Tuning As Ranking

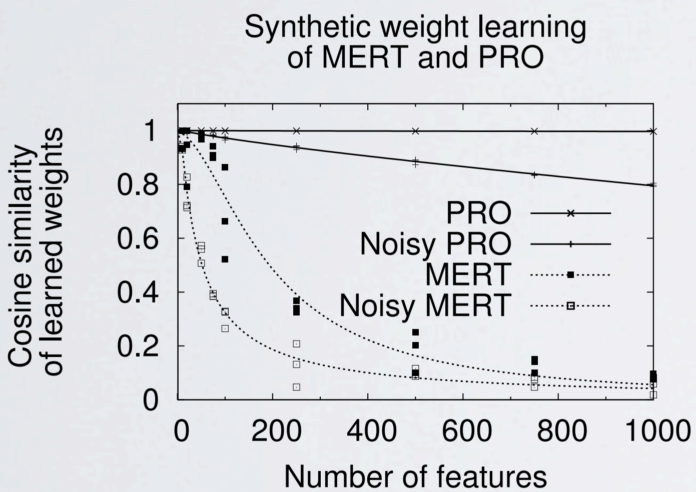
Mark Hopkins
Jonathan May
SDL Language Weaver

EMNLP
July 29, 2011

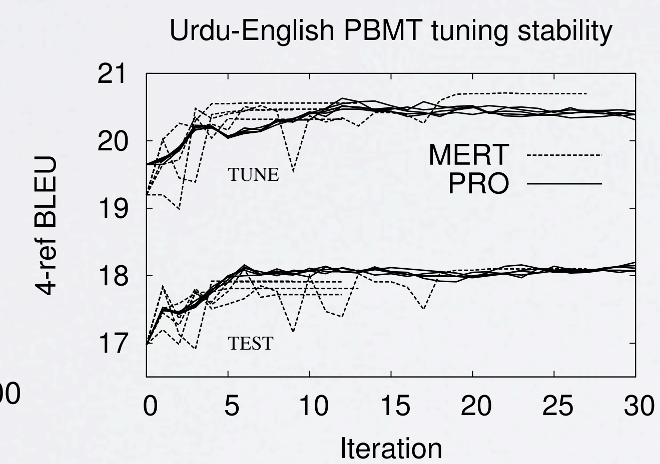
What we did

We replaced **MERT**'s **linear optimization** with a **linear binary classifier**, and fed it **pairs** of translations, effecting a **ranking**

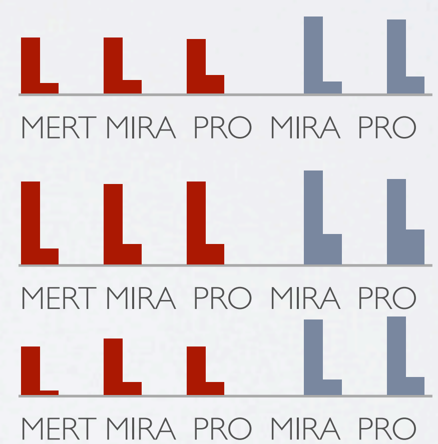
What we found



Scalable to many features



Consistent results



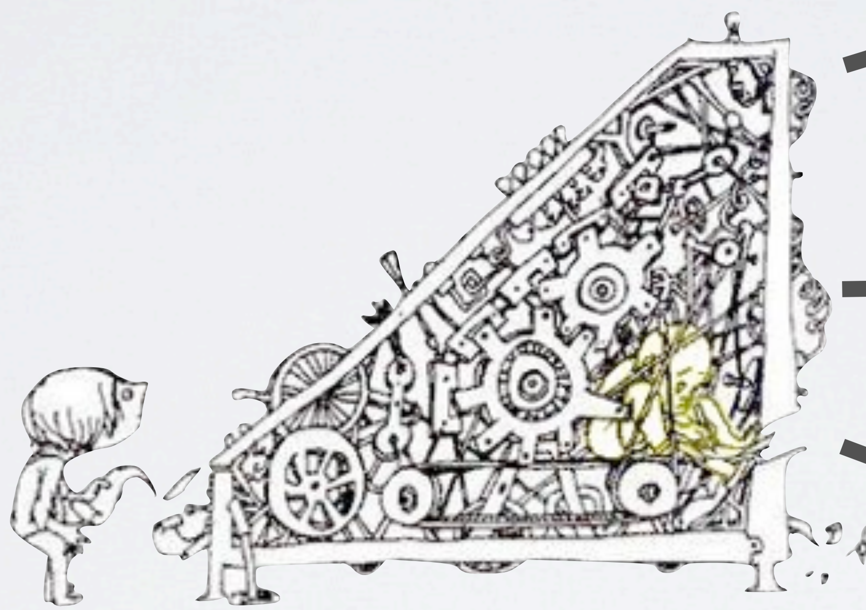
Parity with leading techniques

Very **fast**

Any Questions?

Which is best?

il ne va pas →



→ he goes not

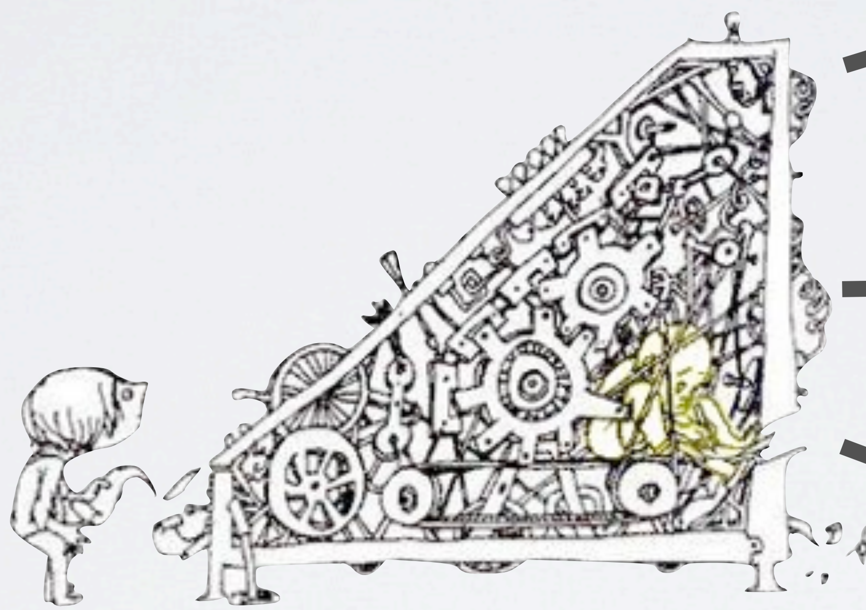
→ he doesn't go

→ she not go

(Image credit: Silverstein, 1981)

Which is best?

il ne va pas



he goes not **0**

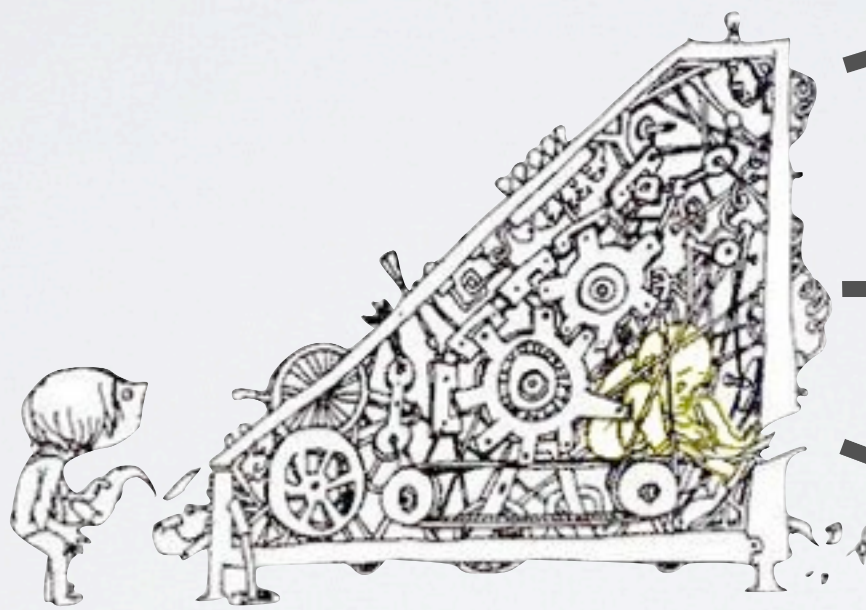
he doesn't go ✓ **2**

she not go **-11**

A good scoring function can tell us

Which is best?

il ne va pas



he goes not **-11**

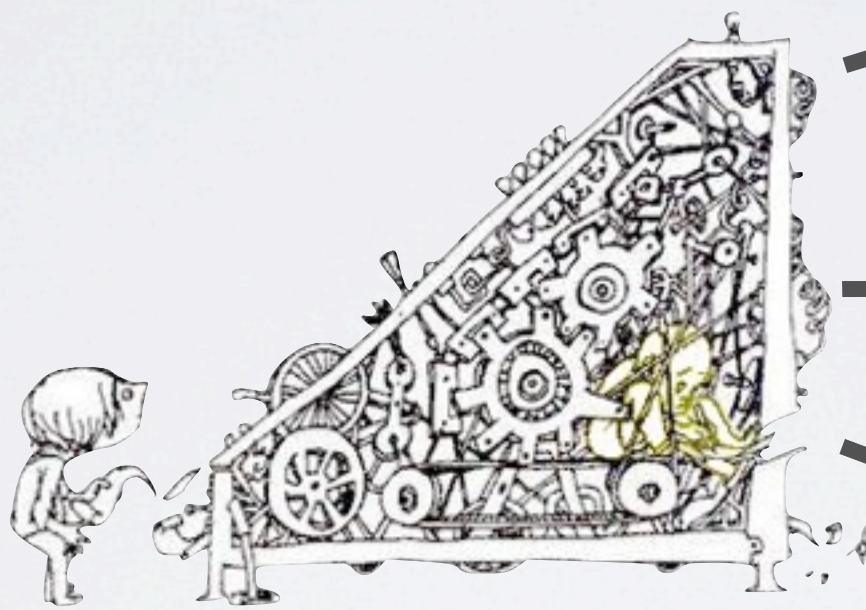
he doesn't go **0**

she not go **2**

We should avoid bad functions

Which is best?

il ne va pas



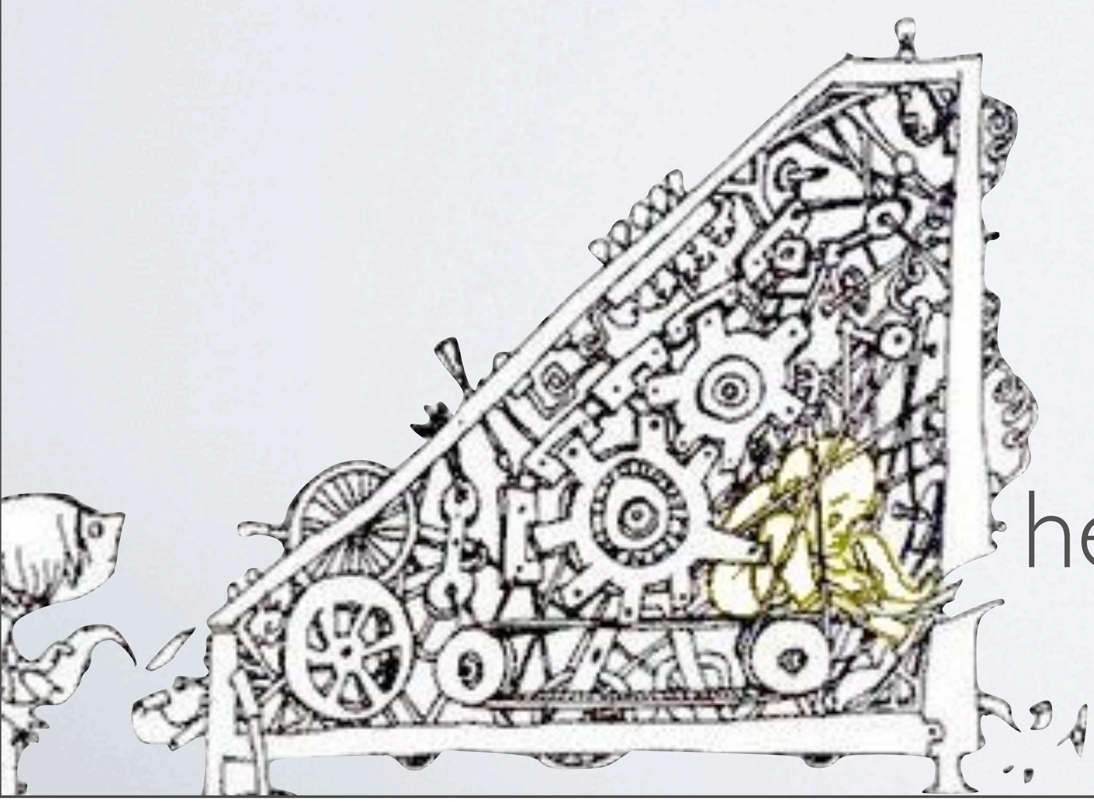
he goes not ??

he doesn't go ??

she not go ??

How do we ensure "proper" scores?

Properties of the translation

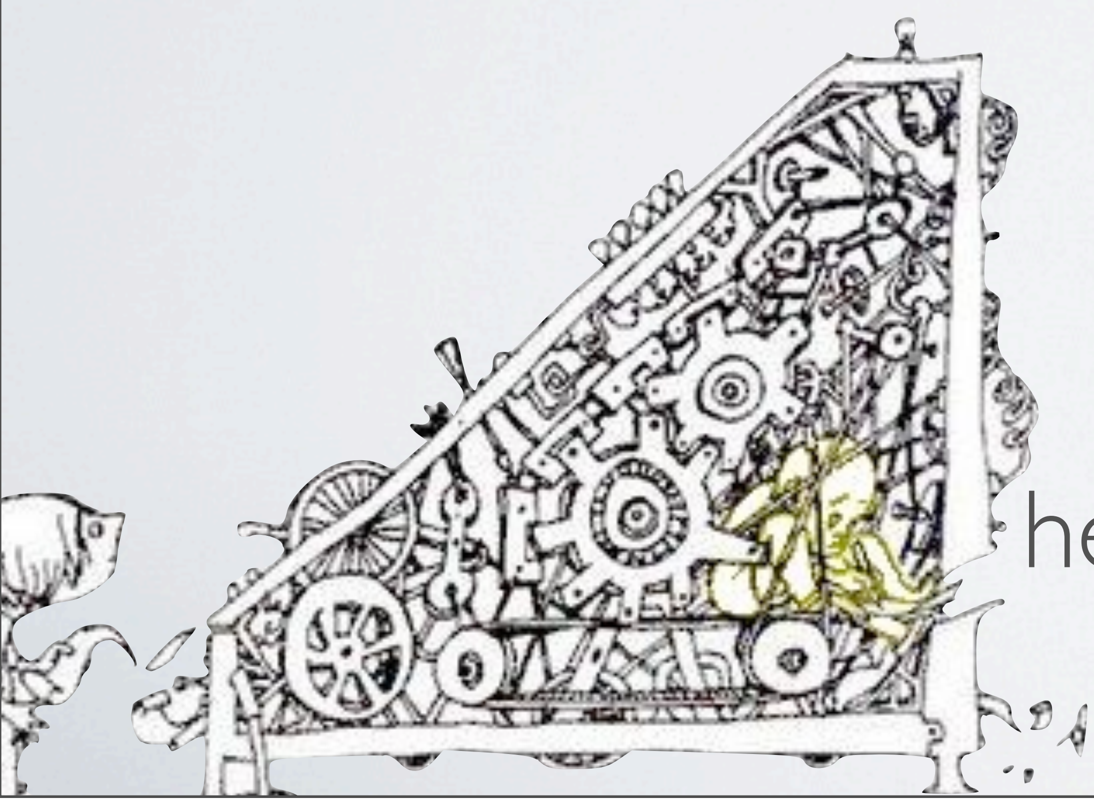


he goes not

Properties of the translation

literal meaning?

2

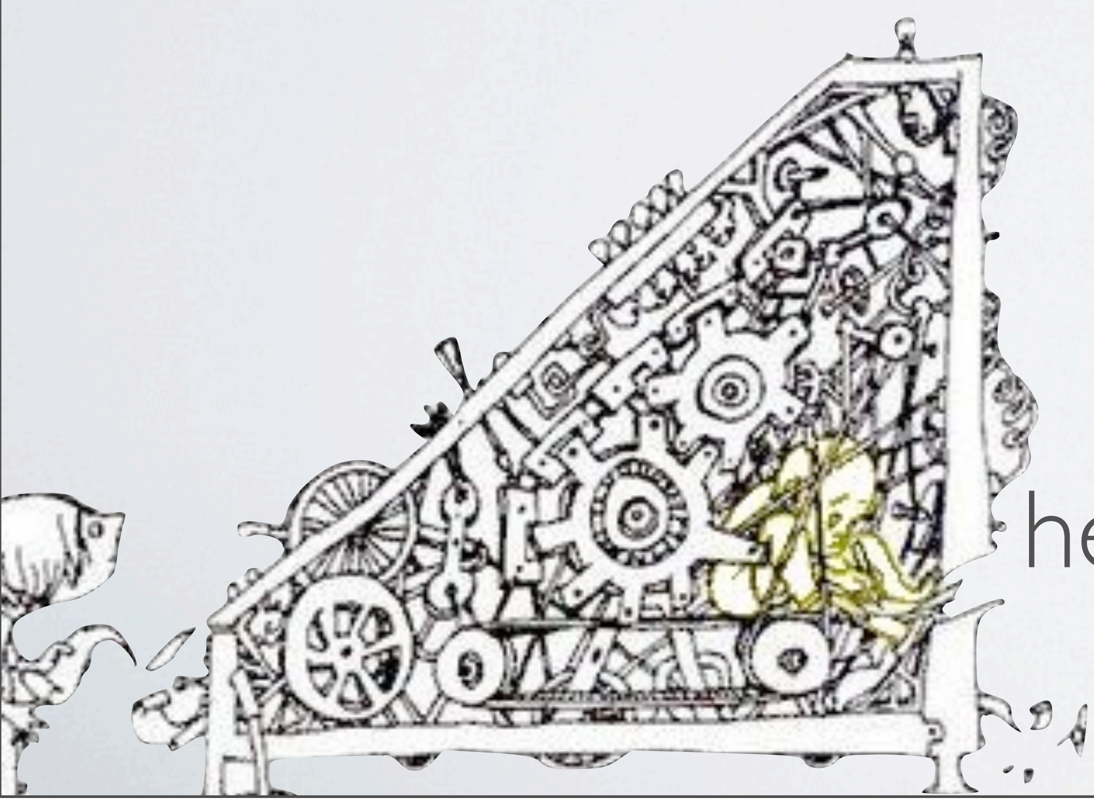


he goes not

Properties of the translation

literal meaning? 2

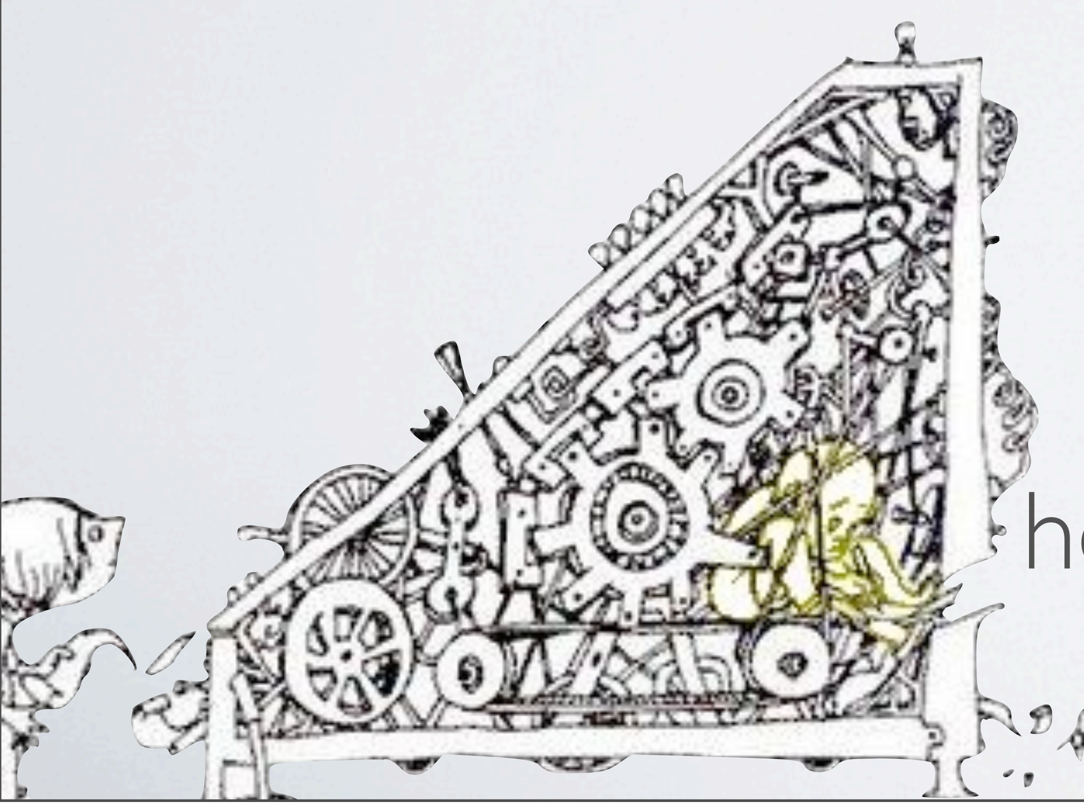
fluency? 4



he goes not

Properties of the translation

literal meaning?	**2**
fluency?	**4**
word count?	**3**
count of "he"?	**1**
count of "coffee"?	**0**
...	
alliterative?	**0**
$2 \leq count(\text{"o"}) \leq 3$?	**1**
how do you feel?	**32.8**

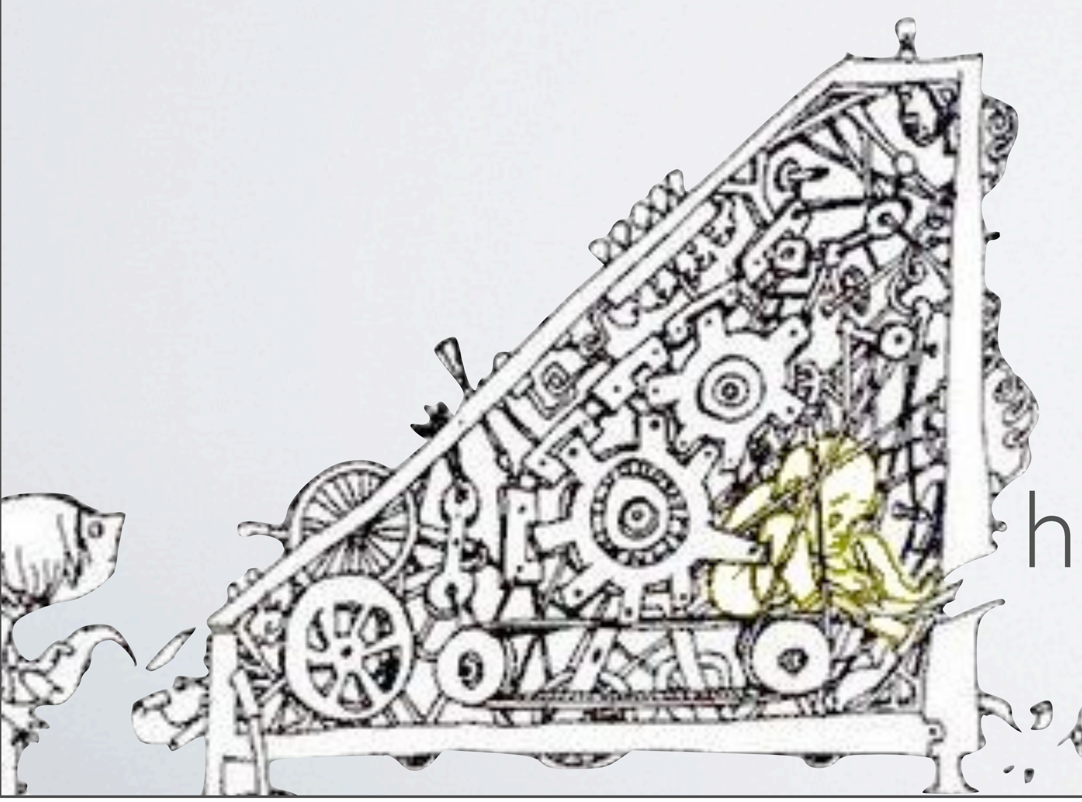


he goes not →

Properties of the translation

Features!

literal meaning?	2
fluency?	4
word count?	3
count of "he"?	1
count of "coffee"?	0
...	
alliterative?	0
$2 \leq$count("o")≤ 3 ?	1
how do you feel?	32.8

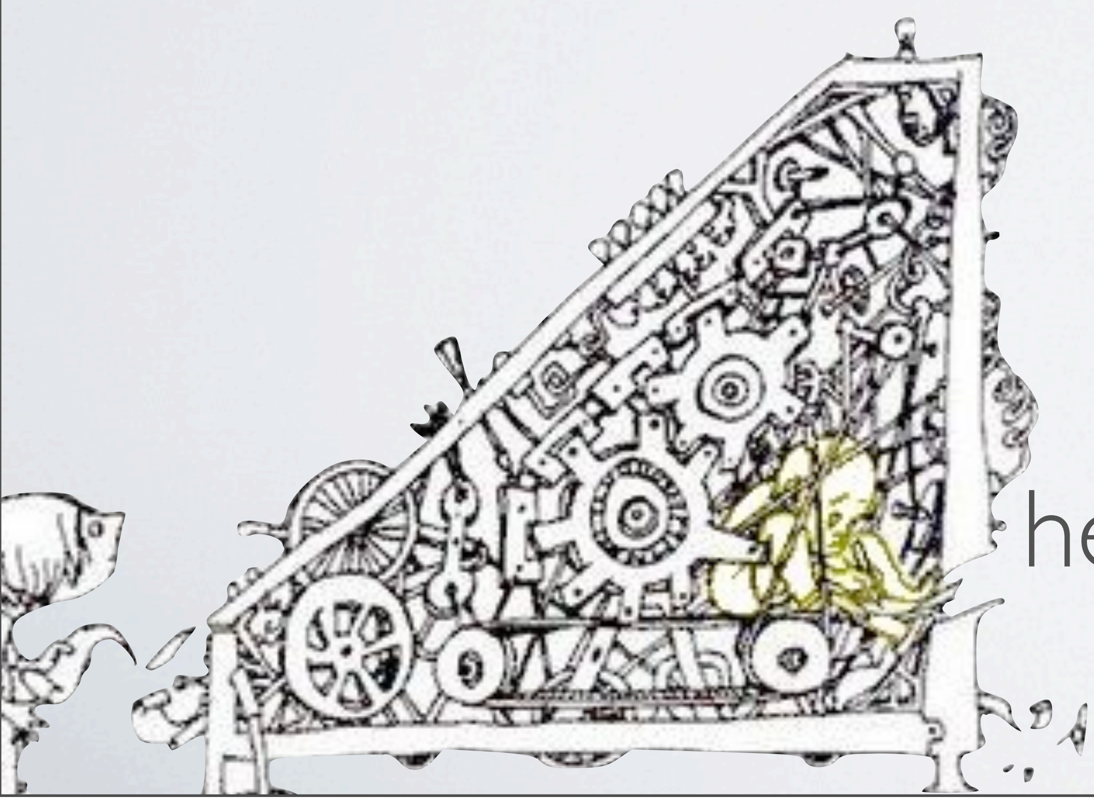


he goes not

Properties of the translation

Features!

f1: **2**
f2: **4**



he goes not

Properties of the translation

Features!

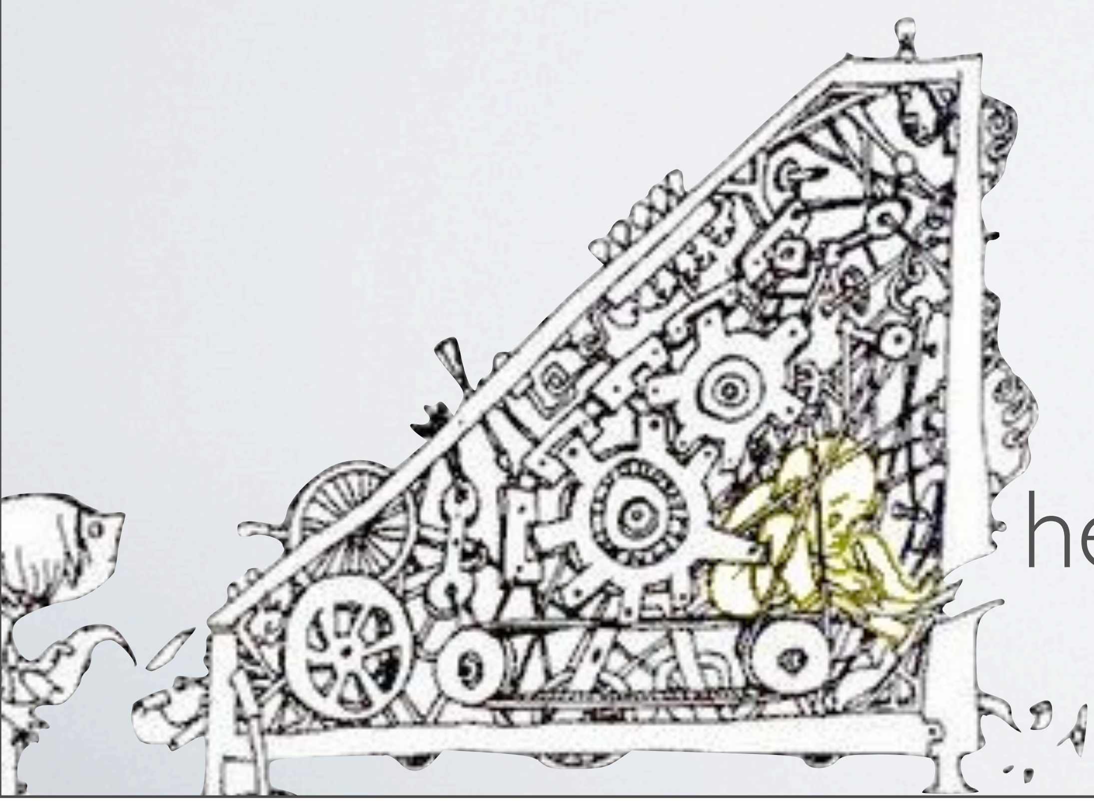
f1:

f2:

Form a weighted sum

$-2 \times 2 + 3 \times 4 = 8$

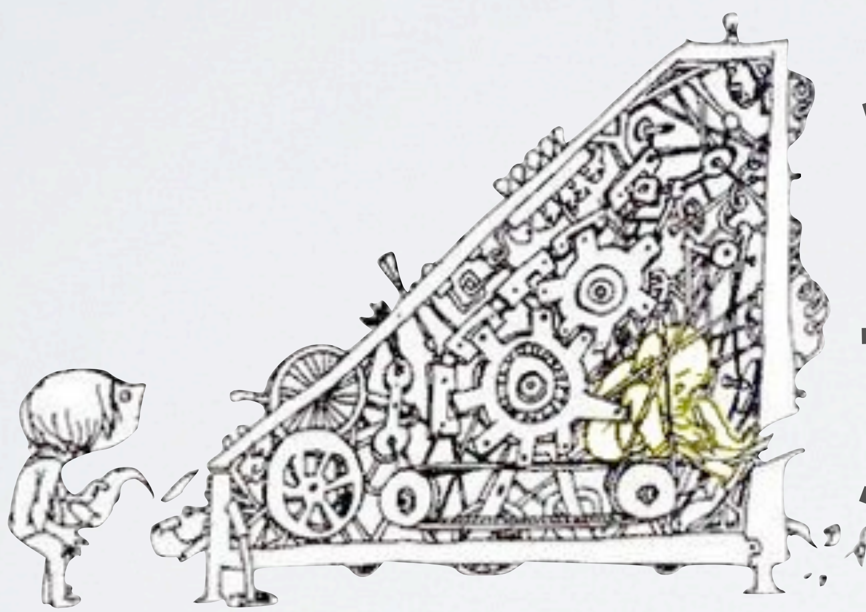
Weights!



he goes not

Translations are feature vectors

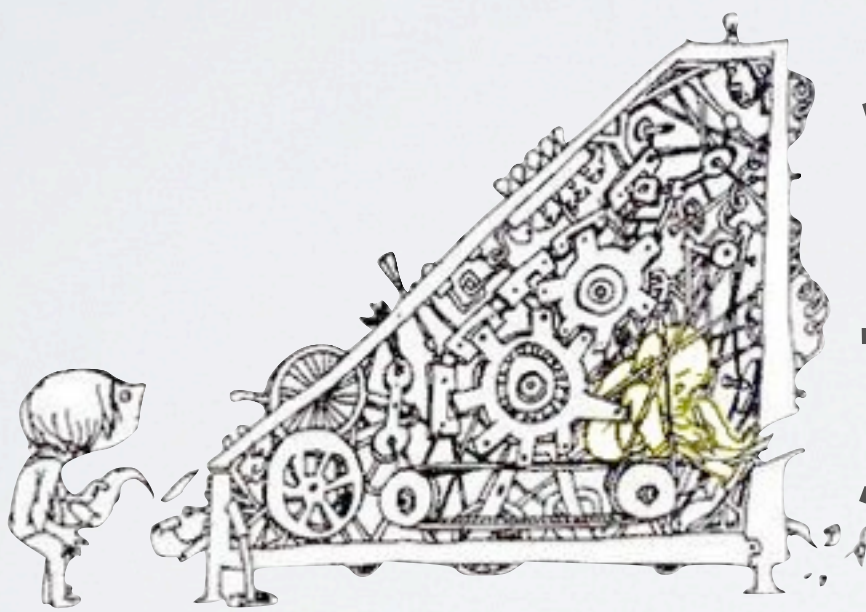
il ne va pas



he goes not	2	4
he doesn’t go	3	8
she not go	6	1

Weight vector determines the score

il ne va pas



2 4 = 8

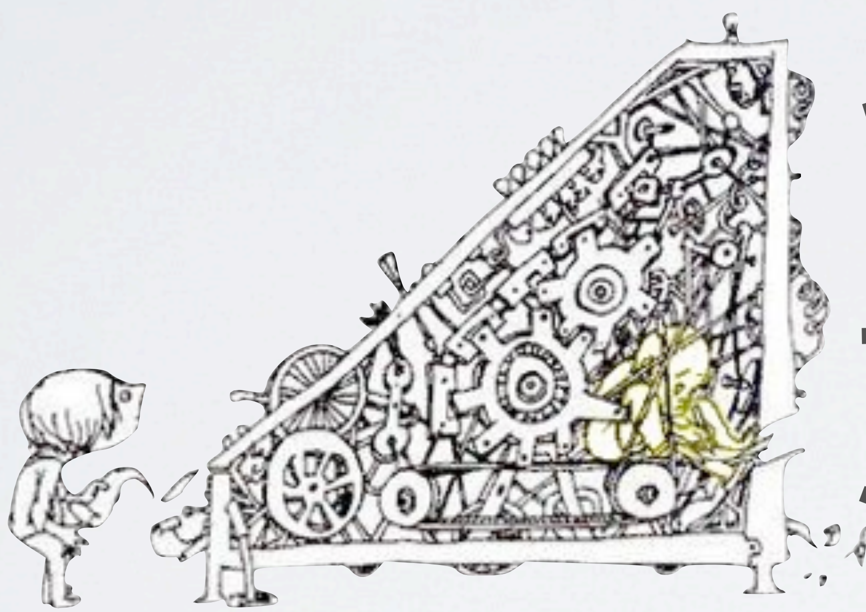
3 8 = 18

6 1 = -9

w: - 2 3

Weight vector determines the score

il ne va pas



2	4	=	8
3	8	=	18
6	1	=	-9

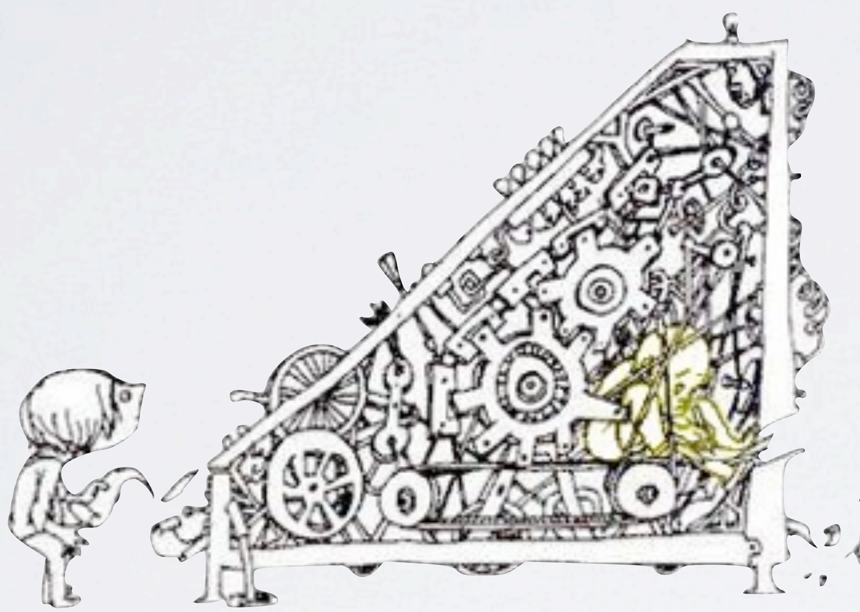
w: - 2 3

model score **features** **weights**

h = **f** ● **w**

Weight vector determines the score

il ne va pas



2	4	=	-2
3	8	=	-8
6	1	=	27

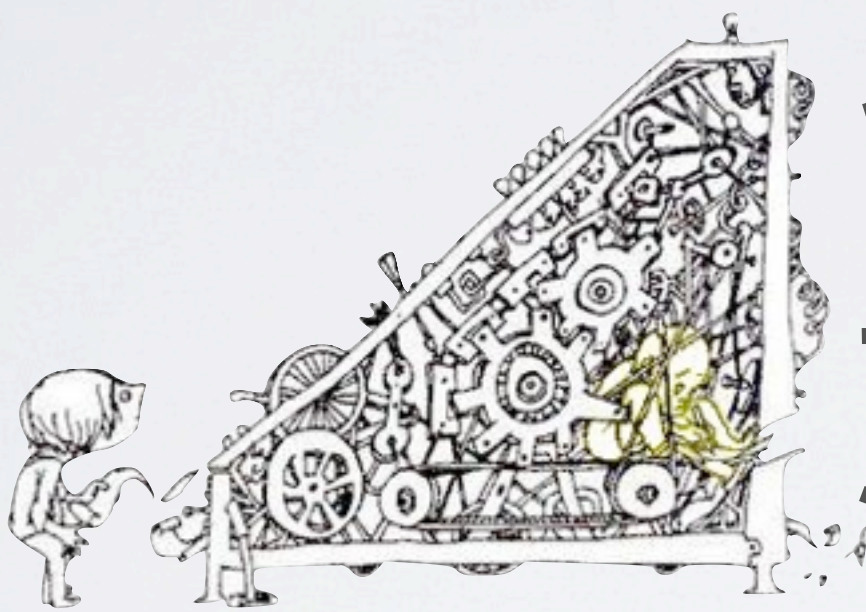
w: 5 -3

model score **features** **weights**

h = **f** ● **w**

Weight vector determines the score

il ne va pas



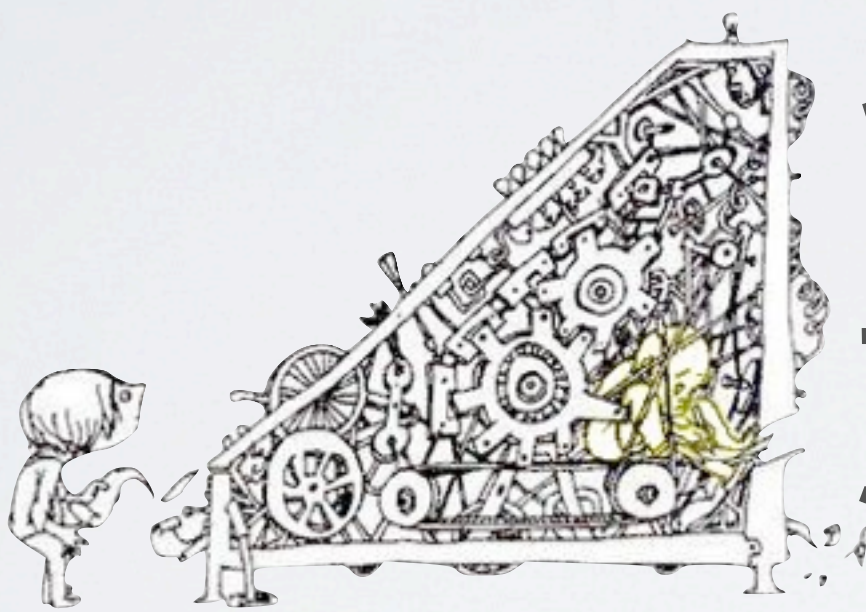
features			model score
2	4	=	-2
3	8	=	-8
6	1	=	27

Tuning is all about choosing this vector

w: 5 -3

Weight vector determines the score

il ne va pas →



features			model score	extrinsic score
2	4	=	-2	.28
3	8	=	-8	.44
6	1	=	27	.12

↑ BLEU+1

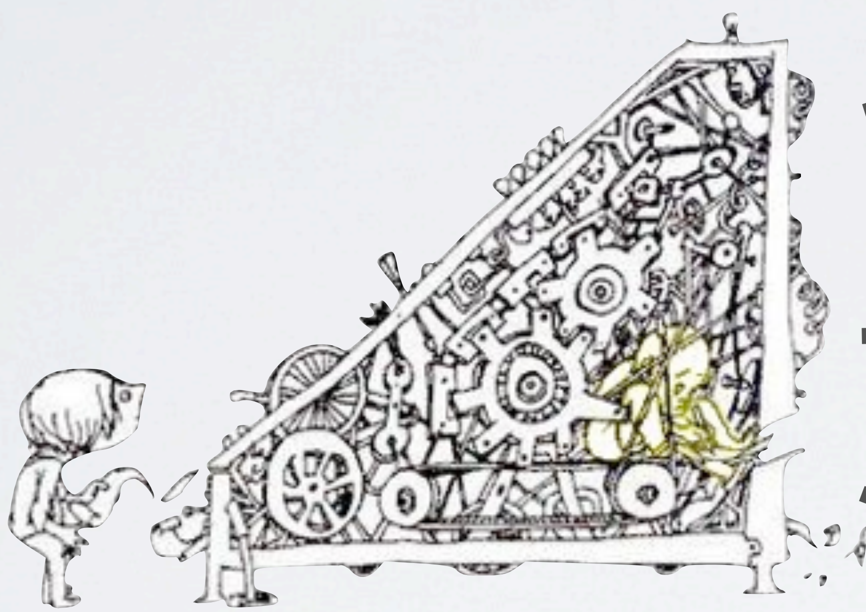
w: 5 -3

We should choose a vector that matches an extrinsic score

(Lin & Och, '04)

Weight vector determines the score

il ne va pas



features

ranks

2	4	=	-2 (B)	.28 (B)
3	8	=	-8 (C)	.44 (A)
6	1	=	27 (A)	.12 (C)

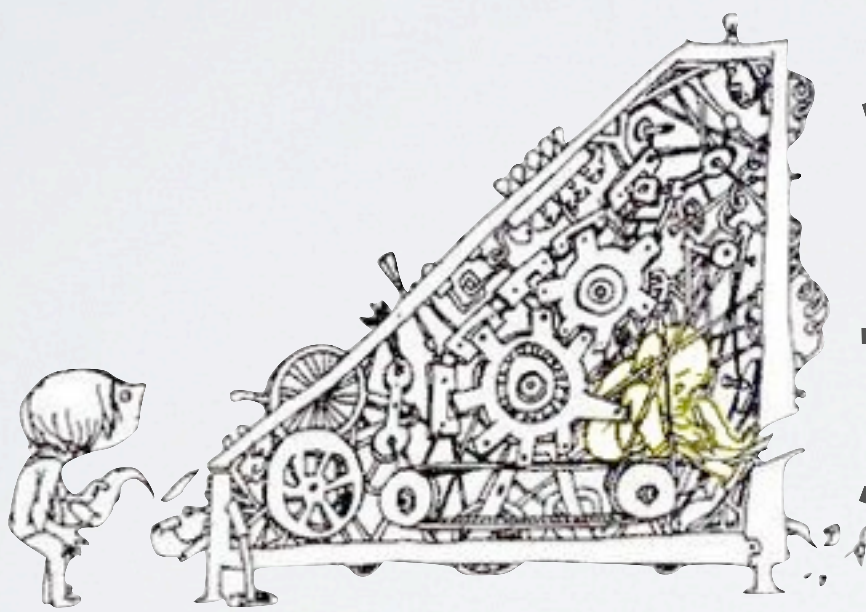
We should choose a vector that matches an extrinsic score

w: 5 -3

Bad match!

Weight vector determines the score

il ne va pas



features

			B — B	
2	4	=	0	.28
			A — A	
3	8	=	2	.44
			C — C	
6	1	=	-11	.12

We should choose a vector that matches an extrinsic score

w: - 2 1

Good match!

The tuning framework that everybody uses

MERT framework

(Och, 2003)

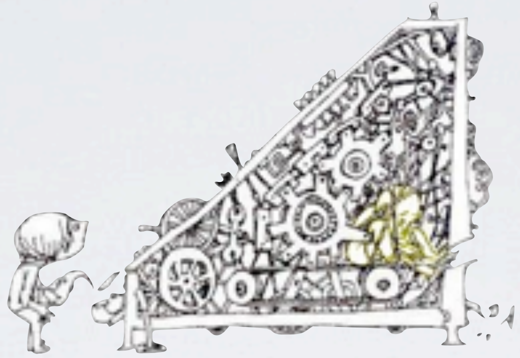
The tuning framework that (almost)* everybody uses

MERT framework

* Not David Chiang

(Och, 2003)

The tuning framework that (almost)* everybody uses



Candidate Generation

Generate n-best per input sentence

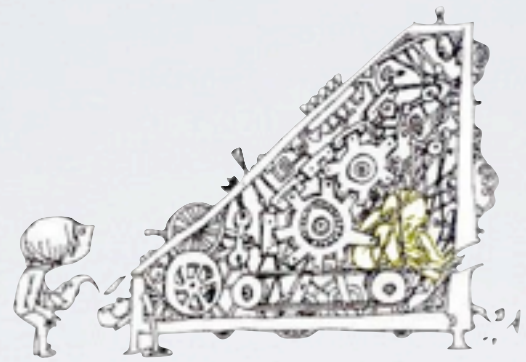
Add to Candidate Pool

MERT framework

* Not David Chiang

(Och, 2003)

The tuning framework that (almost)* everybody uses



Candidate Generation

Generate n-best per input sentence

Add to Candidate Pool

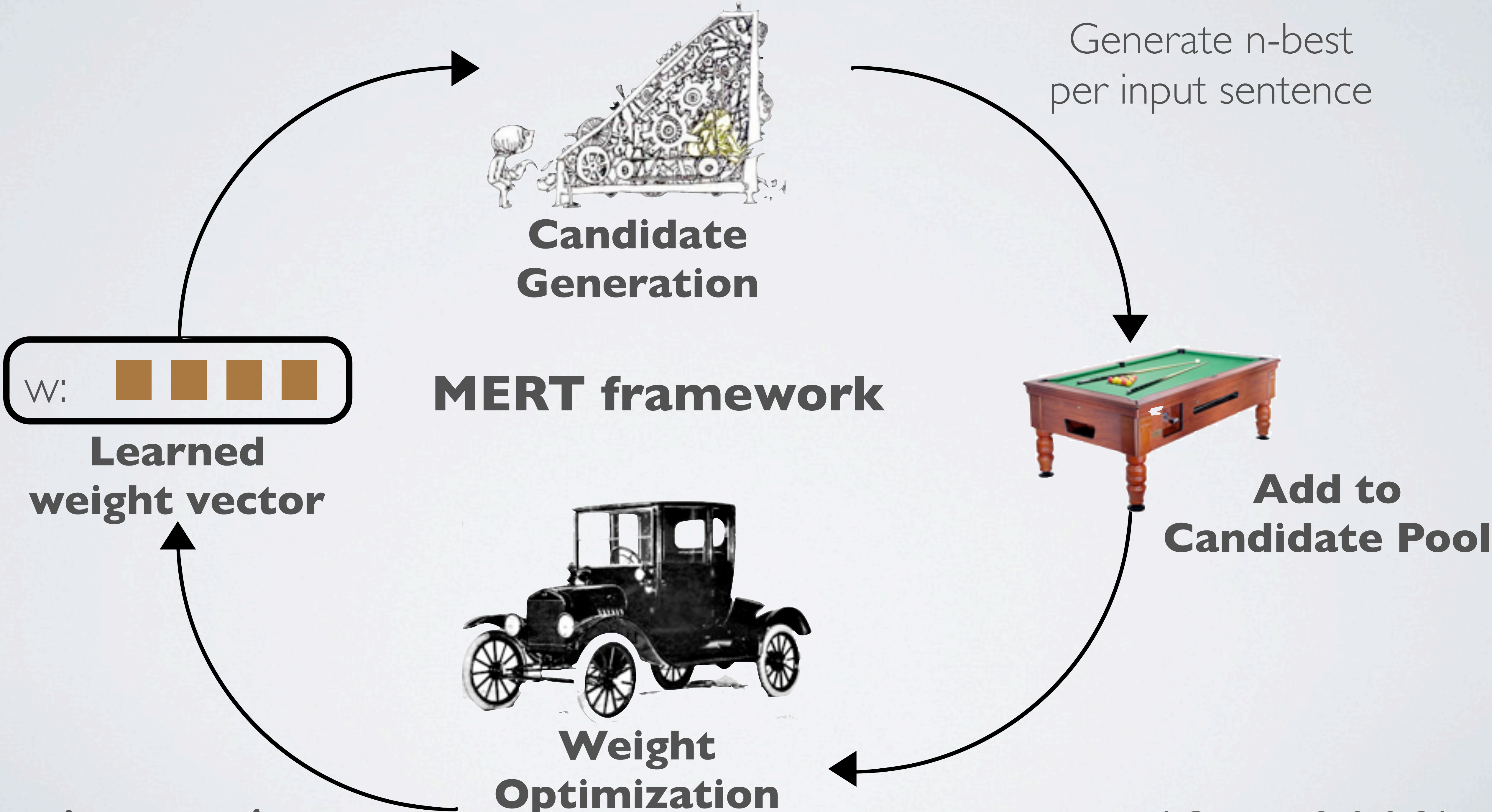
MERT framework

Weight Optimization

* Not David Chiang

(Och, 2003)

The tuning framework that (almost)* everybody uses



* Not David Chiang

(Och, 2003)

How MERT works

	feats		extrins
	2	4	.28
	3	8	.44
S1	6	1	.12
	-3	-3	.15
	1	5	.18
S2	-5	-3	.32

The MERT algorithm works by varying one weight at a time to find a value for that weight that aligns the **best model score** with the **best extrinsic score**

How MERT works

	feats		model	extrins
	2	4	0	.28
	3	8	0	.44
S1	6	1	0	.12
	-3	-3	0	.15
	1	5	0	.18
S2	-5	-3	0	.32

w: 0 0

total extrinsic **.43**

The MERT algorithm works by varying one weight at a time to find a value for that weight that aligns the **best model score** with the **best extrinsic score**

How MERT works

	feats		model	extrins
	2	4	0	.28
	3	8	0	.44
S1	6	1	0	.12
	-3	-3	0	.15
	1	5	0	.18
S2	-5	-3	0	.32

w: 0 0

total extrinsic .43

(MERT can optimize the non-decomposable BLEU; swap these for n-gram component values and determine **total** with the BLEU equation)

How MERT works

	feats		model	extrins
	2	4	0	.28
	3	8	0	.44
S1	6	1	0	.12
	-3	-3	0	.15
	1	5	0	.18
S2	-5	-3	0	.32

w: 0 0

vary hold

total extrinsic **.43**

The MERT algorithm works by varying one weight at a time to find a value for that weight that aligns the **best model score** with the **best extrinsic score**

How MERT works

	feats		model	extrins
	2	4	-2	.28
	3	8	-3	.44
S1	6	1	-6	.12
	-3	-3	3	.15
	1	5	-1	.18
S2	-5	-3	5	.32

w: -1 0

vary hold

total extrinsic **.60**

The MERT algorithm works by varying one weight at a time to find a value for that weight that aligns the **best model score** with the **best extrinsic score**

How MERT works

	feats		model	extrins
	2	4	-2	.28
	3	8	-3	.44
S1	6	1	-6	.12
	-3	-3	3	.15
	1	5	-1	.18
S2	-5	-3	5	.32

w: -1 0

hold vary

total extrinsic **.60**

The MERT algorithm works by varying one weight at a time to find a value for that weight that aligns the **best model score** with the **best extrinsic score**

How MERT works

	feats		model	extrins
	2	4	2	.28
	3	8	5	.44
S1	6	1	-5	.12
	-3	-3	0	.15
	1	5	4	.18
S2	-5	-3	2	.32

w: -1 1

hold vary

total extrinsic **.62**

The MERT algorithm works by varying one weight at a time to find a value for that weight that aligns the **best model score** with the **best extrinsic score**

How MERT works

	feats		model	extrins
	2	4	0	.28
	3	8	2	.44
S1	6	1	-11	.12
	-3	-3	3	.15
	1	5	1	.18
S2	-5	-3	7	.32

w: -2 1

vary hold

total extrinsic .76

The MERT algorithm works by varying one weight at a time to find a value for that weight that aligns the **best model score** with the **best extrinsic score**

How MERT works

	feats				model	extrins
	2	4	0	2	??	.28
	3	8	1	3	??	.44
S1	6	1	1	-5	??	.12
	-3	-3	0	7	??	.15
	1	5	0	17	??	.18
S2	-5	-3	2	6	??	.32

w: -2 1 3 16

This works well for small feature sets, but as the feature space grows, it is hard to find a good position

Synthetic Experiment

random

random

random

"features"

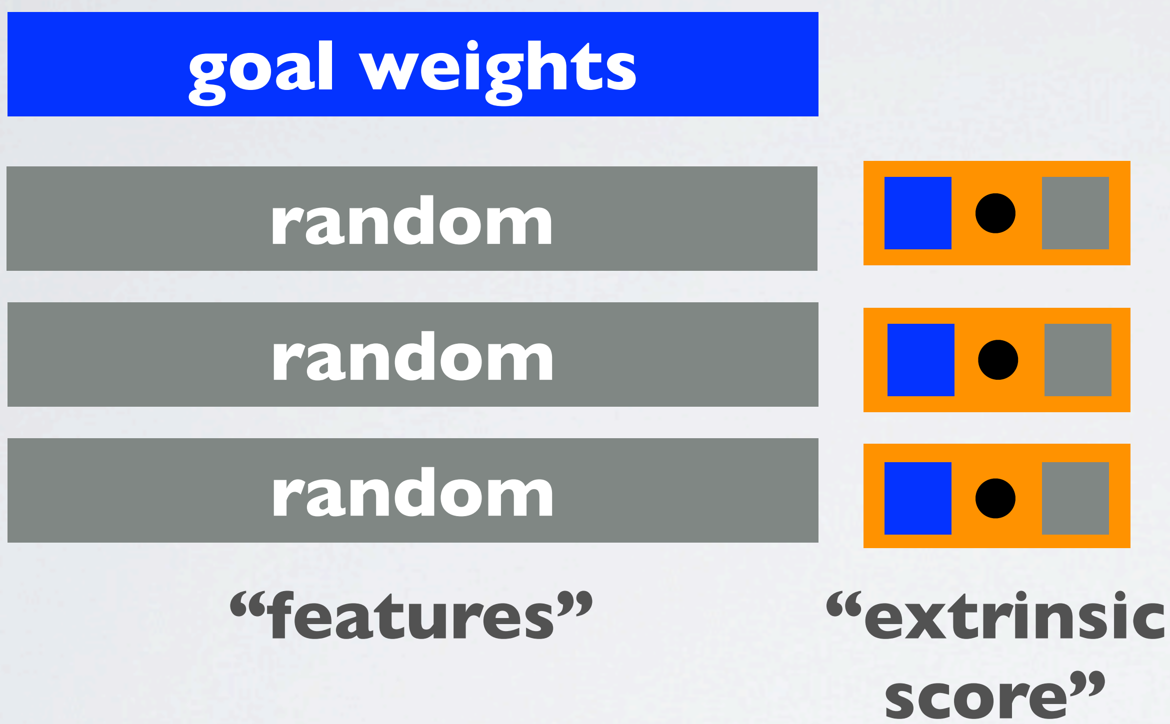
"Candidate pool" of randomly drawn "feature" vectors

Synthetic Experiment

“Candidate pool” of randomly drawn “feature” vectors

How to determine “extrinsic score”?

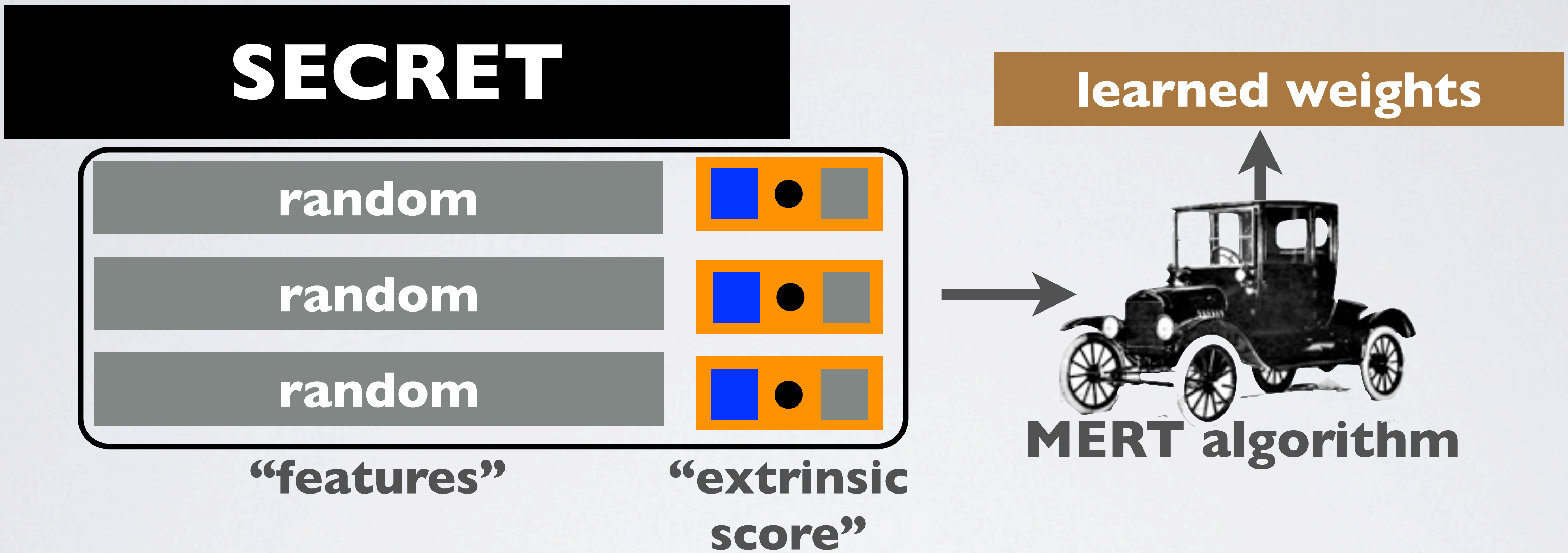
Synthetic Experiment



“Candidate pool” of randomly drawn “feature” vectors

Secret “goal weights” used to calculate extrinsic score

Synthetic Experiment

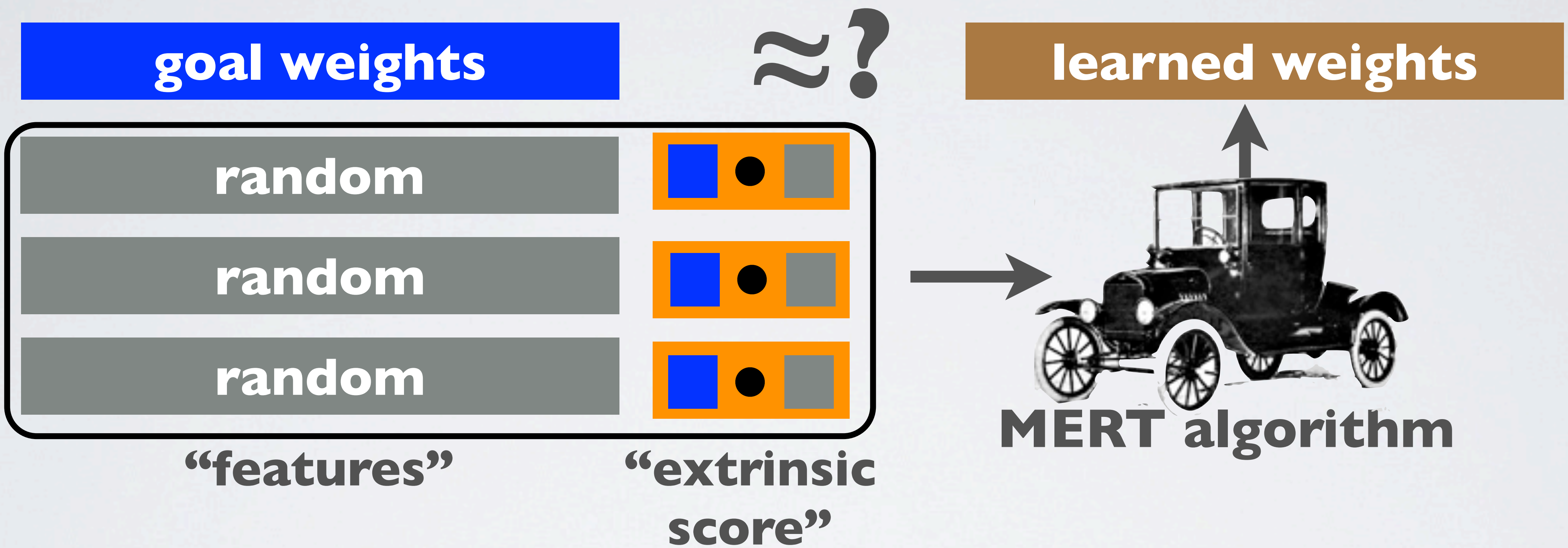


Now use MERT to try and learn the goal weights back

This is linear equation solving

It's much easier than MT tuning

Synthetic Experiment

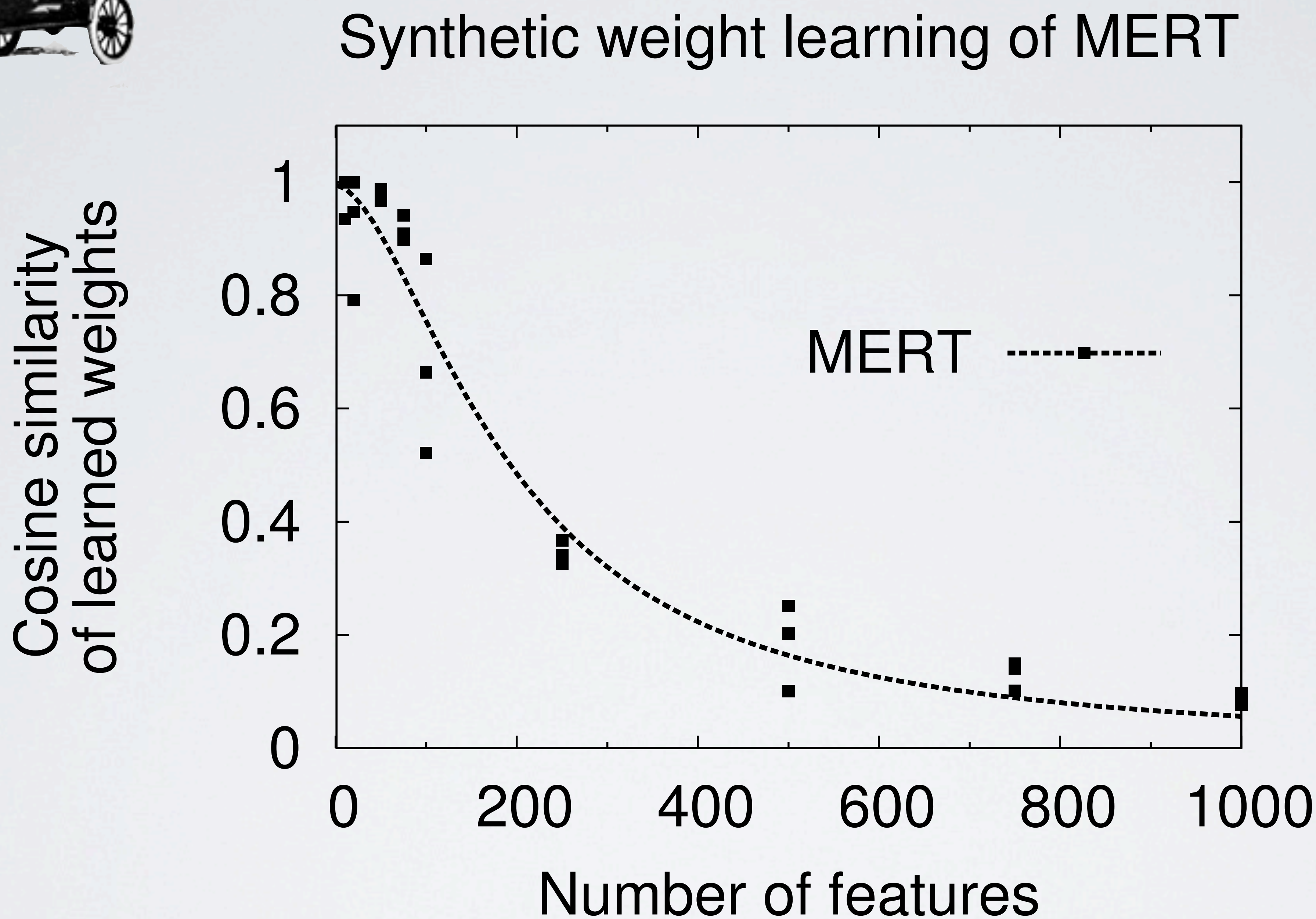


Now use MERT to try and learn the goal weights back

This is linear equation solving

It's much easier than MT tuning

MERT *doesn't scale*



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

MERT only cares about the top-scoring translation

	feats		model	extrins
	2	4	0	B
	3	8	2 ☆	A
S1	6	1	-11	C
	-3	-3	3	C
	1	5	1	B
S2	-5	-3	7 ☆	A

MERT only cares about the top-scoring translation

	feats		model	extrins
	2	4	??	B
	3	8	A	A
S1	6	1	??	C
	-3	-3	??	C
	1	5	??	B
S2	-5	-3	A	A

MERT doesn't care about these

It doesn't care about matching the overall ranking

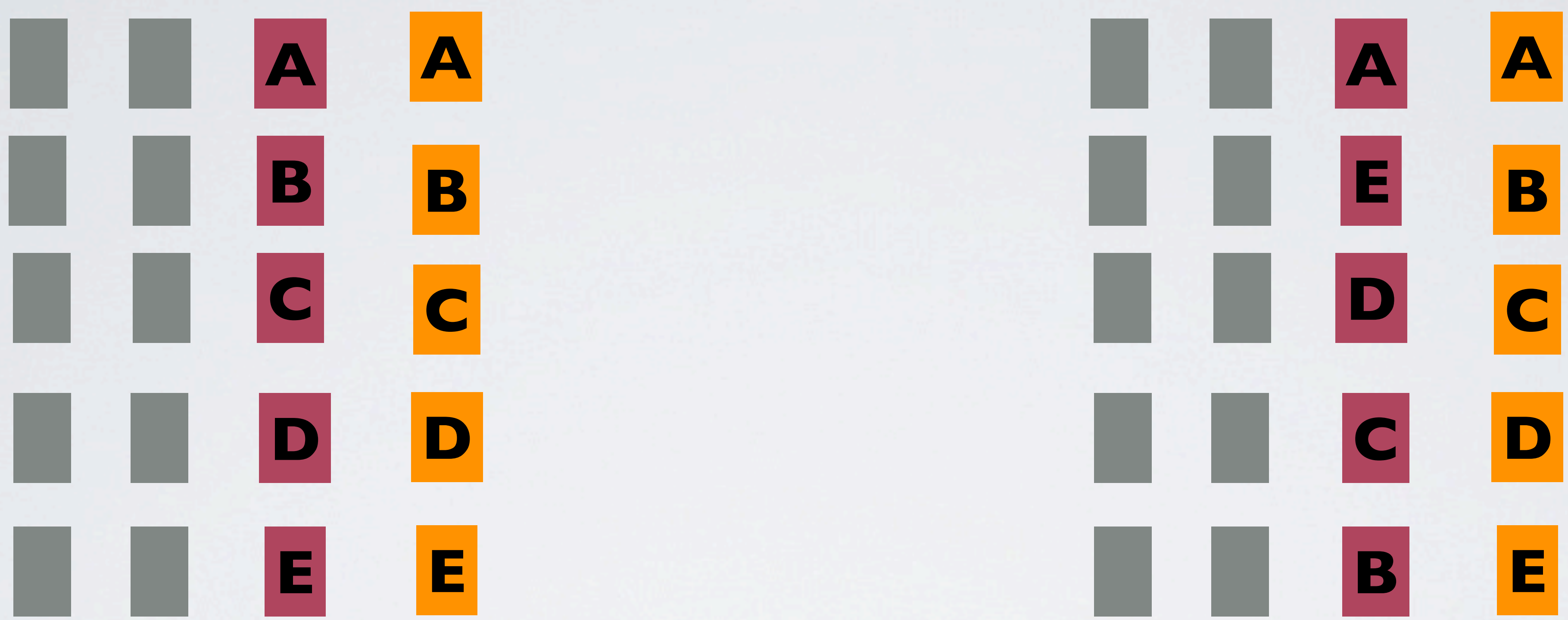
	feats		model	extrins
	2	4	B	B
	3	8	A	A
S1	6	1	C	C
	-3	-3	B	C
	1	5	C	B
S2	-5	-3	A	A

mismatch!

This could lead to poor generalization

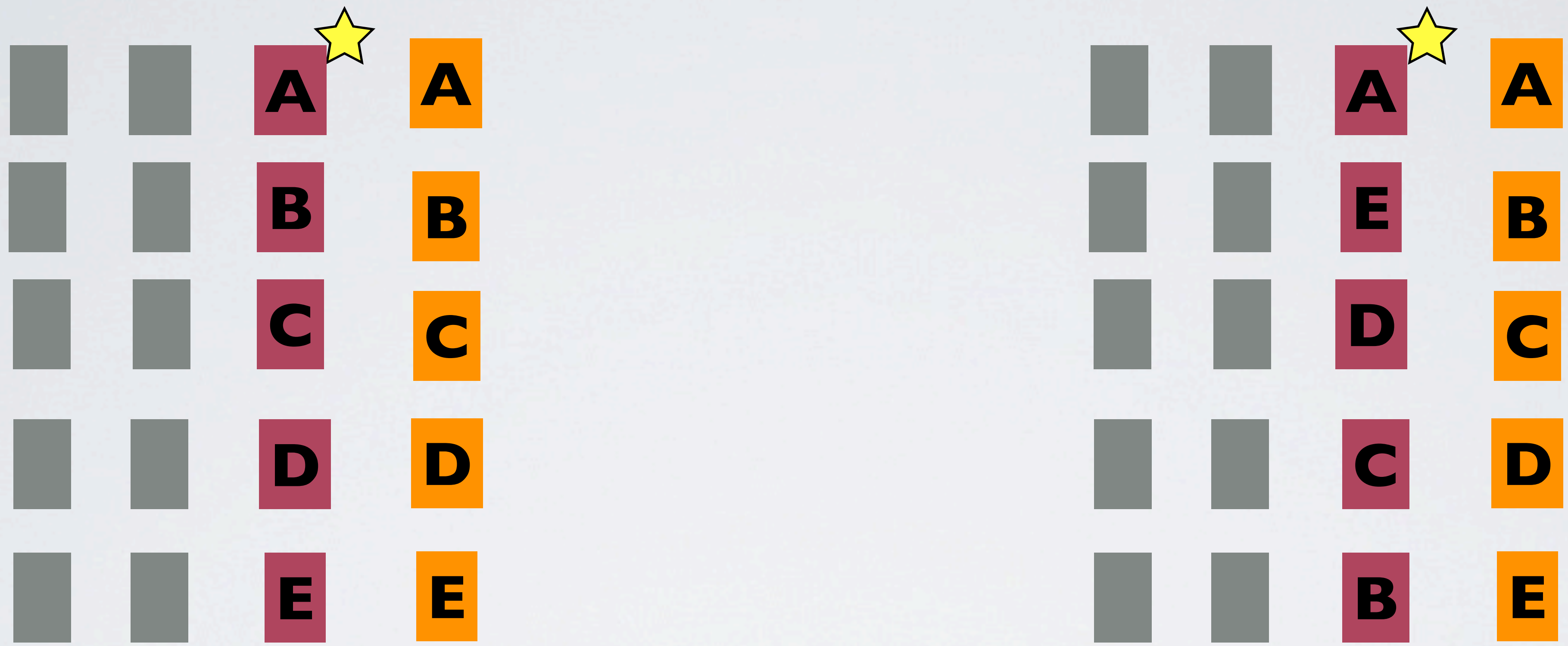
	feats		model	extrins	
	2	4	D	B	
	3	8	A	A	
	6	1	C	C	
	7	-4	J	S	← not good but liked by model
	12	-2	B	G	
	...	...	...	...	
S1	-12	5	Z	D	← good but disliked

We should focus on rank



Recognize that these are different solutions!

We should focus on rank



Recognize that these
are different solutions!
(To MERT they are the
same)

We can describe rank from a **pairwise** perspective

translation a

$\vec{f}_a$ h_a g_a

translation b

$\vec{f}_b$ h_b g_b

For any two translations **a** and **b** of the same sentence

(Herbrich et al., '99)

We can describe rank from a **pairwise** perspective

translation a

$\vec{f}_a$ h_a g_a

translation b

$\vec{f}_b$ h_b g_b

extrinsic

$g_a > g_b$

$\longleftrightarrow$

model

$h_a > h_b$

Model and extrinsic score **order** should agree

We can describe rank from a **pairwise** perspective

translation a

$\vec{f_a}$ h_a g_a

translation b

$\vec{f_b}$ h_b g_b

extrinsic

$g_a > g_b$

$\leftrightarrow$

model

$h_a > h_b$

$\leftrightarrow$ $h_a - h_b > 0$

We can describe rank from a **pairwise** perspective

translation a

$\vec{f}_a$ h_a g_a

translation b

$\vec{f}_b$ h_b g_b

extrinsic **model**

$$g_a > g_b \quad \leftrightarrow \quad h_a > h_b$$

$$\leftrightarrow \quad h_a - h_b > 0$$

$$\leftrightarrow \quad \vec{w} \bullet \vec{f}_a - \vec{w} \bullet \vec{f}_b > 0$$

We can describe rank from a **pairwise** perspective

translation a

$\vec{f_a}$ h_a g_a

translation b

$\vec{f_b}$ h_b g_b

extrinsic

model

$$g_a > g_b \quad \leftrightarrow \quad h_a > h_b$$

$$\leftrightarrow \quad h_a - h_b > 0$$

$$\leftrightarrow \quad \vec{w} \bullet \vec{f_a} - \vec{w} \bullet \vec{f_b} > 0$$

$$\leftrightarrow \quad \vec{w} \bullet \overrightarrow{f_a - f_b} > 0$$

This is a **binary classification** problem

extrinsic **model**

$g_a > g_b \leftrightarrow \vec{w} \bullet \overrightarrow{f_a - f_b} > 0$

This is a **binary classification** problem

extrinsic | **model**

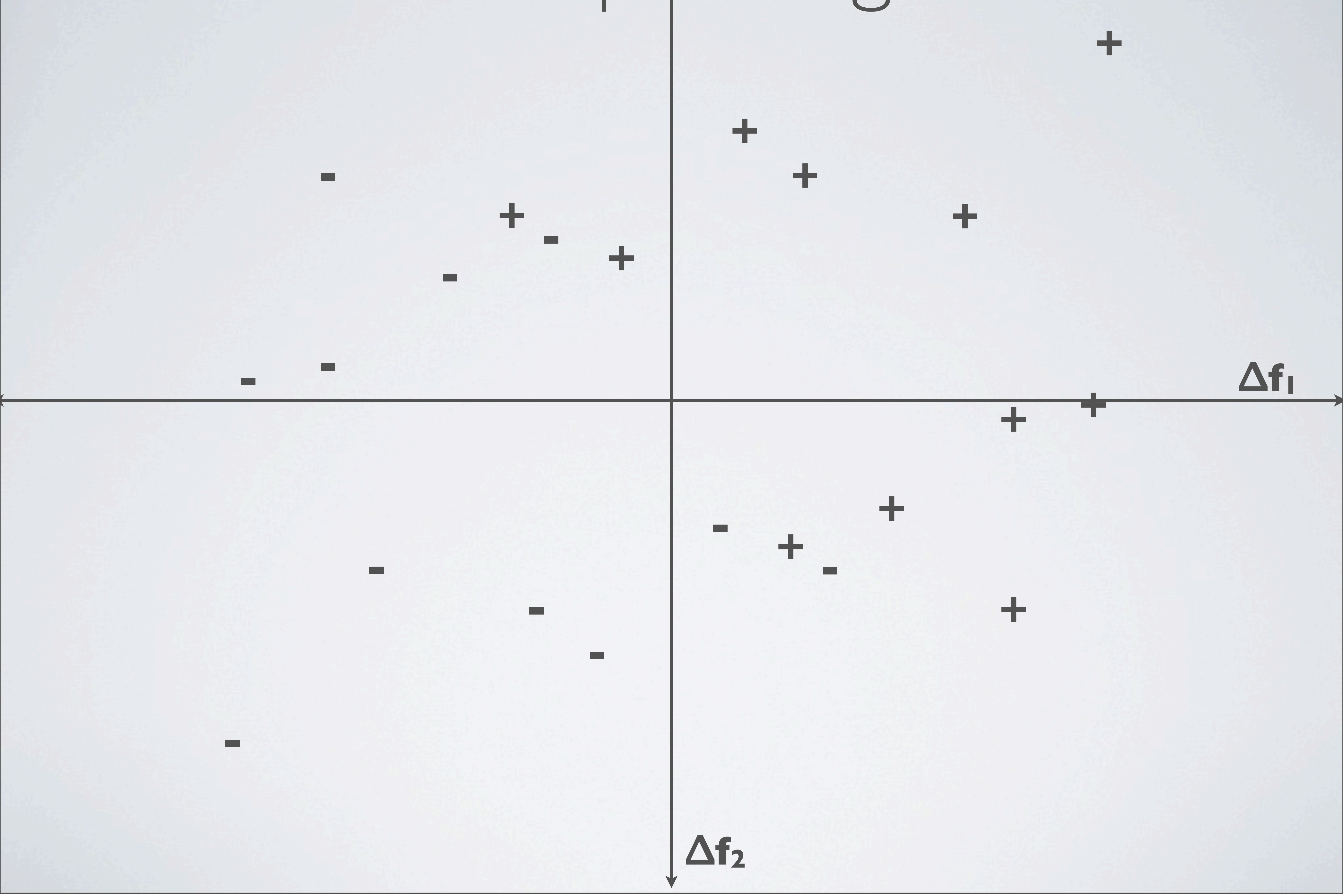
$$g_a > g_b \longleftrightarrow \vec{w} \bullet \overrightarrow{f_a - f_b} > 0$$

label
(+ if a is better,
- if b is better)

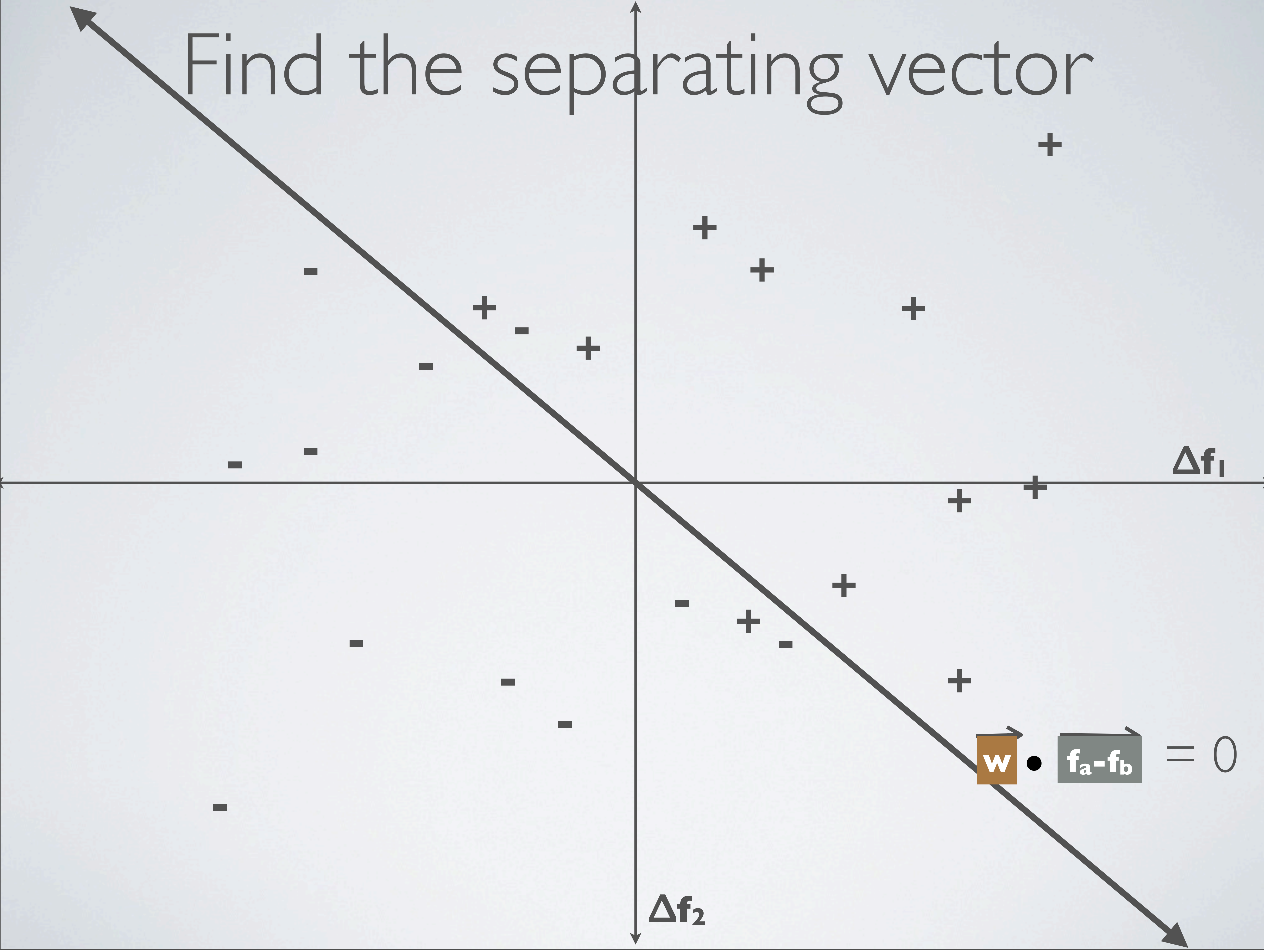
training instance
(difference vector)

$$+\ \overrightarrow{f_a - f_b}$$

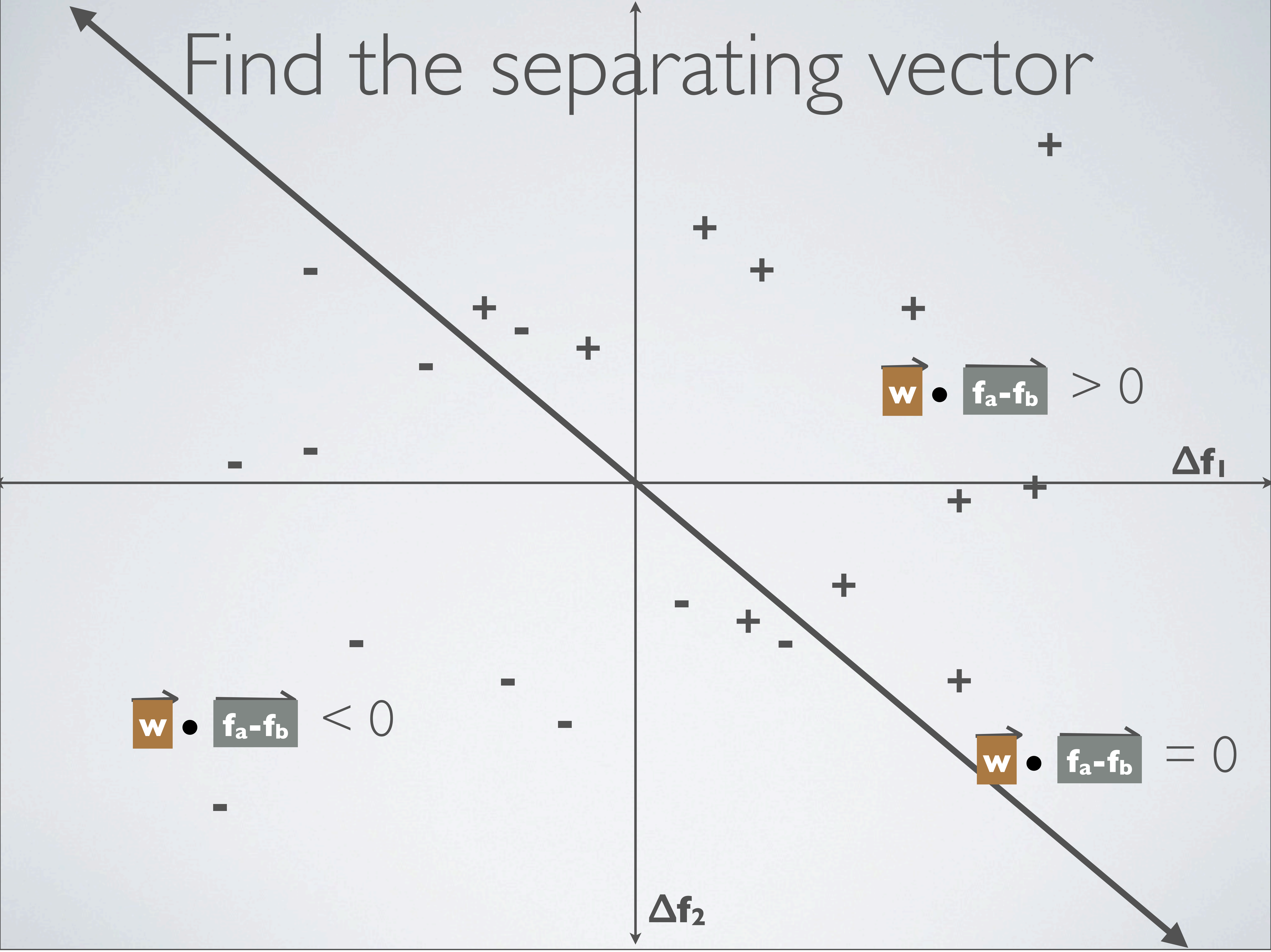
Find the separating vector



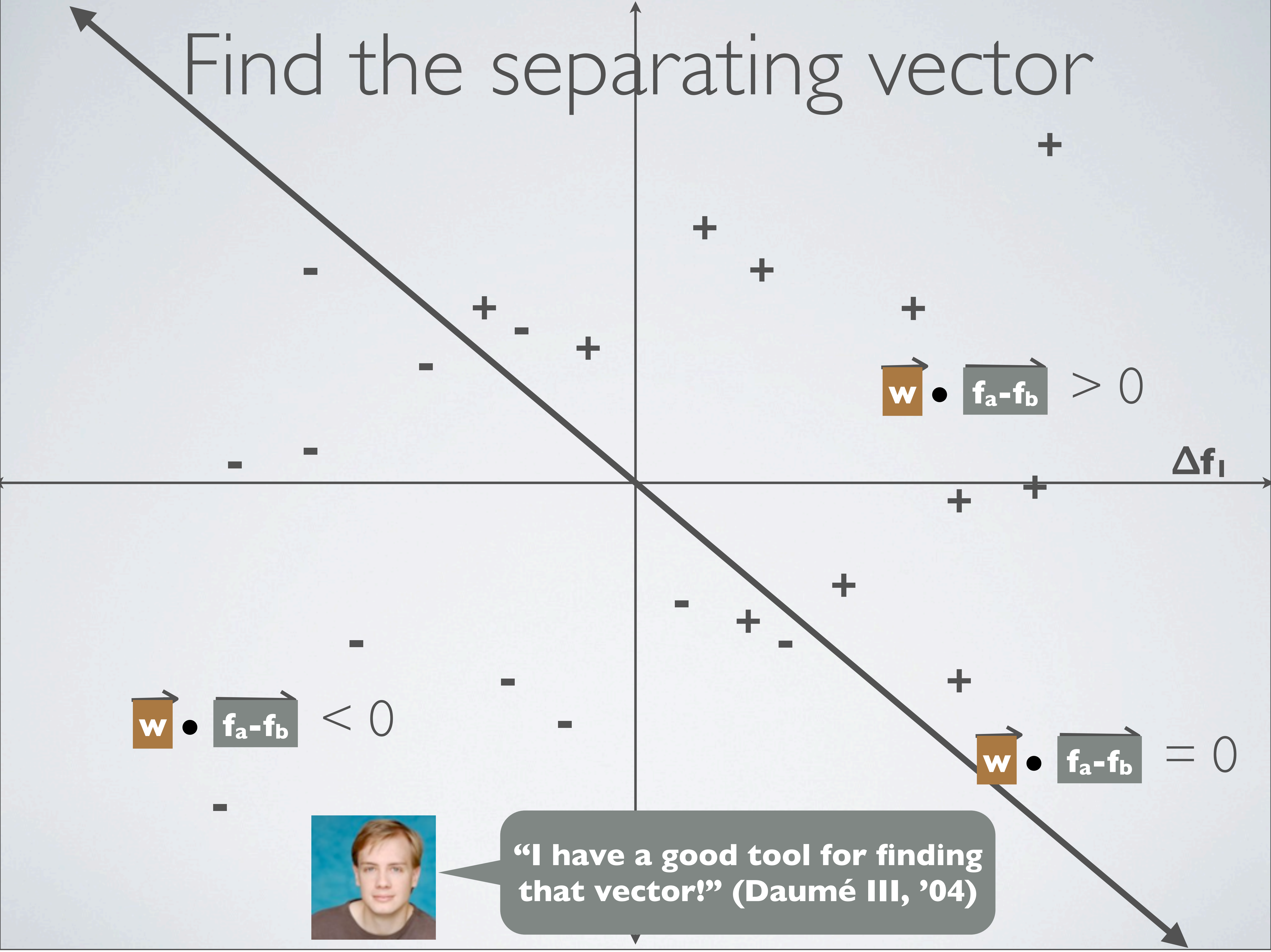
Find the separating vector



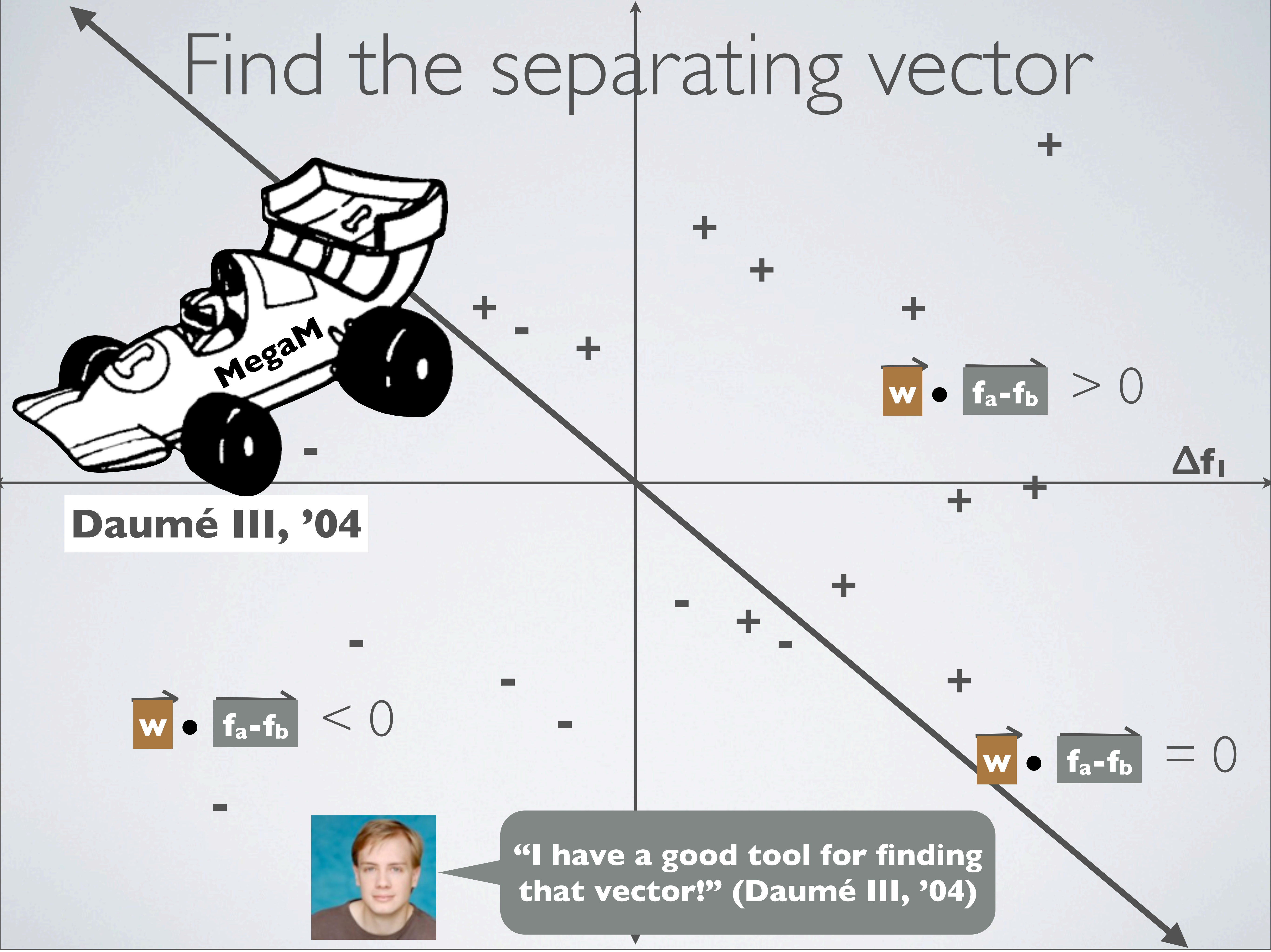
Find the separating vector



Find the separating vector

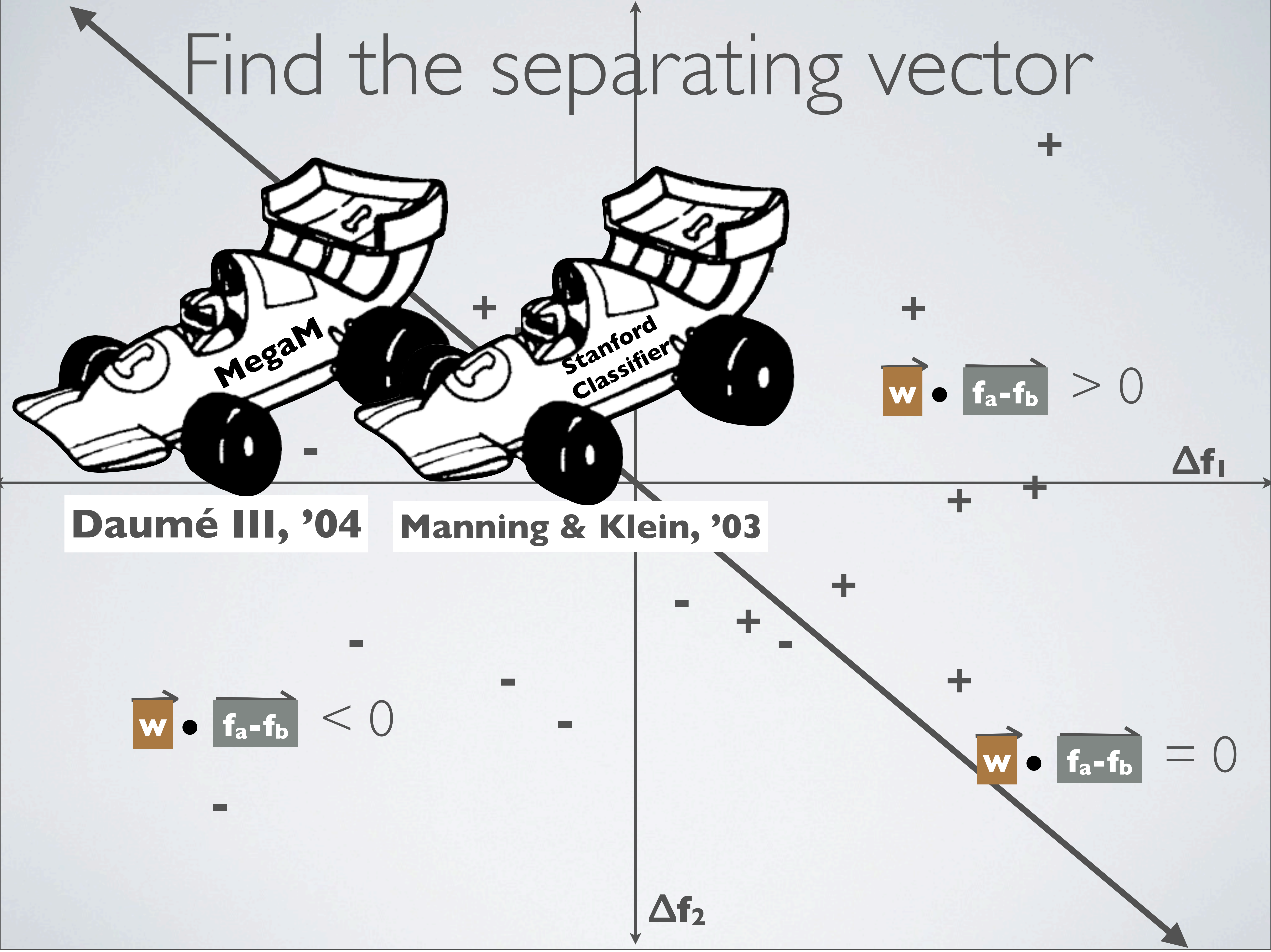


Find the separating vector

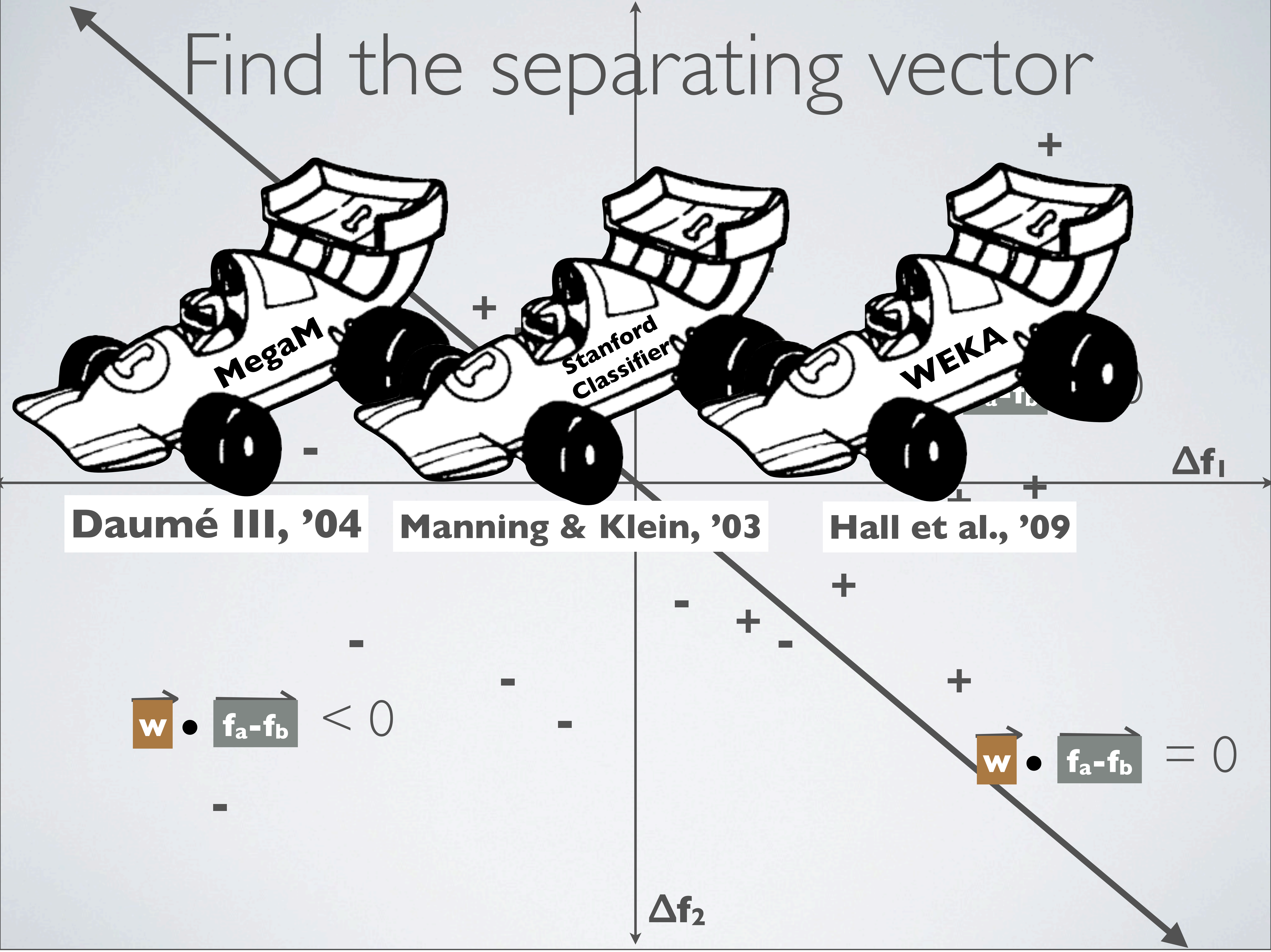


"I have a good tool for finding that vector!" (Daumé III, '04)

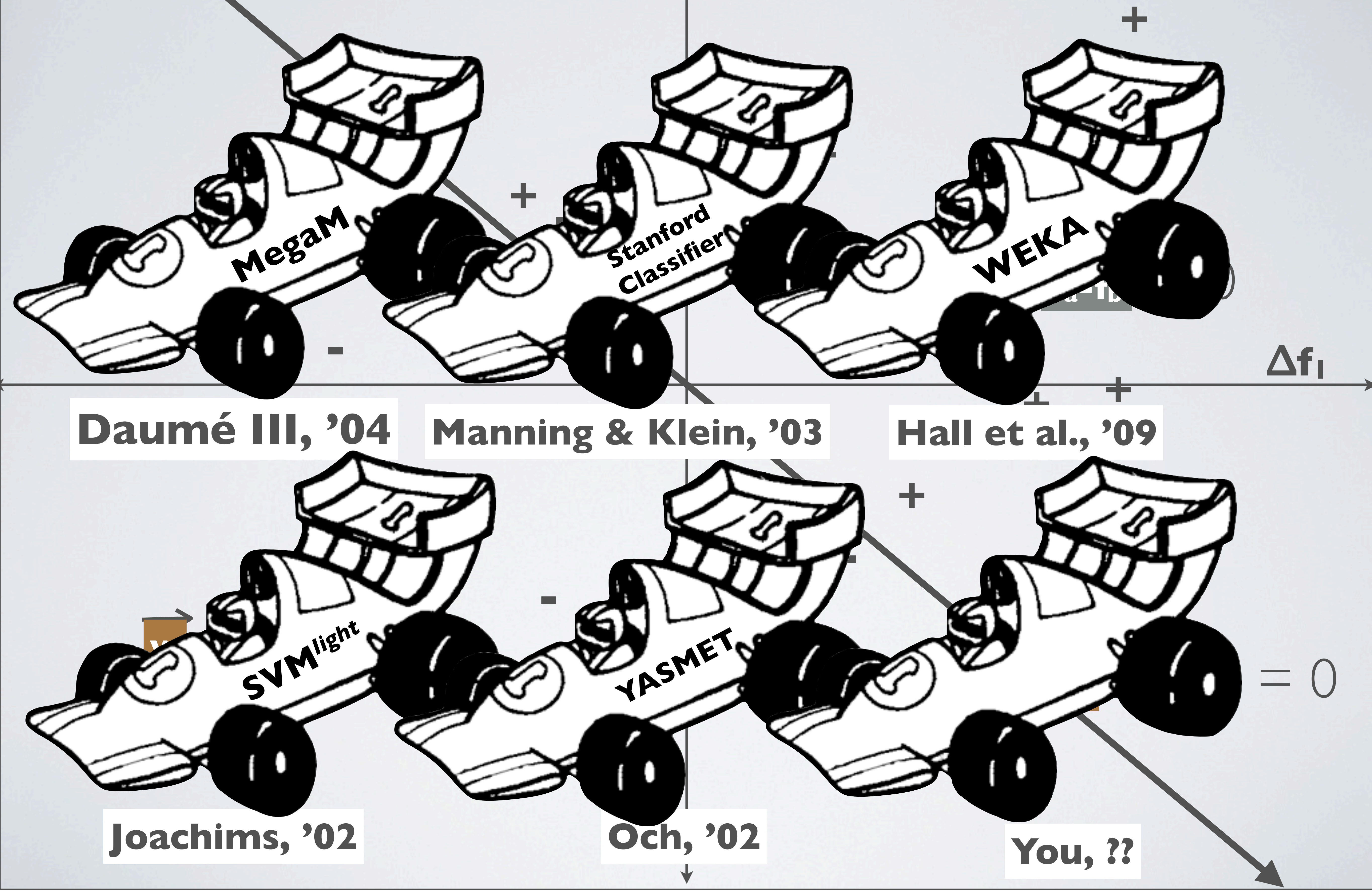
Find the separating vector



Find the separating vector

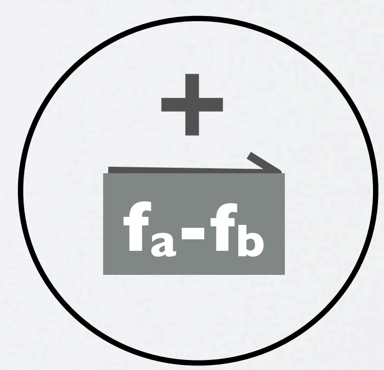


Find the separating vector



Avoid Intractability

- Sample from the pool to avoid blowup
- Focus on difference vectors with large differences
- Add evil twins to ensure balance

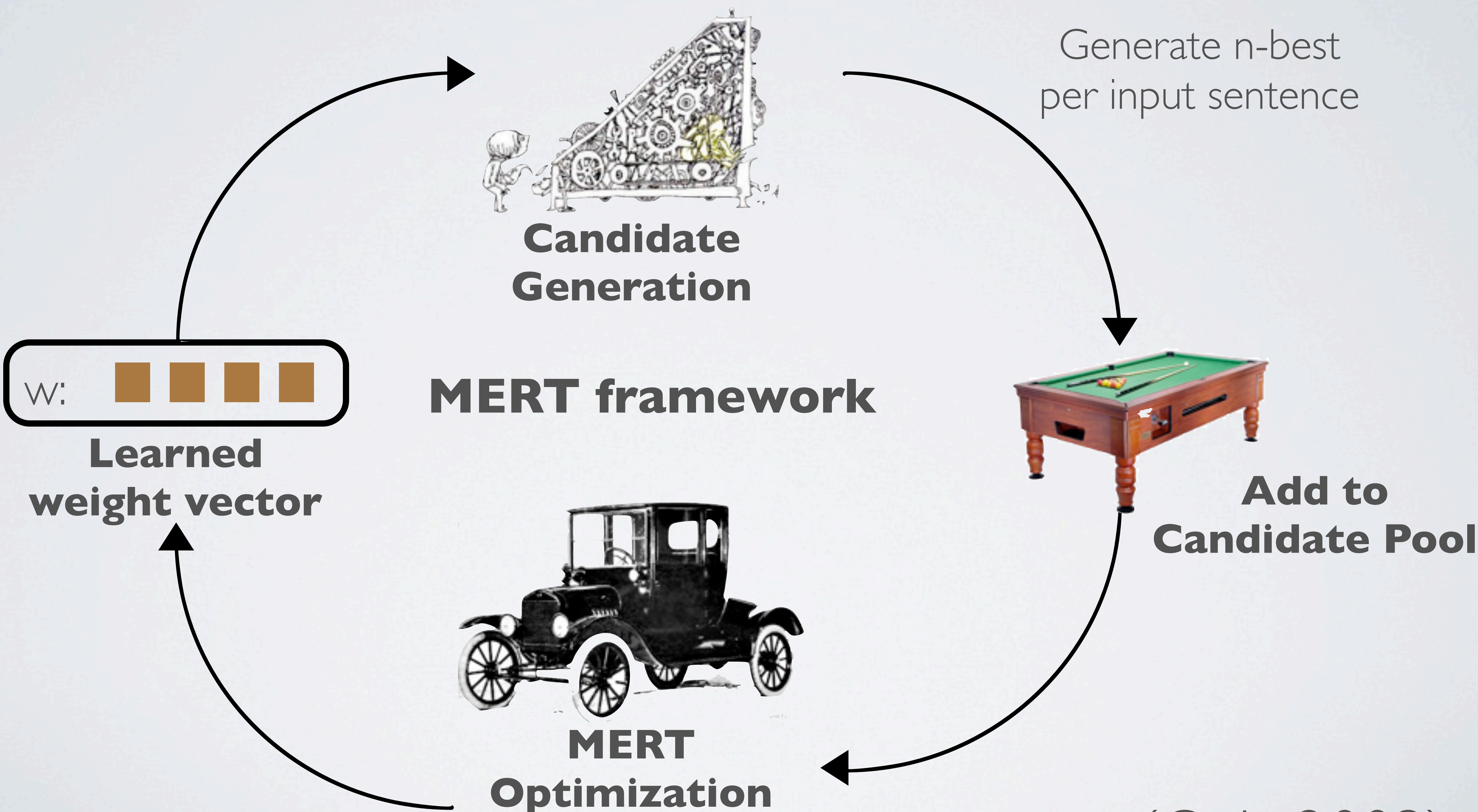


Avoid Intractability

- Sample from the pool to avoid blowup
- Focus on difference vectors with large differences
- Add evil twins to ensure balance

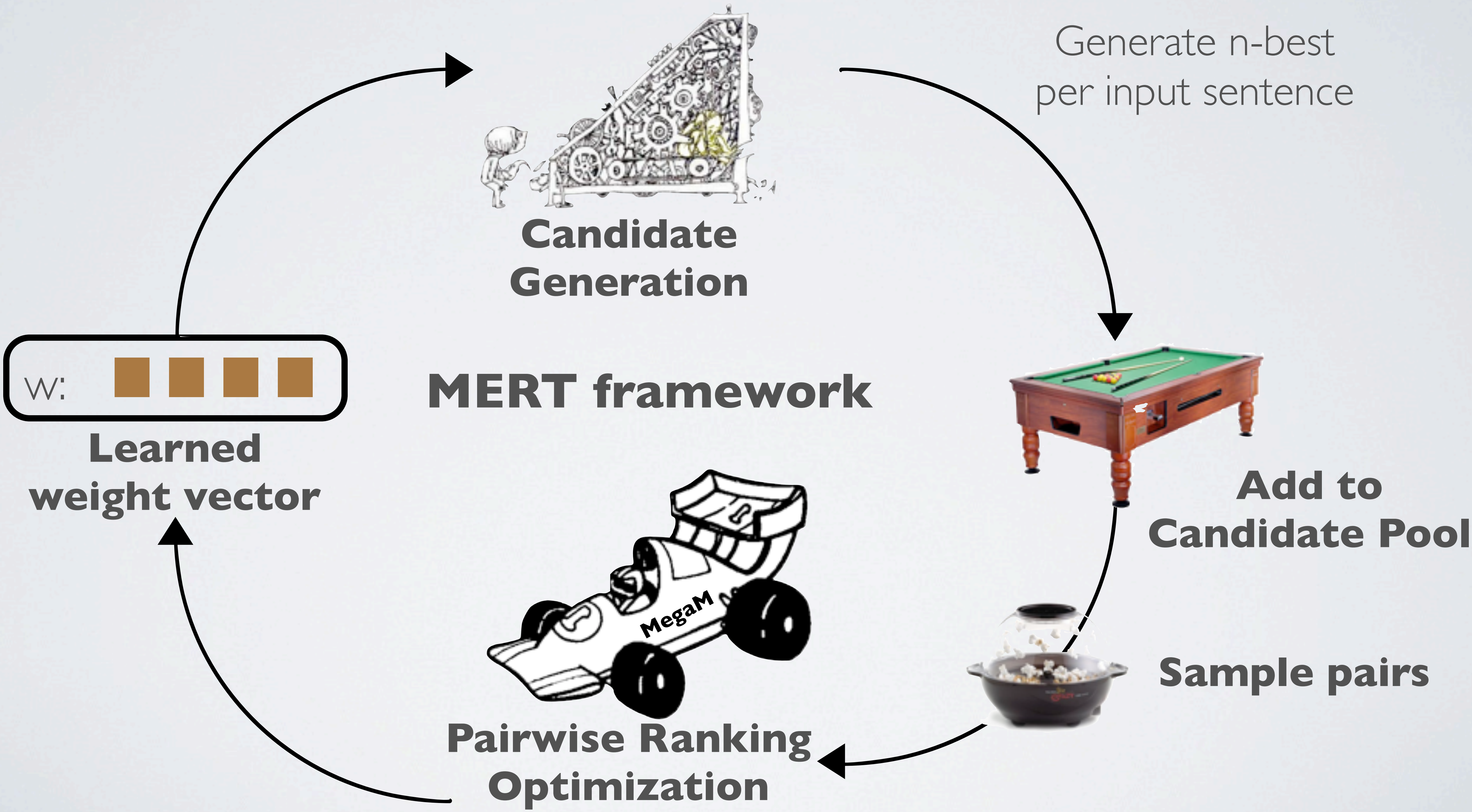
(Nimoy, '68)

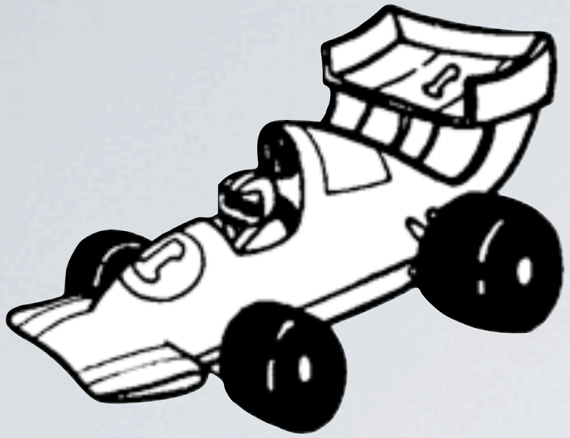
MERT Tuning



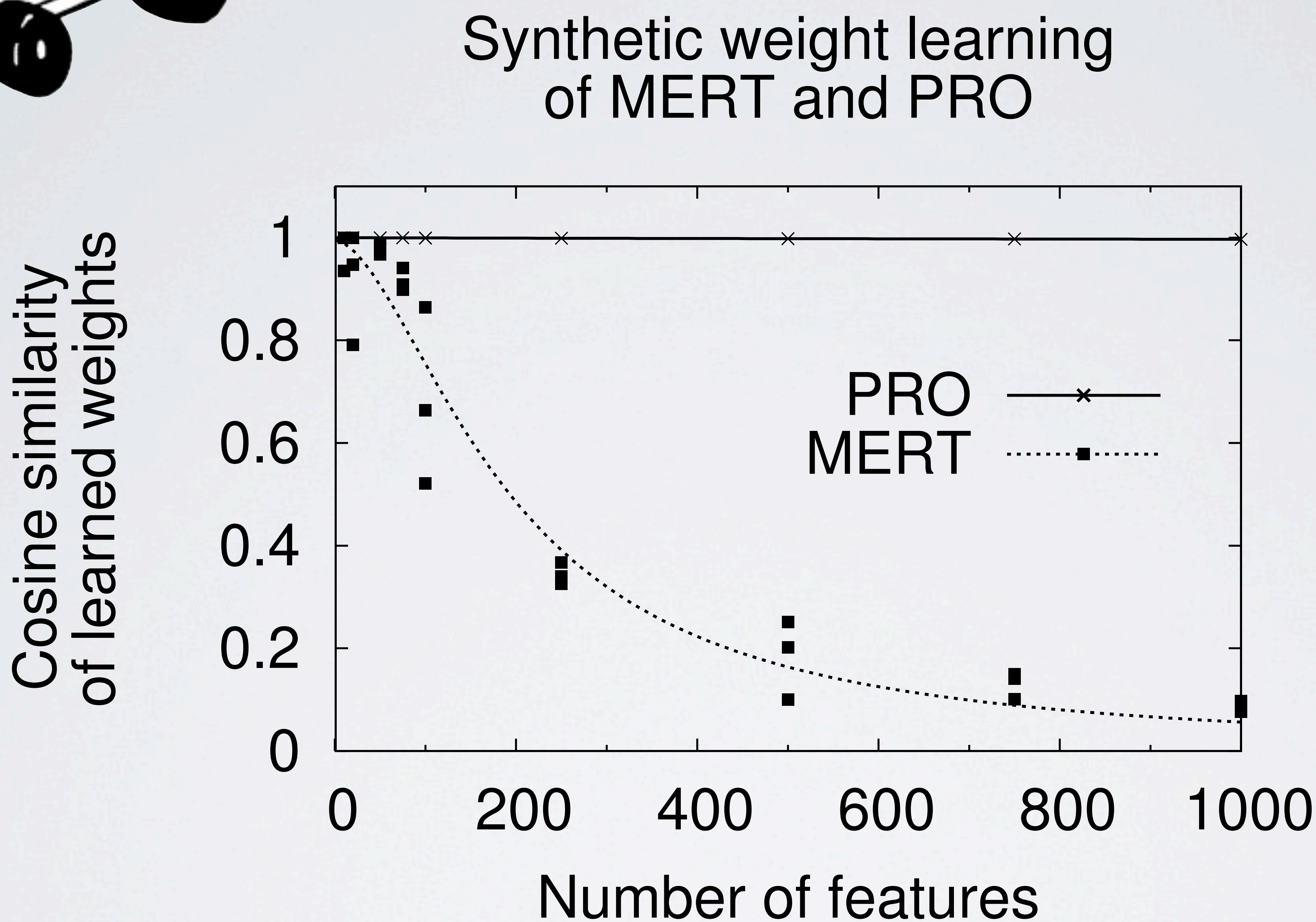
(Och, 2003)

Pairwise Ranking Optimization (PRO) Tuning

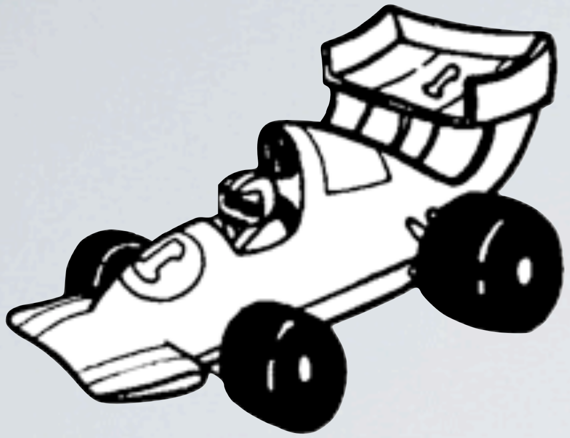




PRO ***scales***

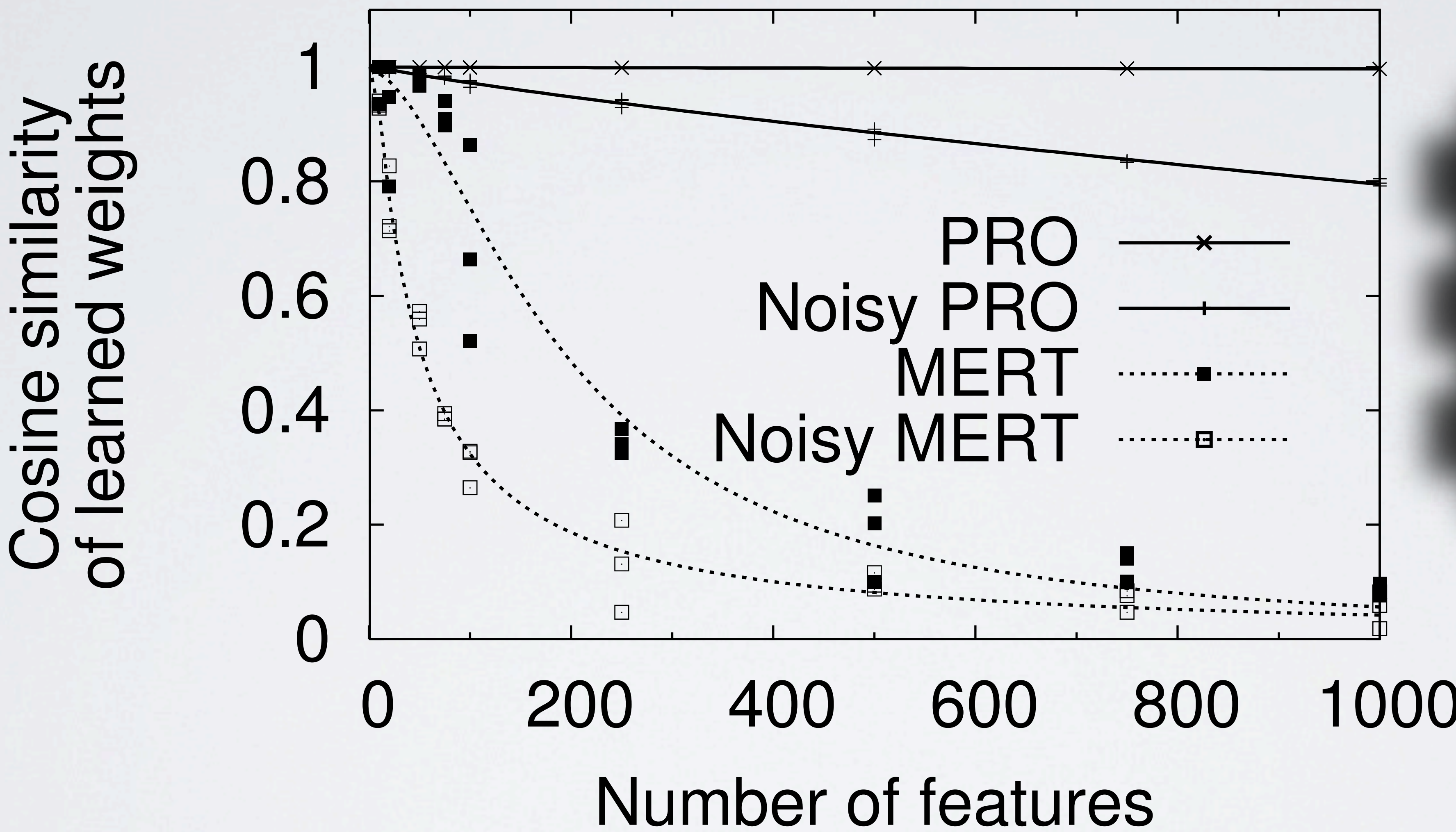


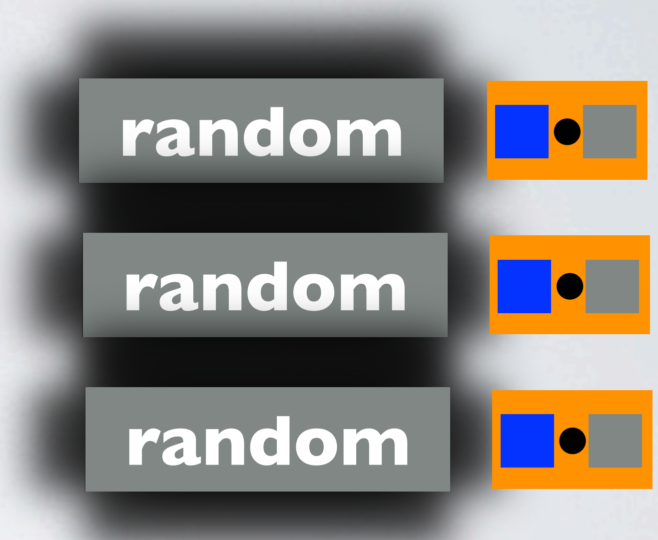
Unlike MERT, PRO is unfazed by
a large number of features in the synthetic test



PRO ***scales***

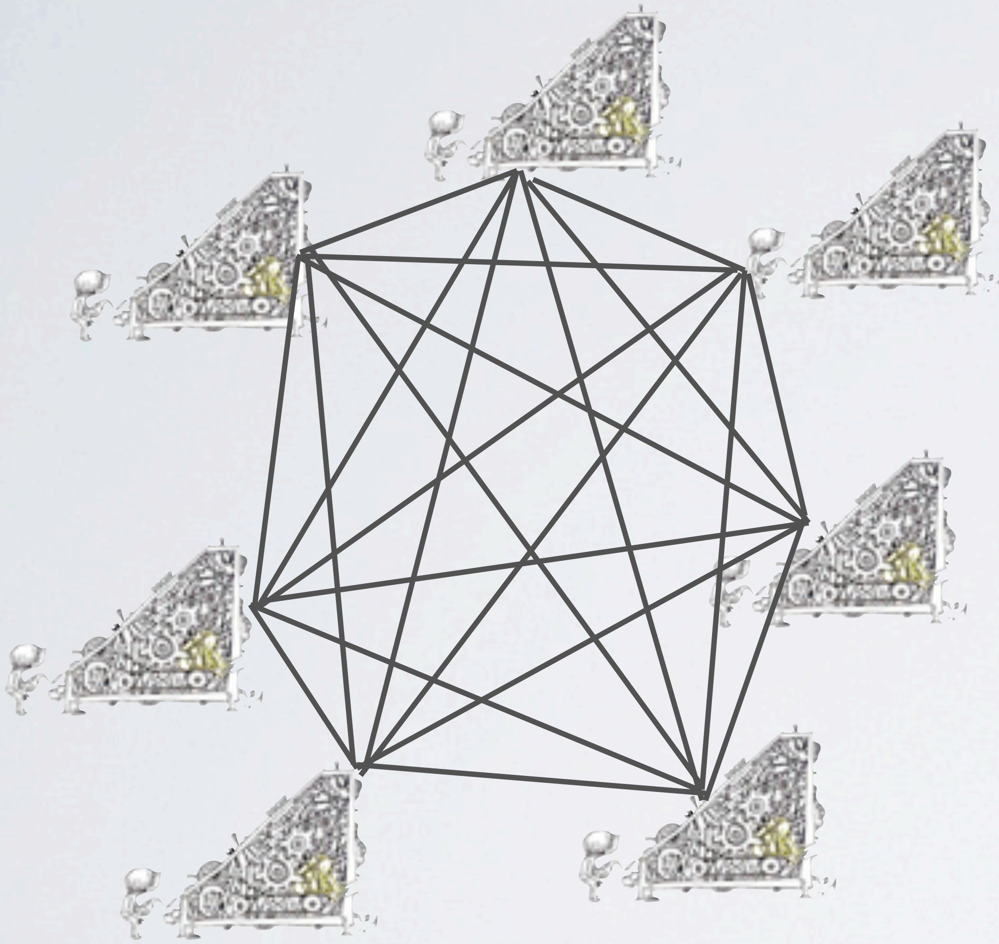
Synthetic weight learning
of MERT and PRO





Adding noise to the synthetic test makes it more difficult but PRO still does quite well compared to MERT

MIRA also scales... but it's ***hard*** to implement



- Like PRO, a discriminative learning algorithm
- Unlike PRO, requires online, simultaneous optimization and decoding
- MIRA tuning must be customized to compute environment (cluster, inter-process communication, reliability concerns)

(Watanabe et al., '07) (Chiang et al., '08, '09)

Unavoidable slide detailing the configuration and data of the experimental conditions....zzzzz

Language	Data (words)			Features			
	Train	Tune	Test	PBMT		SBMT	
				base	ext	base	ext
Urdu-English	2.2M (NIST 2009)	16K (NIST 2008)	18K (NIST 2008)	15	2250	19	277
Arabic-English	175M (NIST 2008)	65K (NIST 03-06/GALE)	47K (NIST 2008)	15	6333	19	352
Chinese-English	173M (GALE 2008)	42K (NIST 03-06)	37K (NIST 2008)	15	1828	19	517

Unavoidable slide detailing the configuration and data of the experimental conditions....zzzzz

Language	Data (words)						
	Train	Tune					
Urdu-English	2.2M (NIST 2009)	16K (NIST 2008)	18K (NIST 2008)	15	2250	19	277
Arabic-English	175M (NIST 2008)	65K (NIST 03-06/GALE)	47K (NIST 2008)	15	6333	19	352
Chinese-English	173M (GALE 2008)	42K (NIST 03-06)	37K (NIST 2008)	15	1828	19	517

Standard large-scale language pairs

Unavoidable slide detailing the configuration and data of the experimental conditions....zzzzz

Language	Data (words)			Features			
	Train	Tune	Test	PBMT		SBMT	
				base	ext	base	ext
Urdu-English	2.[illegible] (NIST [illegible]	[illegible]	[illegible]K ([illegible]2008)	15	2250	19	277
Arabic-English	175M (NIST 2008)	65K (NIST 03-06/GALE)	47K (NIST 2008)	15	6333	19	352
Chinese-English	173M (GALE 2008)	42K (NIST 03-06)	37K (NIST 2008)	15	1828	19	517

State-of-the-art decoders

Unavoidable slide detailing the configuration and data of the experimental conditions...zzzzz

Two feature configurations per decoder

Language	Data (words)			Features			
	Train	Tune	Test	PBMT		SBMT	
				base	ext	base	ext
Urdu-English	2.2M (NIST 2009)	16K (NIST 2008)	18K (NIST 2008)	15	2250	19	277
Arabic-English	175M (NIST 2008)	65K (NIST 03-06/GALE)	47K (NIST 2008)	15	6333	19	352
Chinese-English	173M (GALE 2008)	42K (NIST 03-06)	37K (NIST 2008)	15	1828	19	517

Unavoidable slide detailing the configuration and data of the experimental conditions....zzzzz

Ran MERT, MIRA, PRO

Language	Data (words)			Features			
				PBMT		SBMT	
				base	ext	base	ext
Urdu-English	2.2M (NIST 2009)	16K (NIST 2008)	18K (NIST 2008)	15	2250	19	277
Arabic-English	175M (NIST 2008)	65K (NIST 03-06/GALE)	47K (NIST 2008)	15	6333	19	352
Chinese-English	173M (GALE 2008)	42K (NIST 03-06)	37K (NIST 2008)	15	1828	19	517

Unavoidable slide detailing the configuration and data of the experimental conditions....zzzzz

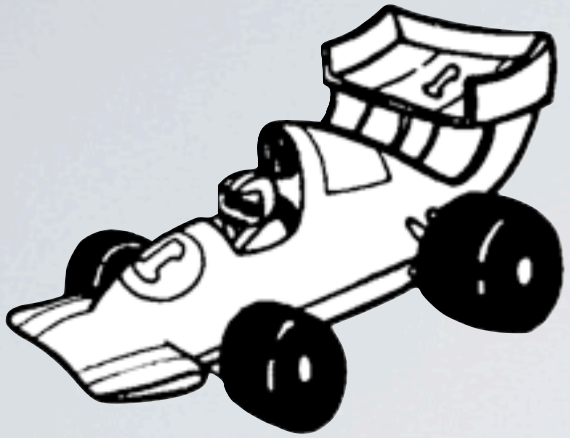
Ran MIRA, PRO (MERT doesn't scale)

Language	Data (words)			Features			
				PBMT		SBMT	
				base	ext	base	ext
Urdu-English	2.2M (NIST 2009)	16K (NIST 2008)	18K (NIST 2008)	15	2250	19	277
Arabic-English	175M (NIST 2008)	65K (NIST 03-06/GALE)	47K (NIST 2008)	15	6333	19	352
Chinese-English	173M (GALE 2008)	42K (NIST 03-06)	37K (NIST 2008)	15	1828	19	517

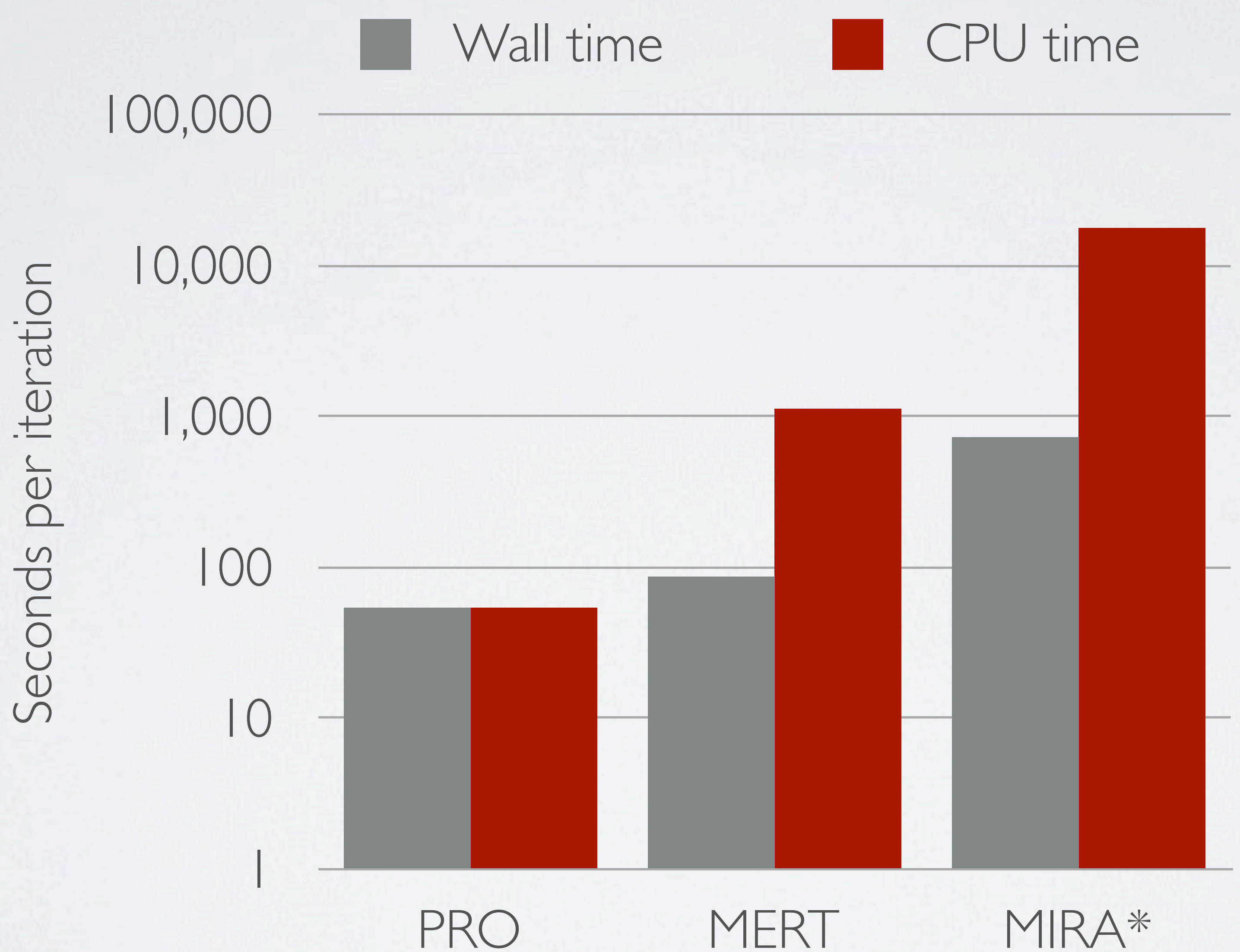
Unavoidable slide detailing the configuration experimental

Report 4-reference, detokenized, mixed-case BLEU

Language	Data (words)			Features			
	Train	Tune	Test	PBMT		SBMT	
				base	ext	base	ext
Urdu-English	2.2M (NIST 2009)	16K (NIST 2008)	18K (NIST 2008)	15	2250	19	277
Arabic-English	175M (NIST 2008)	65K (NIST 03-06/GALE)	47K (NIST 2008)	15	6333	19	352
Chinese-English	173M (GALE 2008)	42K (NIST 03-06)	37K (NIST 2008)	15	1828	19	517

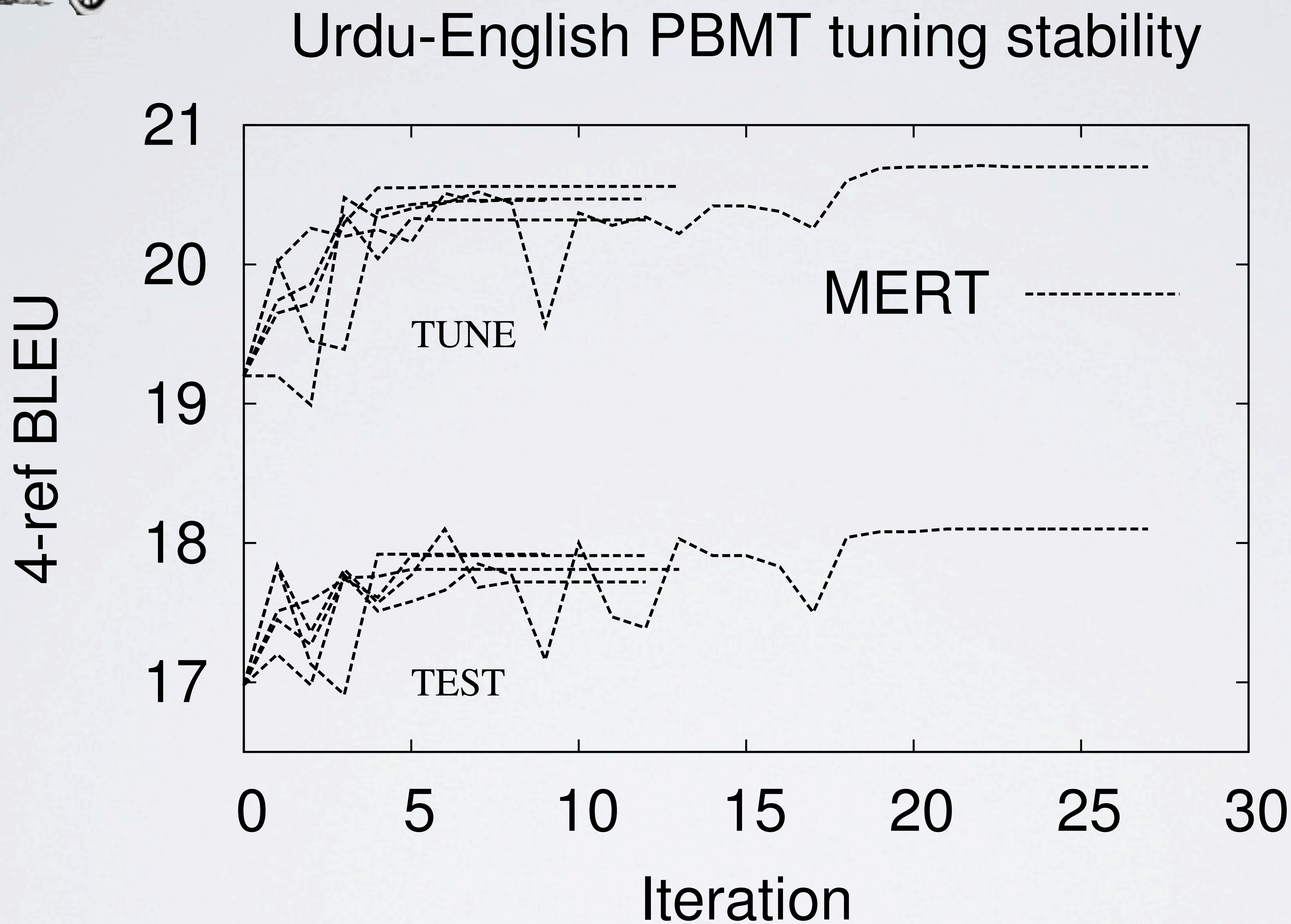


PRO is ***fast***



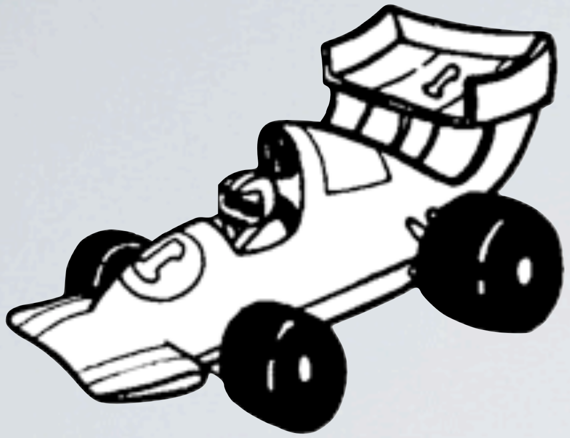
* Your implementation of MIRA may be faster

MERT is ***unstable***

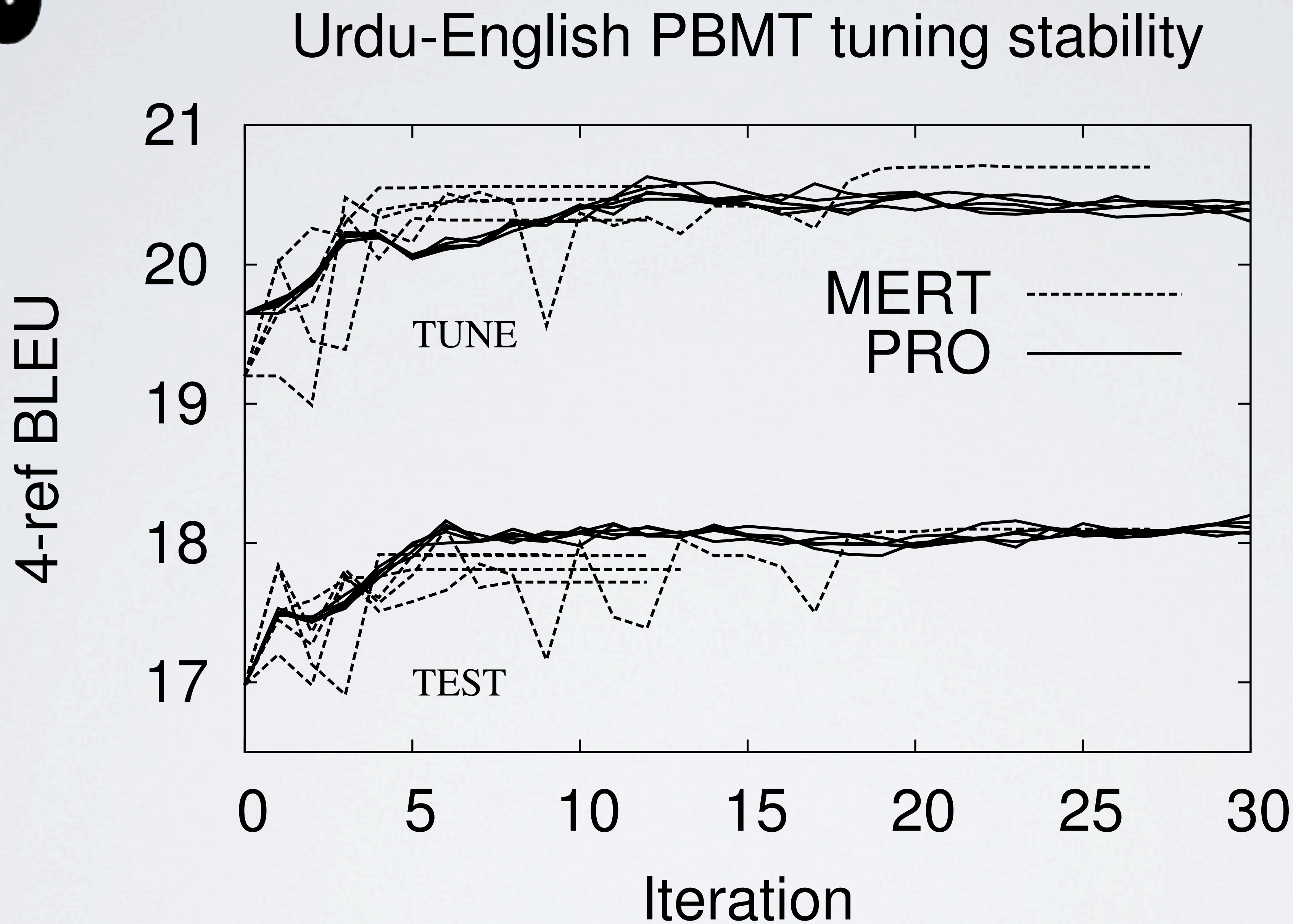


Result from five identical runs

(Clark et al., 2011)

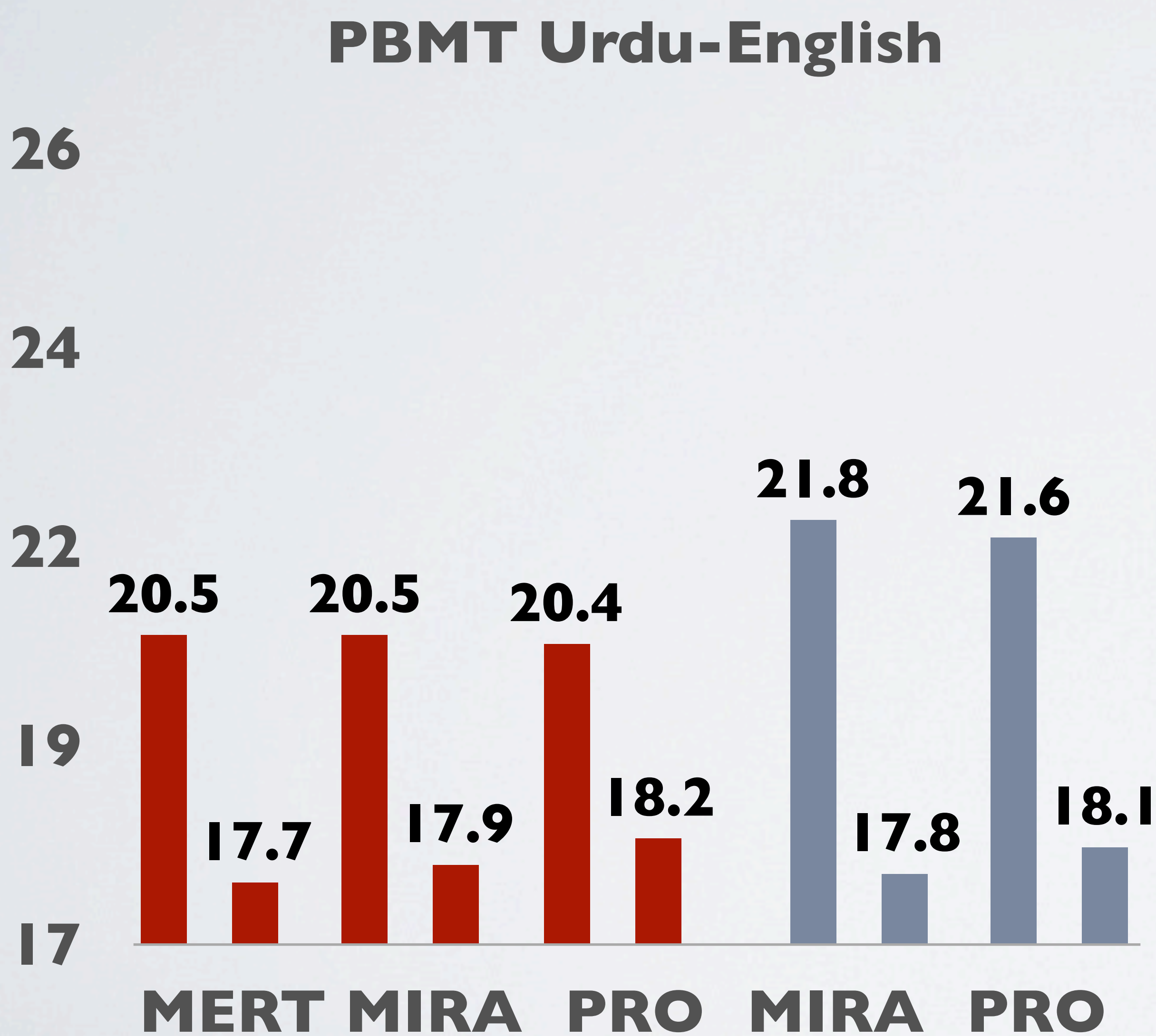


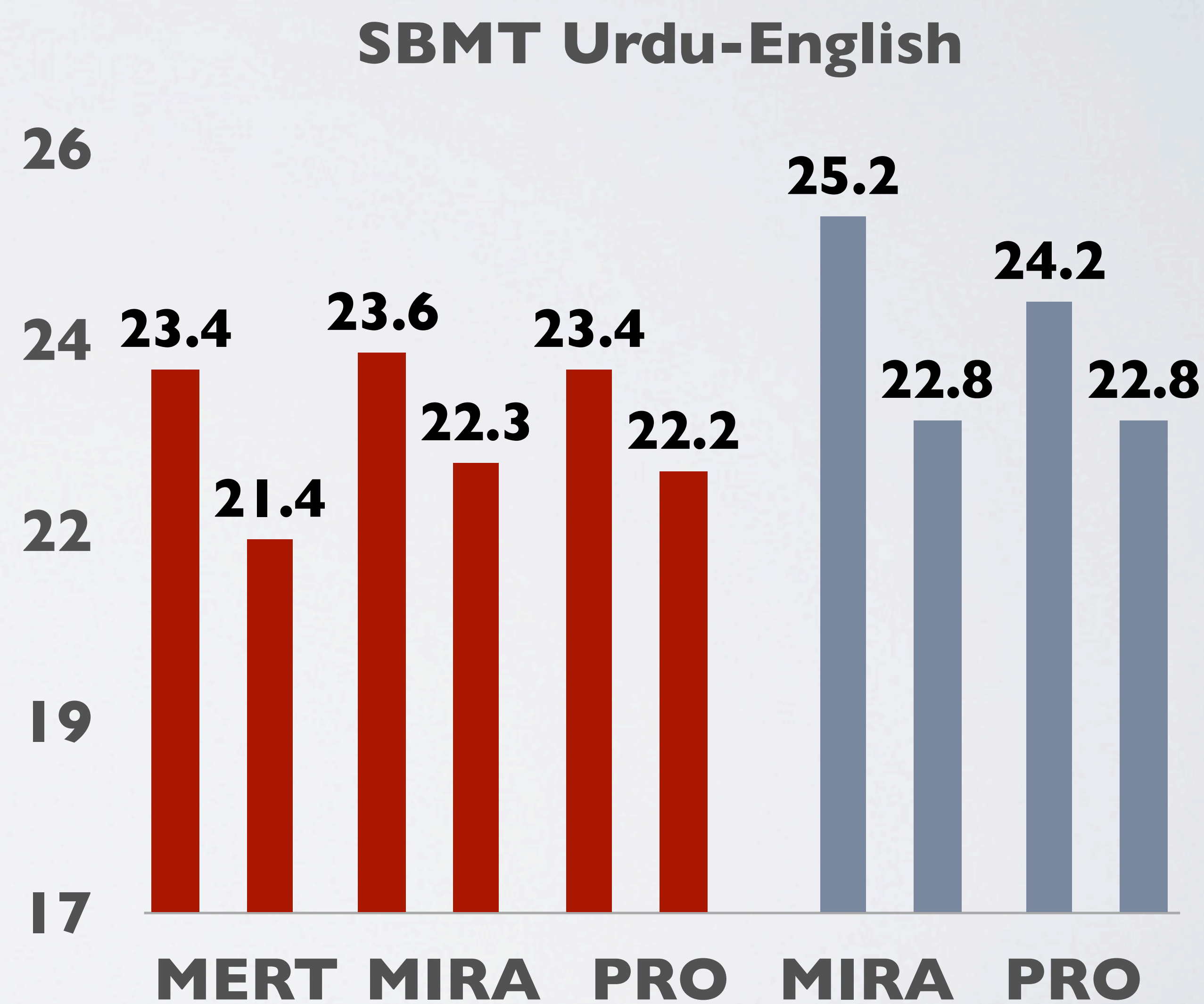
PRO is ***stable***



Result from five identical runs

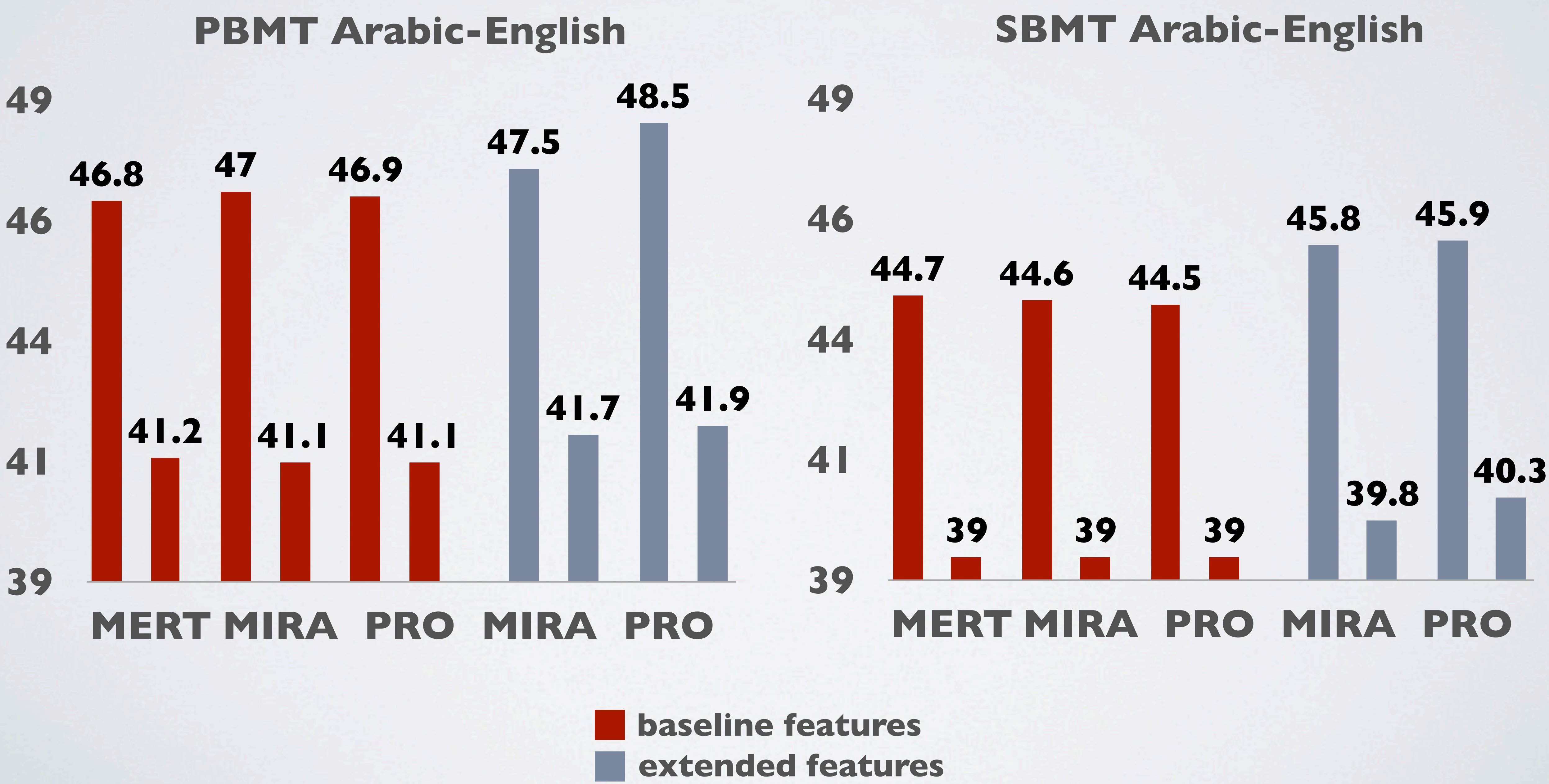
MERT vs. MIRA vs. ***PRO***



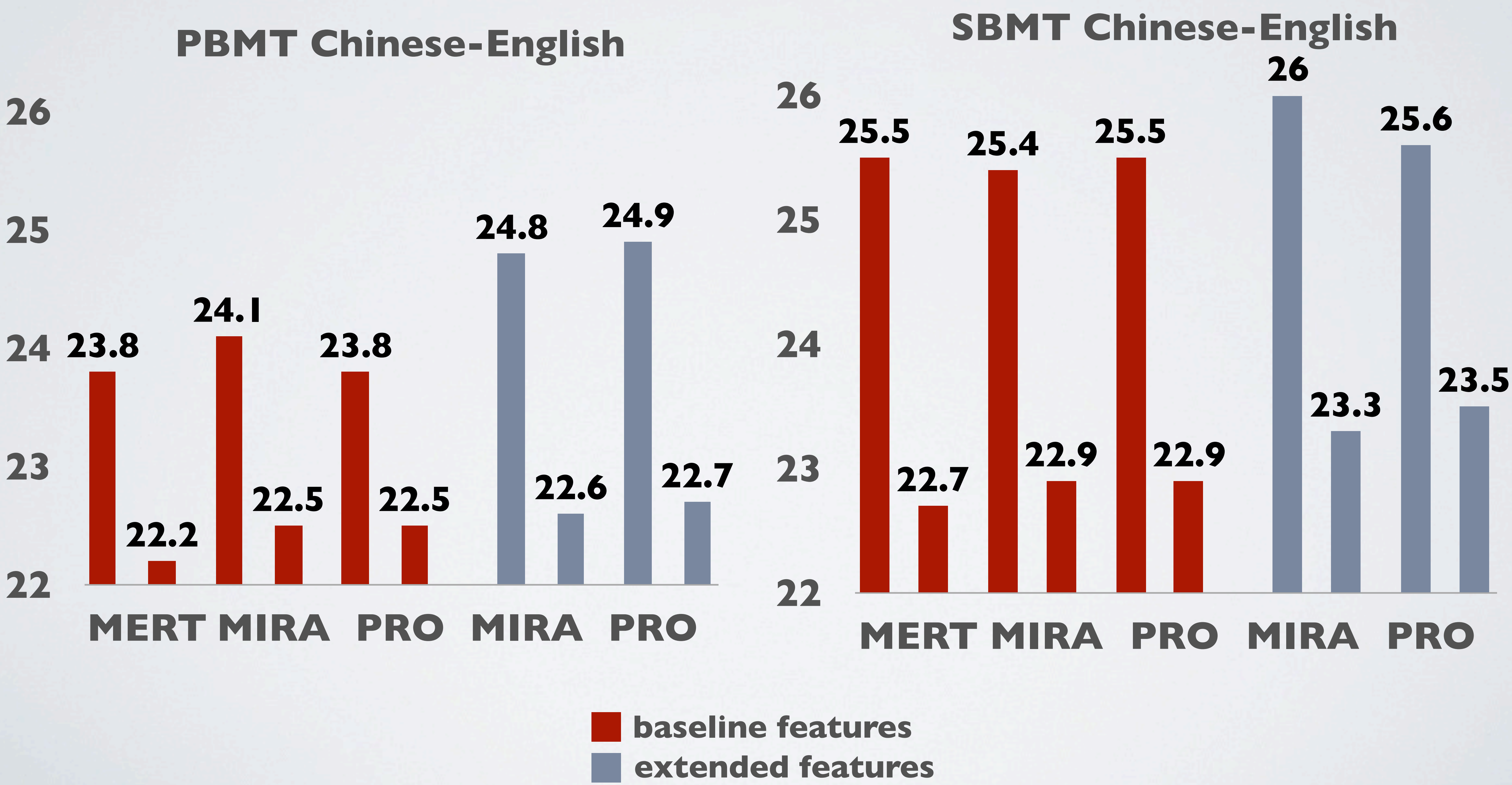


baseline features
extended features

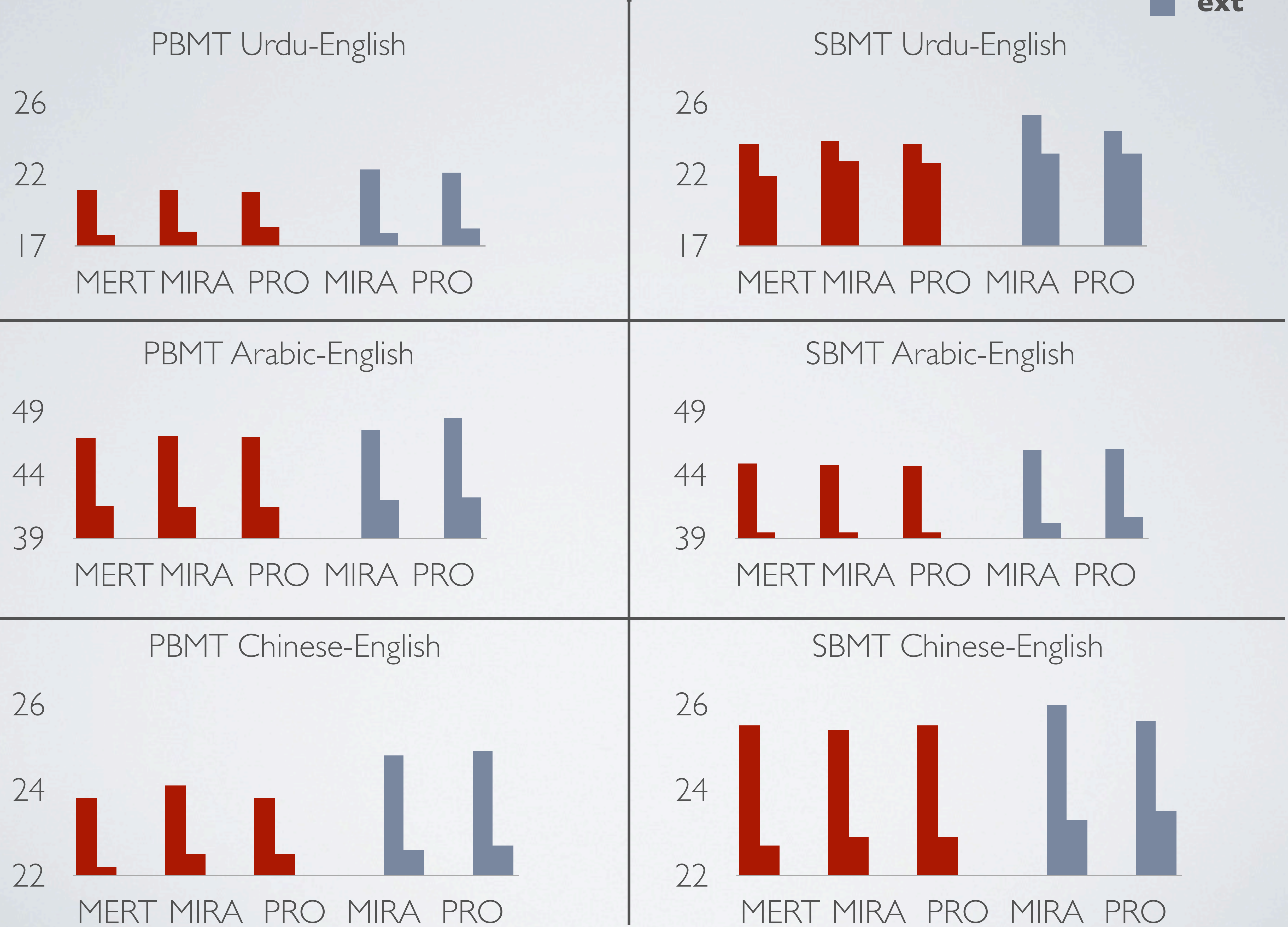
MERT vs. MIRA vs. ***PRO***



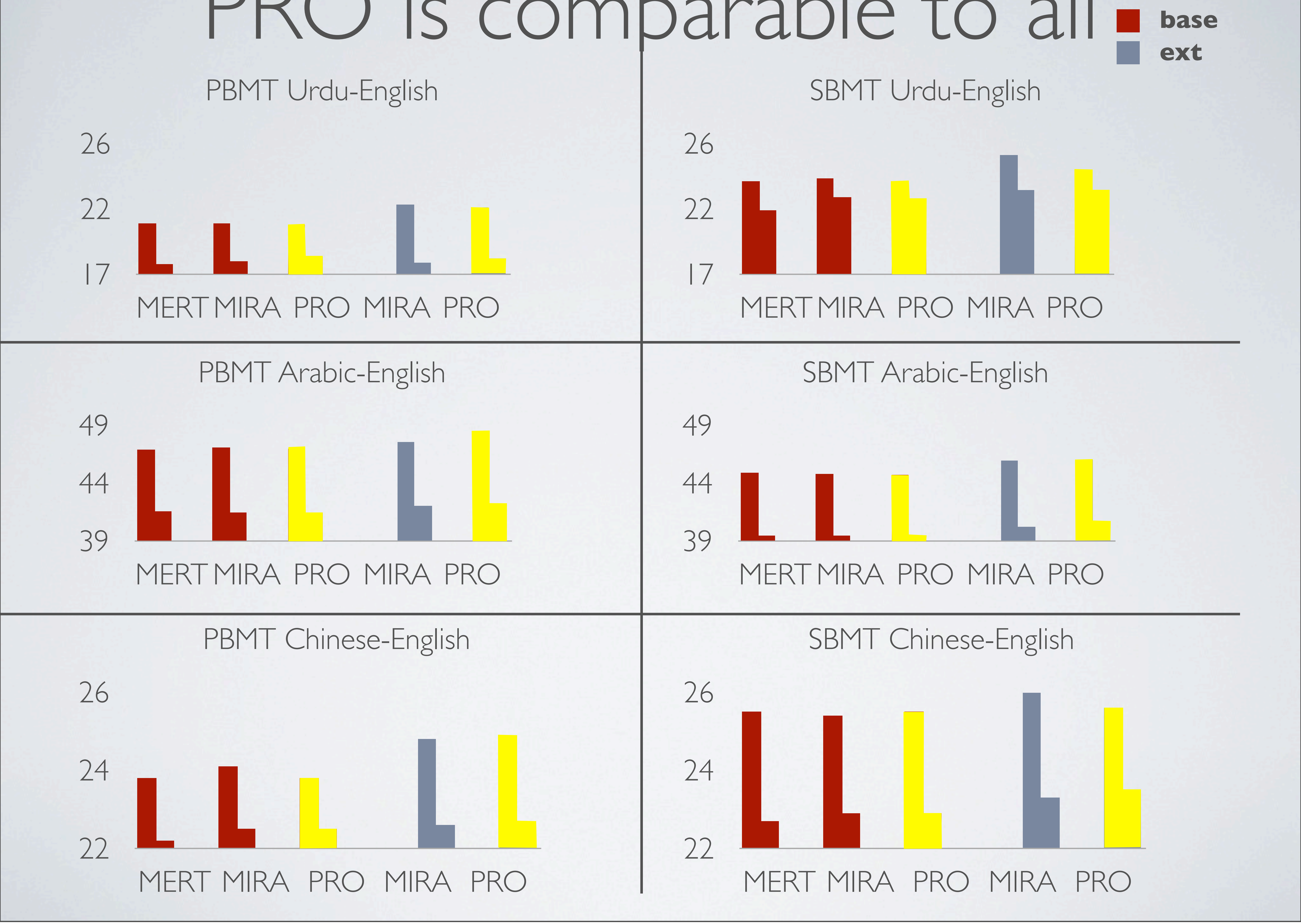
MERT vs. MIRA vs. *PRO*



PRO is comparable to all



PRO is comparable to all



Related Work

SampleRank
(Culotta, '08, Wick et al., '09, Roth et al., '10)

Similar approach, with guided search through pool space (See Haddow et al. in WMT)

Classifier-based Weight Learning
(Tillmann & Zhang, '05, Och & Ney, '02 Ittycheriah & Roukos, '05, Xiong et al., '06)

Various approaches using classifiers to learn MT feature weights -- these do not use the difference vector approach

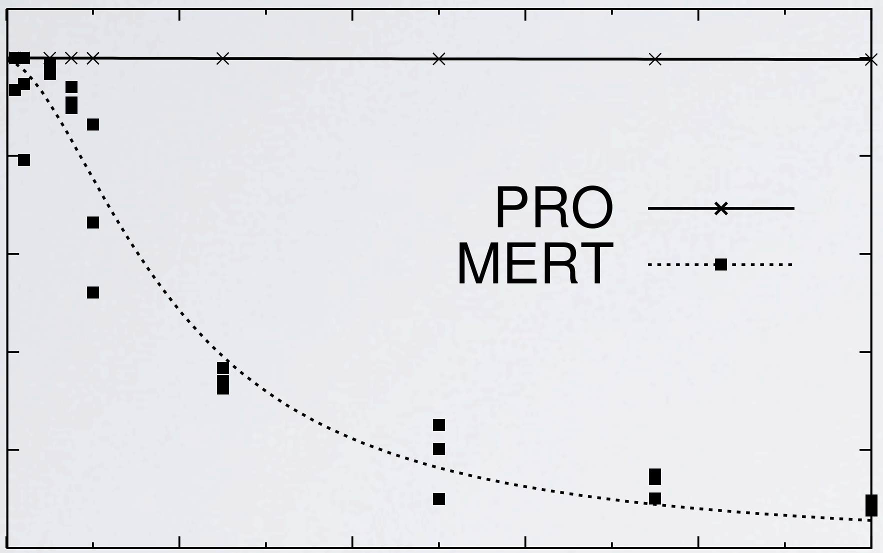
Discriminative Re-ranking
(Shen et al., '04, Cowan et al., '06, Watanabe et al., '06)

Changing the n-best list after decoding using similar techniques to ours

Why Use **PRO?**

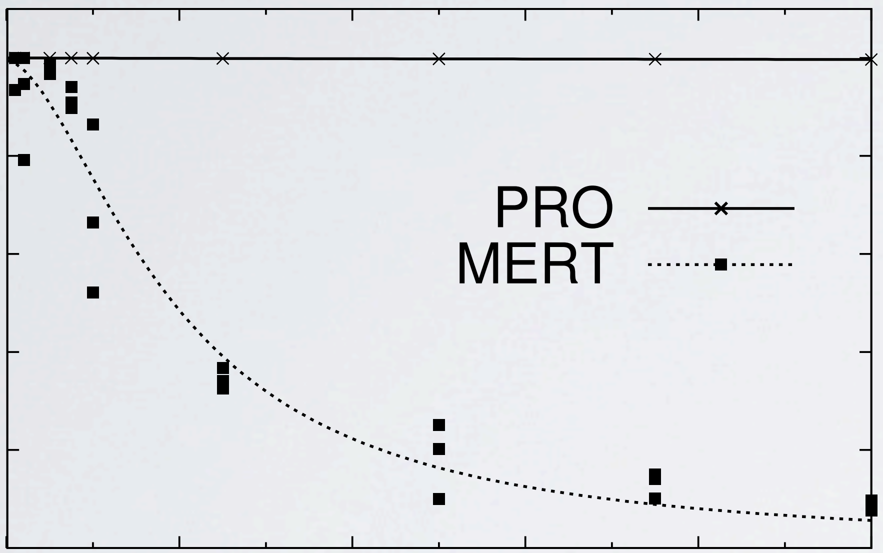
Why Use **PRO?**

It's **scalable**

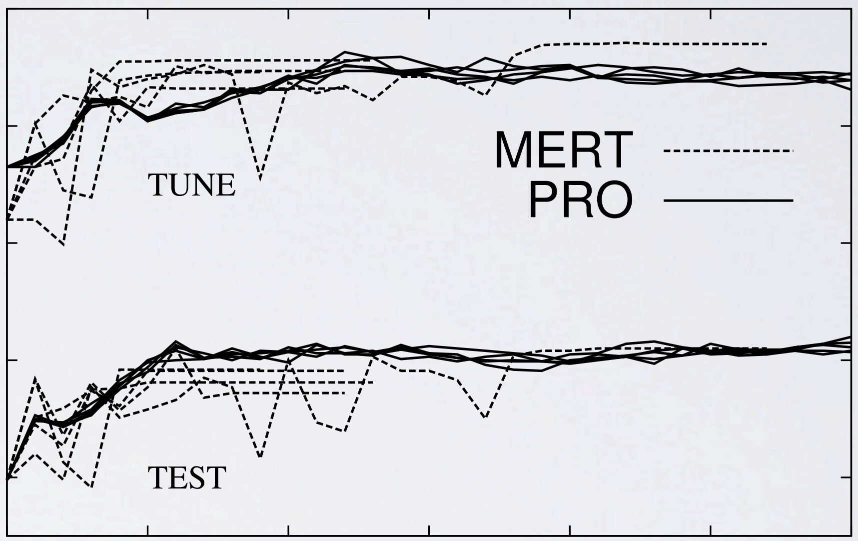


Why Use **PRO?**

It's **scalable**

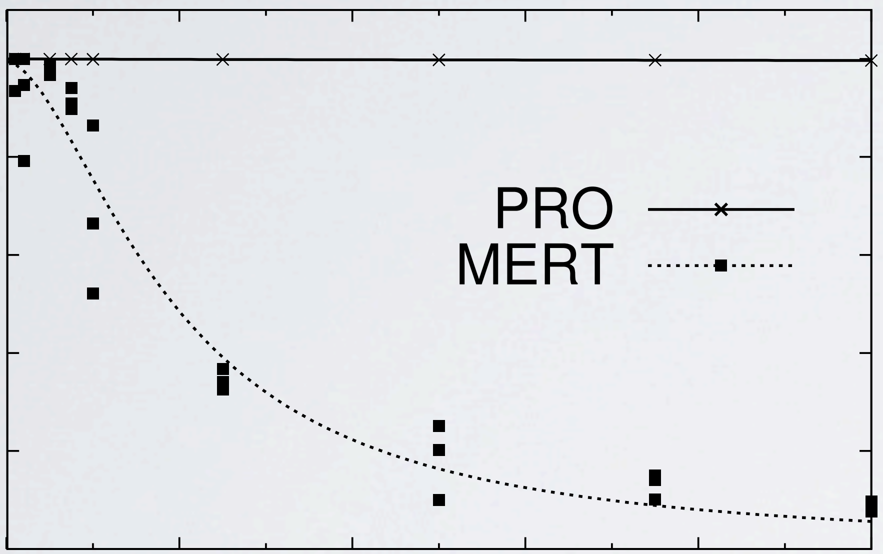


It's **stable**

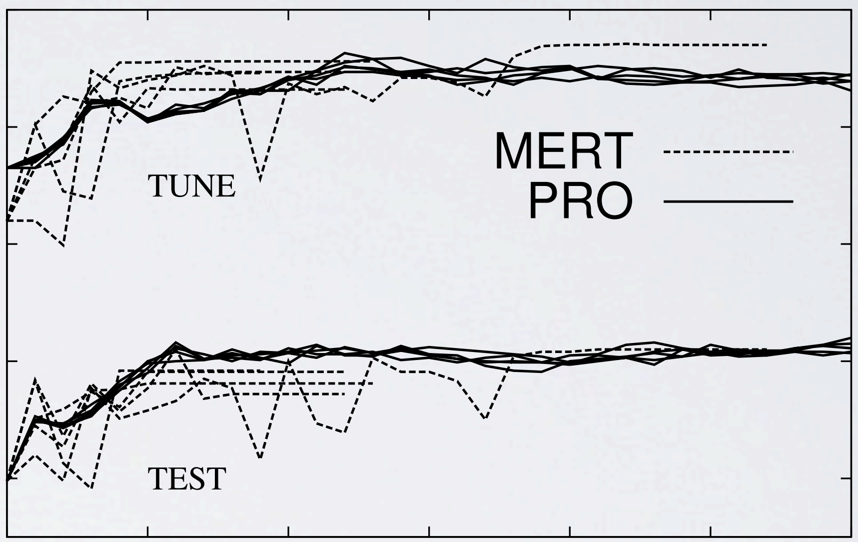


Why Use **PRO?**

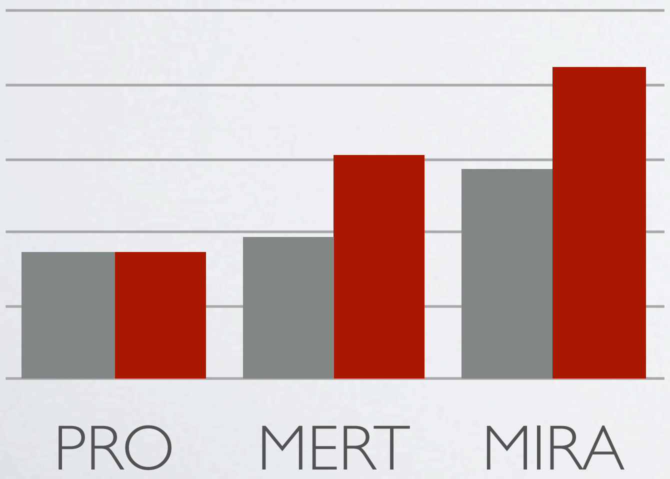
It's **scalable**



It's **stable**

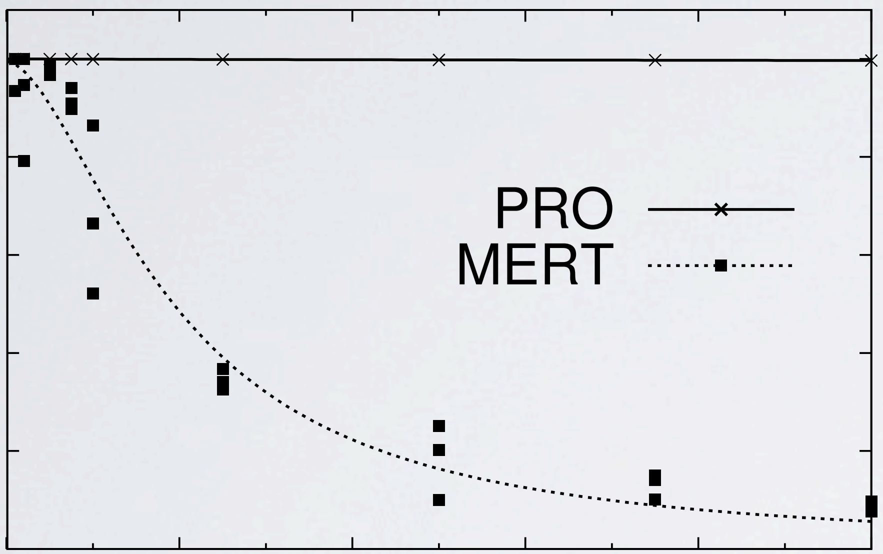


It's **fast**

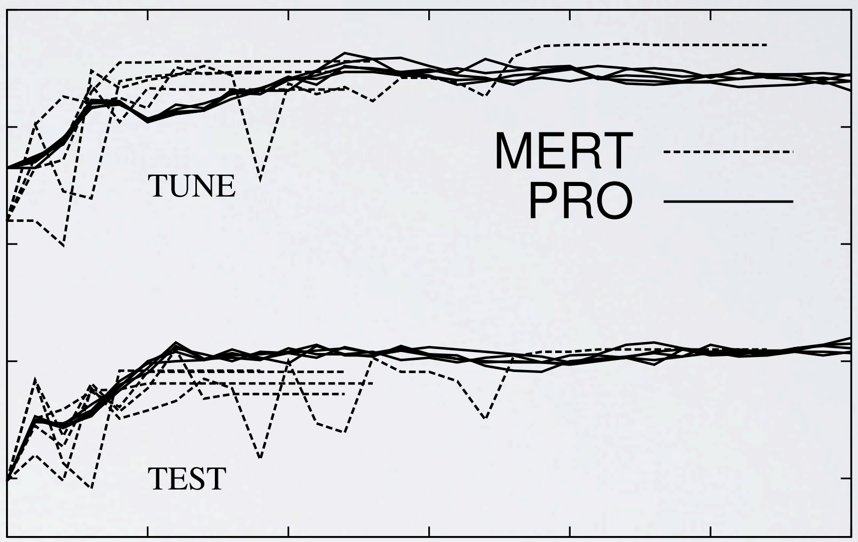


Why Use **PRO?**

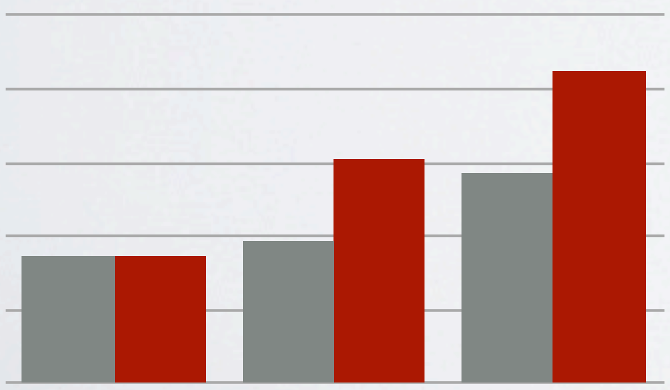
It's **scalable**



It's **stable**



It's **fast**

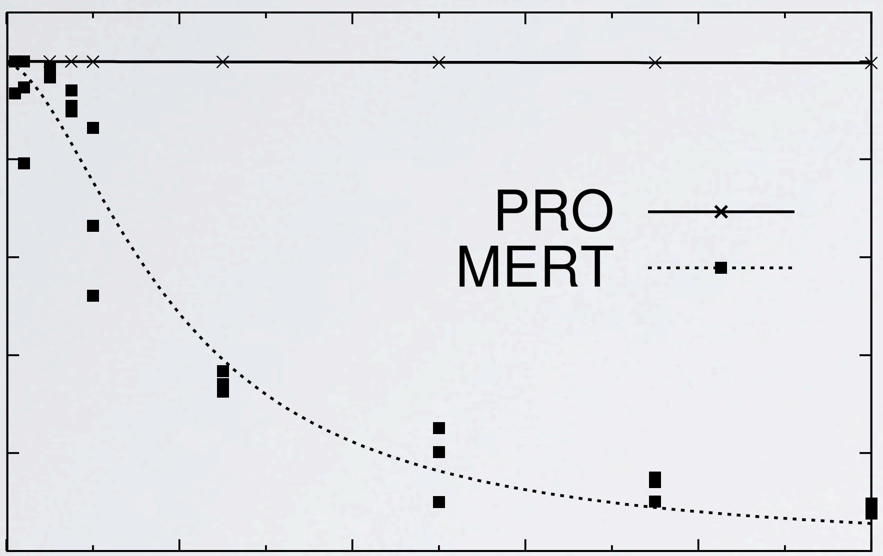


It's **easy**

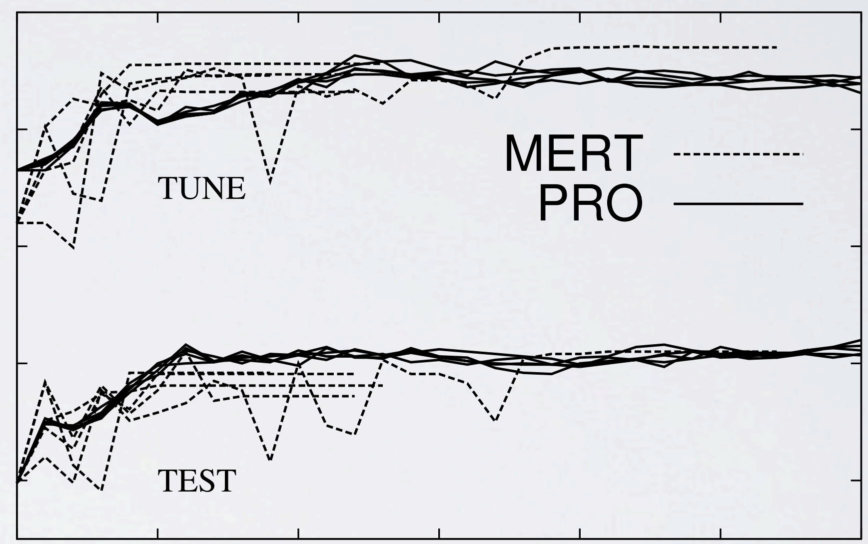
At least **three** external implementations prior to this talk

Why Use **PRO?**

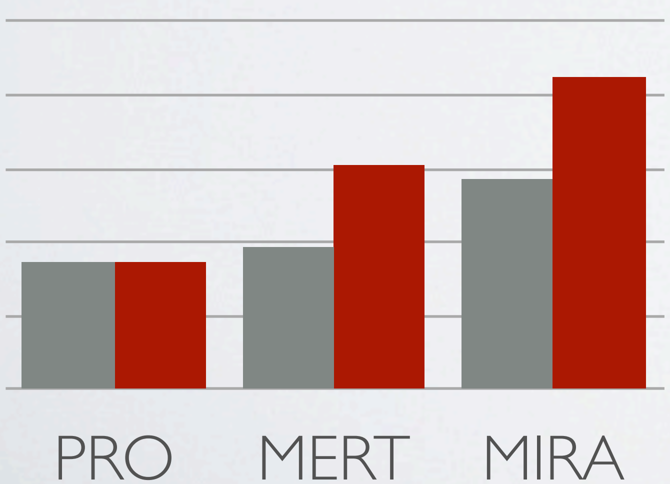
It's **scalable**



It's **stable**



It's **fast**

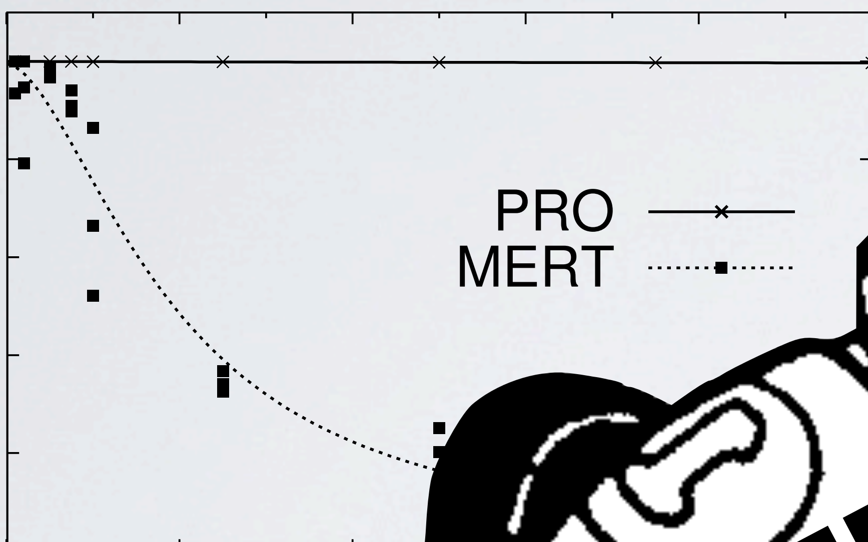


It's **easy**

https://github.com/redpony/cdec/tree/master/pro-train

Why Use **PRO?**

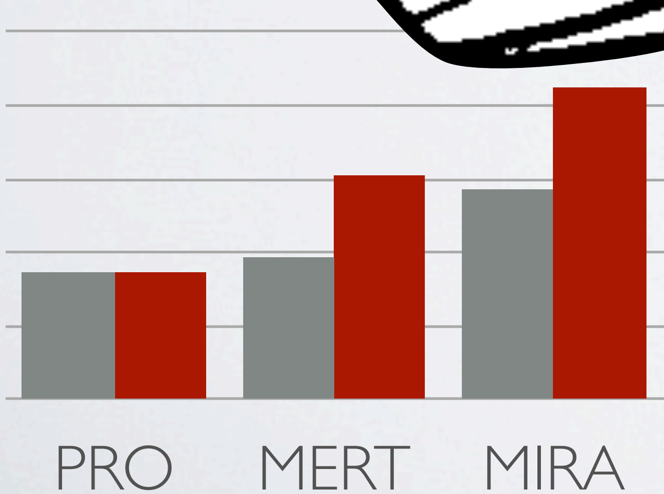
It's **scalable**



THANKS!

It's **easy**

It's **fa**



"Including mine!" (Dyer, P.C.)

https://github.com/redpony/cdec/tree/master/pro-train