TTIC 31190:
Natural Language Processing

Kevin Gimpel
Spring 2018

Lecture 17:
Machine Translation;
Semantics

Roadmap

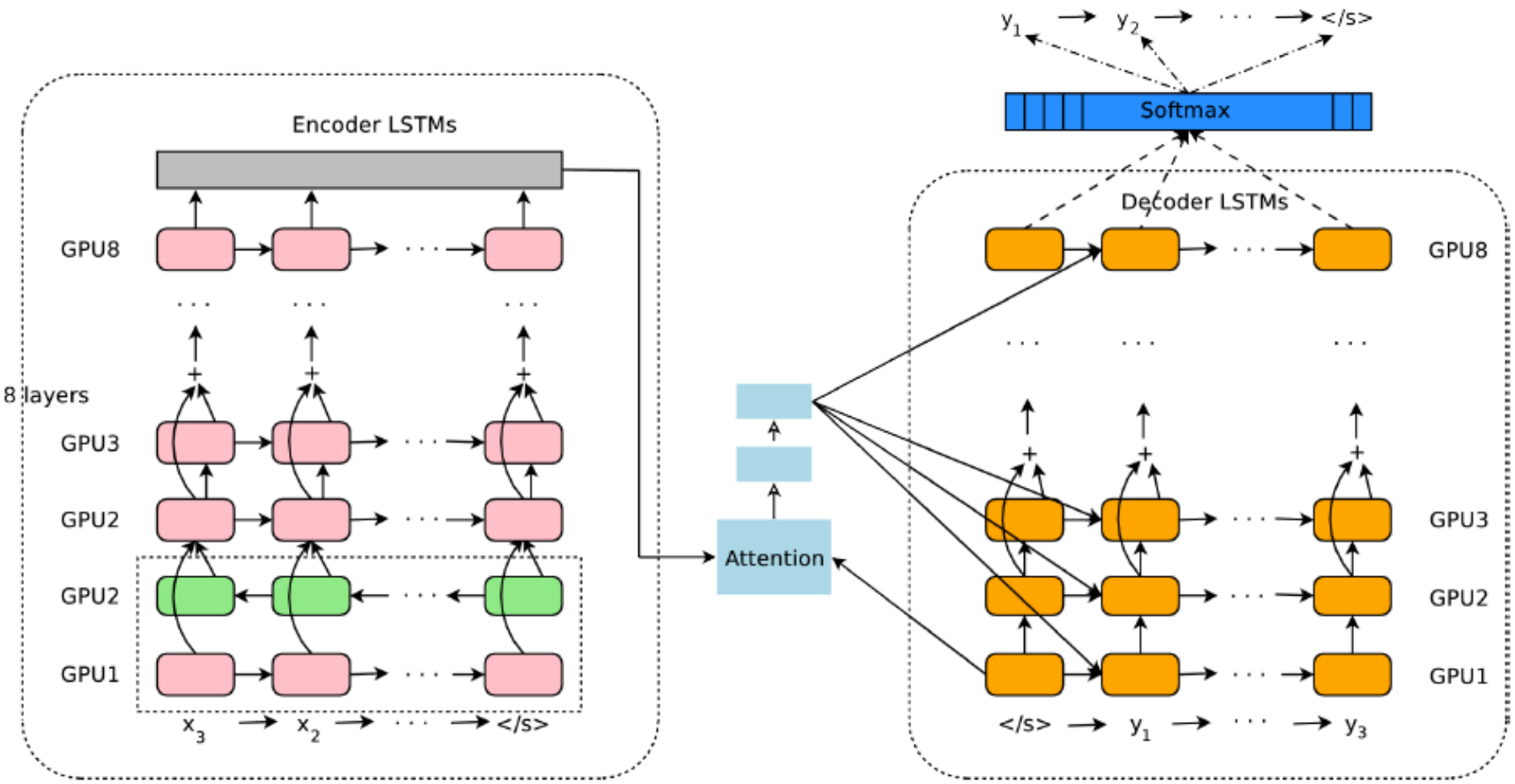
- words, morphology, lexical semantics
- text classification
- simple neural methods for NLP
- language modeling and word embeddings
- recurrent/recursive/convolutional networks in NLP
- sequence labeling, HMMs, dynamic programming
- syntax and syntactic parsing
- machine translation
- semantics

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
yonghui,schuster,zhifengc,qvl,mnorouzi@google.com

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

Google's NMT System



Google's NMT System

In our experience with large-scale translation tasks, simple stacked LSTM layers work well up to 4 layers, barely with 6 layers, and very poorly beyond 8 layers.

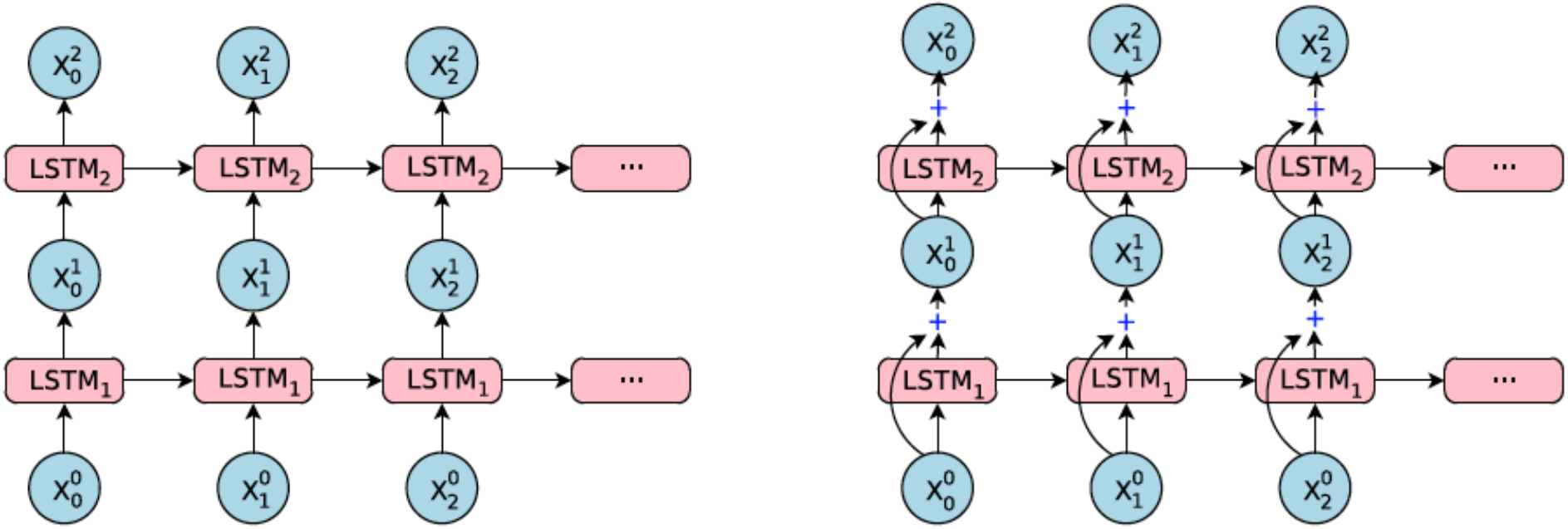


Figure 2: The difference between normal stacked LSTM and our stacked LSTM with residual connections. On the left: simple stacked LSTM layers [41]. On the right: our implementation of stacked LSTM layers with residual connections. With residual connections, input to the bottom LSTM layer ($\mathbf{x_i^0}$'s to LSTM$_1$) is element-wise added to the output from the bottom layer ($\mathbf{x_i^1}$'s). This sum is then fed to the top LSTM layer (LSTM$_2$) as the new input.

Google's NMT System

Here is an example of a word sequence and the corresponding wordpiece sequence:

- **Word**: Jet makers feud over seat width with big orders at stake
- **wordpieces**: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

In the above example, the word "Jet" is broken into two wordpieces "_J" and "et", and the word "feud" is broken into two wordpieces "_fe" and "ud". The other words remain as single wordpieces. "_" is a special character added to mark the beginning of a word.

they use a procedure that deterministically segments any character sequence into wordpieces

vocab: 8k-32k wordpieces

they first learn a "wordpiece model"

Google's NMT System

Table 5: Single model results on WMT En→De (newstest2014)

Model	BLEU	CPU decoding time per sentence (s)
Word	23.12	0.2972
Character (512 nodes)	22.62	0.8011
WPM-8K	23.50	0.2079
WPM-16K	24.36	0.1931
WPM-32K	24.61	0.1882
Mixed Word/Character	24.17	0.3268
PBMT [6]	20.7	
RNNSearch [37]	16.5	
RNNSearch-LV [37]	16.9	
RNNSearch-LV [37]	16.9	
Deep-Att [45]	20.6	

Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu,
Zhifeng Chen, Nikhil Thorat
melvinp,schuster,qvl,krikun,yonghui,zhifengc,nsthorat@google.com

Fernanda Viégas, Martin Wattenberg, Greg Corrado,
Macduff Hughes, Jeffrey Dean

To be able to make use of multilingual data within a single system, we propose one simple modification to the input data, which is to introduce an artificial token at the beginning of the input sentence to indicate the target language the model should translate to. For instance, consider the following English→Spanish pair of sentences:

```
Hello, how are you? -> ¿Hola como estás?
```

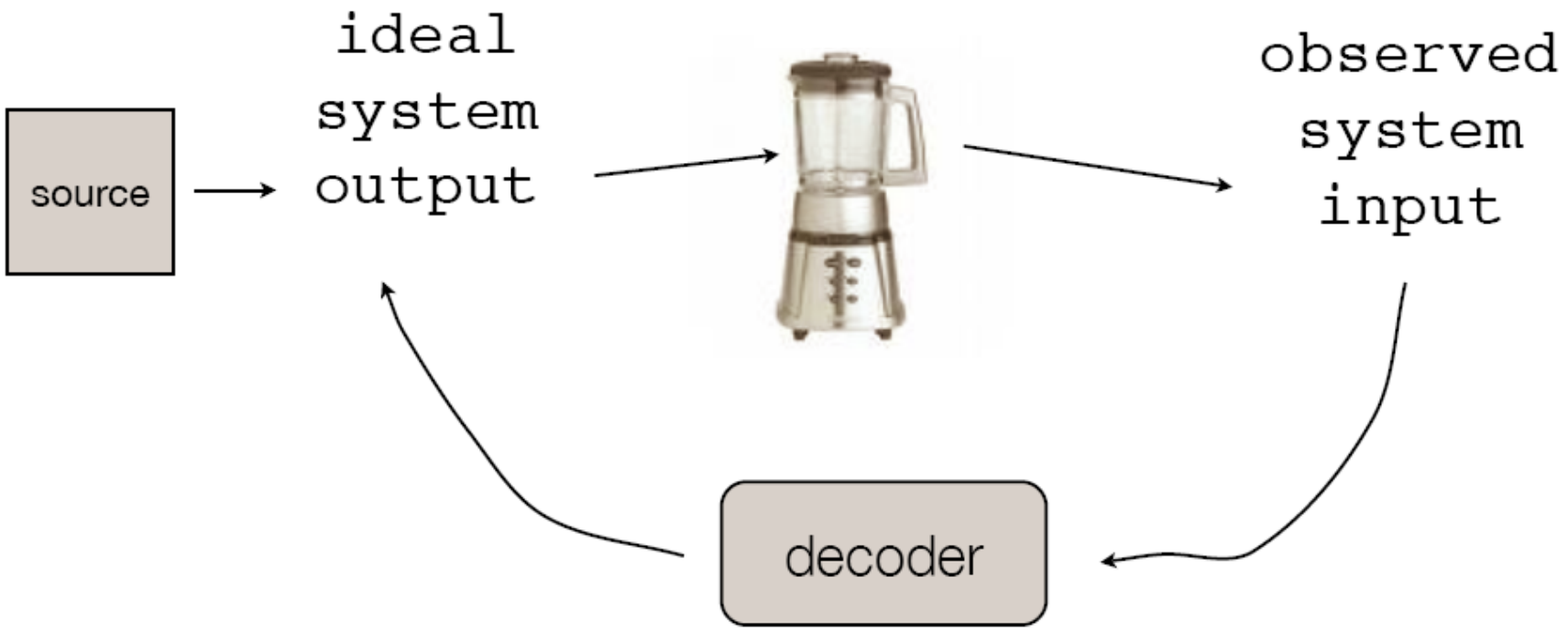
It will be modified to:

```
<2es> Hello, how are you? -> ¿Hola como estás?
```

Table 5: Portuguese→Spanish BLEU scores using various models.

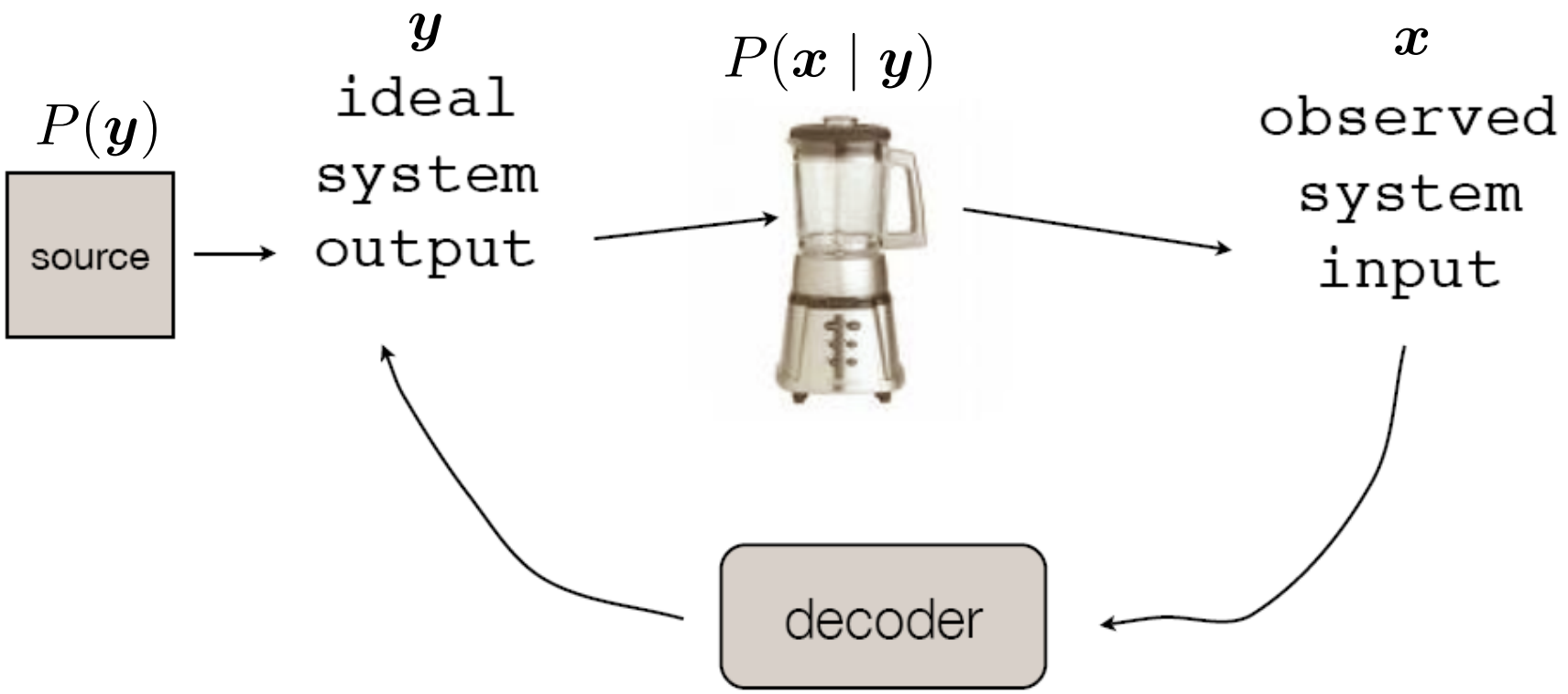
	Model	BLEU
(a)	PBMT bridged	28.99
(b)	NMT bridged	30.91
(c)	NMT Pt→Es	31.50
(d)	Model 1 (Pt→En, En→Es)	21.62
(e)	Model 2 (En↔{Es, Pt})	24.75
(f)	Model 2 + incremental training	31.77

Noisy Channel Model



from Noah Smith

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



$$y^* = \operatorname*{argmax}_y P(y \mid x)$$

$$= \operatorname*{argmax}_y \frac{P(x \mid y)P(y)}{P(x)}$$

$$= \operatorname*{argmax}_y P(x \mid y)P(y)$$

from Noah Smith

Modeling for the Noisy Channel

- we need to model two probability distributions:
  - $P(\boldsymbol{y})$ should favor fluent translations
  - $P(\boldsymbol{x} \mid \boldsymbol{y})$ should favor accurate/faithful translations

Modeling for the Noisy Channel

- we need to model two probability distributions:
  - $P(\boldsymbol{y})$ should favor fluent translations
  - $P(\boldsymbol{x} \mid \boldsymbol{y})$ should favor accurate/faithful translations

- let's start with $P(\boldsymbol{y})$
  - how do we compute the probability of an English sentence?
  - language modeling is an important part of MT

Noisy Channel

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

Noisy Channel

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

predicted translation

source sentence

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

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

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

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

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

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

"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":

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

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":

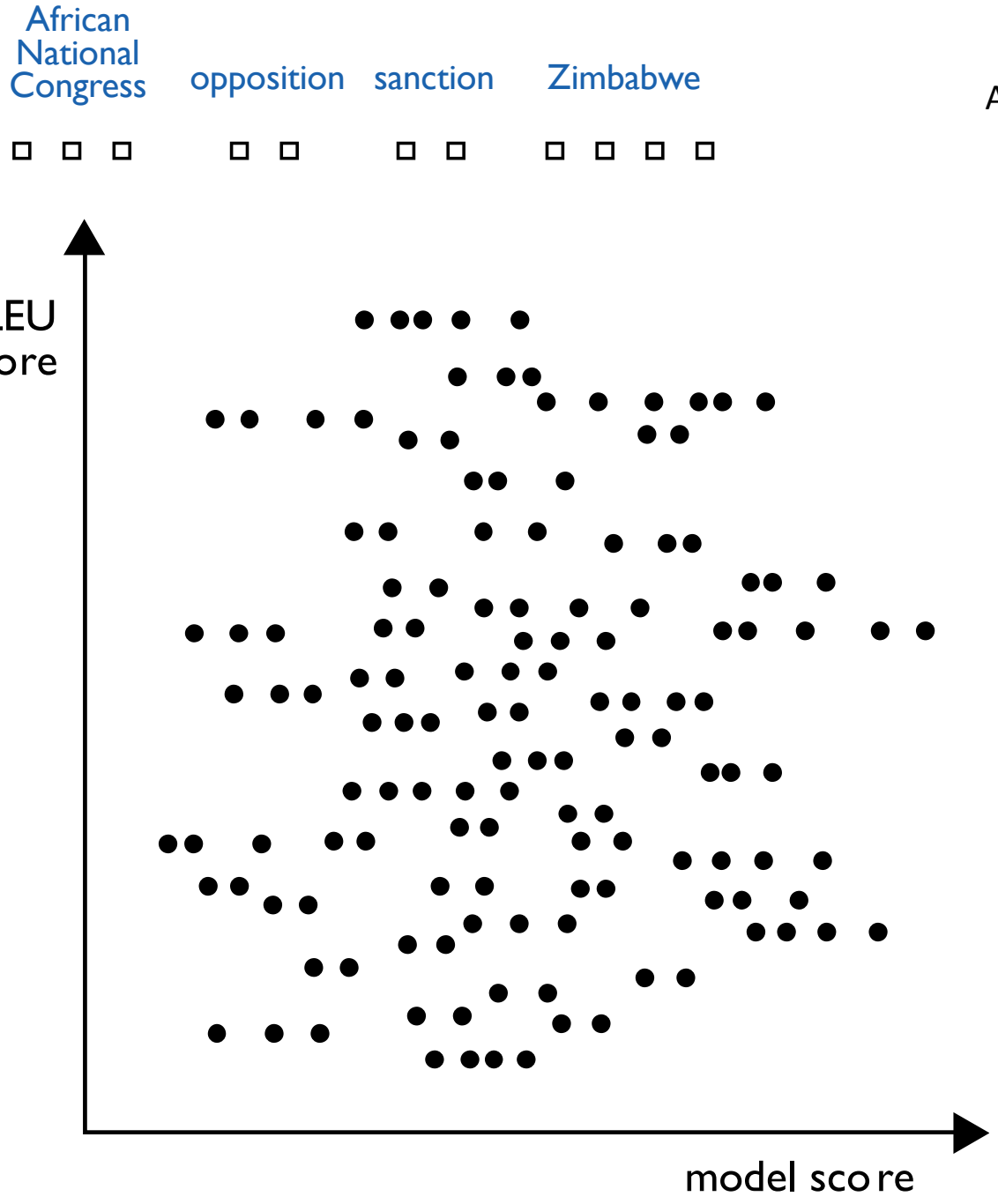
$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y}) + \gamma |\boldsymbol{y}|$$

"reverse translation model feature":

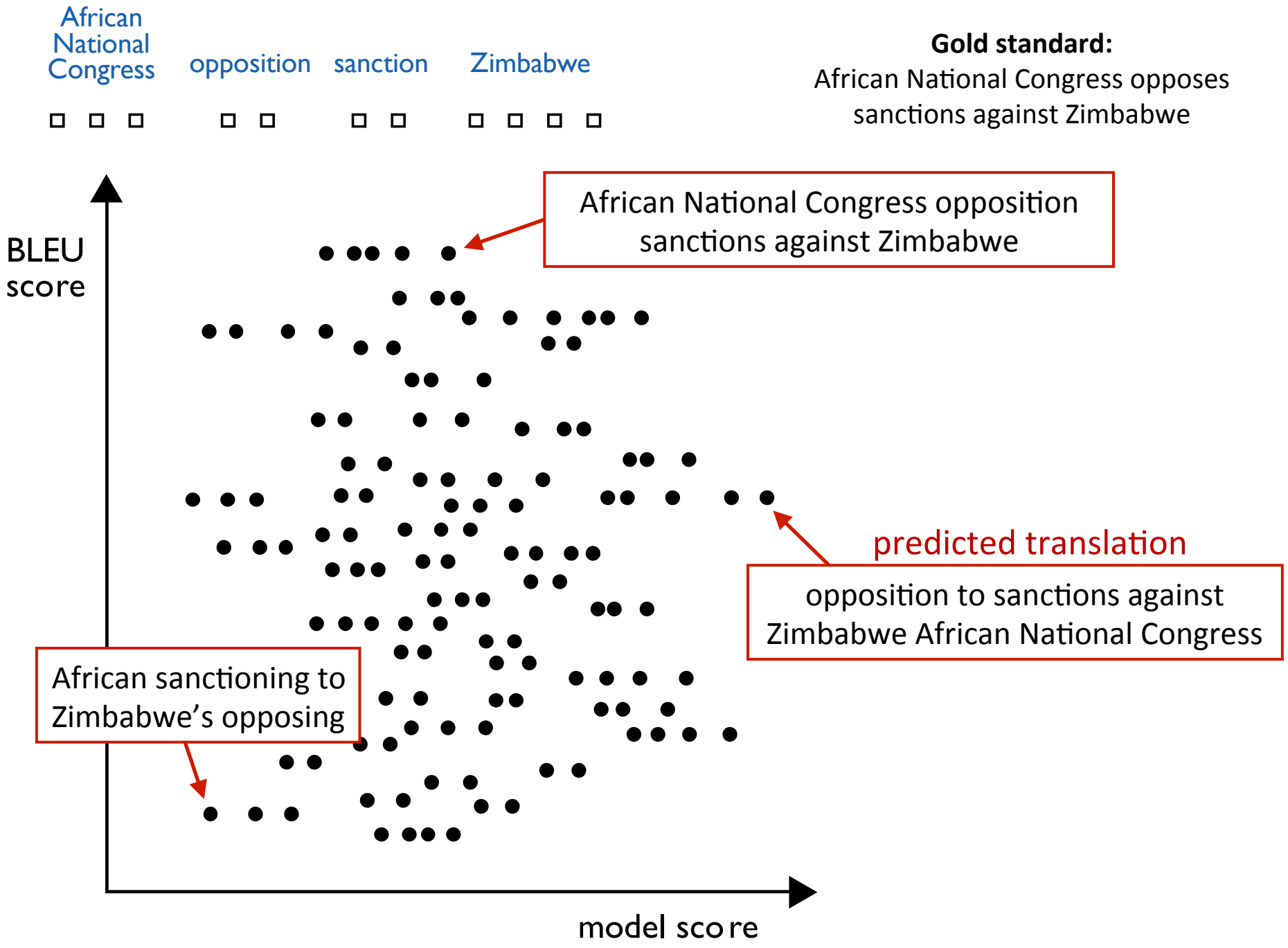
$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \ \alpha \log P(\boldsymbol{x} \mid \boldsymbol{y}) + \beta \log P(\boldsymbol{y}) + \gamma |\boldsymbol{y}| + \delta \log P(\boldsymbol{y} \mid \boldsymbol{x})$$


African National Congress
opposition
sanction
Zimbabwe

Gold standard:
African National Congress opposes sanctions against Zimbabwe

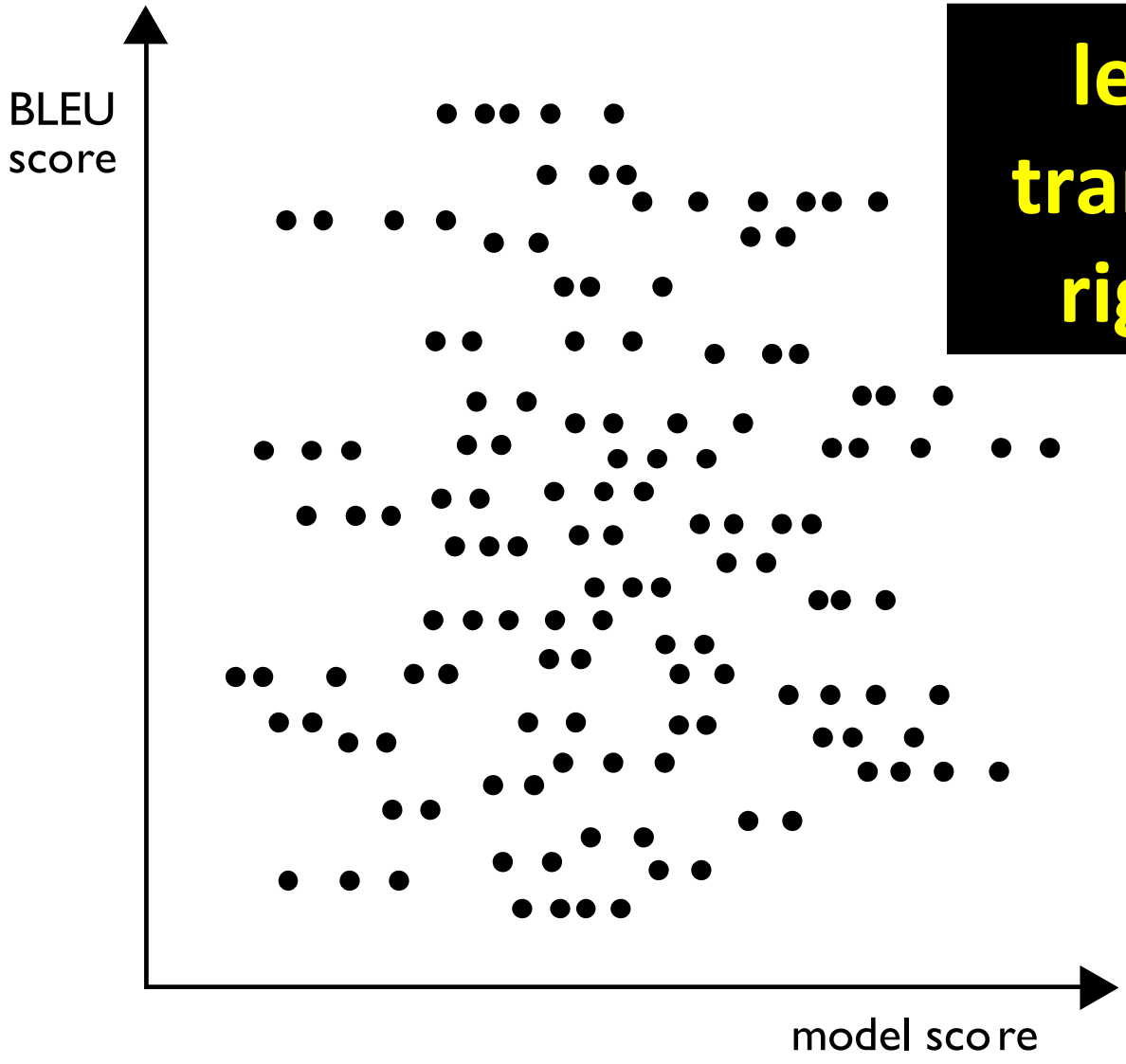


Gold standard:
African National Congress opposes sanctions against Zimbabwe


African National Congress
opposition
sanction
Zimbabwe
Gold standard:
African National Congress opposes sanctions against Zimbabwe
BLEU score
African National Congress opposition sanctions against Zimbabwe
predicted translation
opposition to sanctions against Zimbabwe African National Congress
African sanctioning to Zimbabwe's opposing
model score

African National Congress opposition sanction Zimbabwe

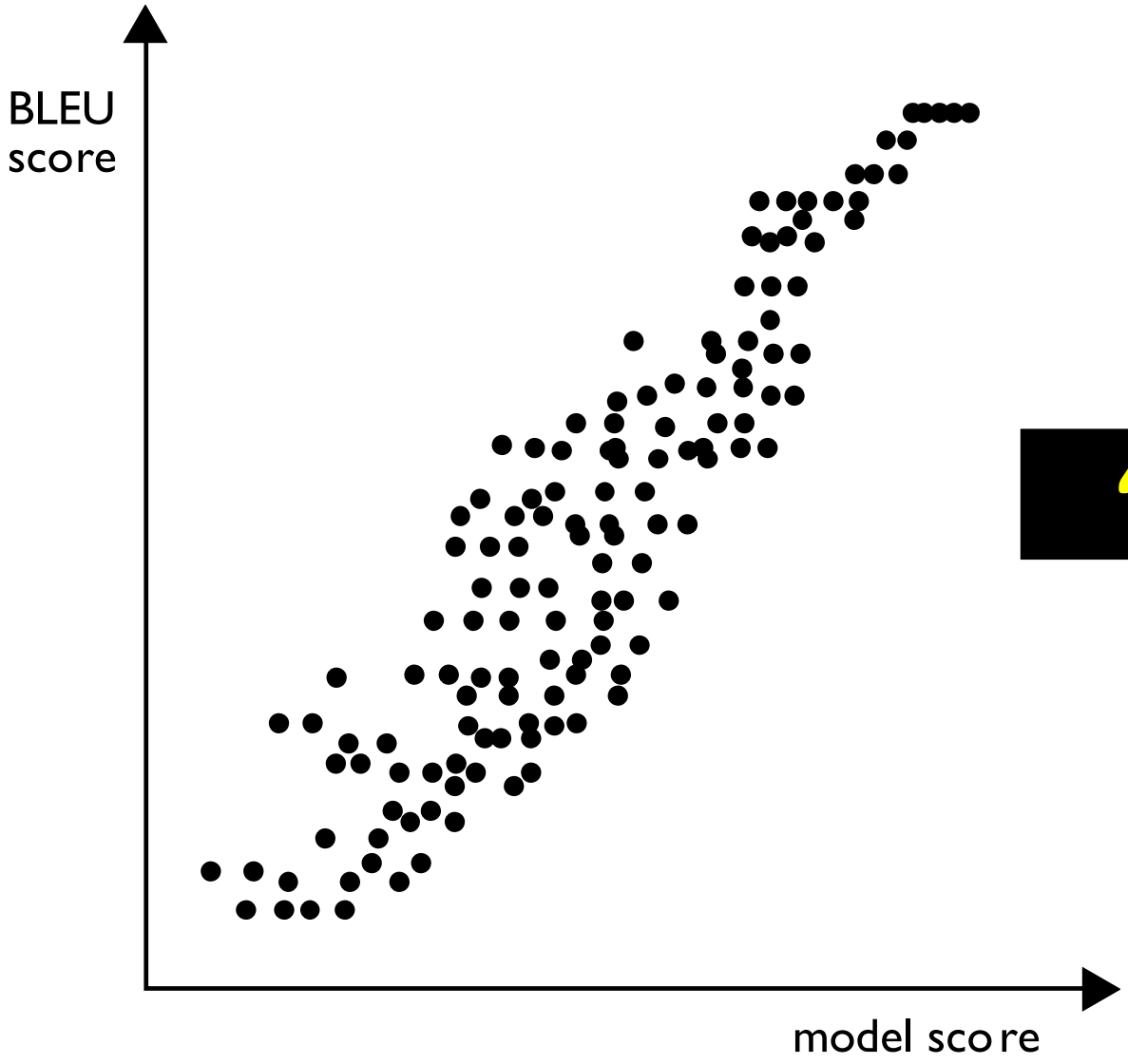
Gold standard:
African National Congress opposes sanctions against Zimbabwe



learning moves translations left or right in this plot

African National Congress opposition sanction Zimbabwe

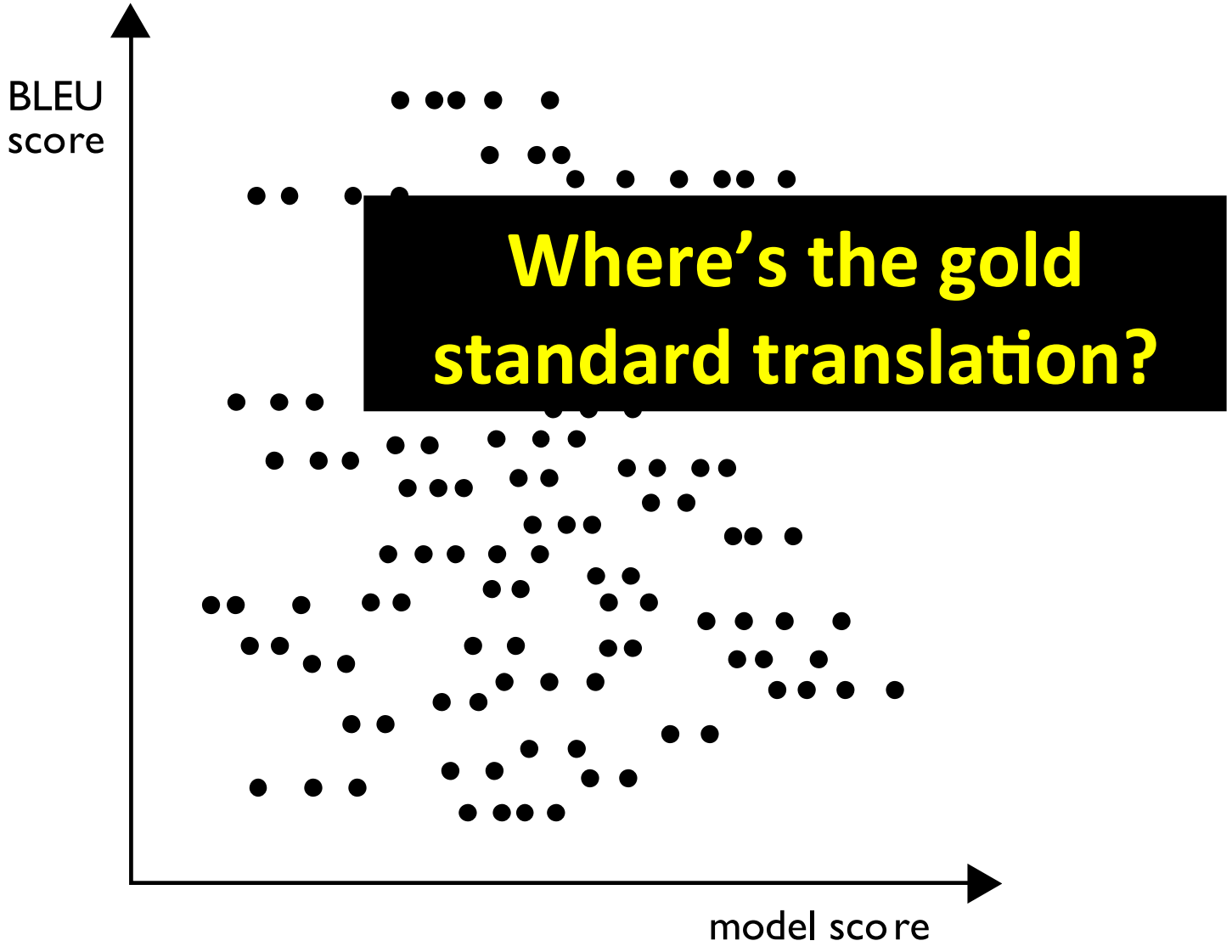
Gold standard:
African National Congress opposes sanctions against Zimbabwe



"ideal" model

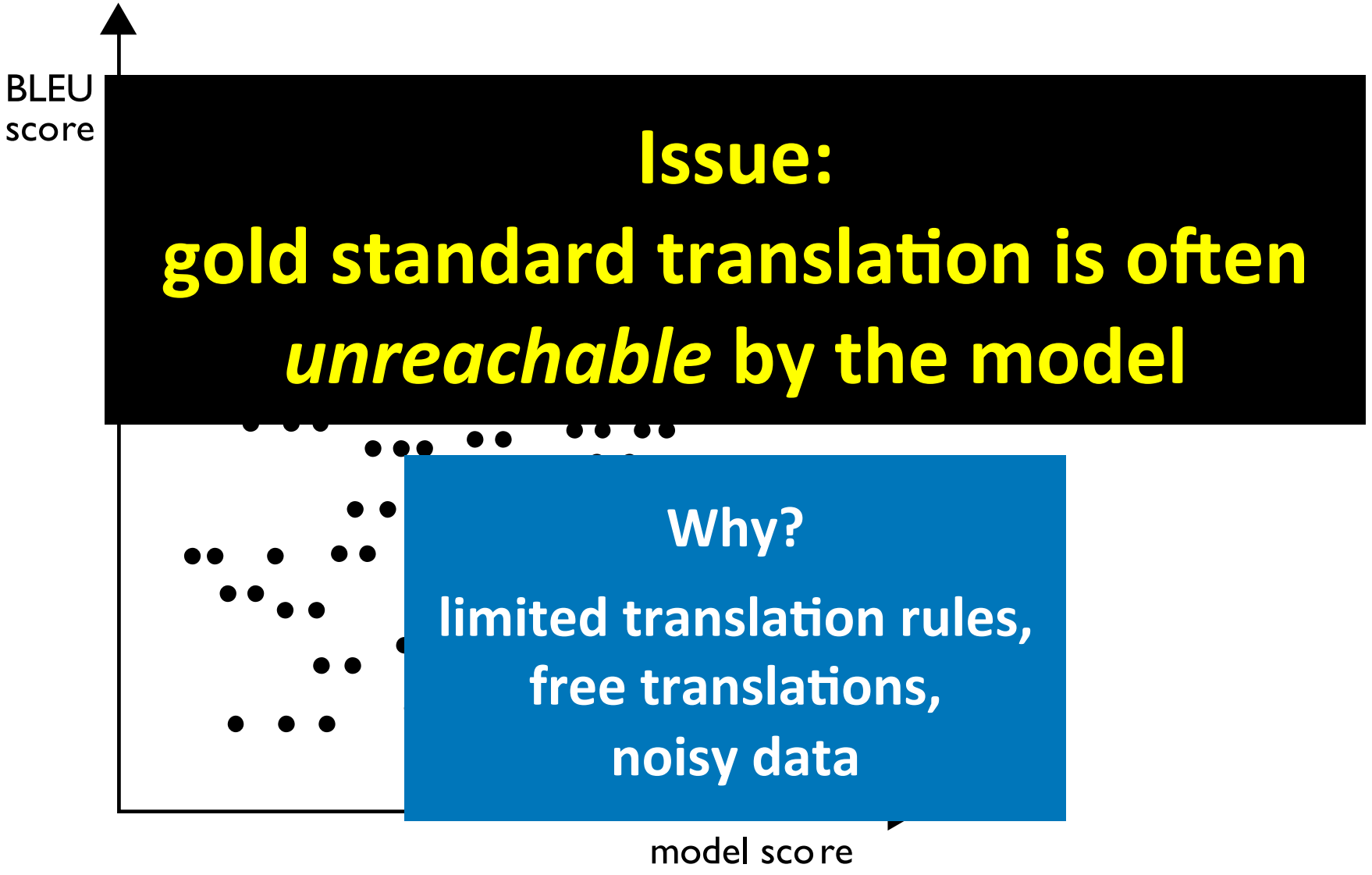
African National Congress opposition sanction Zimbabwe

Gold standard:
African National Congress opposes sanctions against Zimbabwe



African National Congress opposition sanction Zimbabwe

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

Free Translations

Machine translation:
Sharon's office said, leader of the main opposition Labor Party has admitted defeat and congratulatory telephone calls to Sharon.

Human-generated translation:
According to a representative of Sharon's office, the leader of the main opposition Labor Party has admitted defeat and made the obligatory congratulating telephone call to Sharon.

Even if gold standard translation was reachable by model, we might not want to learn from it directly

to Sharon.

Applicable to other tasks:
summarization
image caption generation

issue: gold standard translation is often unreachable by the model

name		where used
cost ("0-1")	$\text{cost}(y, \text{classify}(\boldsymbol{x}, \boldsymbol{\theta}))$	intractable, but underlies "direct error minimization"
perceptron	$-\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})$	perceptron algorithm (Rosenblatt, 1958)
hinge	$-\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} (\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta}) + \text{cost}(y, y'))$	support vector machines, other large-margin algorithms
log	$-\log p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$ $= \text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \log \sum_{y' \in \mathcal{L}} \exp\{\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})\}$	logistic regression, conditional random fields, maximum entropy models

intractable, but it doesn't need to compute model score of gold standard!

name		where used
cost ("0-1")	$\text{cost}(y, \text{classify}(\boldsymbol{x}, \boldsymbol{\theta}))$	intractable, but underlies "direct error minimization"
perceptron	$-\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})$	perceptron algorithm (Rosenblatt, 1958)
hinge	$-\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} (\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta}) + \text{cost}(y, y'))$	support vector machines, other large-margin algorithms
log	$-\log p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$ $= \text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \log \sum_{y' \in \mathcal{L}} \exp\{\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})\}$	logistic regression, conditional random fields, maximum entropy models

MERT, Och (2003)

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)

"h

minimize the cost of the decoder output

intractable in general – how can we solve it?

$$\min_{\boldsymbol{\theta}} \quad \text{cost}\left(\underbrace{\{\boldsymbol{y}^{(i)}\}_{i=1}^{N}}_{\text{references}}, \underbrace{\left\{\operatorname*{argmax}_{\langle \boldsymbol{y},\boldsymbol{h}\rangle \in \mathcal{T}_{\boldsymbol{x}^{(i)}}} \boldsymbol{\theta}^{\top} \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h})\right\}_{i=1}^{N}}_{\text{decoder outputs}}\right)$$

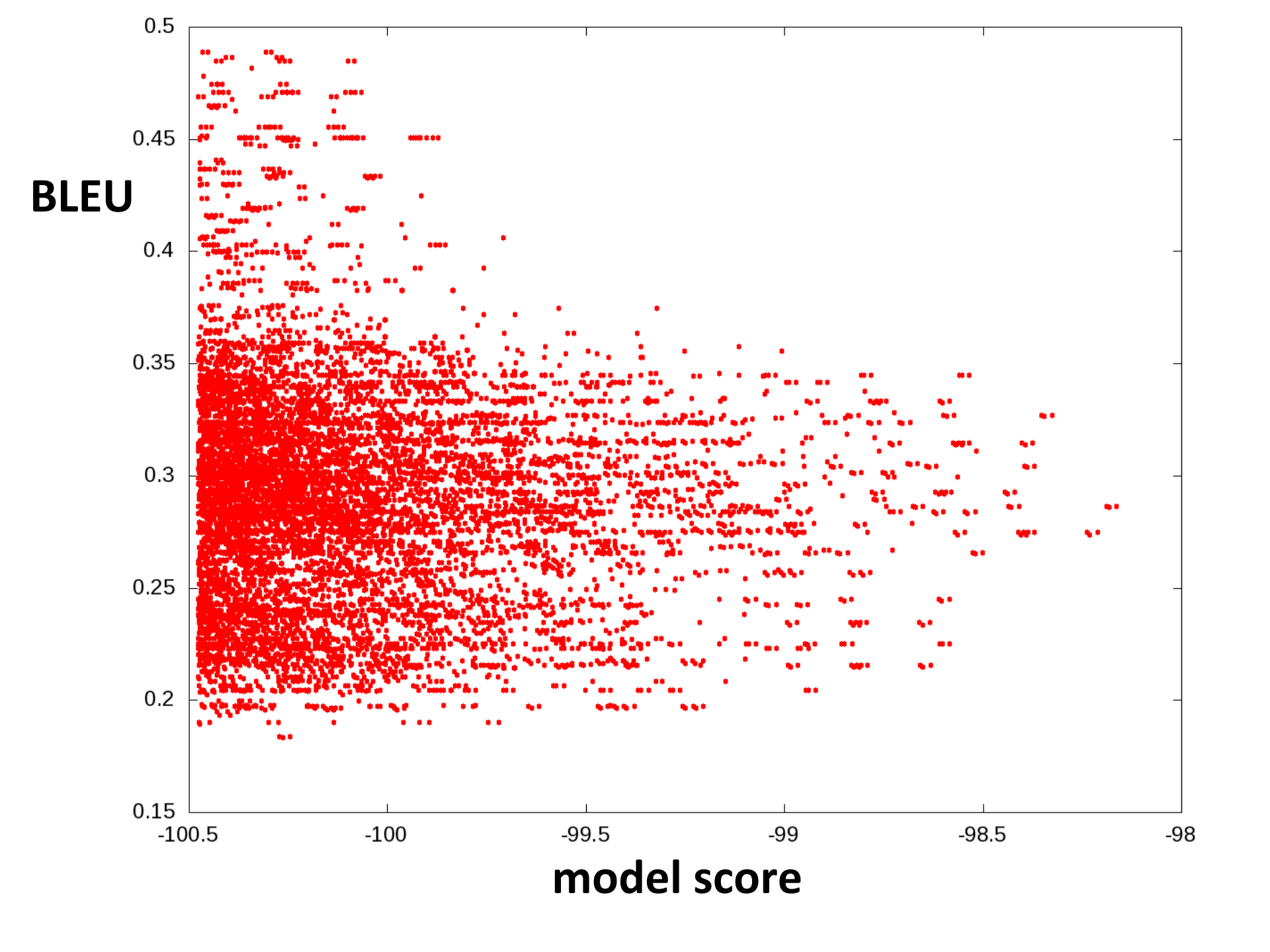
Minimum Error Rate Training (MERT)

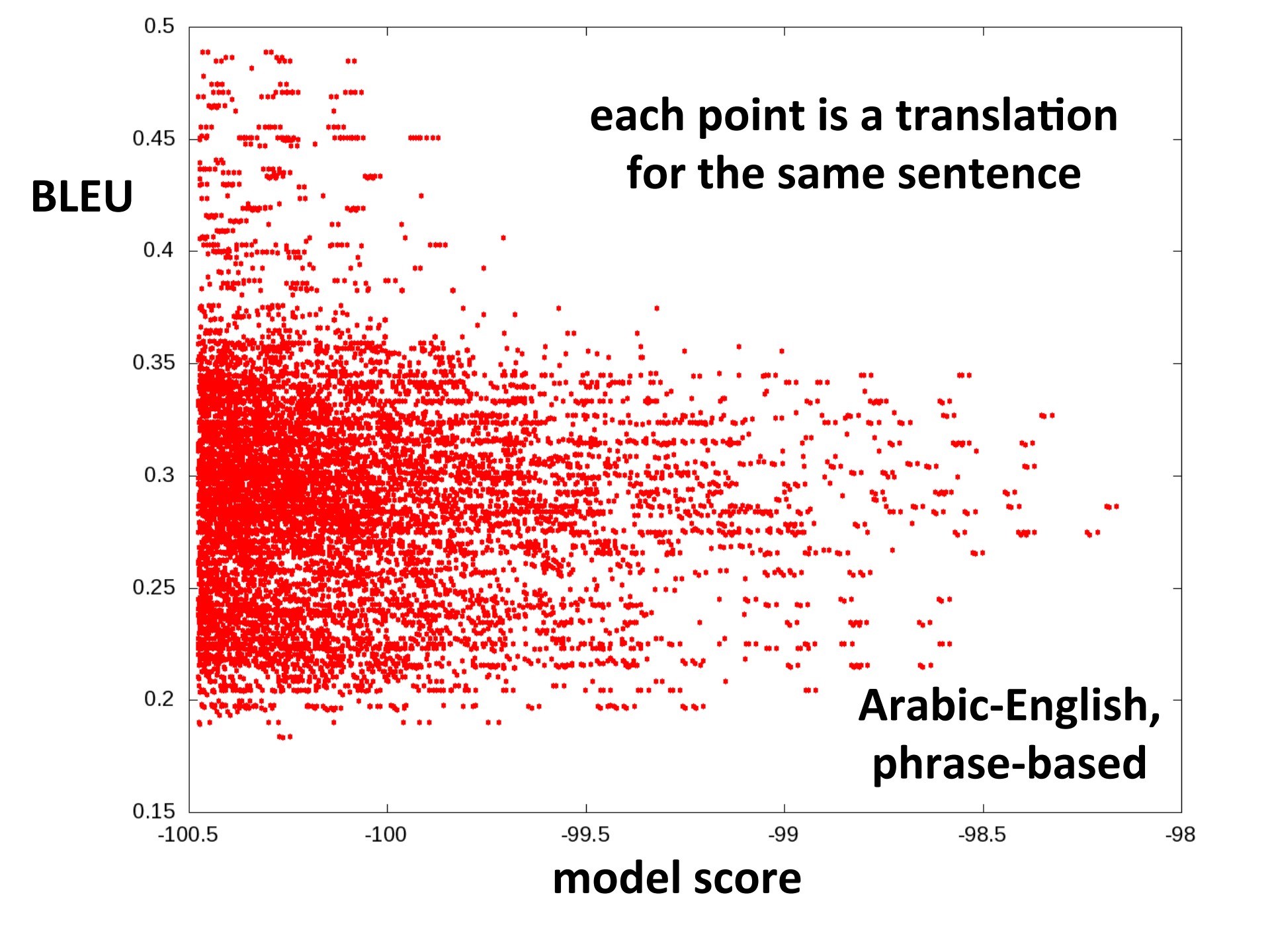
minimize the cost of the decoder output

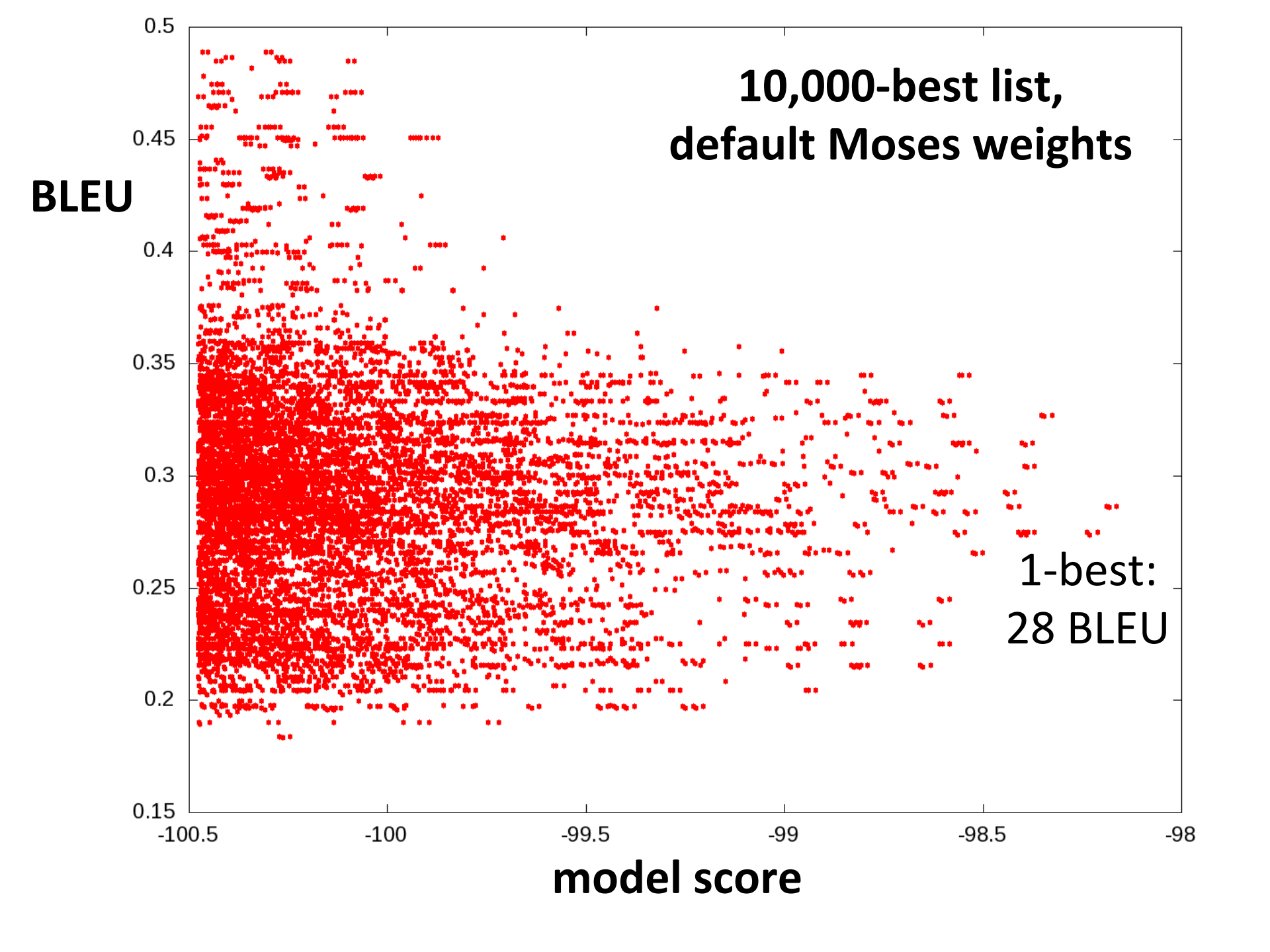
intractable in general – how can we solve it?

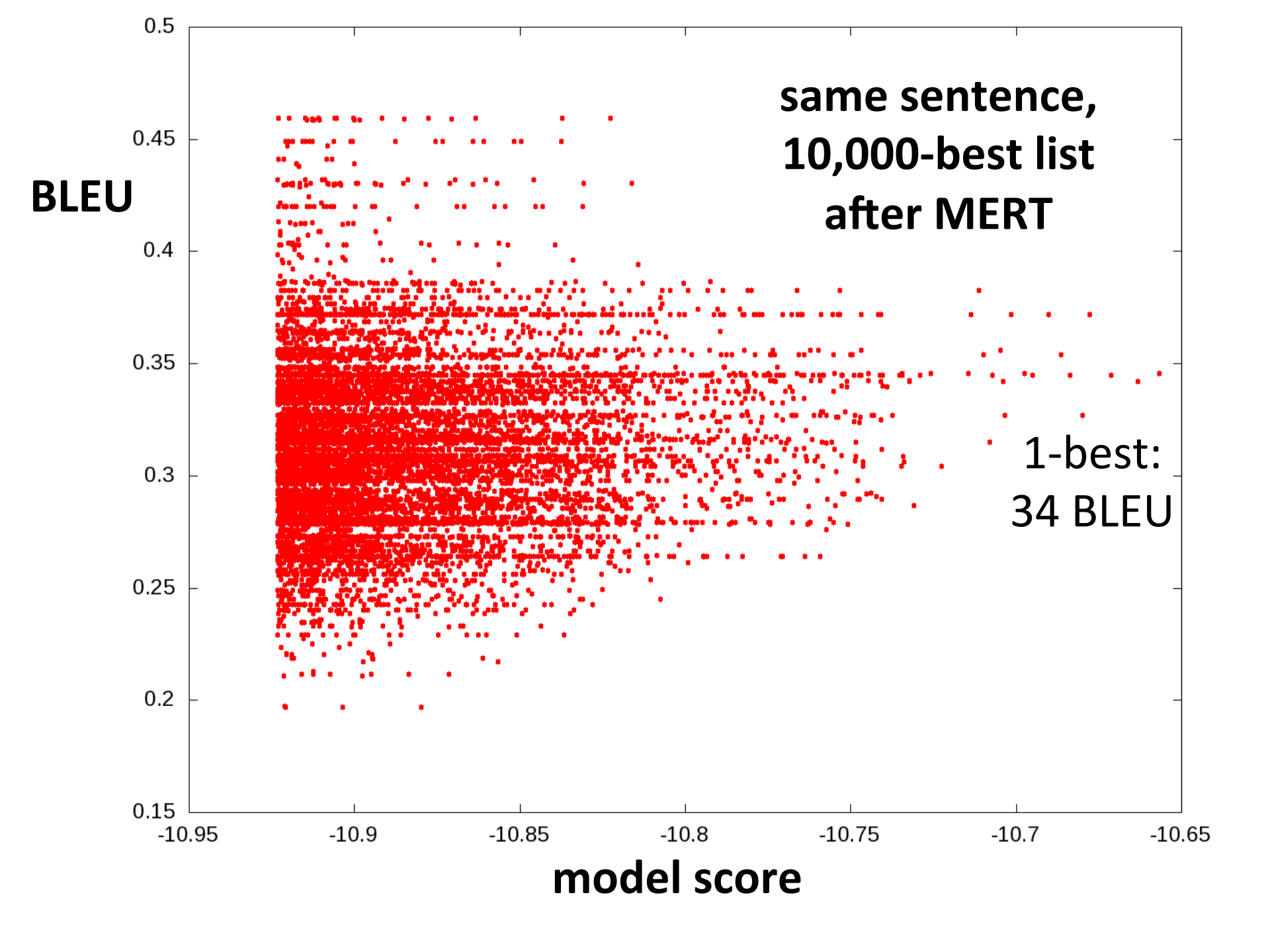
$$\min_{\theta} \quad \text{cost}\left(\{\boldsymbol{y}^{(i)}\}_{i=1}^{N}, \left\{ \underset{\boldsymbol{y},\boldsymbol{h}}{\text{argmax}} \quad \boldsymbol{\theta}^{\top}\boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}, \boldsymbol{h}) \right\}^{N}\right)$$

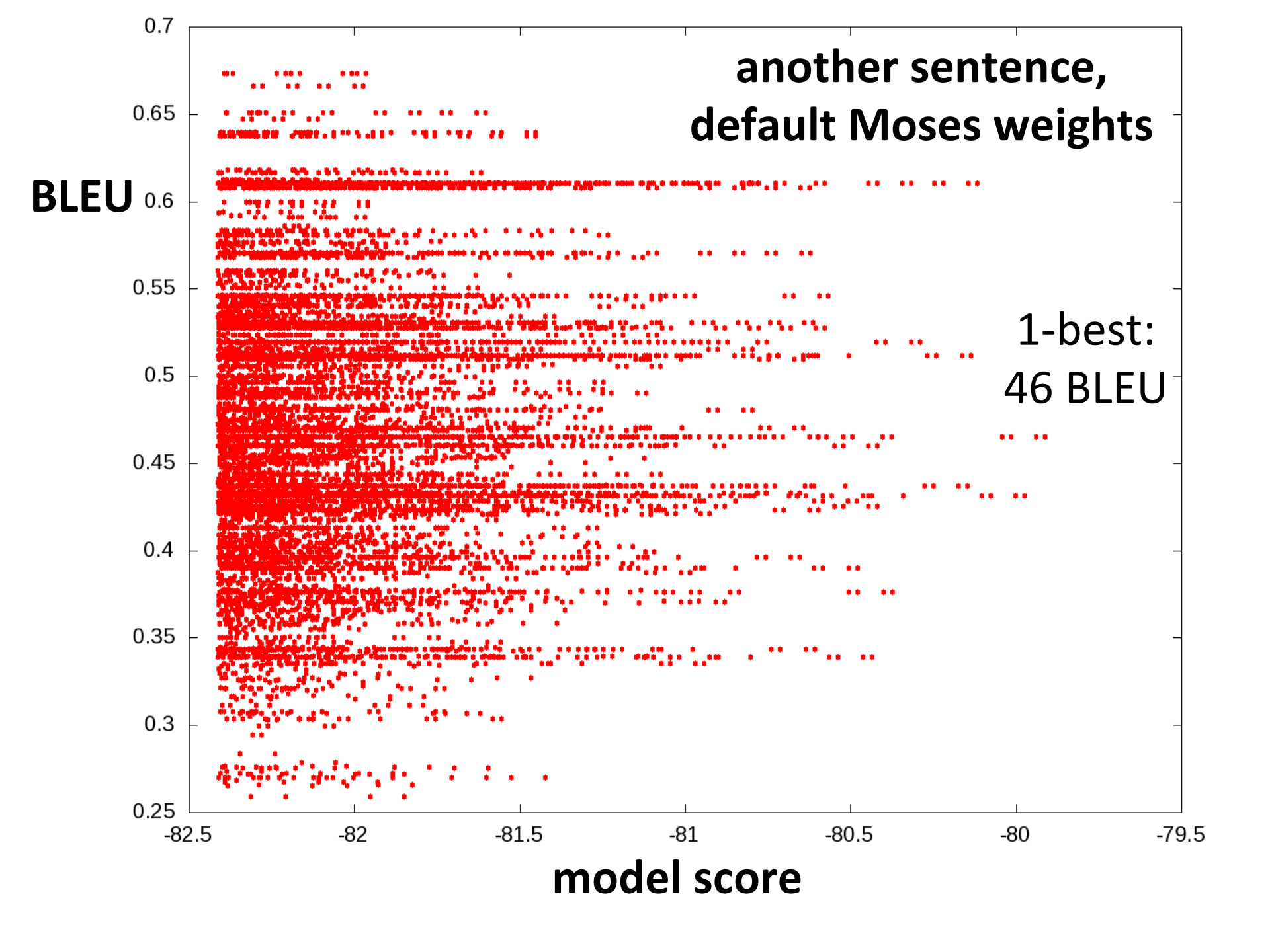
generate k-best lists of translations,
approximately minimize cost on k-best lists,
repeat with new parameters
(pool k-best lists across iterates)

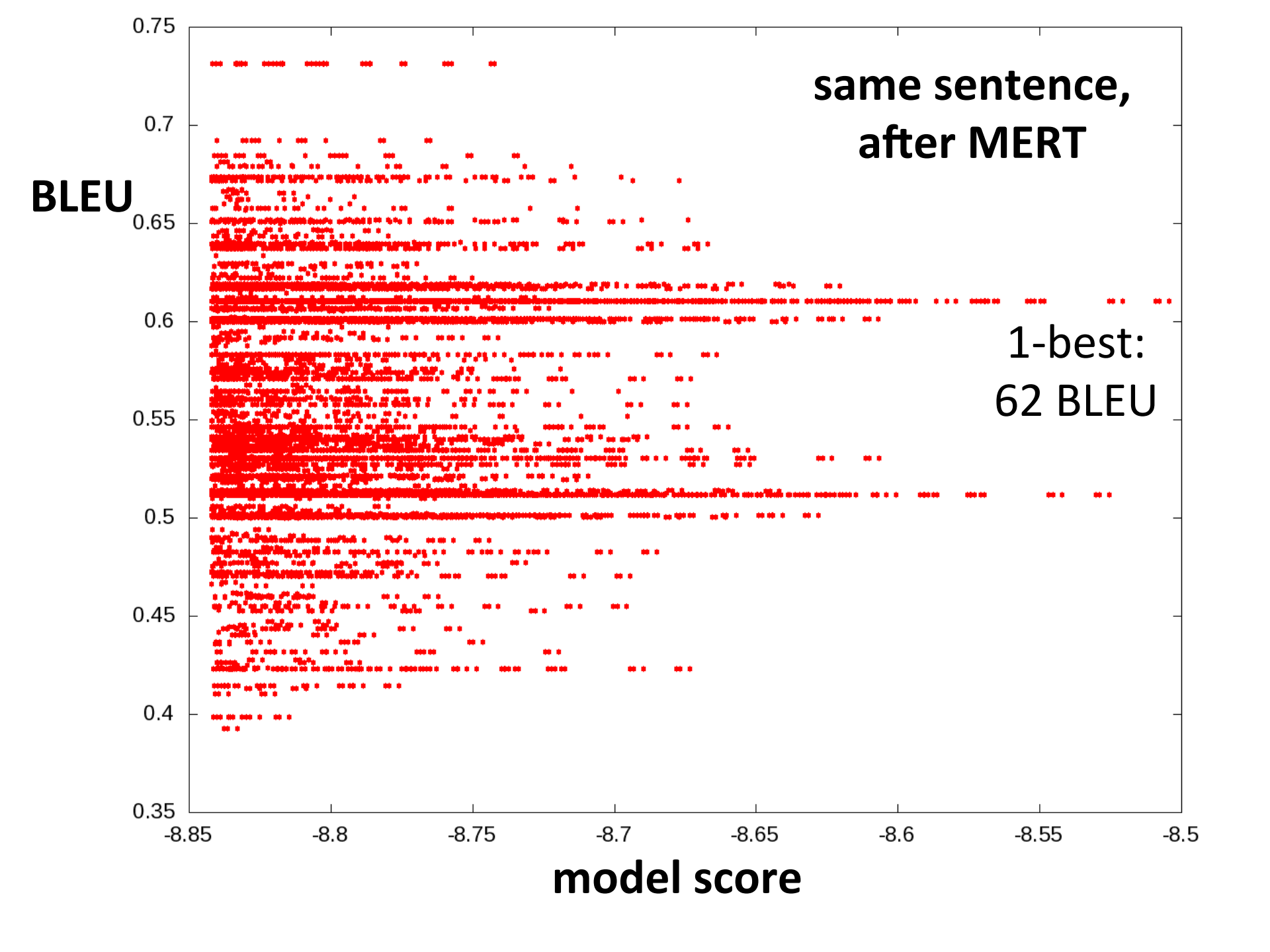

BLEU
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
-100.5
-100
-99.5
-99
-98.5
-98
model score

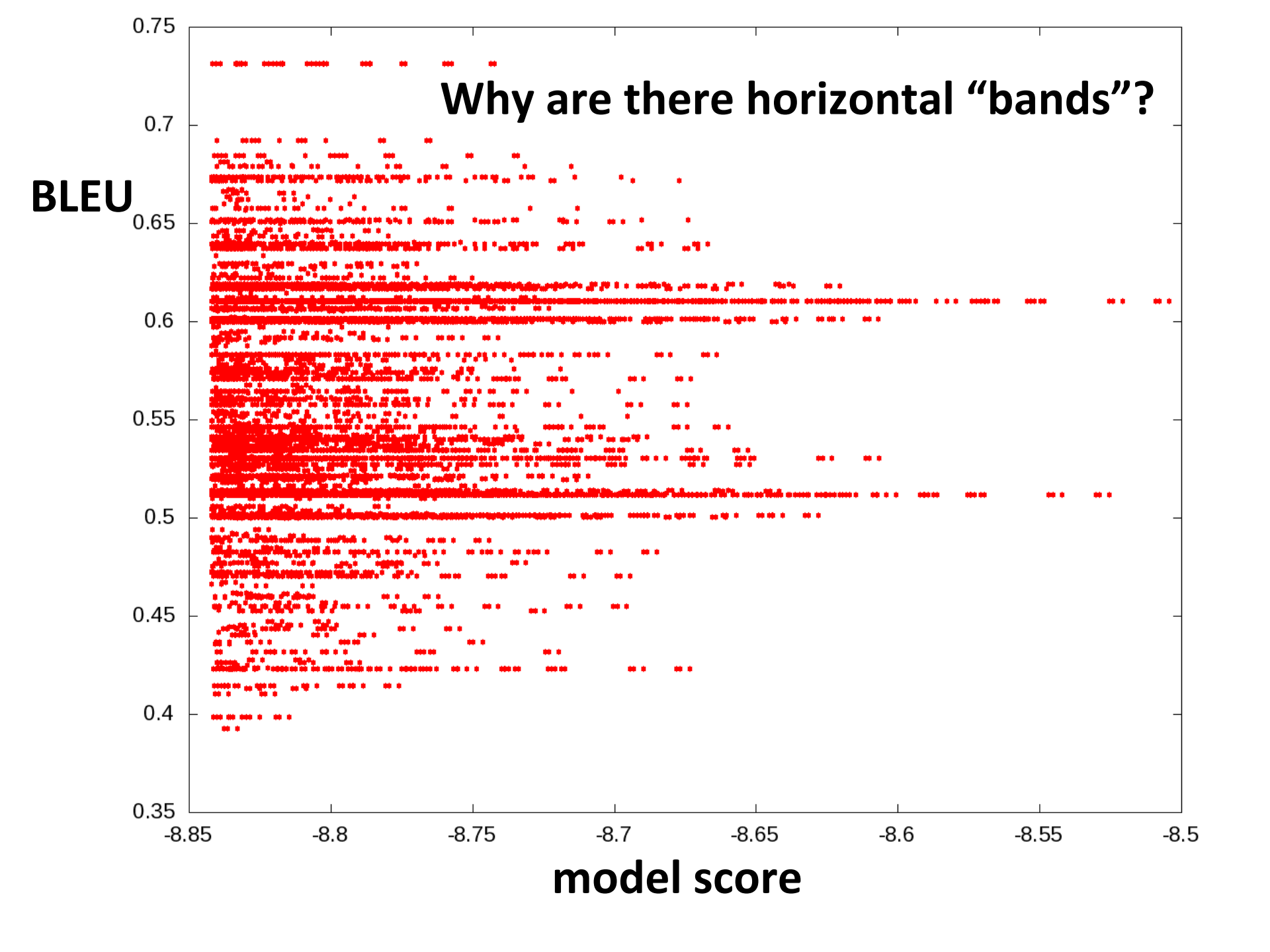

BLEU
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
-100.5
-100
-99.5
-99
-98.5
-98
model score
each point is a translation
for the same sentence
Arabic-English,
phrase-based

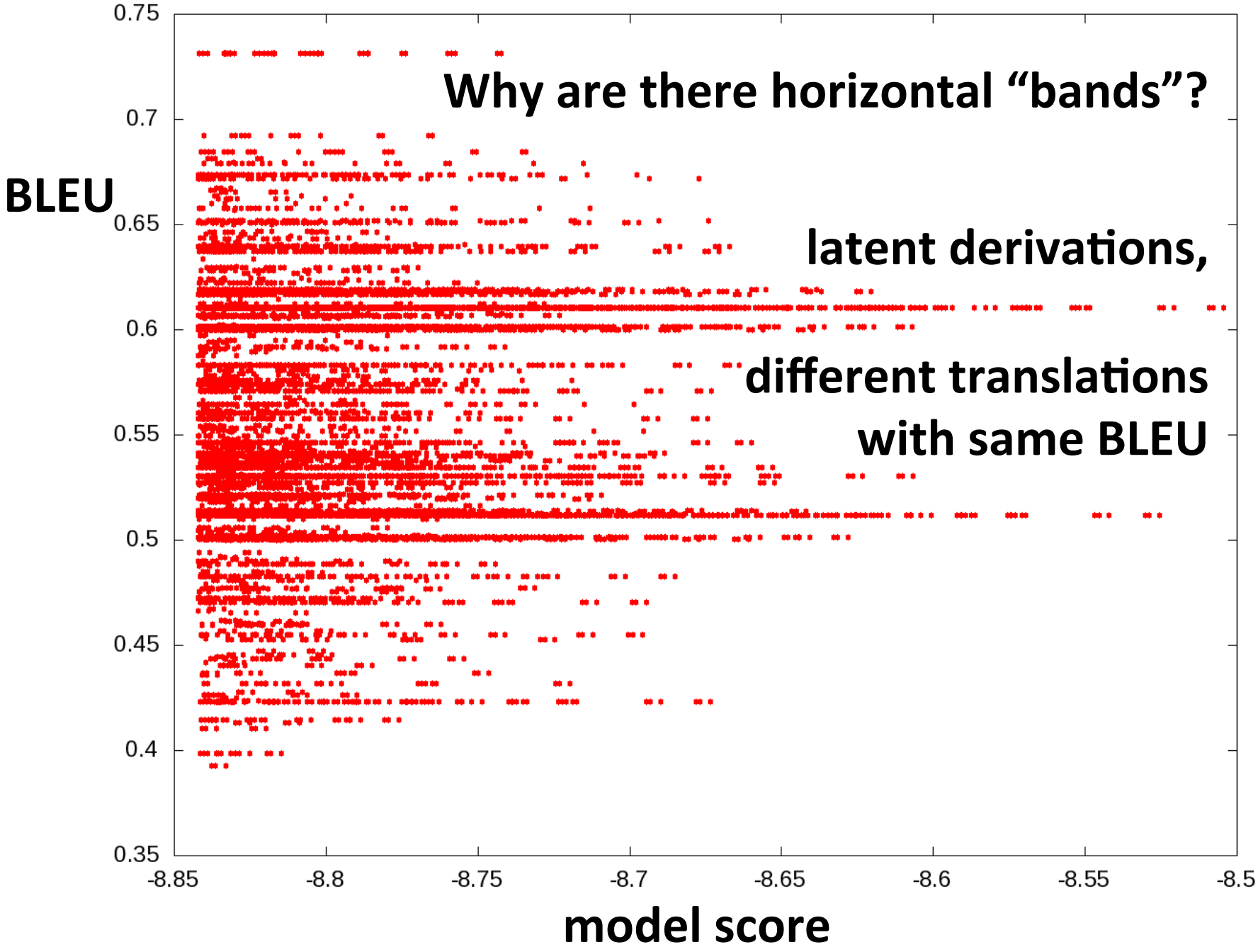

10,000-best list,
default Moses weights
BLEU
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
-100.5
-100
-99.5
-99
-98.5
-98
model score
1-best:
28 BLEU


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
model score
1-best:
46 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


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


Why are there horizontal “bands”?
BLEU
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
model score


Why are there horizontal “bands”?
latent derivations,
different translations
with same 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

Minimum Error Rate Training (MERT)

What are some issues with this loss function?

- Discontinuous & non-convex → optimization relies on randomized search
- No regularization → leads to overfitting

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

Many researchers tried to improve MERT:

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

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

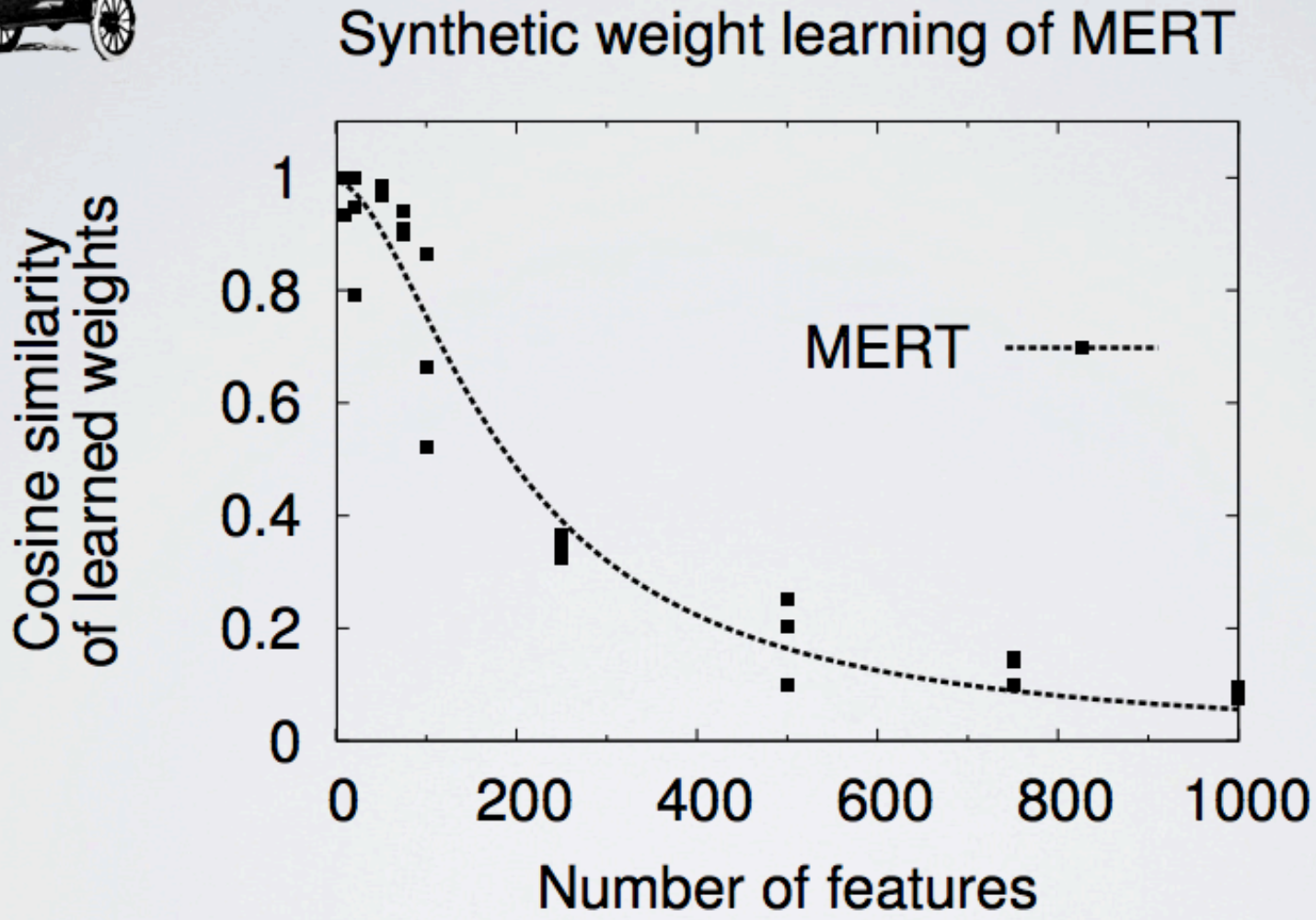
Stabilizing MERT (Foster & Kuhn, 2009)

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

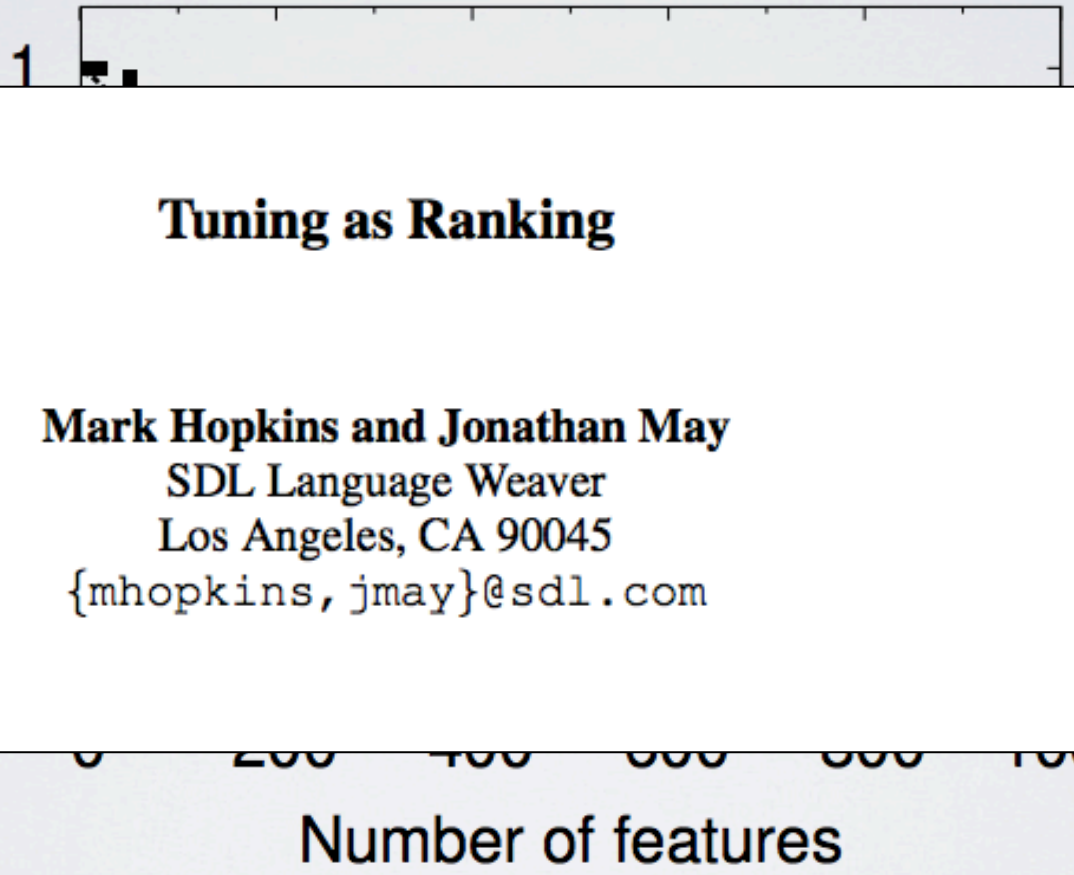
MERT ***doesn't scale***



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

MERT *doesn't scale*

Synthetic weight learning of MERT

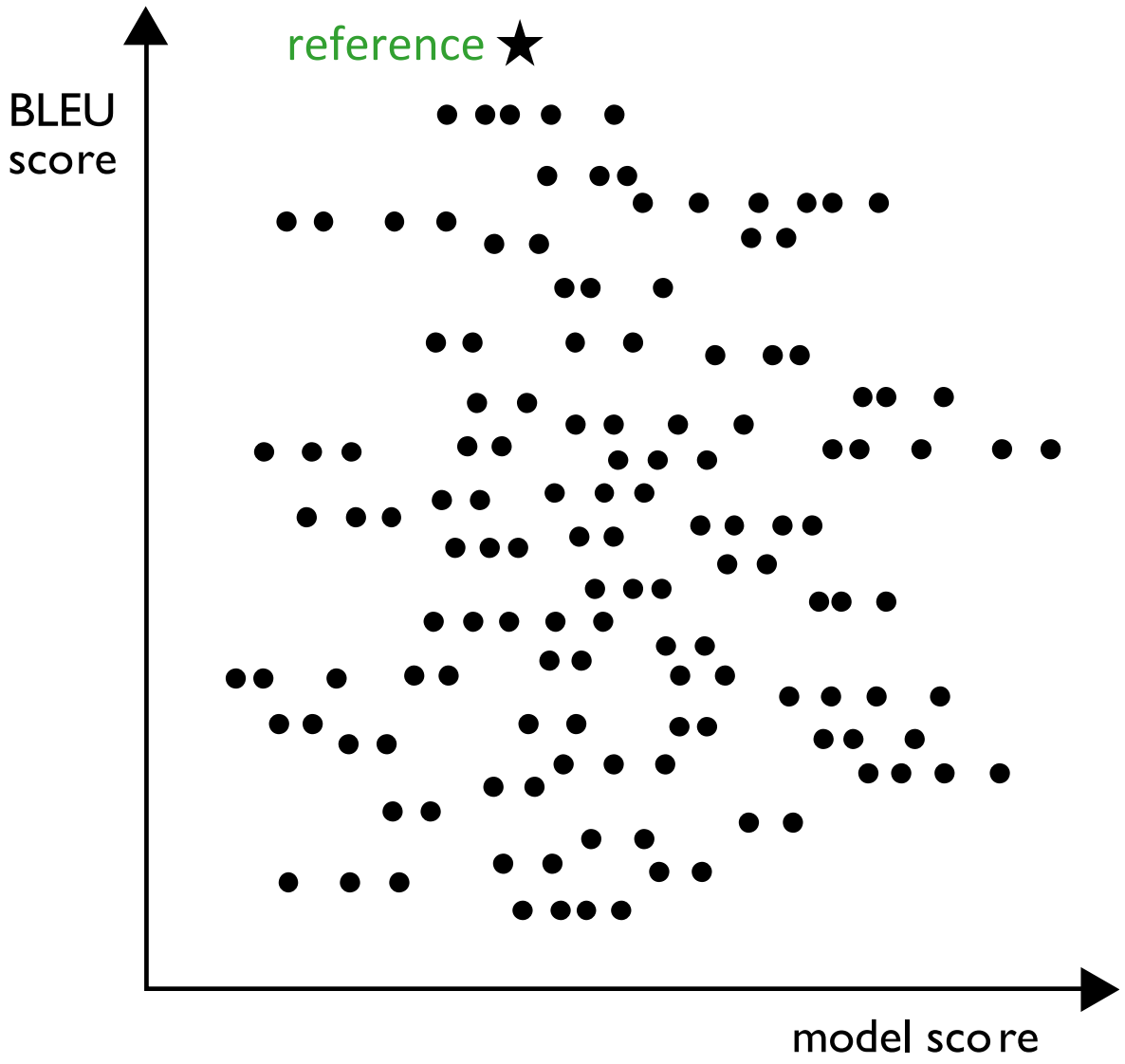


Tuning as Ranking

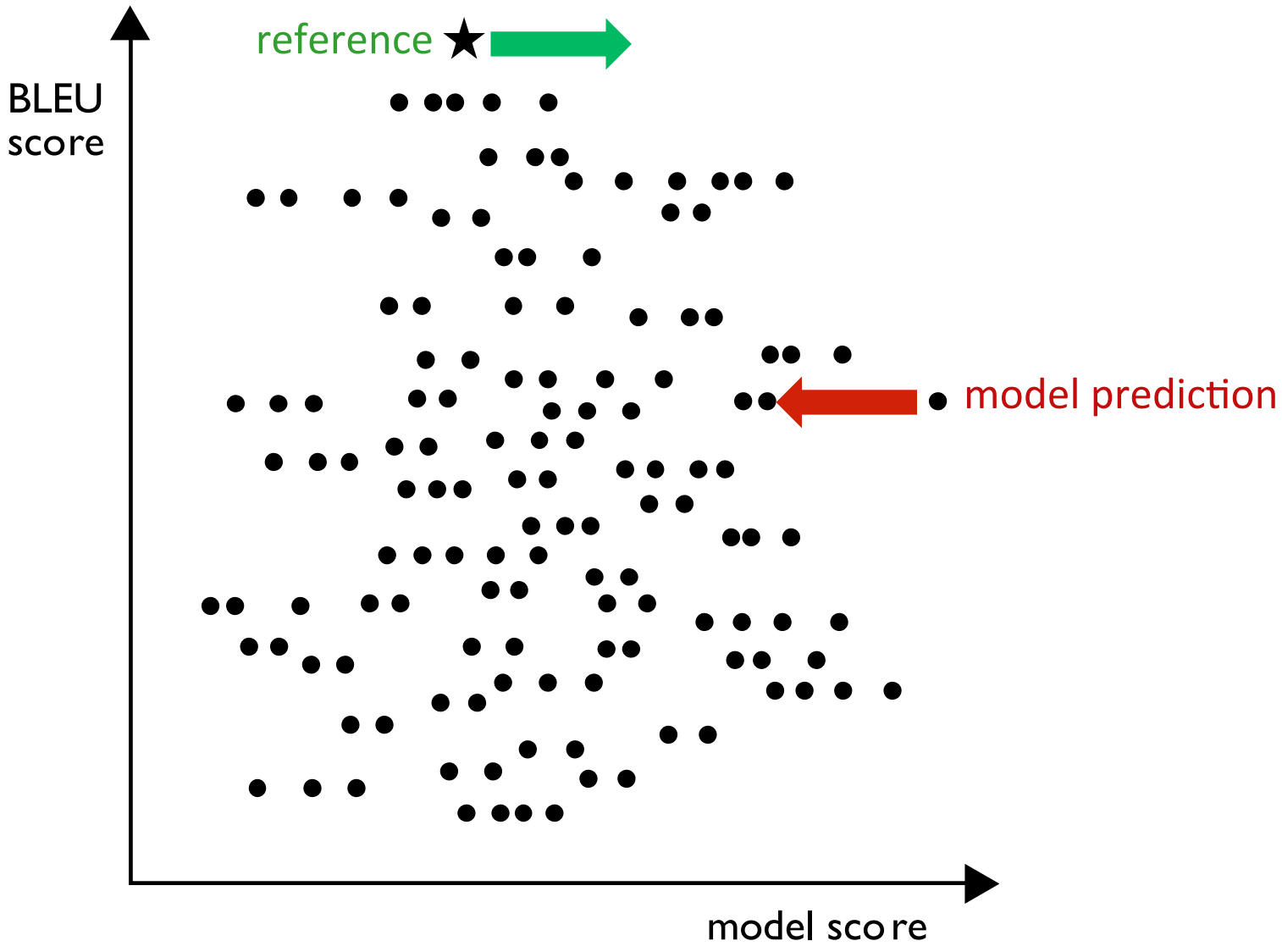
Mark Hopkins and Jonathan May
SDL Language Weaver
Los Angeles, CA 90045
{mhopkins, jmay}@sdl.com

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

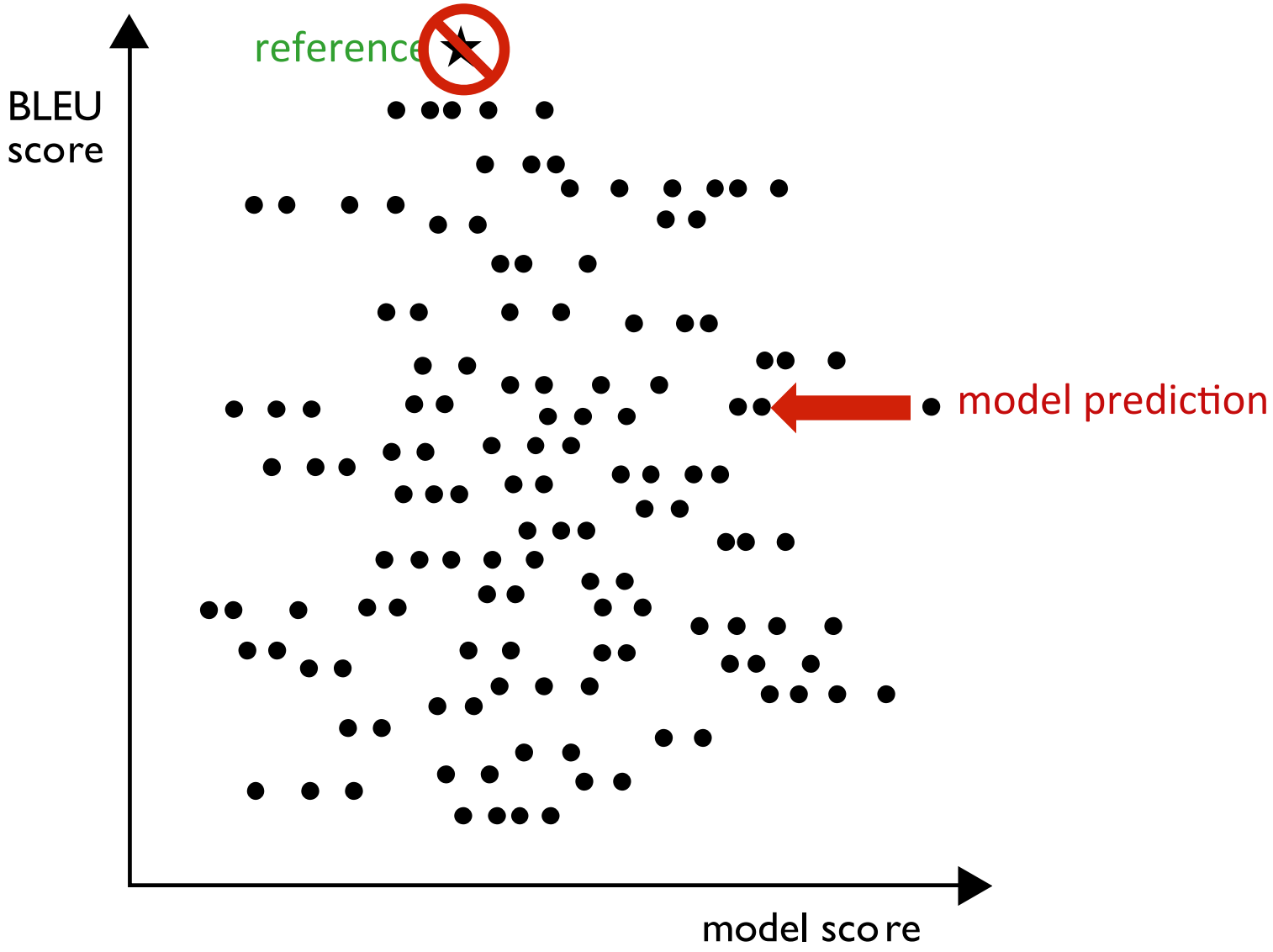
Perceptron Loss



Perceptron Loss

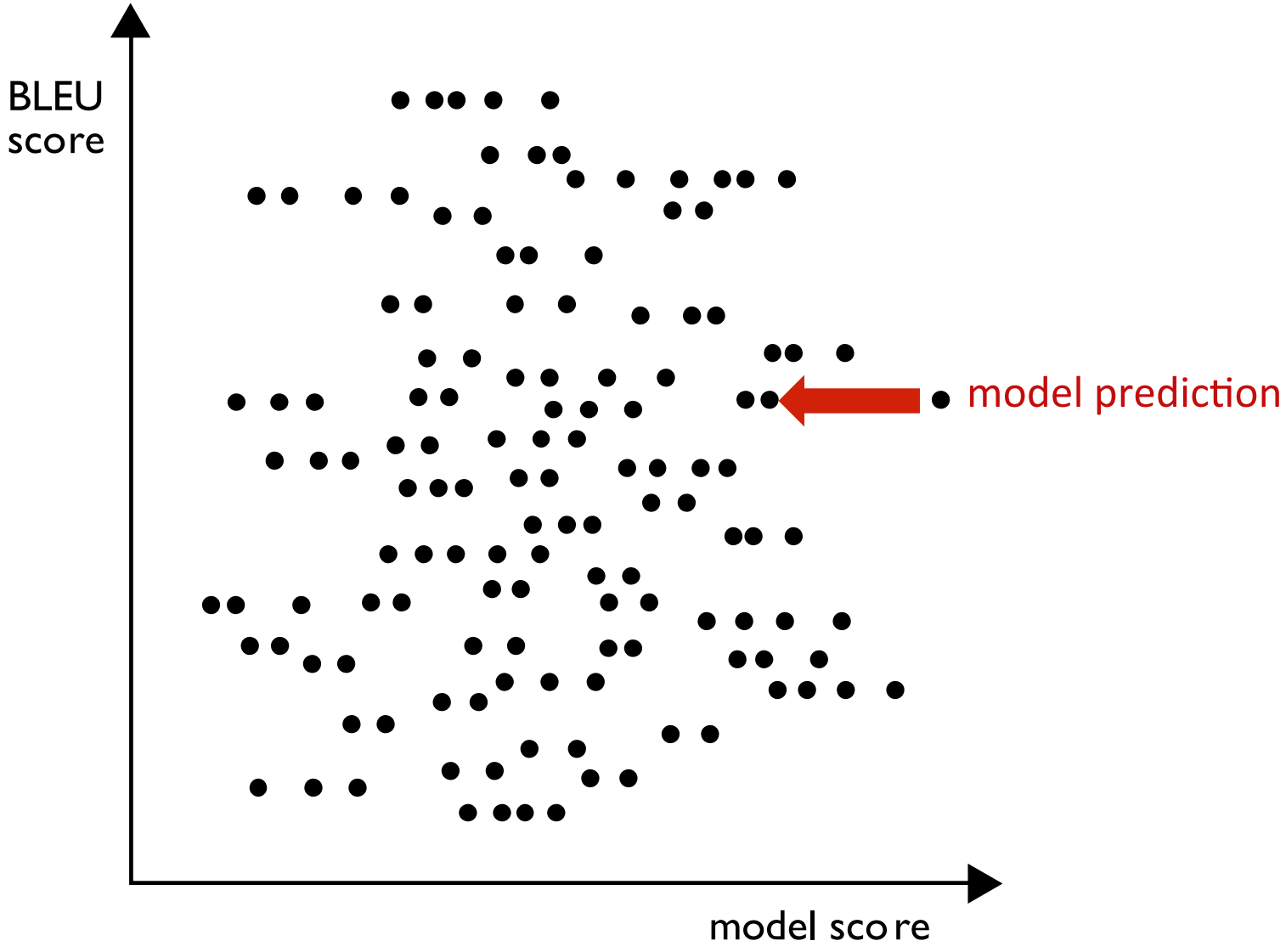


Perceptron Loss for MT?



k-Best Perceptron for MT

(Liang et al., 2006)

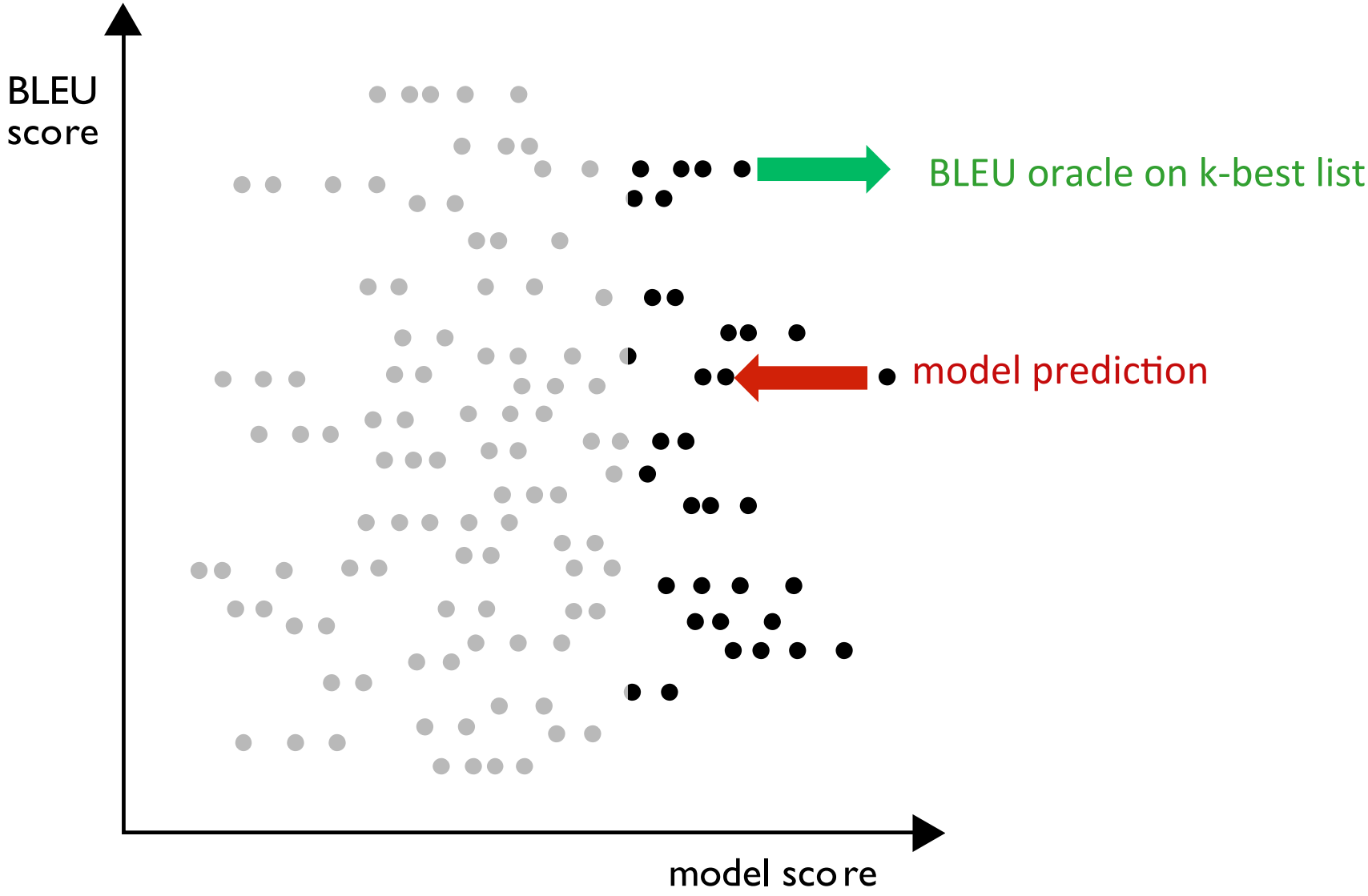


k-Best Perceptron for MT

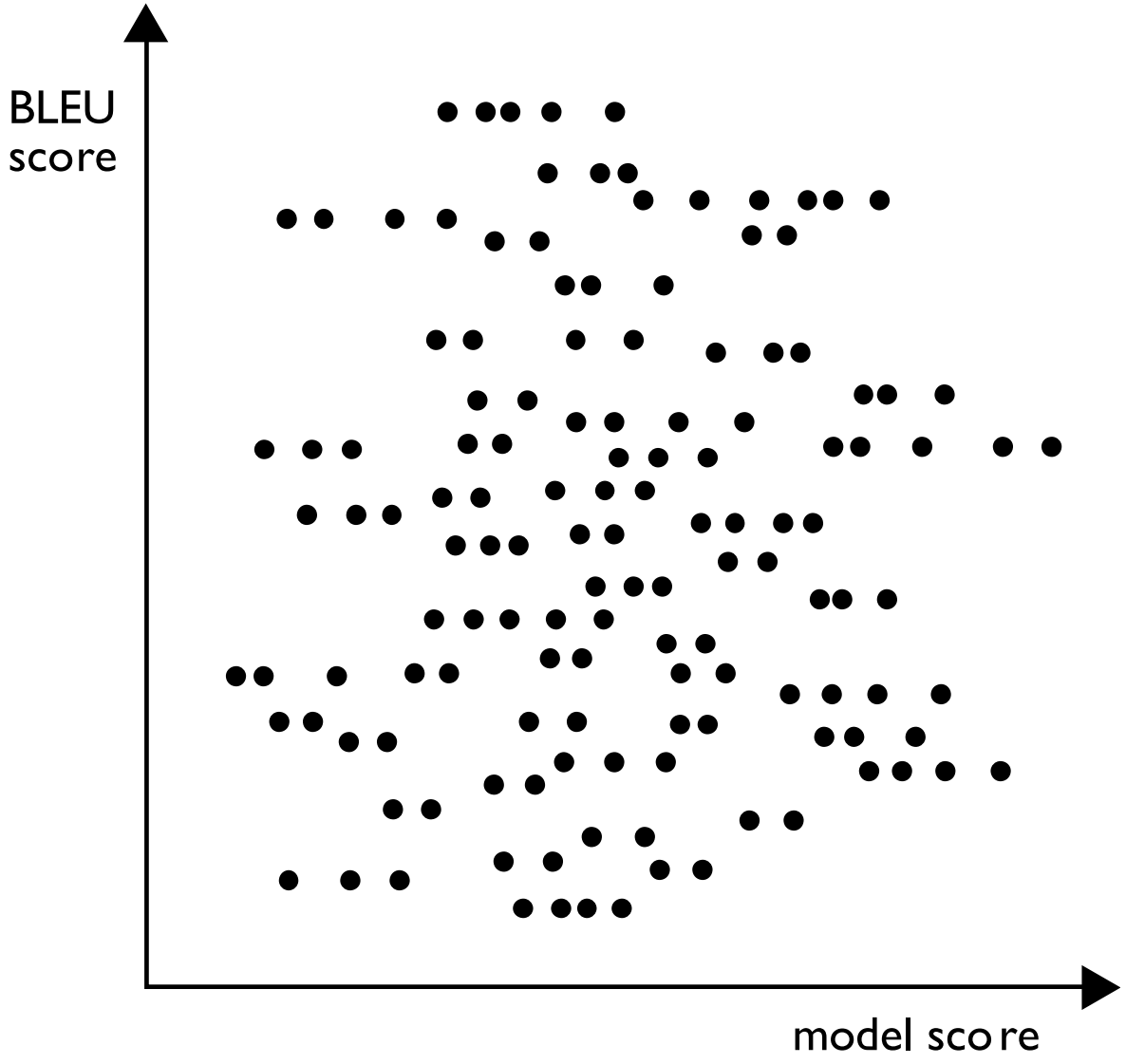
(Liang et al., 2006)

k-Best Perceptron for MT

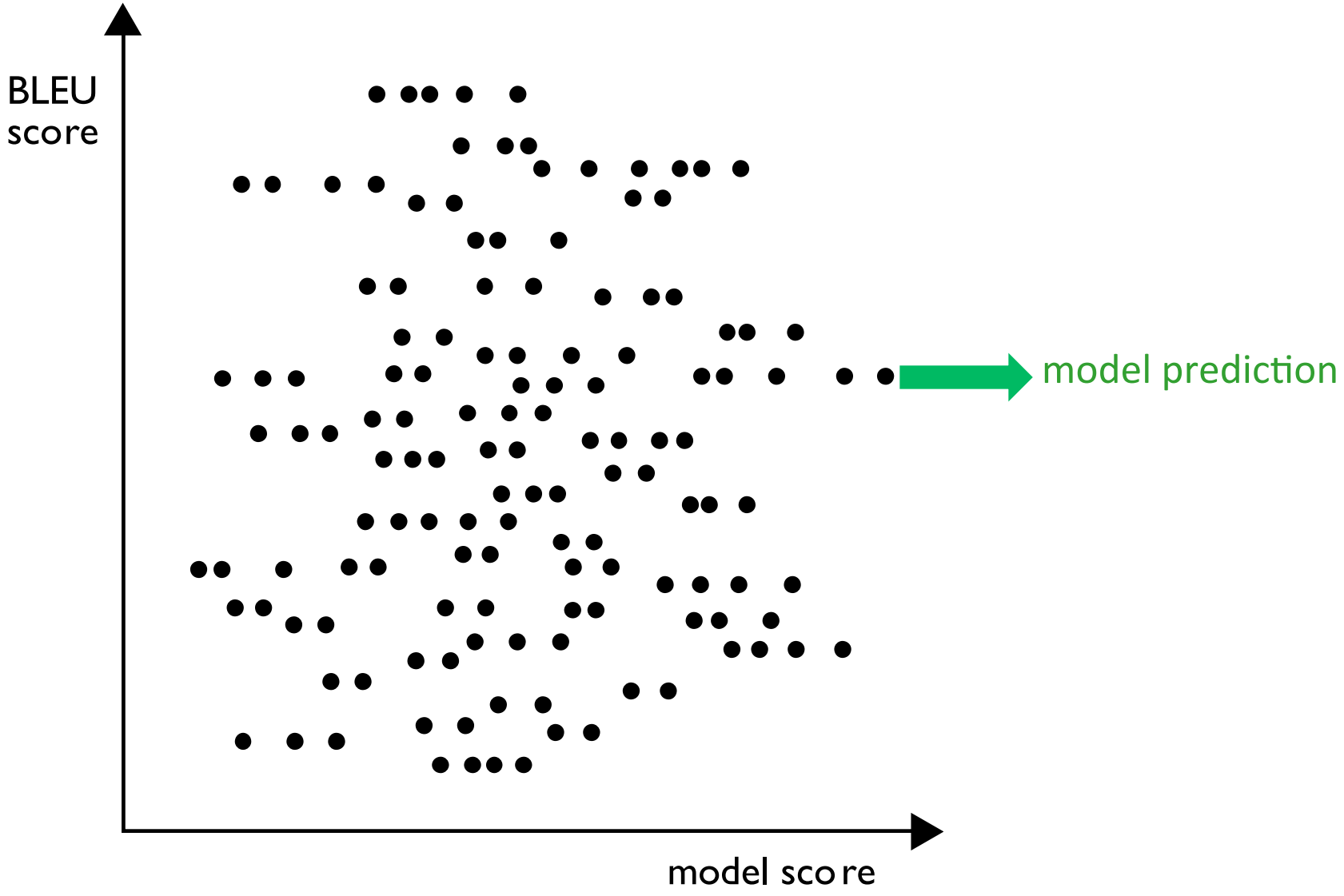
(Liang et al., 2006)



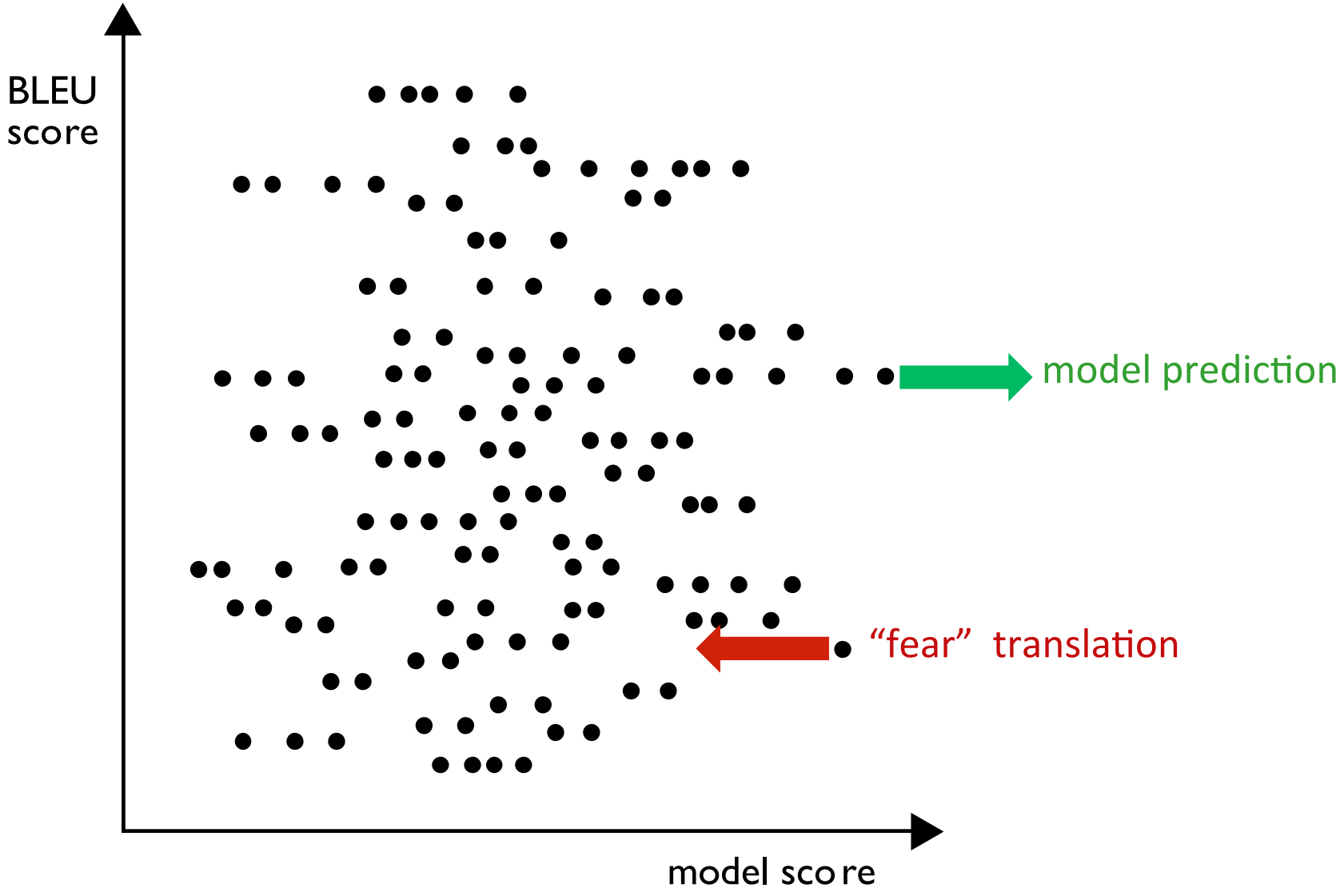
Ramp Loss Minimization



Ramp Loss Minimization

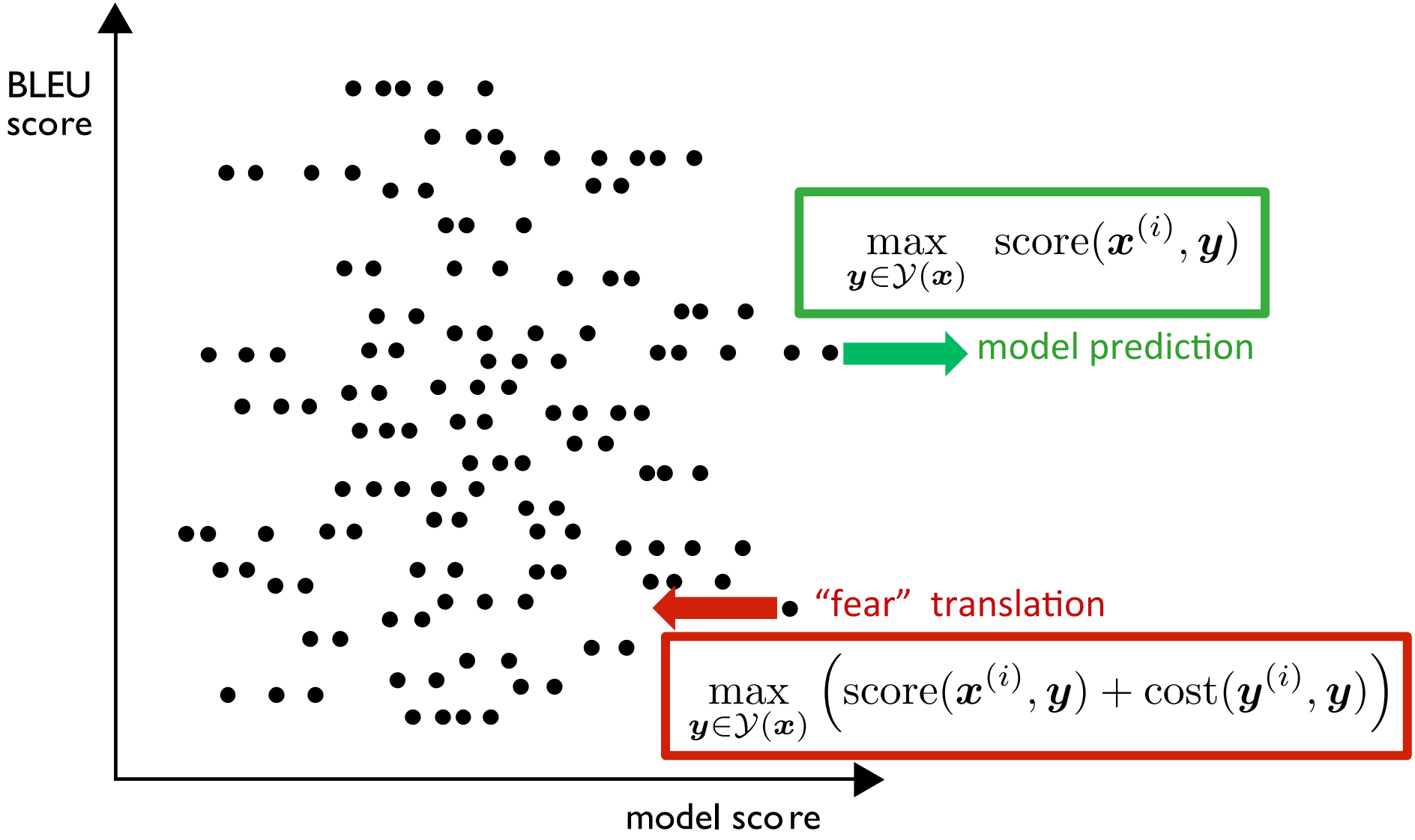


Ramp Loss Minimization



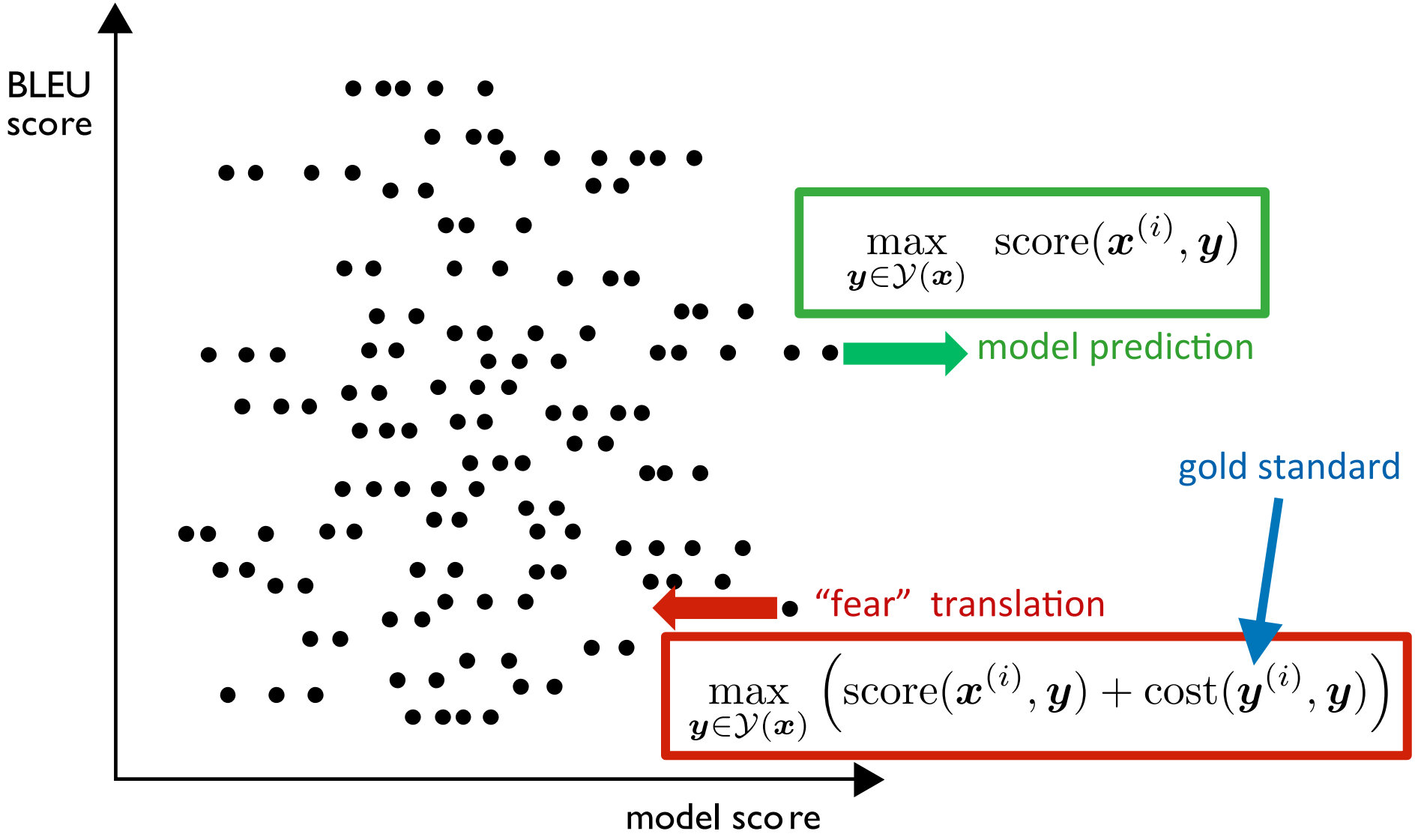
“Fear” Ramp Loss

(Do et al., 2008)



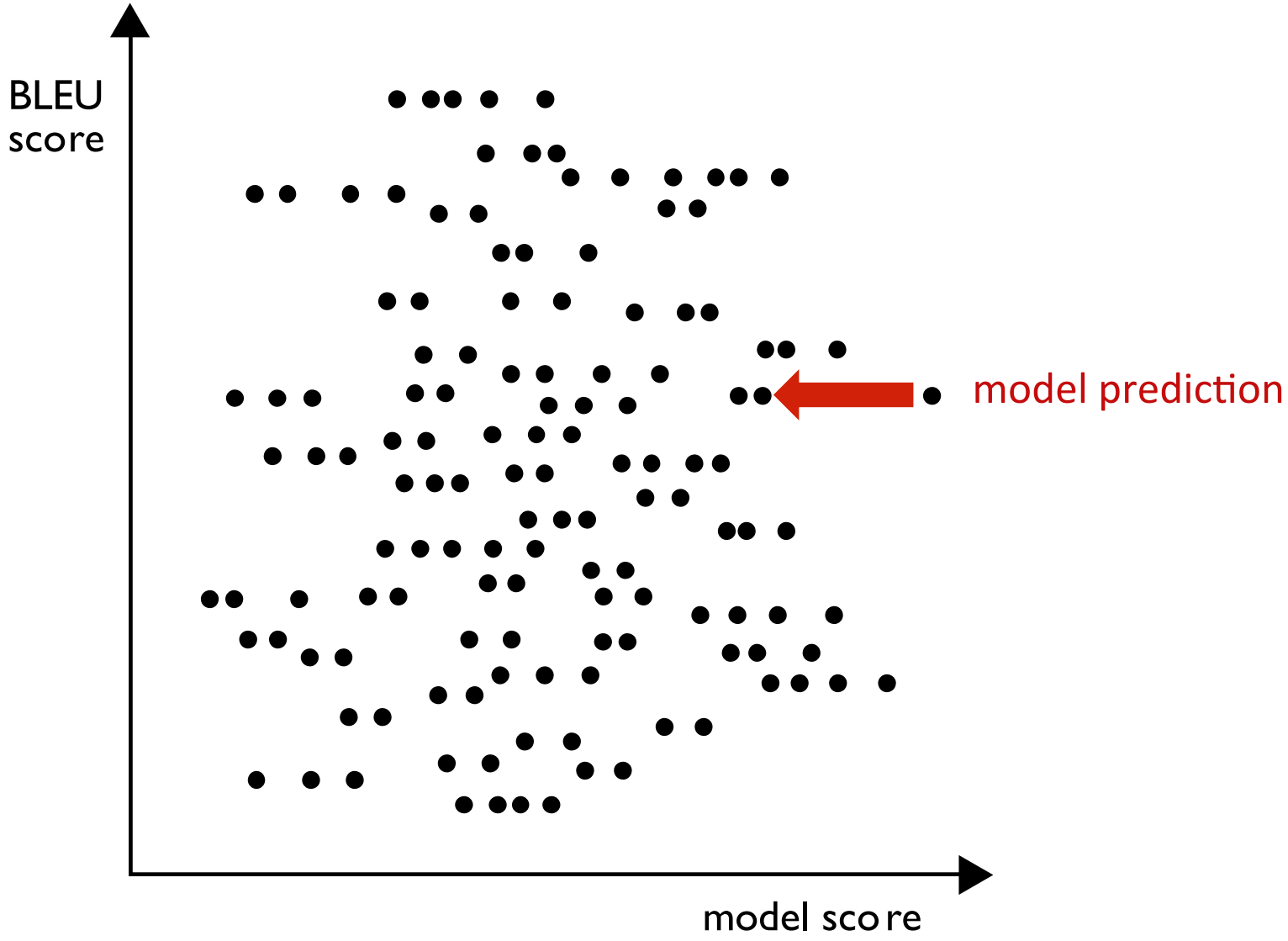
“Fear” Ramp Loss

(Do et al., 2008)



“Hope” Ramp Loss

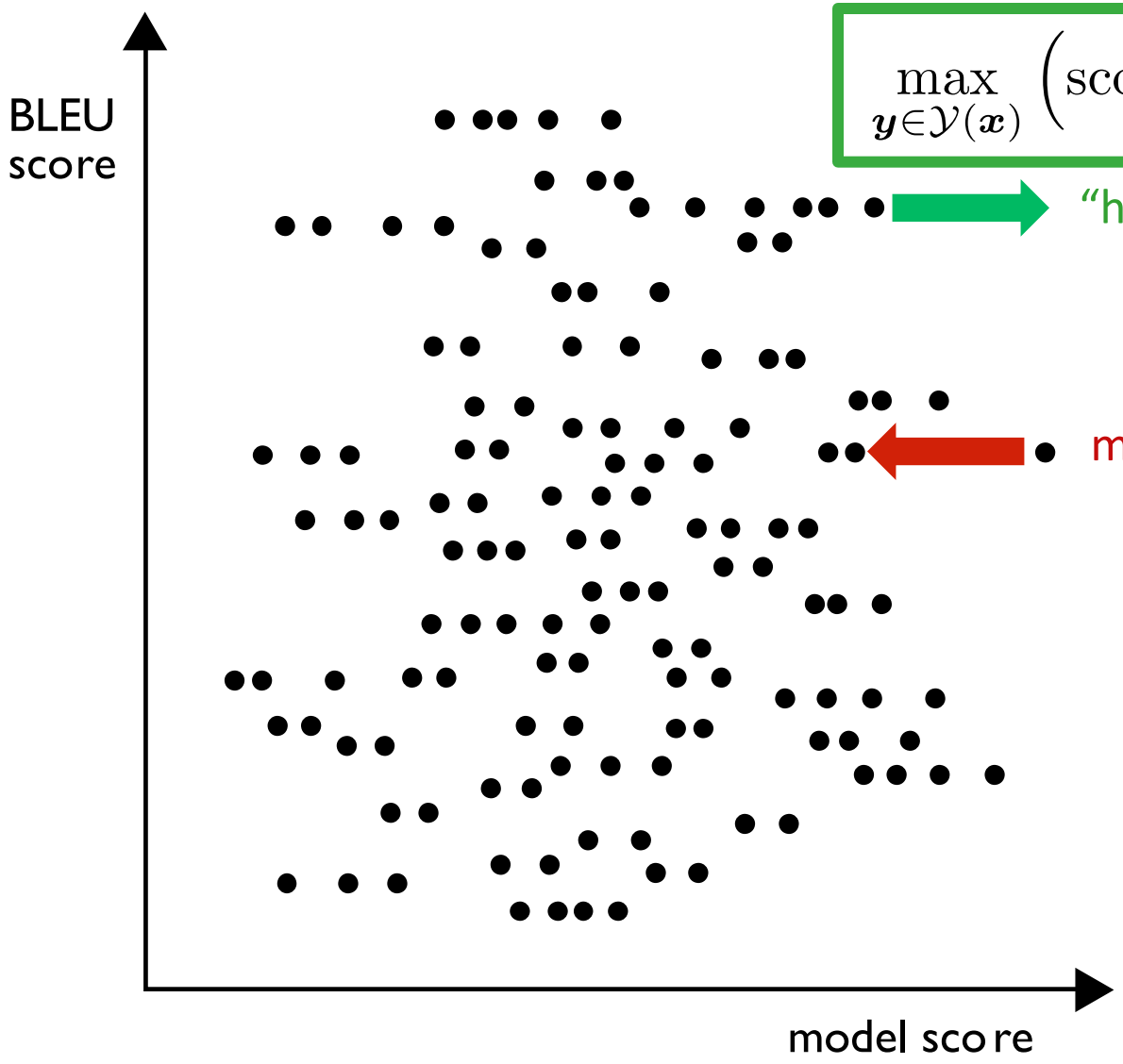
(McAllester & Keshet, 2011; Liang et al., 2006)



“Hope” Ramp Loss

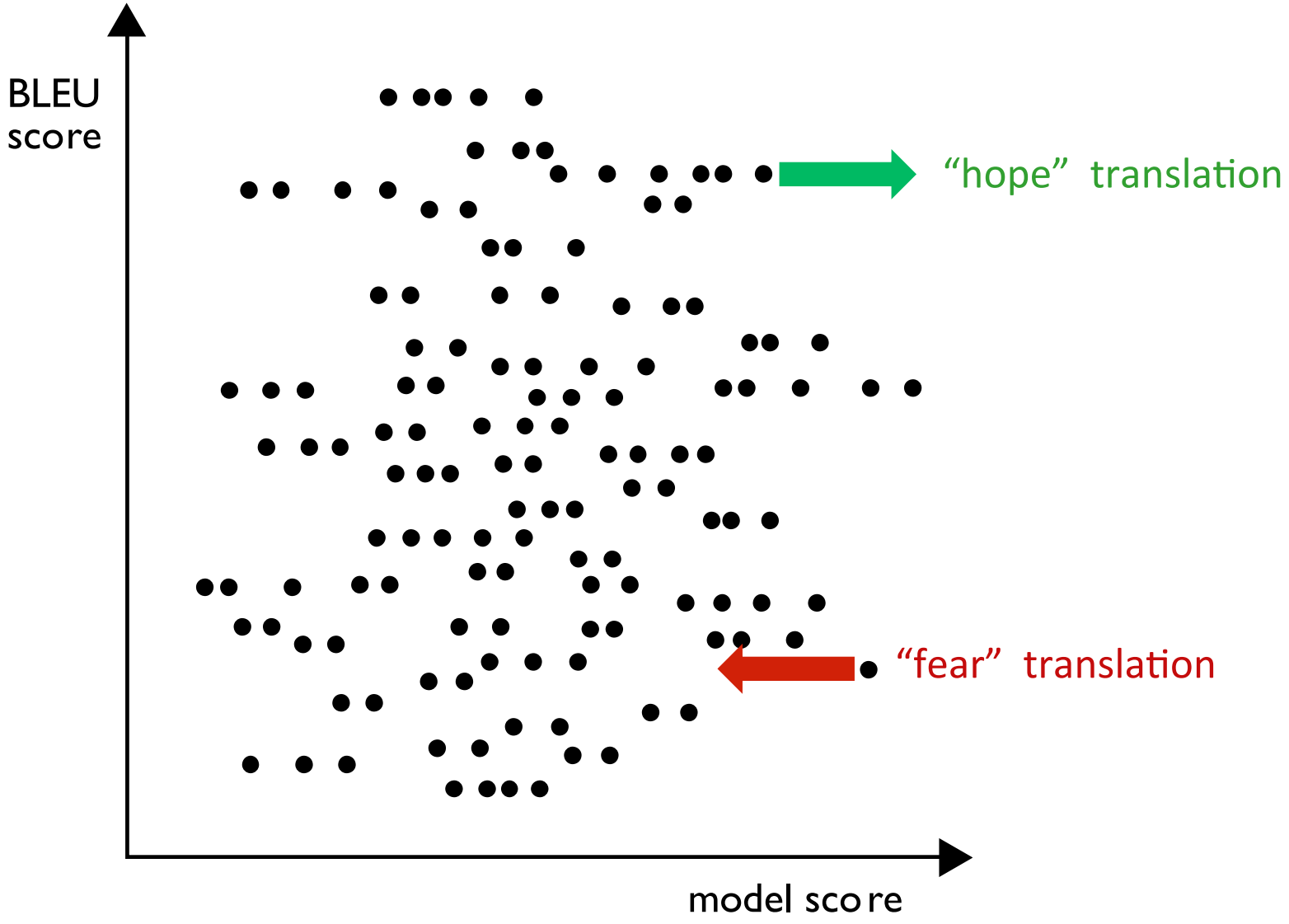
(McAllester & Keshet, 2011; Liang et al., 2006)

$$\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \left(\text{score}(\boldsymbol{x}^{(i)}, \boldsymbol{y}) - \text{cost}(\boldsymbol{y}^{(i)}, \boldsymbol{y}) \right)$$



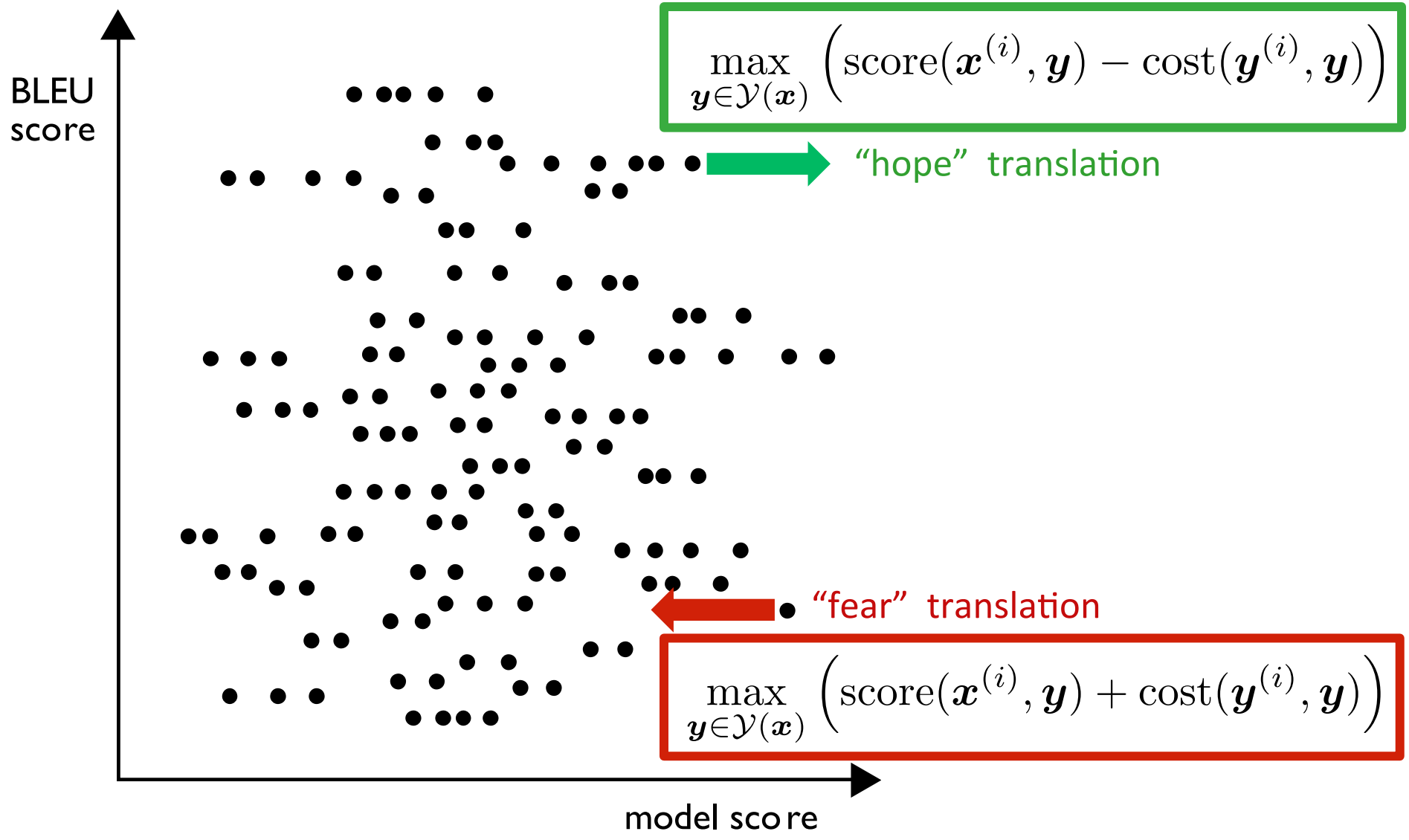
“Hope-Fear” Ramp Loss

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



“Hope-Fear” Ramp Loss

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



Experiments

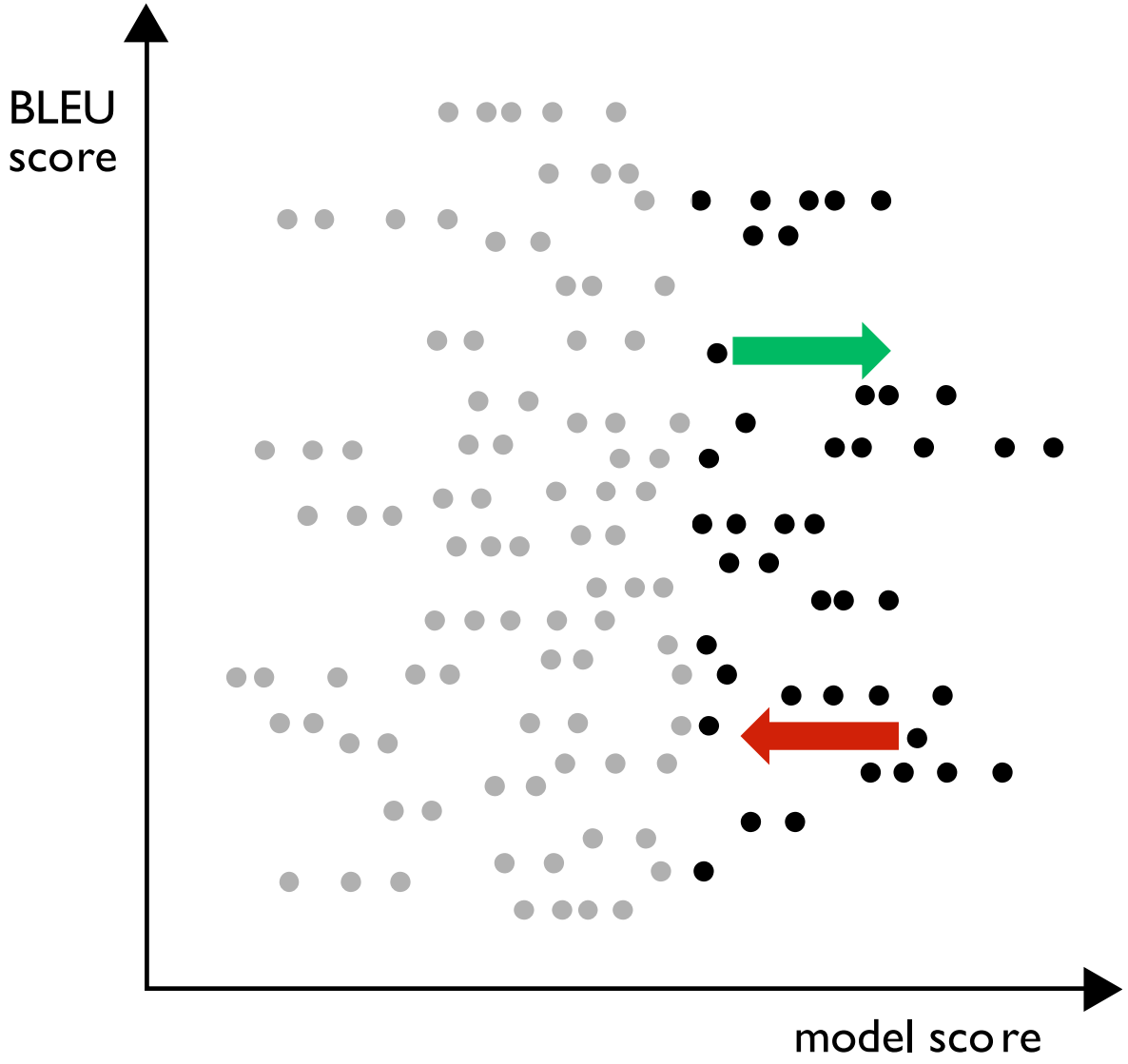
(Gimpel, 2012)

averages over 8 test sets across 3 language pairs

	Moses	Hiero
	(%BLEU)	(%BLEU)
MERT	35.9	37.0
Fear Ramp (away from bad)	34.9	34.2
Hope Ramp (toward good)	35.2	36.0
Hope-Fear Ramp (toward good + away from bad)	35.7	37.0

Pairwise Ranking Optimization

(Hopkins & May, 2011)



Roadmap

- words, morphology, lexical semantics
- text classification
- simple neural methods for NLP
- language modeling and word embeddings
- recurrent/recursive/convolutional networks in NLP
- sequence labeling, HMMs, dynamic programming
- syntax and syntactic parsing
- machine translation
- semantics

Semantics

- we talked a lot about lexical semantics:
  - word sense, WordNet, word embeddings/clusters

- how about semantics beyond the word level?

Compositional Semantics

- “how should the meanings of words combine to create the meaning of something larger?”
- there’s currently a lot of work in producing vector representations of sentences and documents
- simplest case: how should two word vectors be combined to create a vector for a bigram?
- lots of work in this area in the neural era, but earlier work began ~2007

Evaluating Compositional Semantics

- compute similarity of two bigrams under your model, then compute correlation with human judgments:

		BigramSim
television programme	tv set	5.8
training programme	education course	5.7
bedroom window	education officer	1.3

(Mitchell and Lapata, 2010)

COGNITIVE SCIENCE
A Multidisciplinary Journal

Cognitive Science 34 (2010) 1388–1429

ISSN: 0364-0213 print / 1551-6709 online
DOI: 10.1111/j.1551-6709.2010.01106.x

Composition in Distributional Models of Semantics

Jeff Mitchell, Mirella Lapata

School of Informatics, University of Edinburgh

Received 6 May 2009; received in revised form 22 March 2010; accepted 25 March 2010

Abstract

Vector-based models of word meaning have become increasingly popular in cognitive science. The appeal of these models lies in their ability to represent meaning simply by using distributional information under the assumption that words occurring within similar contexts are semantically similar. Despite their widespread use, vector-based models are typically directed at representing words in isolation, and methods for constructing representations for phrases or sentences have received little attention in the literature. This is in marked contrast to experimental evidence (e.g., in sentential priming) suggesting that semantic similarity is more complex than simply a relation between isolated words. This article proposes a framework for representing the meaning of word combinations in vector space. Central to our approach is vector composition, which we operationalize in terms of additive and multiplicative functions. Under this framework, we introduce a wide range of composition models that we evaluate empirically on a phrase similarity task.

Bigram Composition Functions

J. Mitchell, M. Lapata/Cognitive Science 34 (2010)

Table 5
Composition functions considered in our experiments

Model	Function
Additive	$p_i = u_i + v_i$
Kintsch	$p_i = u_i + v_i + n_i$
Multiplicative	$p_i = u_i \cdot v_i$
Tensor product	$p_{i,j} = u_i \cdot v_j$
Circular convolution	$p_i = \sum_j u_j \cdot v_{i-j}$
Weighted additive	$p_i = \alpha v_i + \beta u_i$
Dilation	$p_i = v_i \sum_j u_j u_j + (\lambda - 1) u_i \sum_j u_j v_j$
Head only	$p_i = v_i$
Target unit	$p_i = v_i(t_1 t_2)$

Bigram Similarity Results

J. Mitchell, M. Lapata/Cognitive Science 34 (2010)

Table 6
Correlation coefficients of model predictions with subject similarity ratings (Spearman's ρ) using a simple semantic space

Model	Adjective–Noun	Noun–Noun	Verb–Object
Additive	.36	.39	.30
Kintsch	.32	.22	.29
Multiplicative	.46	.49	.37
Tensor product	.41	.36	.33
Convolution	.09	.05	.10
Weighted additive	.44	.41	.34
Dilation	.44	.41	.38
Target unit	.43	.34	.29
Head only	.43	.17	.24
Humans	.52	.49	.55

Why does multiplication work?

- these vectors are built from co-occurrence counts (like in the first part of Assignment 1)
- so element-wise multiplication is like performing an AND operation on context counts

- when using skip-gram word vectors (or other neural network-derived vectors), addition often works better

A Comparison of Vector-based Representations for Semantic Composition

William Blacoe and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
w.b.blacoe@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we address the problem of modeling compositional meaning for phrases and sentences using distributional methods. We experiment with several possible combinations of representation and composition, exhibiting varying degrees of sophistication. Some are shallow while others operate over syntactic structure, rely on parameter learning, or require access to very large corpora. We find that shallow approaches are as good as more computationally intensive alternatives with regards to two particular tests: (1) phrase similarity and (2) paraphrase detection. The sizes of the involved training corpora and the

word sense discrimination (Schütze, 1998), language modeling (Bellegarda, 2000), and the identification of analogical relations (Turney, 2006).

While much research has been directed at the most effective ways of constructing representations for individual words, there has been far less consensus regarding the representation of larger constructions such as phrases and sentences. The problem has received some attention in the connectionist literature, particularly in response to criticisms of the ability of connectionist representations to handle complex structures (Smolensky, 1990; Plate, 1995). More recently, several proposals have been put forward for computing the meaning of word combina-

Results

	dim.	c.m.	Adj-N	N-N	V-Obj
SDS (BNC)	2000	+	0.37	0.38	0.28
	2000	⊙	**0.48**	**0.50**	**0.35**
	100	RAE	0.31	0.30	0.28
NLM (BNC)	50	+	0.28	0.26	0.24
	50	⊙	0.26	0.22	0.18
	100	RAE	0.19	0.24	0.28

Table 3: Correlation coefficients of model predictions with subject similarity ratings (Spearman's ρ); columns show dimensionality: fixed or varying (see Section 2.1), composition method: + is additive vector composition, ⊙ is component-wise multiplicative vector composition, RAE is Socher et al. (2011a)'s recursive auto-encoder.

SDS = simple distributional semantic

NLM = neural language model

- currently there is a lot of work on designing functional architectures for bigram, phrase, and sentence similarity
  - e.g., word averaging, recurrent neural networks, LSTMs, recursive neural networks, etc.
- our recent results find that, for sentence similarity, word averaging is a surprisingly strong baseline

TOWARDS UNIVERSAL PARAPHRASTIC SENTENCE EMBEDDINGS

John Wieting **Mohit Bansal** **Kevin Gimpel** **Karen Livescu**
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{jwieting,mbansal,kgimpel,klivescu}@ttic.edu

Model	Pavlick et al. (2015) (test)
PARAGRAM-PHRASE	60.0
iRNN	60.0
projection	58.4
DAN	60.1
RNN	60.3
LSTM (o.g.)	60.9
LSTM (no o.g.)	**61.3**
skip-thought	39.3
GloVe	44.8
PARAGRAM-SL999	55.3

on similar data to training data, LSTM does best

word averaging

adding layers to word averaging

Dataset	50%	75%	Max	PP	iRNN	proj.	DAN	RNN	LSTM (o.g.)	LSTM (no o.g.)	ST	GloVe	PSL
STS 2012 Average	54.5	59.5	70.3	58.7	58.4	**60.0**	56.0	48.1	46.4	51.0	30.8	52.5	52.8
STS 2013 Average	45.3	51.4	65.3	55.8	56.7	**56.8**	54.2	44.7	41.5	45.2	24.8	42.3	46.4
STS 2014 Average	64.7	71.4	76.7	70.9	70.9	**71.3**	69.5	57.7	51.5	59.8	31.4	54.2	59.5
STS 2015 Average	70.2	75.8	80.2	**75.8**	75.6	74.8	72.7	57.2	56.0	63.9	31.0	52.7	60.0
2014 SICK	71.4	79.9	82.8	71.6	71.2	**71.6**	70.7	61.2	59.0	63.9	49.8	65.9	66.4
2015 Twitter	49.9	52.5	61.9	52.9	52.9	52.8	**53.7**	45.1	36.1	47.6	24.7	30.3	36.3

but when evaluating on other datasets, word averaging models do best!

"You can't cram the meaning of a whole %&!$# sentence into a single $&!#* vector!"

--Ray Mooney

"You can't map all sentences into a cold, sterile space of meaningless, uninterpretable dimensions.
Symbolic representations can encode meaning much more efficiently."

--my interpretation

"You can't cram the meaning of a whole %&!$# sentence into a single $&!#* vector!"

--Ray Mooney

Why must we choose?

Neural architectures for text understanding can combine discrete (symbolic) and continuous representations

Syntax and Semantics

- **syntax**: rules, principles, processes that govern sentence structure of a language
- **semantics**: what the sentence means

- we saw syntactic parsing, which produces a syntactic structure of a sentence
  - helps to disambiguate attachments, coordinations, sometimes word sense
- now we'll look at semantic parsing, which roughly means "produce a semantic structure of a sentence"

Several Kinds of Semantic Parsing

- semantic role labeling (SRL)
- frame-semantic parsing
- "semantic parsing" (first-order logic)
- abstract meaning representation (AMR)
- dependency-based compositional semantics