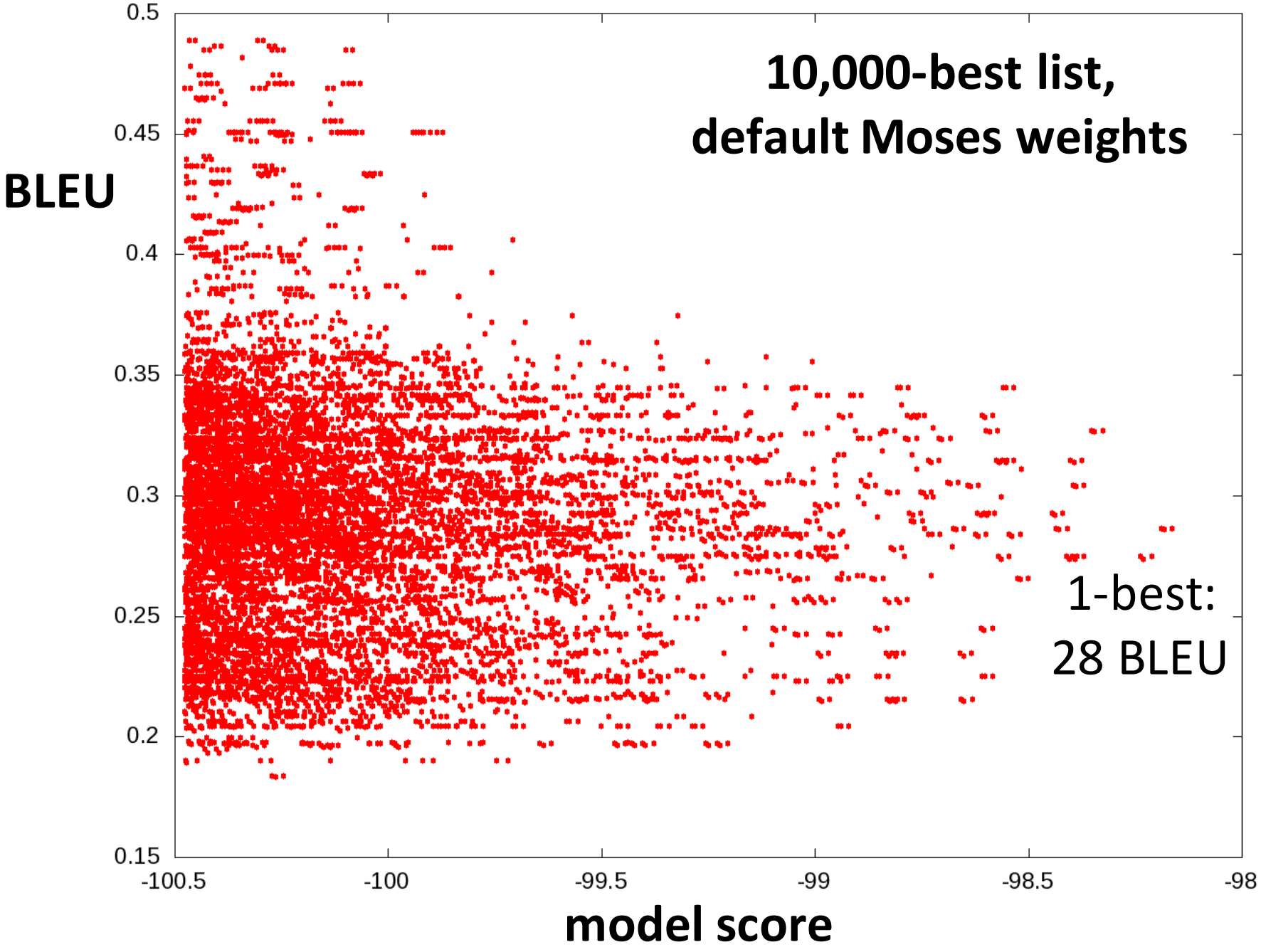
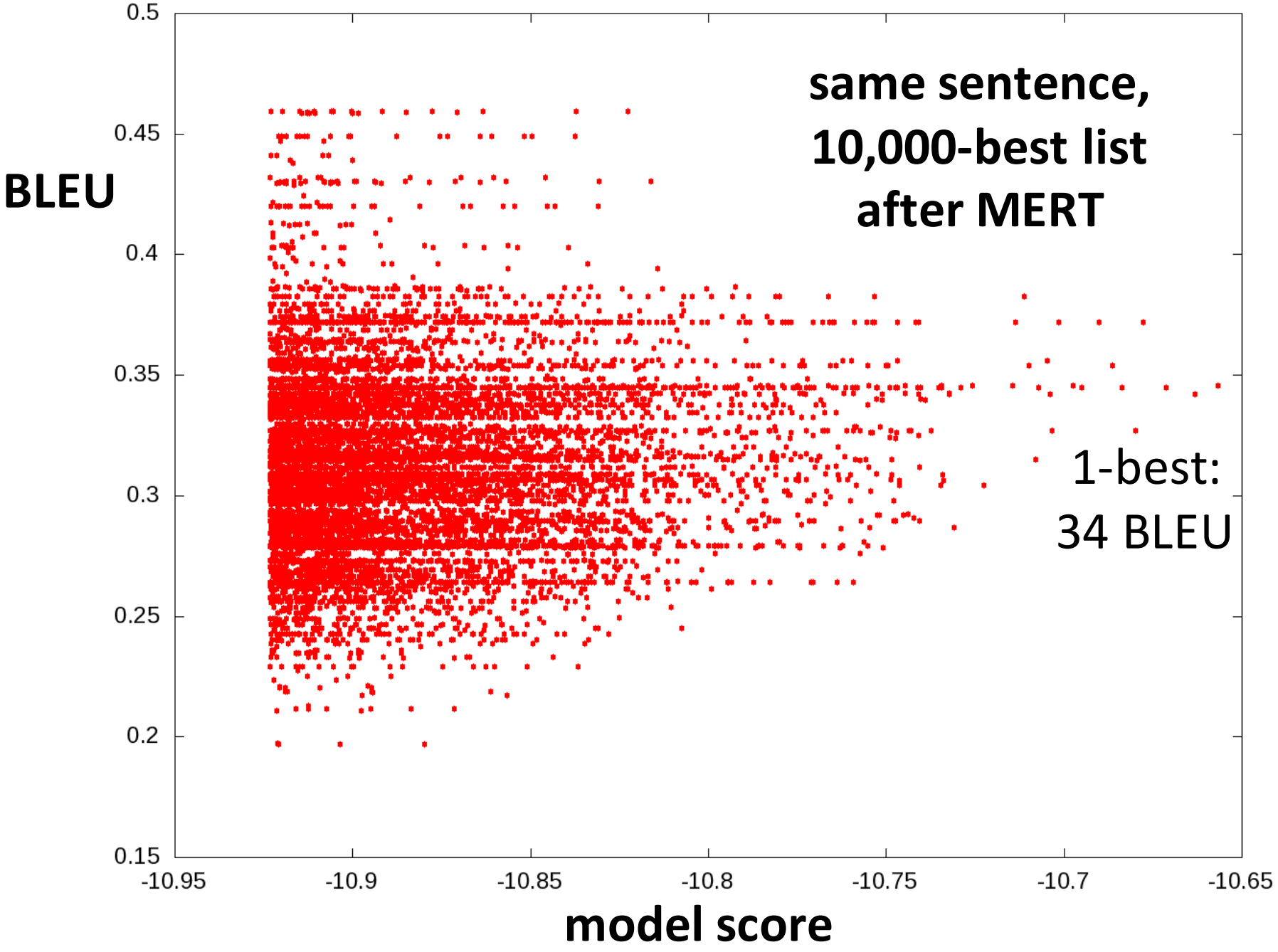
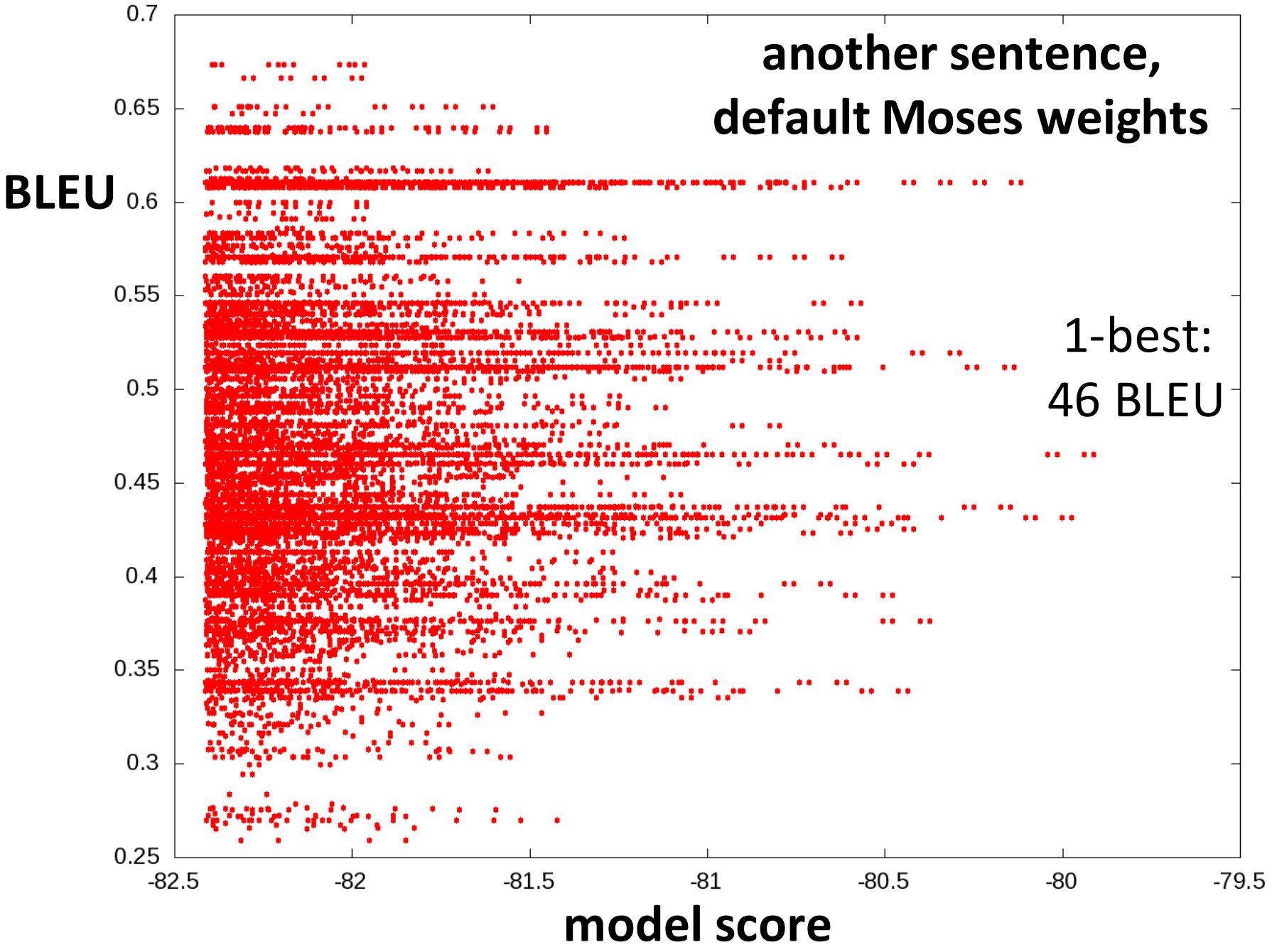


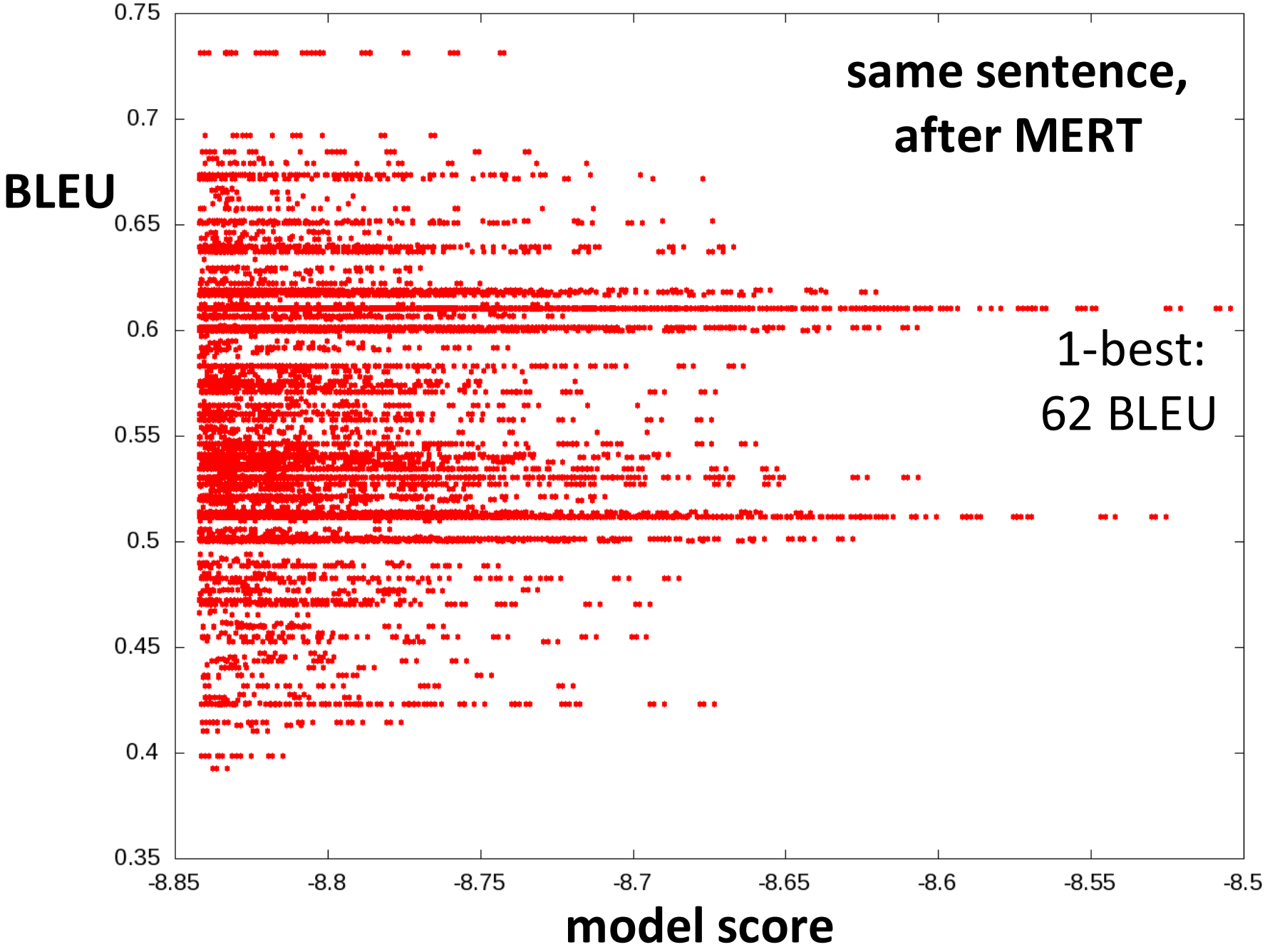
Figure 1: The loss surface for a machine translation system: while other parameters are held constant, we vary the weights on the distortion and word penalty features. Note the piecewise constant regions with several local maxima.

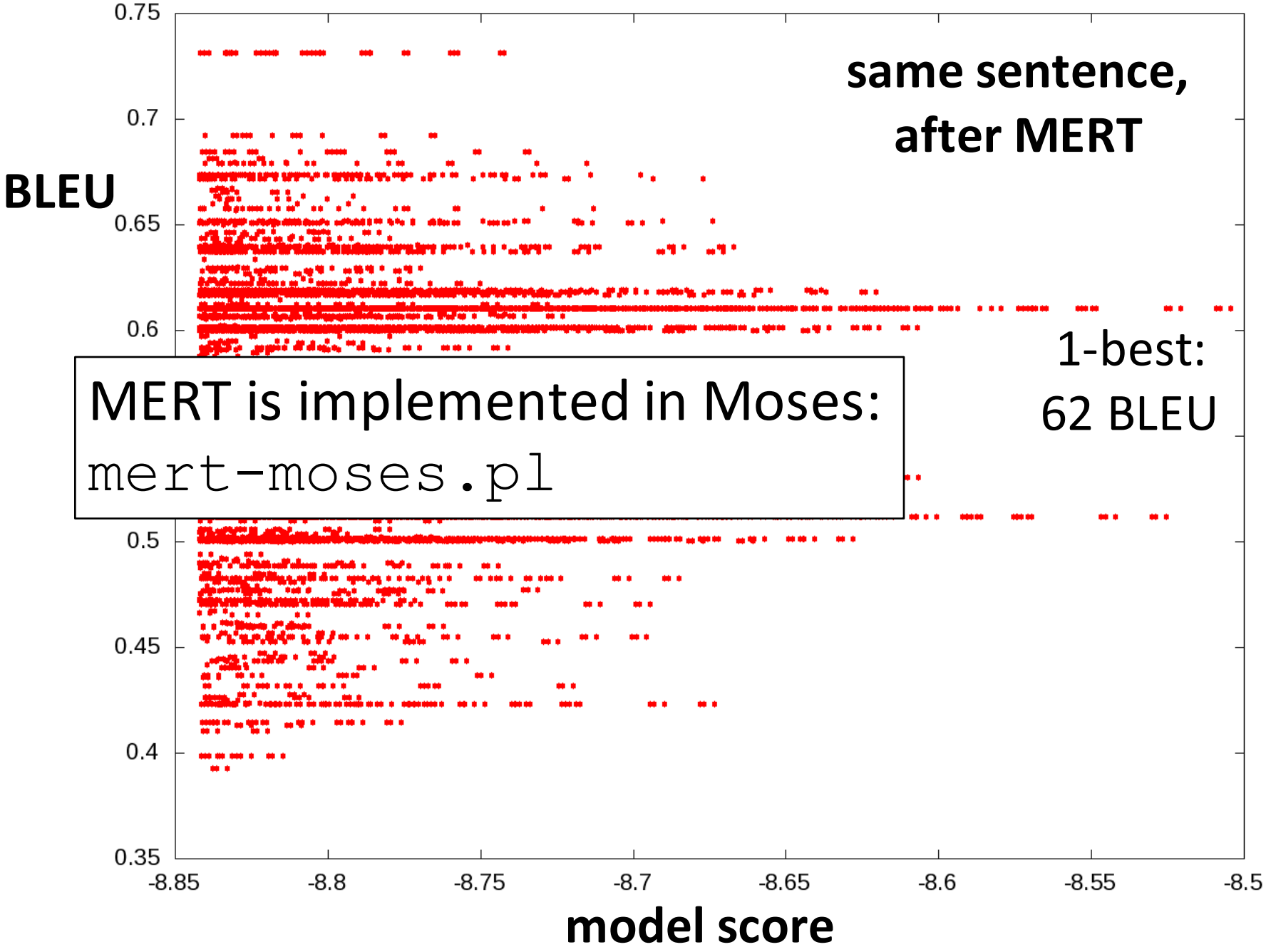

BLEU
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
-100.5
-100
-99.5
-99
-98.5
-98
model score


10,000-best list,
default Moses weights
BLEU
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
-100.5
-100
-99.5
-99
-98.5
-98
model score
1-best:
28 BLEU


same sentence,
10,000-best list
after MERT
1-best:
34 BLEU
BLEU
model score
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
-10.95
-10.9
-10.85
-10.8
-10.75
-10.7
-10.65


another sentence,
default Moses weights
BLEU
1-best:
46 BLEU
model score
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
-82.5
-82
-81.5
-81
-80.5
-80
-79.5


same sentence,
after MERT
1-best:
62 BLEU
BLEU
model score
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
-8.85
-8.8
-8.75
-8.7
-8.65
-8.6
-8.55
-8.5


same sentence,
after MERT
BLEU
1-best:
62 BLEU
MERT is implemented in Moses:
mert-moses.pl
0.75
0.7
0.65
0.6
0.5
0.45
0.4
0.35
-8.85
-8.8
-8.75
-8.7
-8.65
-8.6
-8.55
-8.5
model score

$$\max_{\boldsymbol{\theta}} \ \mathrm{BLEU}(\mathrm{references}, \{\mathrm{translate}(\boldsymbol{x}_i, \boldsymbol{\theta})\}_{i=1}^{N})$$

Problems with the MERT objective function?

- Discontinuous & non-convex → optimization relies on randomized search
- No regularization → frequently overfits to tuning set

As a result, MERT is only effective for very small models (<40 parameters)

Smoothing the Objective

$$\max_{\boldsymbol{\theta}} \ \text{BLEU}\left(\text{references}, \{\operatorname{argmax}_{\boldsymbol{y}} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\}_{i=1}^{N}\right)$$

first, convert to sentence-level objective:

$$\max_{\boldsymbol{\theta}} \ \sum_{i=1}^{N} \text{BLEU}^{+1}\left(\text{references}^{(i)}, \operatorname{argmax}_{\boldsymbol{y}} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\right)$$

then, smooth using expectation under log-linear distribution:

$$\max_{\boldsymbol{\theta}} \ \sum_{i=1}^{N} \sum_{\boldsymbol{y}} \text{BLEU}^{+1}\left(\text{references}^{(i)}, \boldsymbol{y}\right) \frac{\exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\}^{\alpha}}{\sum_{\boldsymbol{y}'} \exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}')\}^{\alpha}}$$

alpha = “smoothness factor”

Smoothing the Objective

$$\max_{\boldsymbol{\theta}} \ \text{BLEU}\left(\text{references}, \{\operatorname*{argmax}_{\boldsymbol{y}} \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\}_{i=1}^{N}\right)$$

first, convert to sentence-level objective:

$$\max_{\boldsymbol{\theta}} \ \sum_{i=1}^{N} \text{BLE} \qquad \text{x}\, \boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\Big)$$

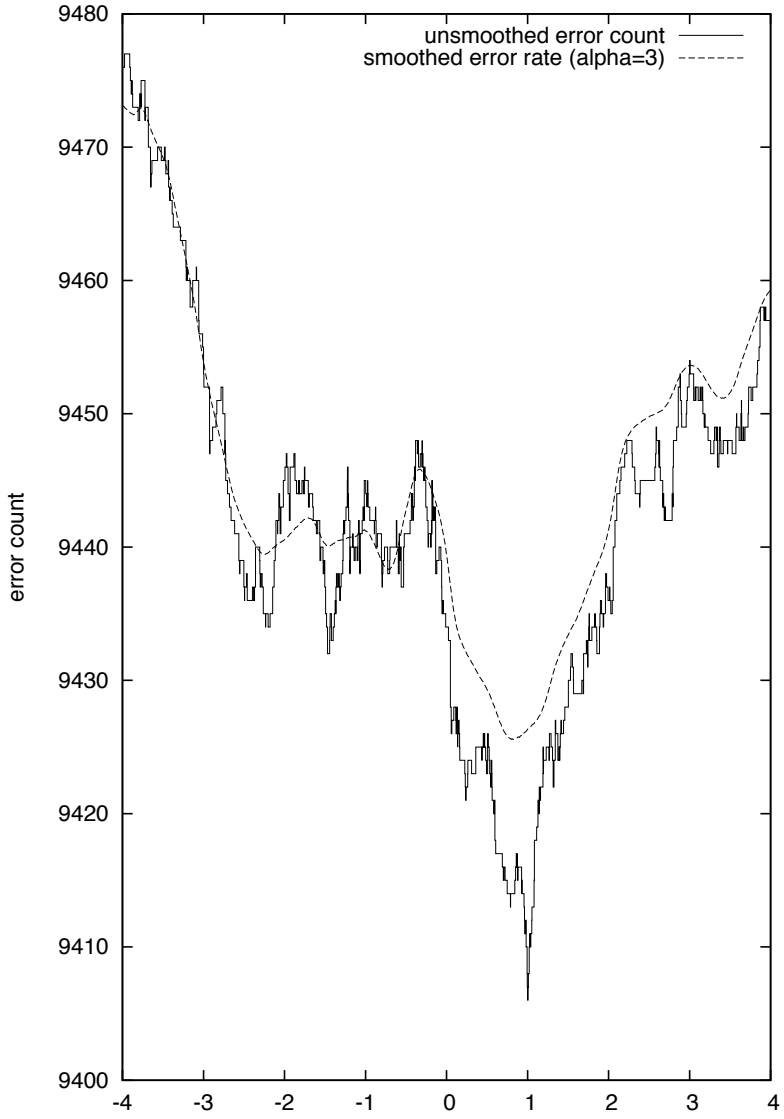
th on under
lo

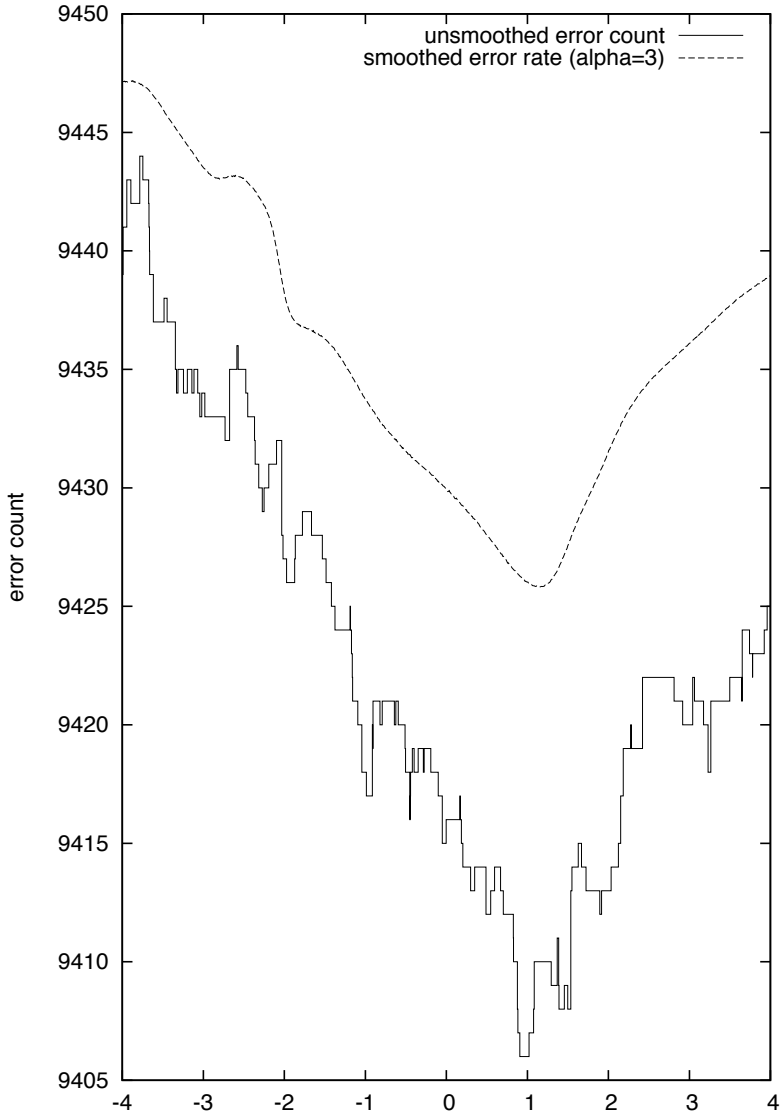
This is often called "risk" or "Bayes risk"

$$\max_{\boldsymbol{\theta}} \ \sum_{i=1}^{N} \sum_{\boldsymbol{y}} \text{BLEU}^{+1}\left(\text{references}^{(i)}, \boldsymbol{y}\right) \frac{\exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y})\}^{\alpha}}{\sum_{\boldsymbol{y}'} \exp\{\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}')\}^{\alpha}}$$

alpha = "smoothness factor"

Smoothing Error Surfaces (Och, 2003)





Many other researchers tried to improve MERT:

Regularization and Search for MERT (Cer et al., 2008)

Random Restarts in MERT for MT (Moore & Quirk, 2008)

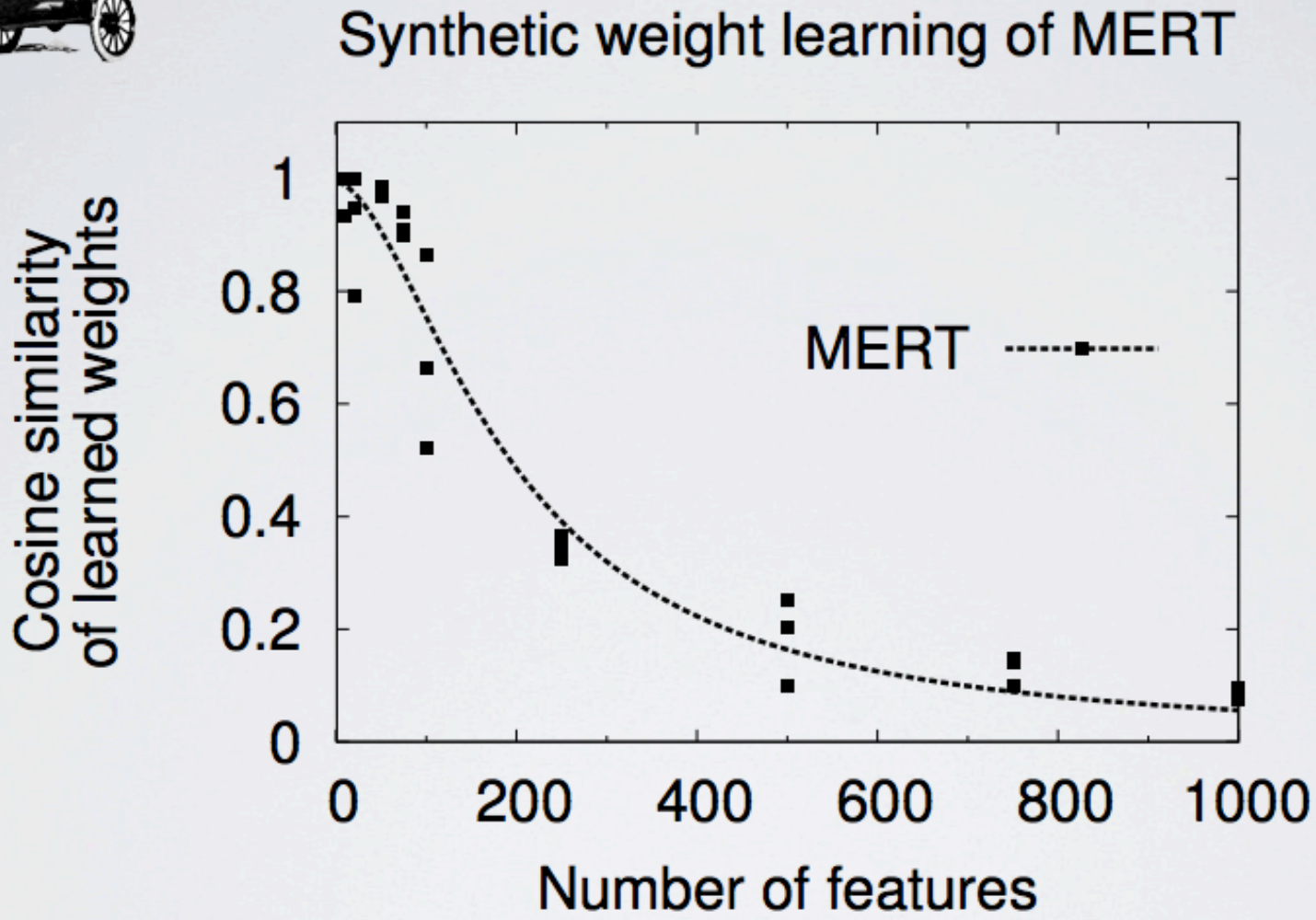
Stabilizing MERT (Foster & Kuhn, 2009)

Issues remain:

Better Hypothesis Testing for Statistical MT: Controlling for Optimizer Instability (Clark et al., 2011)

They suggest running MERT 3-5 times due to its instability

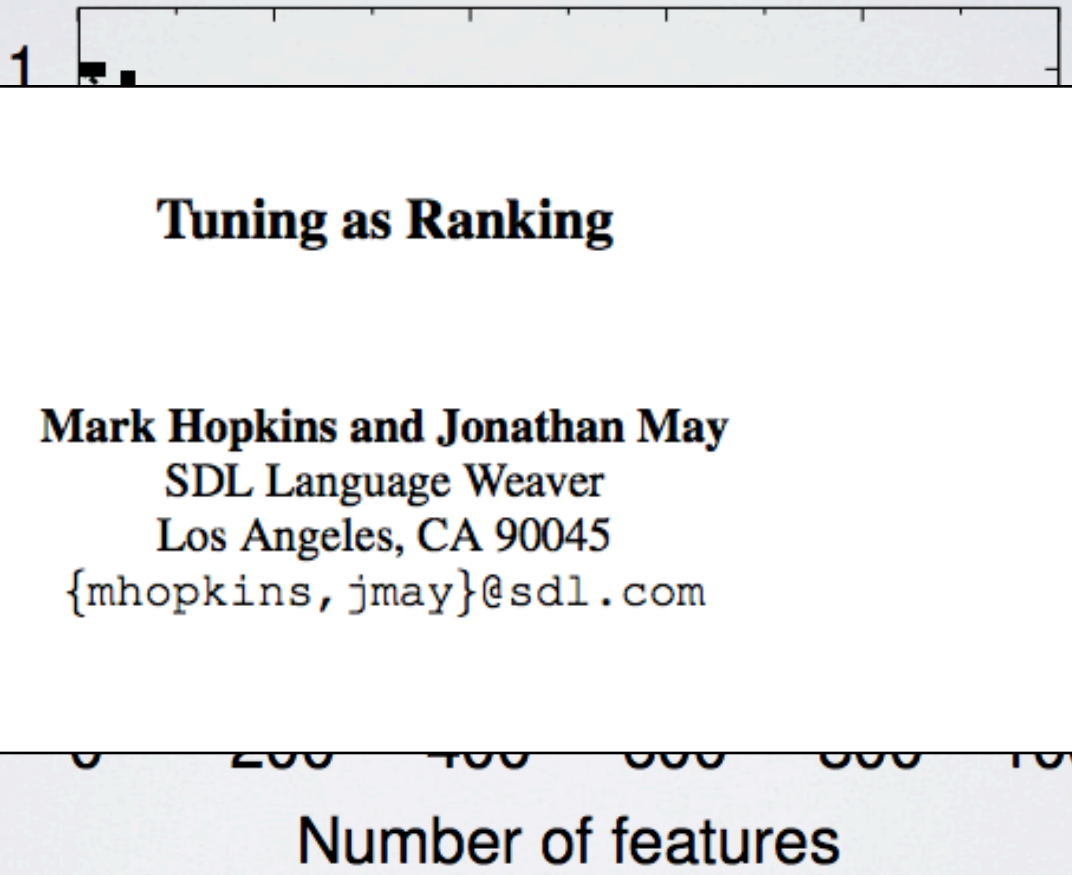
MERT ***doesn't scale***



The synthetic experiment in ideal conditions validates what has long been accepted as truth

MERT *doesn't scale*

Synthetic weight learning of MERT



Tuning as Ranking

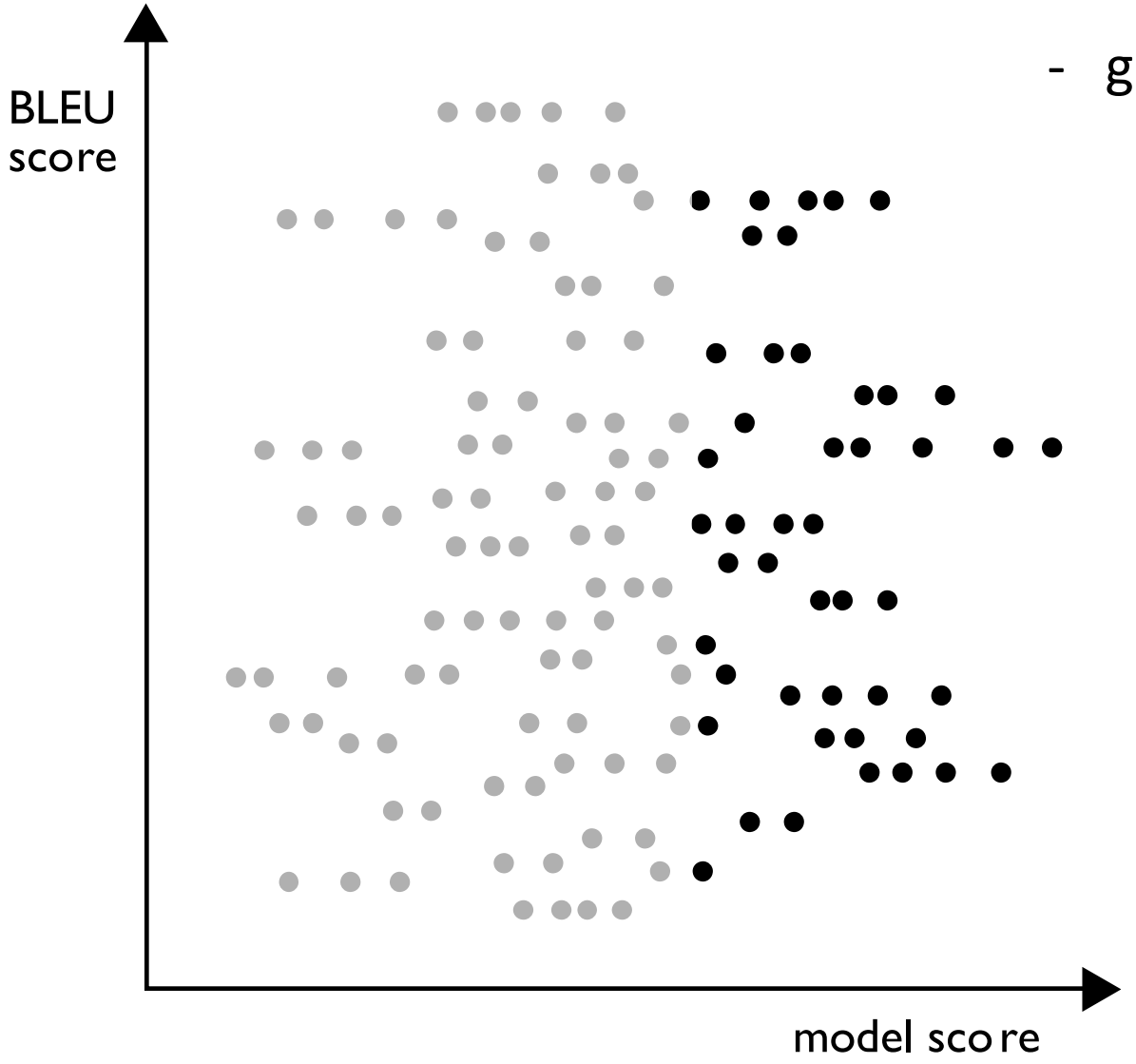
Mark Hopkins and Jonathan May
SDL Language Weaver
Los Angeles, CA 90045
{mhopkins, jmay}@sdl.com

The synthetic experiment in ideal conditions validates what has long been accepted as truth

Pairwise Ranking Optimization

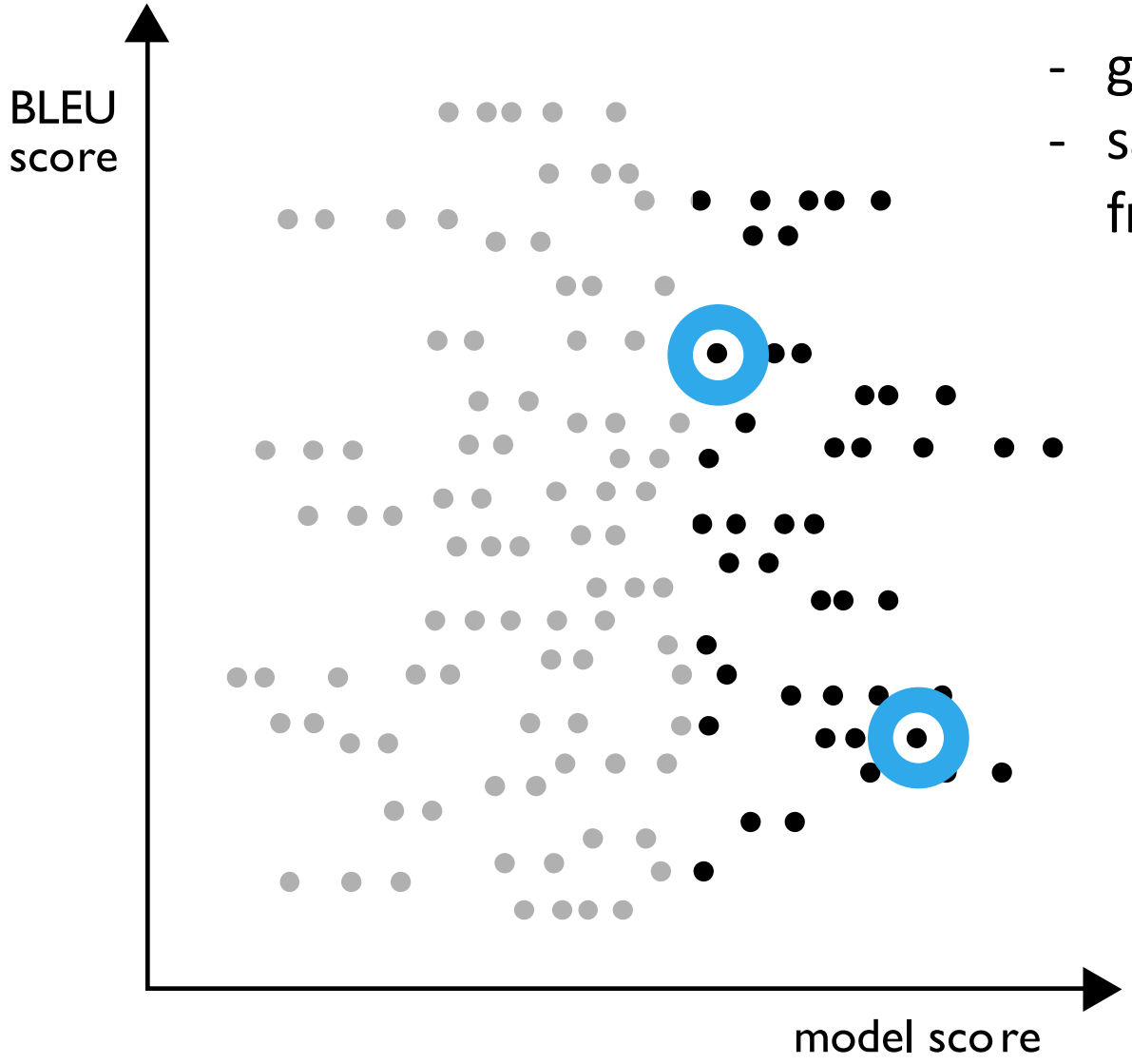
(Hopkins & May, 2011)

- generate k-best lists



Pairwise Ranking Optimization

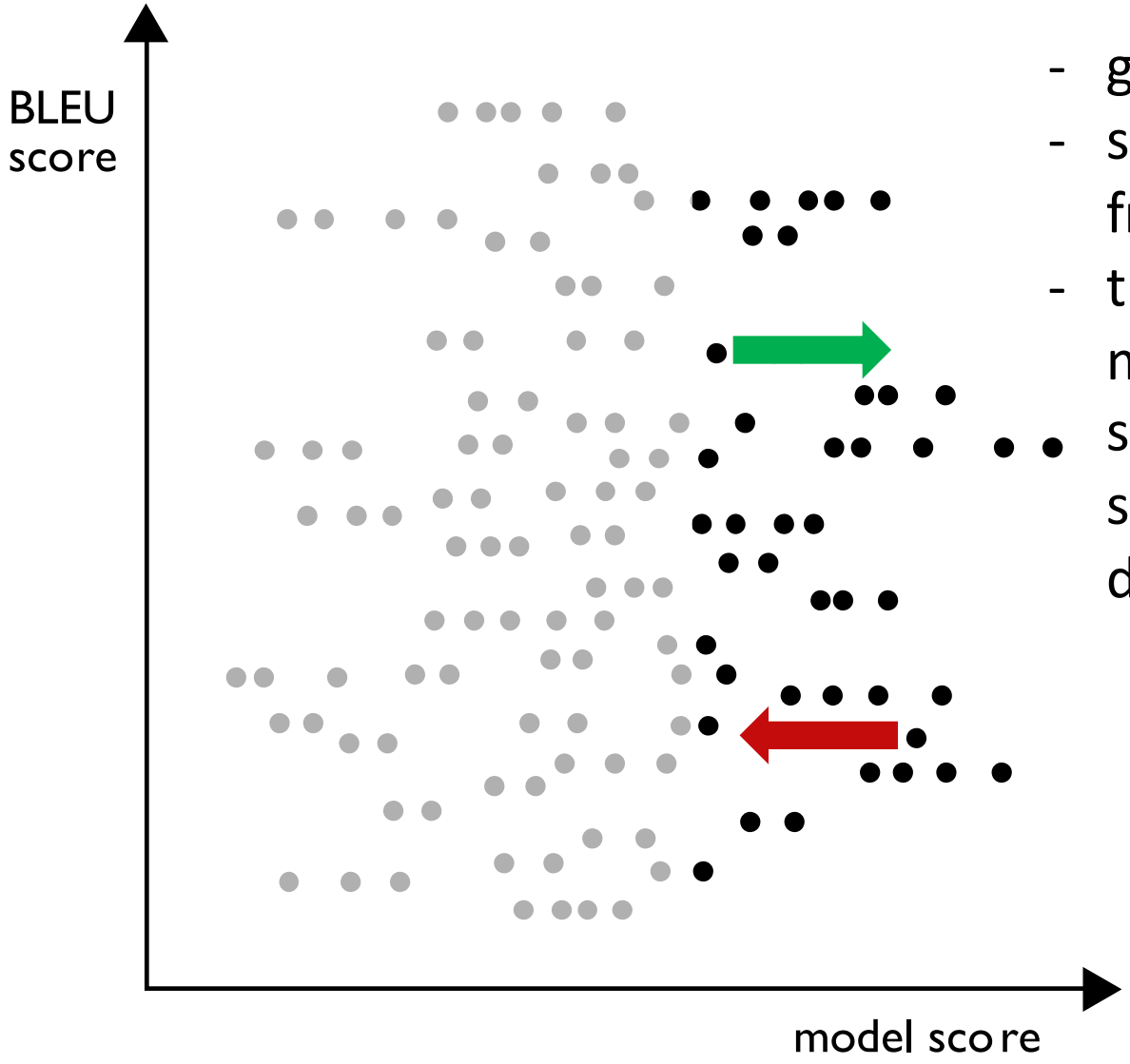
(Hopkins & May, 2011)



- generate k-best lists
- sample translation pairs from k-best list

Pairwise Ranking Optimization

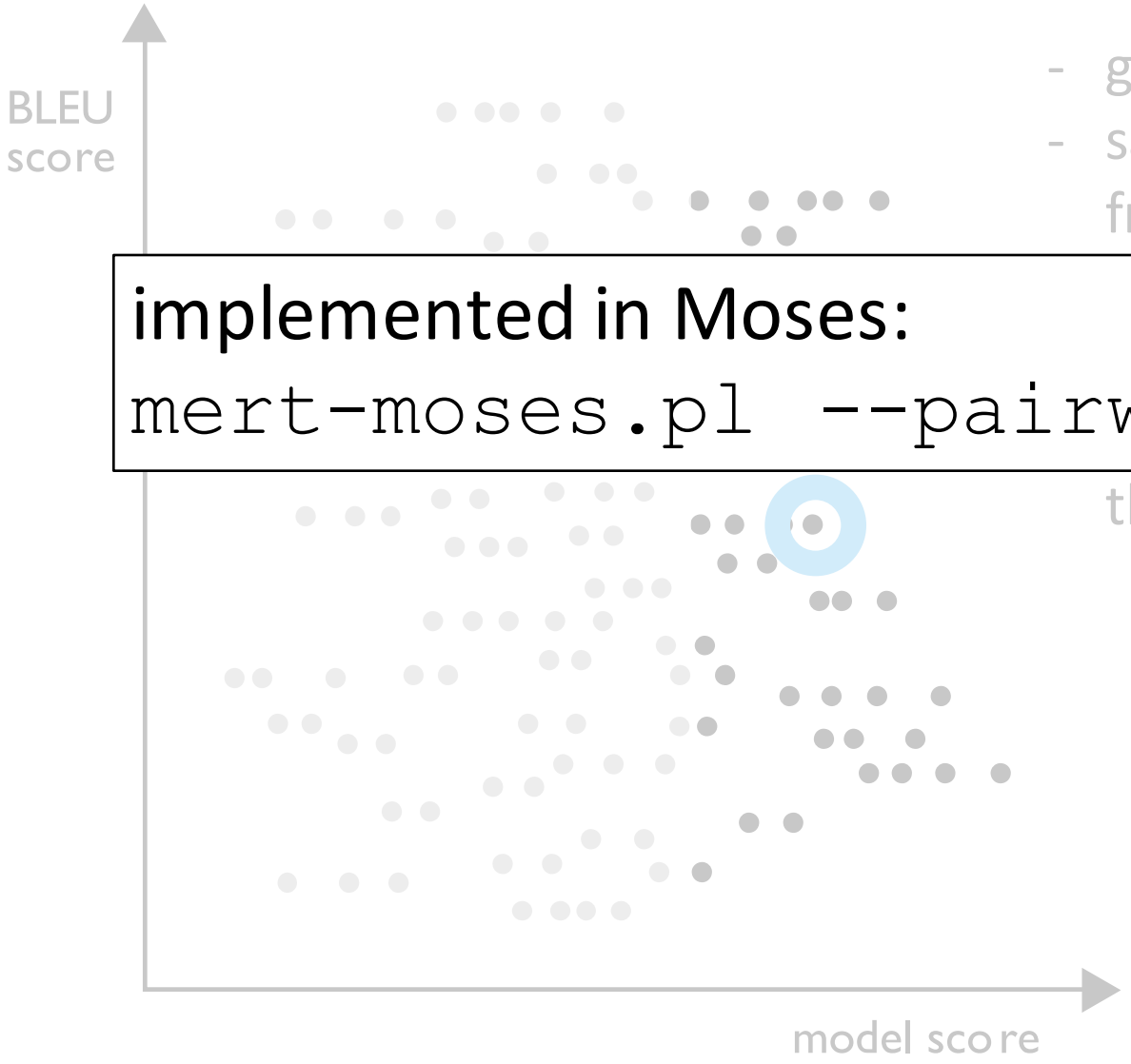
(Hopkins & May, 2011)



- generate k-best lists
- sample translation pairs from k-best list
- train a binary classifier to make the pairs' model score difference have same sign as the BLEU difference

Pairwise Ranking Optimization

(Hopkins & May, 2011)



- generate k-best lists
- sample translation pairs from k-best list

to

the BLEU difference

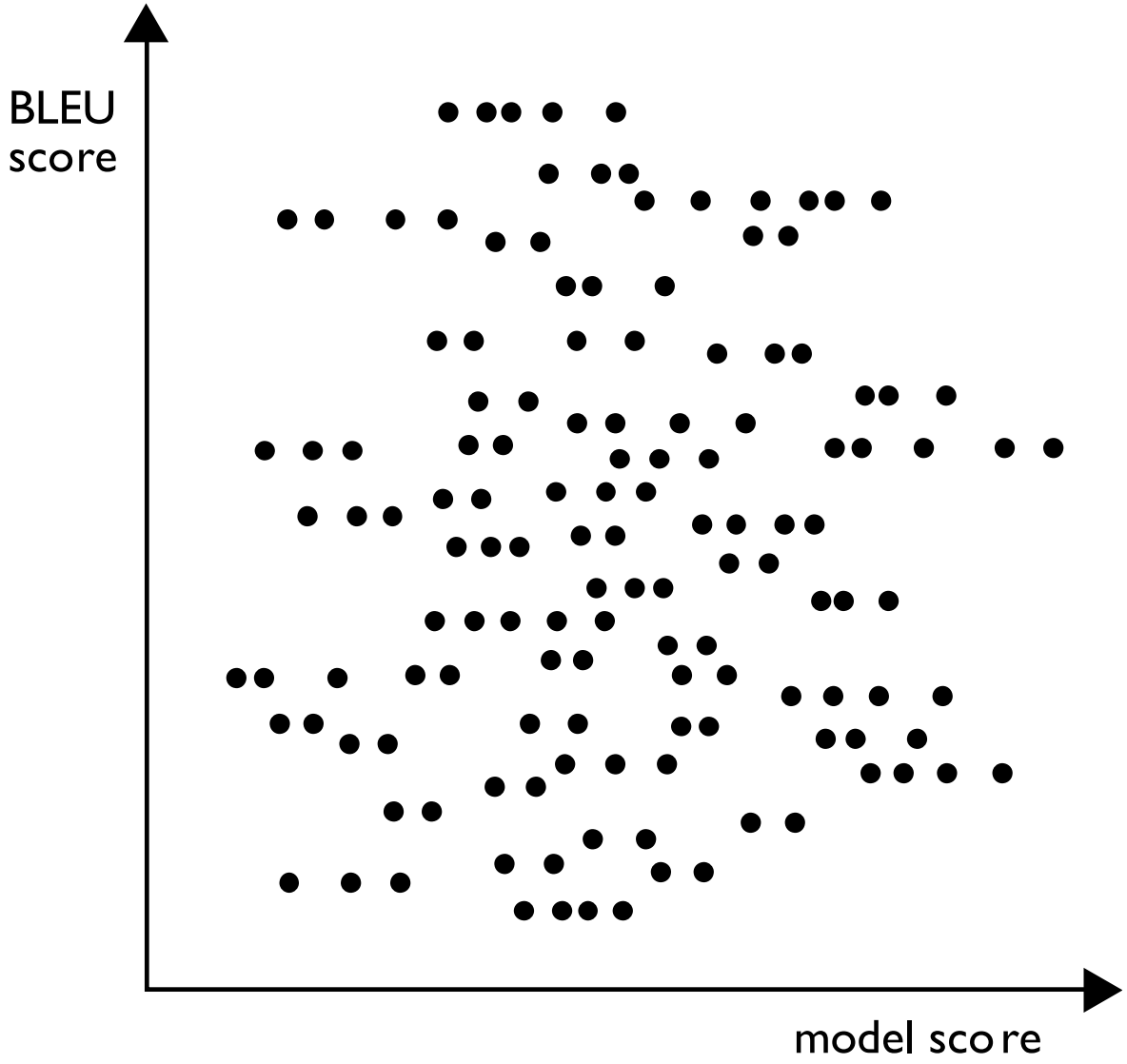
implemented in Moses:

```
mert-moses.pl --pairwise-ranked
```

How about standard machine learning algorithms for structured prediction?

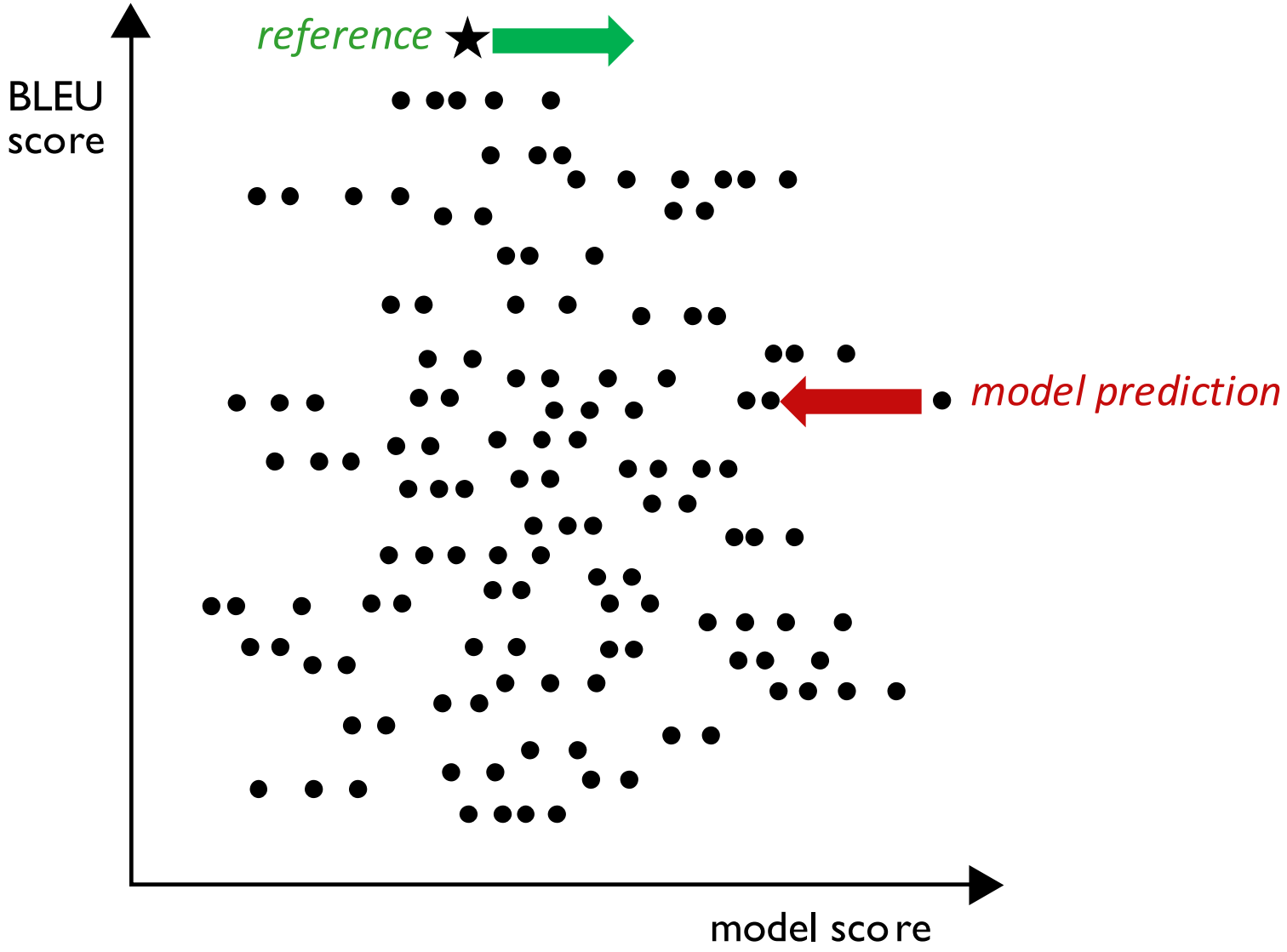
Structured Perceptron?

(Collins, 2002)

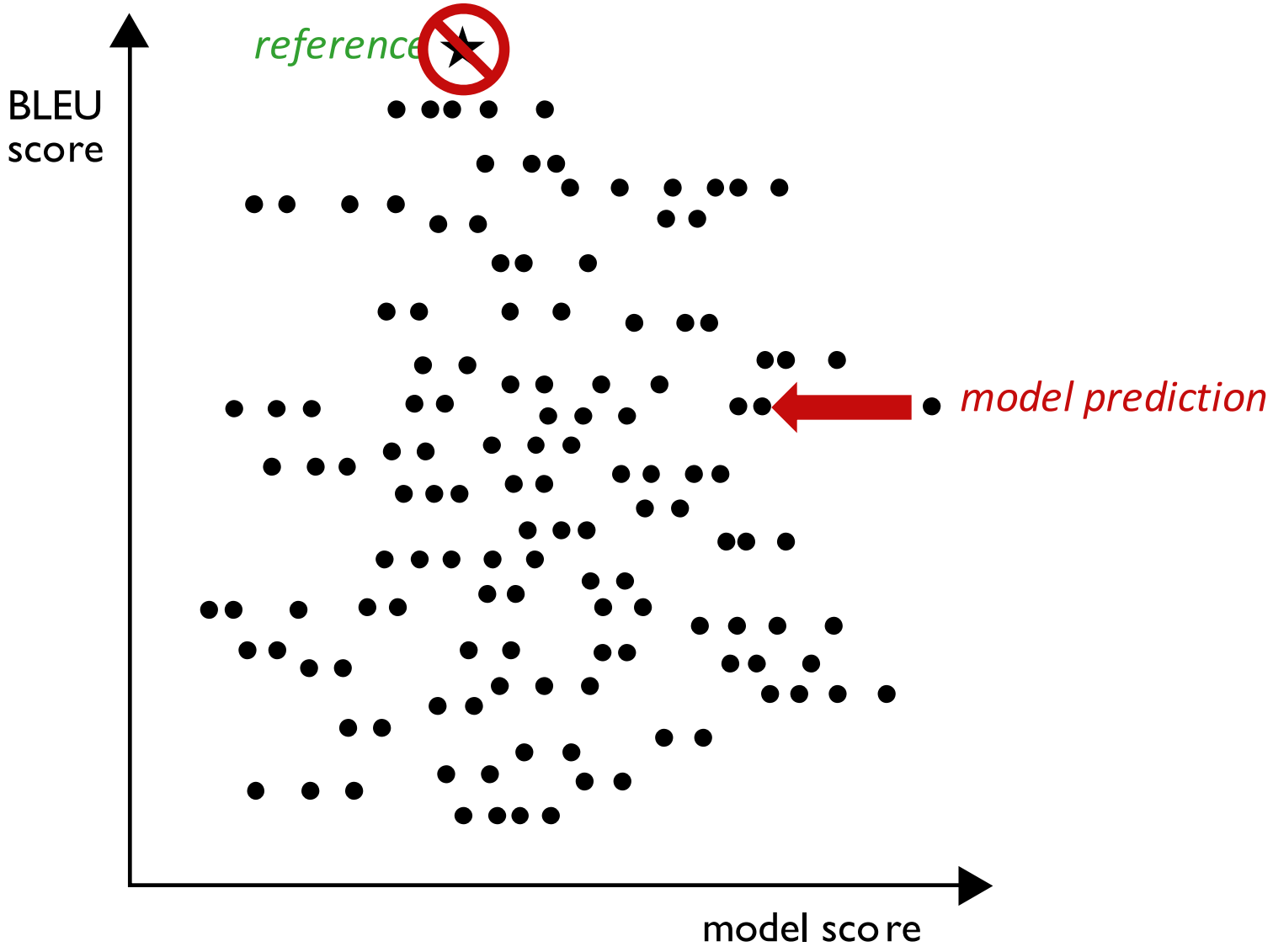


Structured Perceptron

(Collins, 2002)

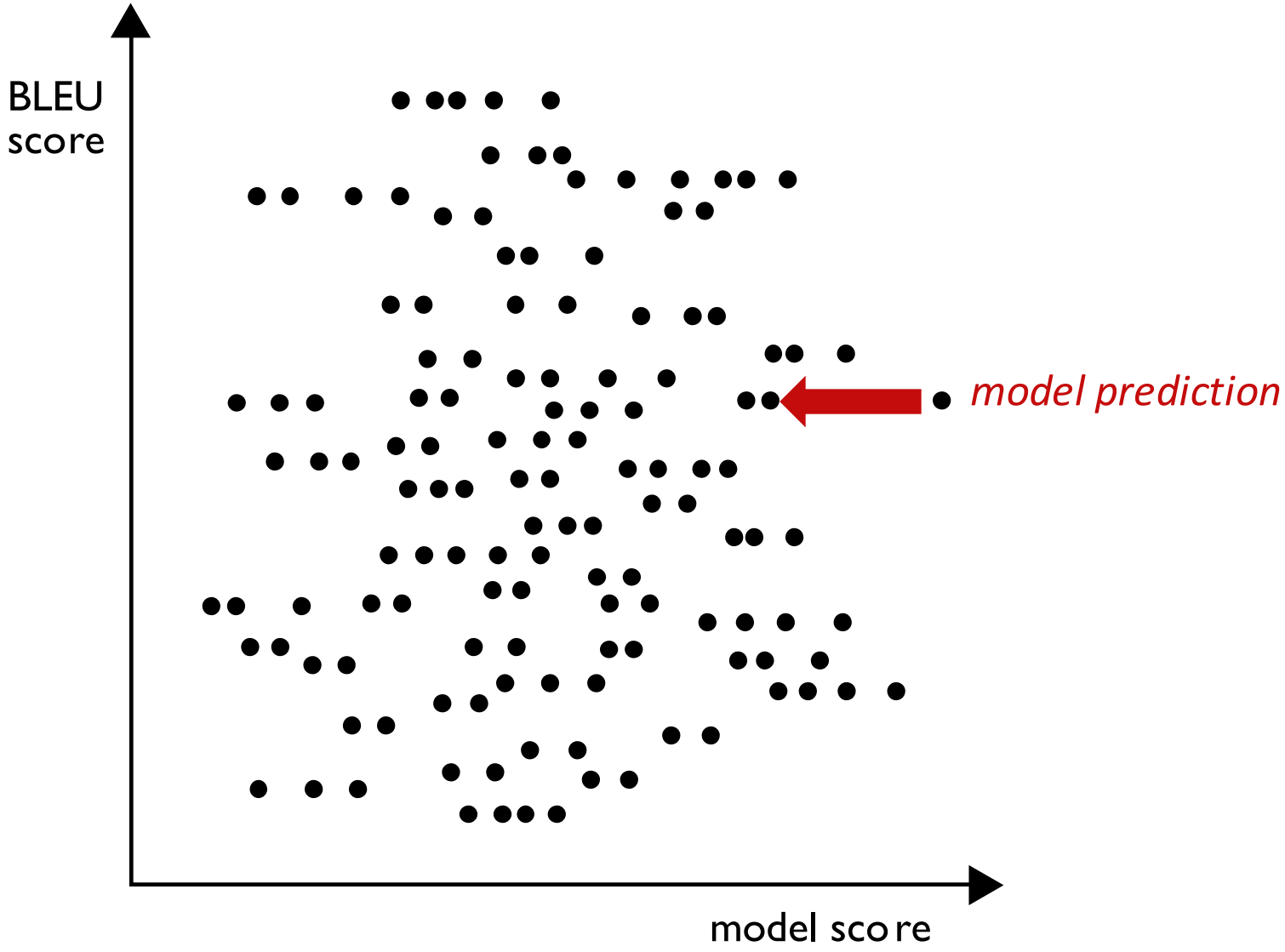


Structured Perceptron for MT?



k-Best Perceptron for MT

(Liang et al., 2006)

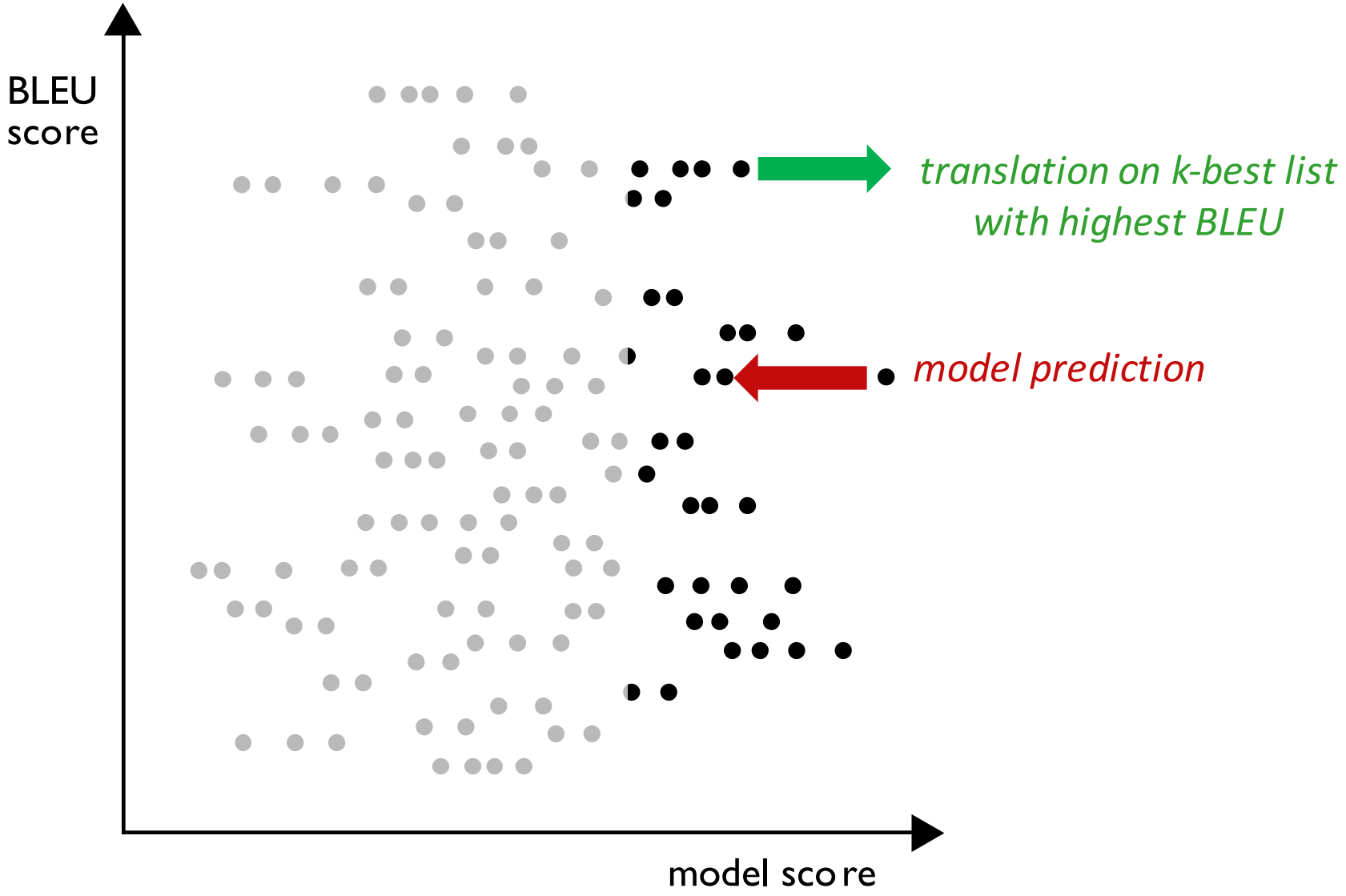


k-Best Perceptron for MT

(Liang et al., 2006)

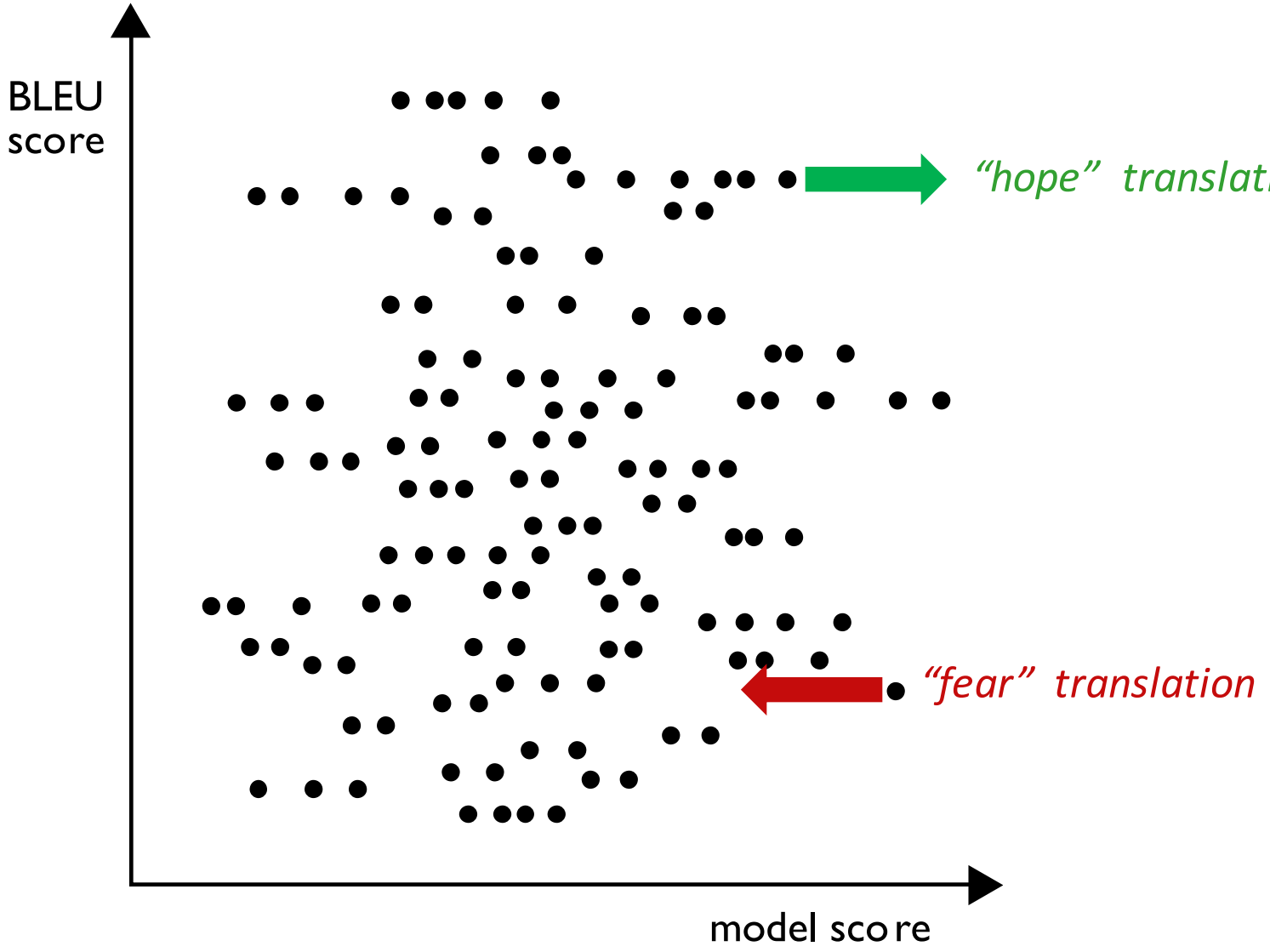
k-Best Perceptron for MT

(Liang et al., 2006)



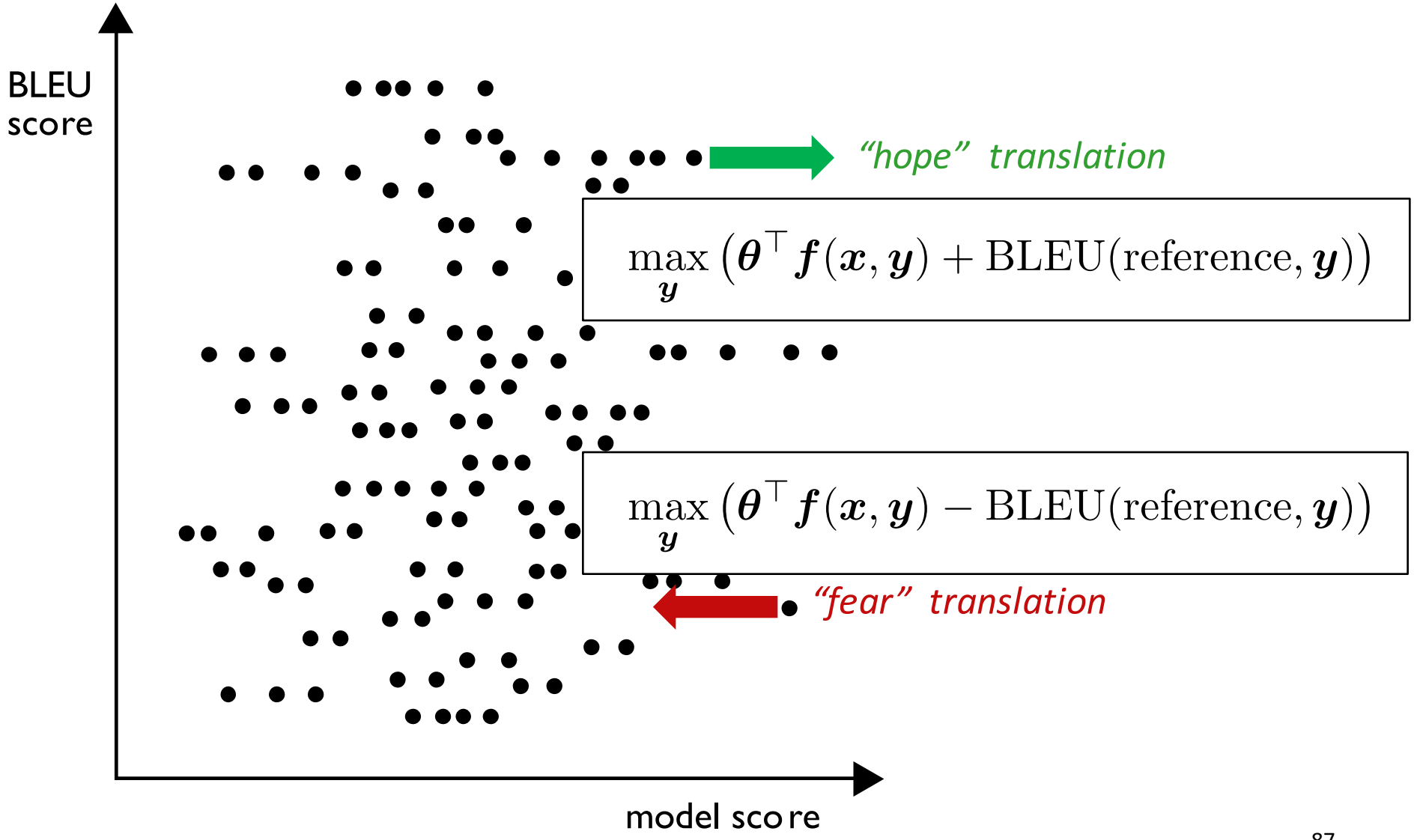
“Hope-Fear” MIRA

(Chiang et al., 2008; 2009; Cherry & Foster, 2012; Chiang, 2012)



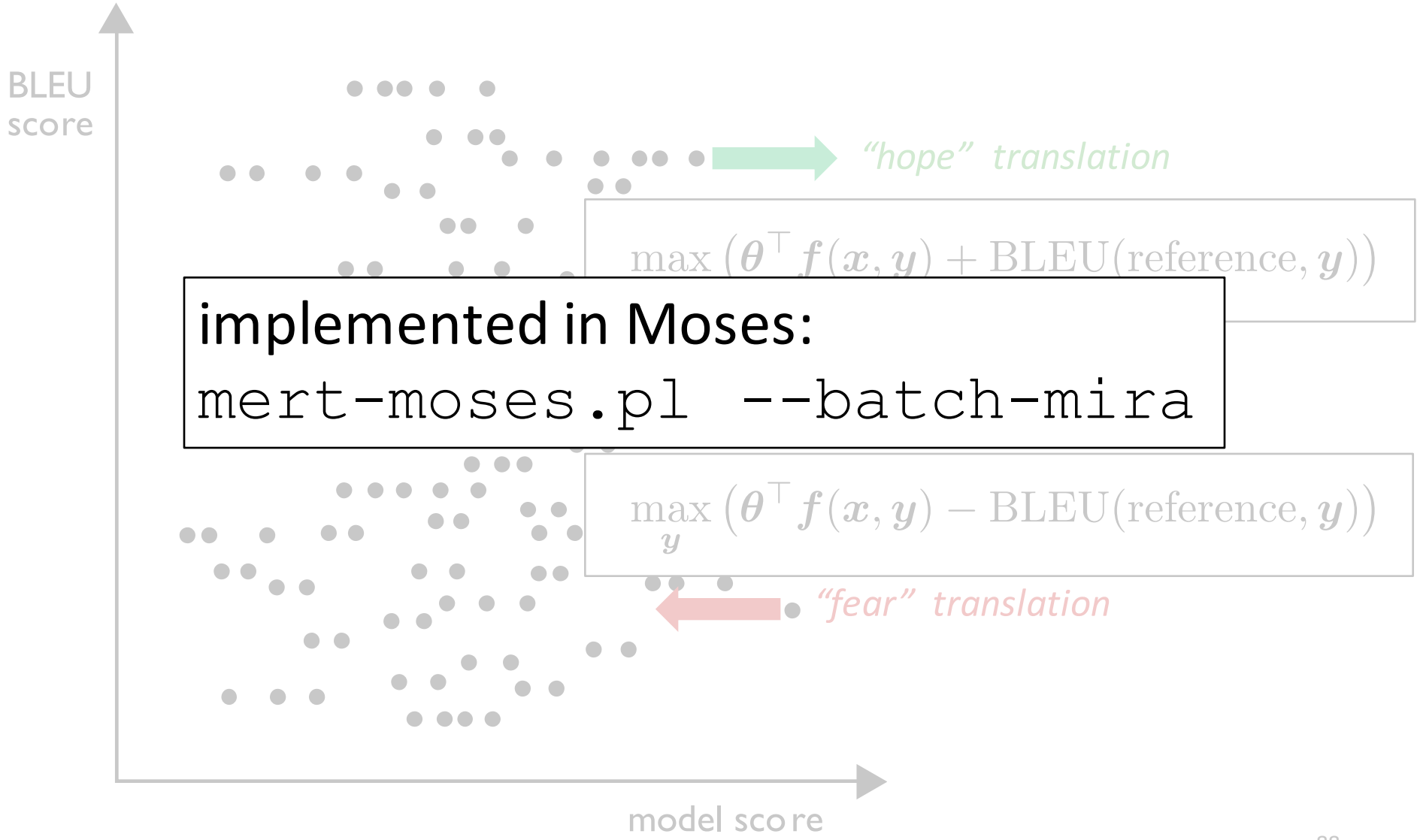
"Hope-Fear" MIRA

(Chiang et al., 2008; 2009; Cherry & Foster, 2012; Chiang, 2012)



“Hope-Fear” MIRA

(Chiang et al., 2008; 2009; Cherry & Foster, 2012; Chiang, 2012)



MT Experiments

(Gimpel, 2012)

averages over 8 test sets across 3 language pairs

	Moses %BLEU	Hiero %BLEU
MERT	**35.9**	**37.0**
PRO	**35.9**	36.9
Bayes Risk	35.6	36.4
Fear MIRA	34.9	34.2
Hope MIRA	35.2	36.0
Hope-Fear MIRA	35.7	**37.0**

Questions?

2002:
conditional log-likelihood

Proceedings of the 40th Annual Meeting of the Association for
Computational Linguistics (ACL), Philadelphia, July 2002, pp. 295-302.

Discriminative Training and Maximum Entropy Models for Statistical Machine Translation

Franz Josef Och and **Hermann Ney**
Lehrstuhl für Informatik VI, Computer Science Department
RWTH Aachen - University of Technology
D-52056 Aachen, Germany
{och,ney}@informatik.rwth-aachen.de

2002:
conditional log-likelihood

Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 295-302.

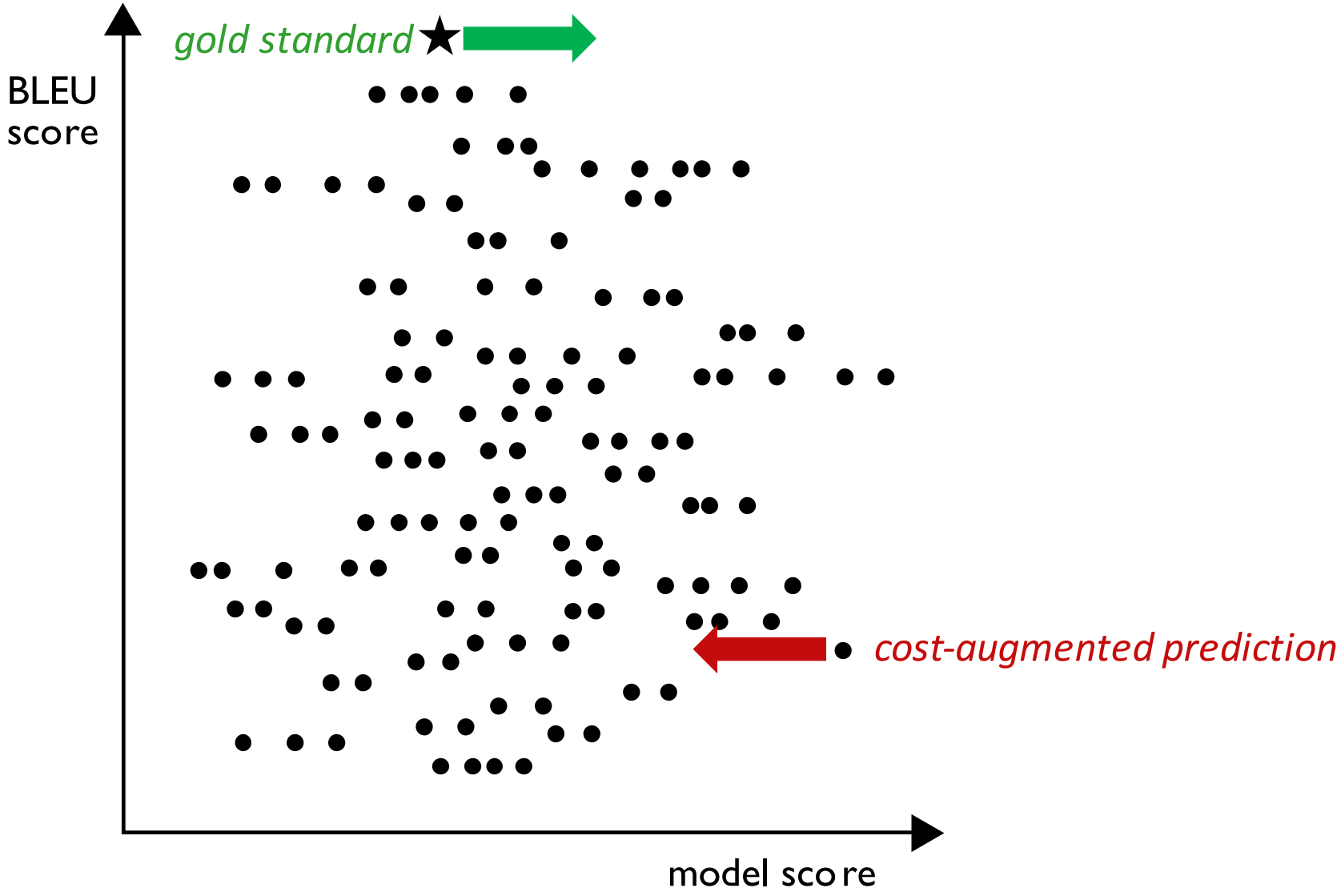
Discriminative Training and Maximum Entropy Models for Statistical Machine Translation

Table 2: Effect of maximum entropy training for alignment template approach (WP: word penalty feature, CLM: class-based language model (five-gram), MX: conventional dictionary).

	objective criteria [%]					subjective criteria [%]	
	SER	WER	PER	mWER	BLEU	SSER	IER
Baseline($\lambda_m = 1$)	86.9	42.8	33.0	37.7	43.9	35.9	39.0
ME	81.7	40.2	28.7	34.6	49.7	32.5	34.8
ME+WP	80.5	38.6	26.9	32.4	54.1	29.9	32.2
ME+WP+CLM	78.1	38.3	26.9	32.1	55.0	29.1	30.9
ME+WP+CLM+MX	77.8	38.4	26.8	31.9	55.2	28.8	30.9

Discriminative

Hinge Loss


gold standard
BLEU score
cost-augmented prediction
model score