SMT Model Building with MapReduce

Chris Dyer, Alex Mont, Aaron Cordova, Jimmy Lin

A brief intoduction

Statistical machine translation in a (idealized) nutshell:

$$\hat{e} = \arg\max_{e} P(e \mid f)$$

$$\hat{e} = \arg\max_{e} P(f \mid e) P(e)$$

We consider two decompositions:

-Word-based models (used for word alignment)

-Phrase-based models (used for translation)

A brief intoduction

Statistical machine translation in a (idealized) nutshell:

$$\hat{e} = \arg\max_e P(e \mid f)$$

$$\hat{e} = \arg\max_e P(f \mid e) P(e)$$

We consider two decompositions:

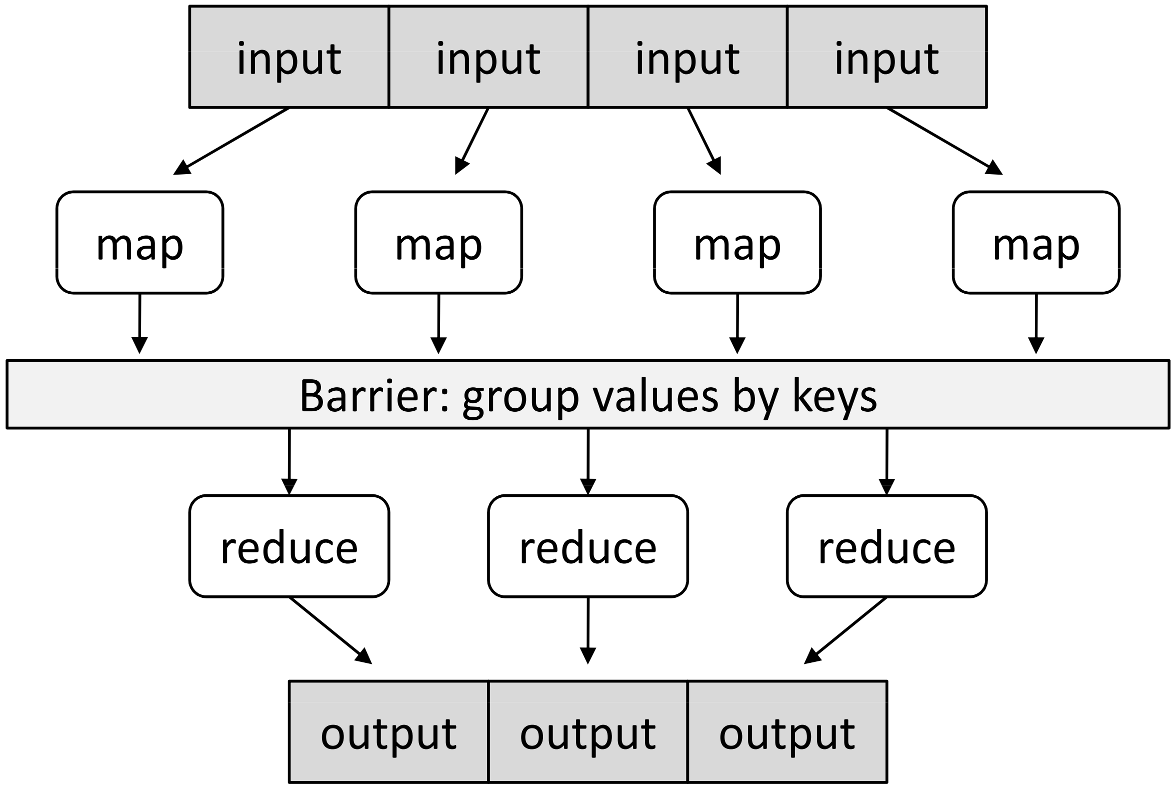
-Word-based models (used for word alignment)
-Phrase-based models (used for translation)

How do we estimate the parameters efficiently?

Outline

- MapReduce
- SMT & the SMT pipeline
- MapReducing relative frequencies
- Experimental results
- Future directions

MapReduce



User supplies these functions:

map	`(k1,v1)`	$\rightarrow$ `list(k2,v2)`
reduce	`(k2,list(v2))`	$\rightarrow$ `list(v2)`

MapReduce: example

Count the words: "Hello, world!" for MapReduce

```
Map(input):
  for each w in input:
    emit <w,1>
```

MapReduce: example

Count the words: "Hello, world!" for MapReduce

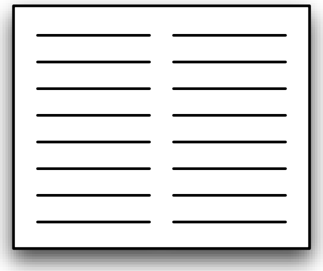
Map(*input*):
  for each *w* in *input*:
    emit <*w*,1>

Reduce(*key, values*):
  sum = 0
  for each *val* in *values*:
    sum += *val*
  emit(*key*, *sum*)

MapReduce

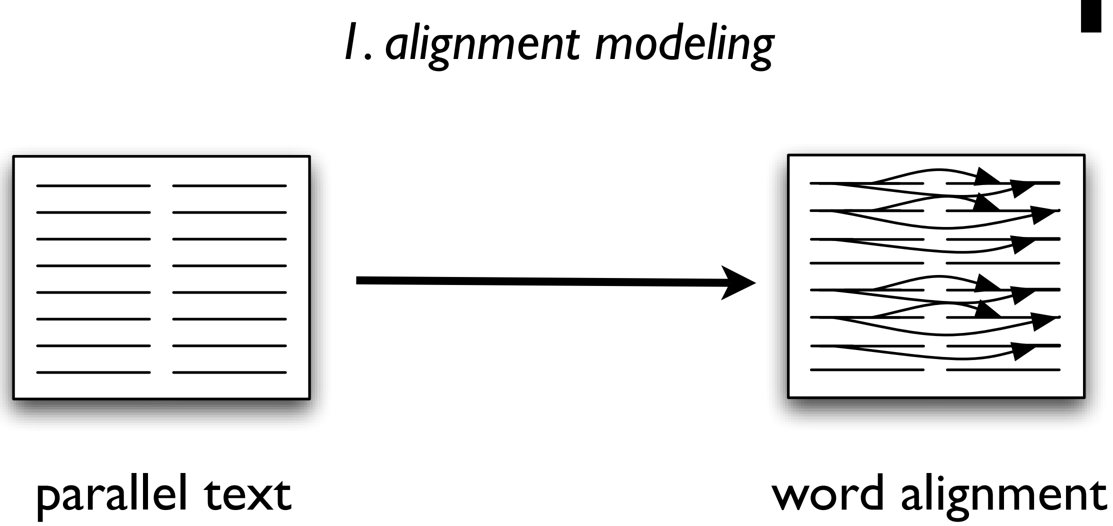
- Benefits
  - Highly scalable
  - Fault tolerant
  - Hides details of concurrence from user
  - Runs on commodity hardware
    - Store massive logical files across small disks!

The Phrase-Based SMT Pipeline

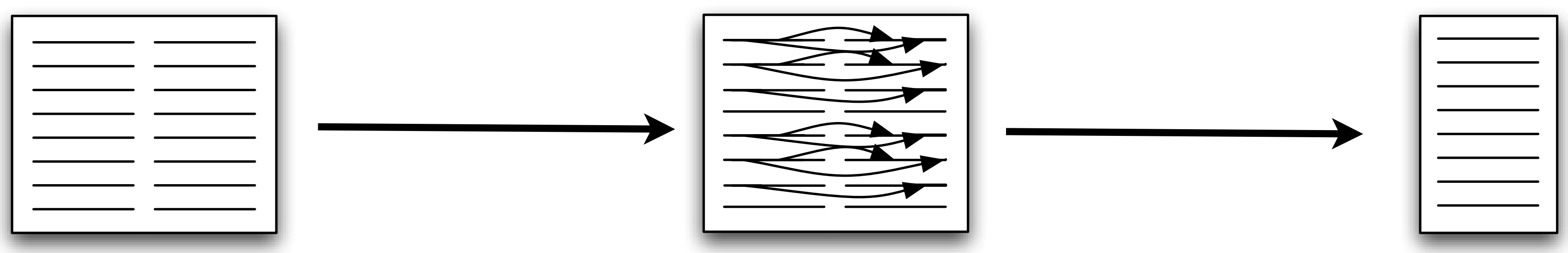


parallel text

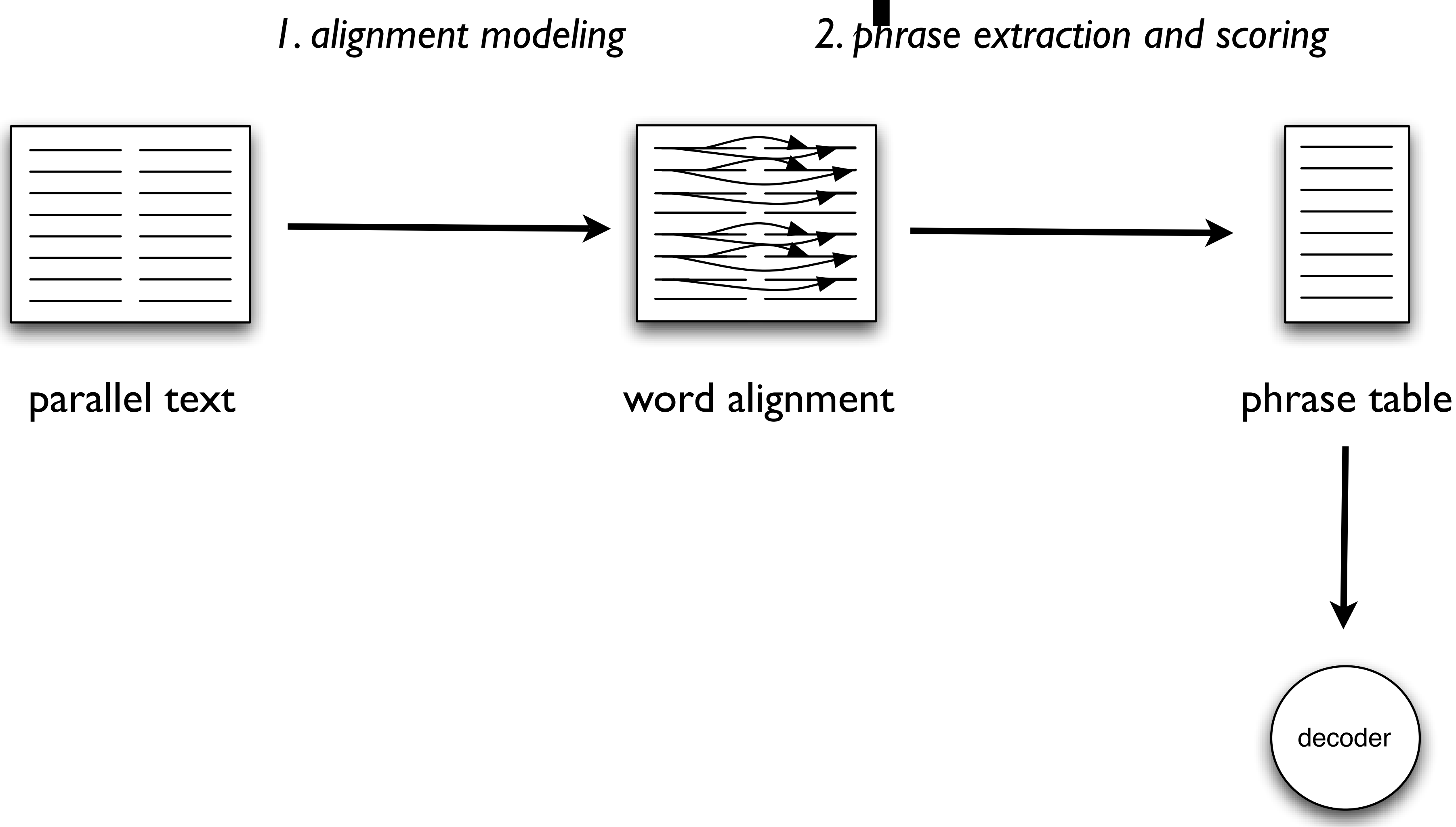
The Phrase-Based SMT Pipeline



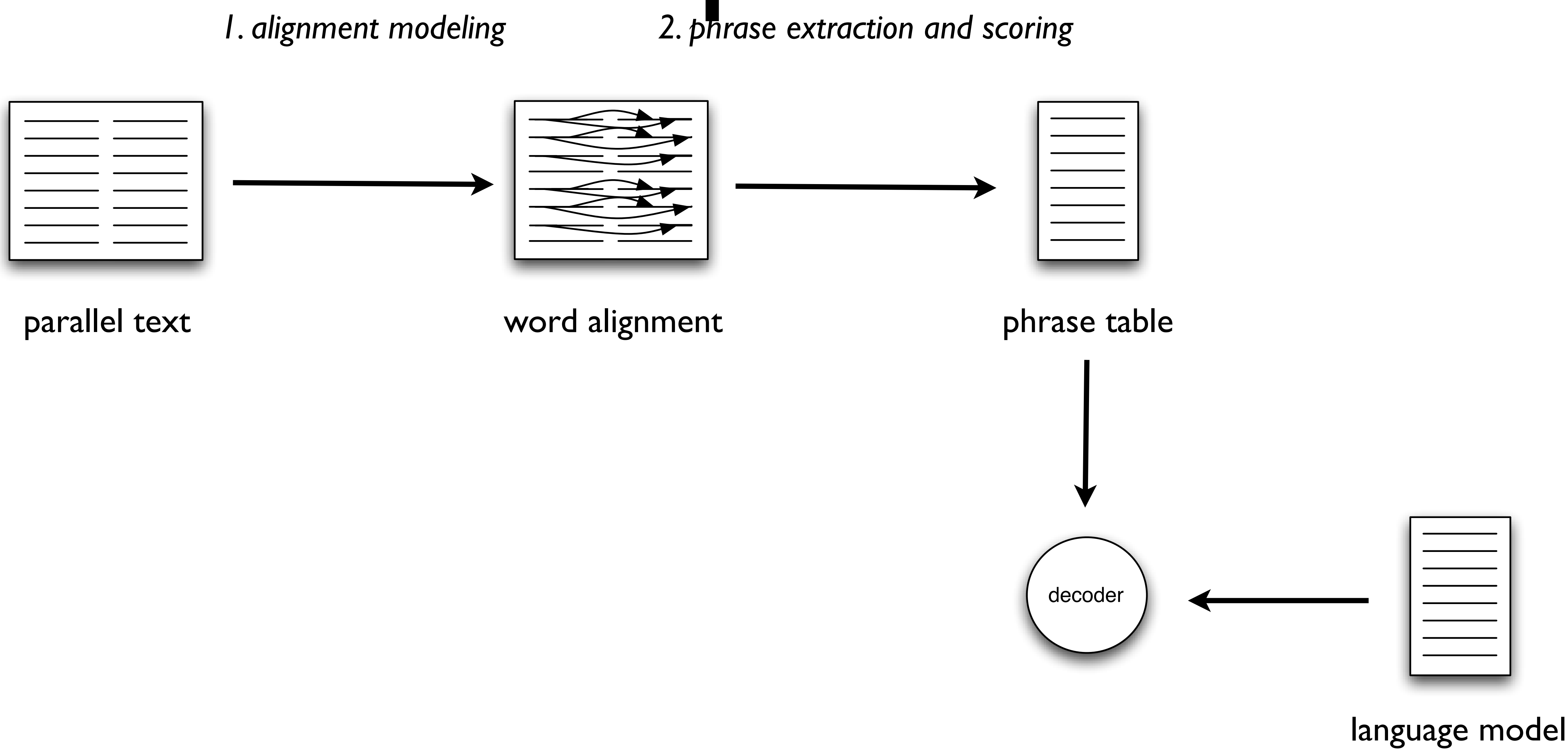
The Phrase-Based SMT Pipeline



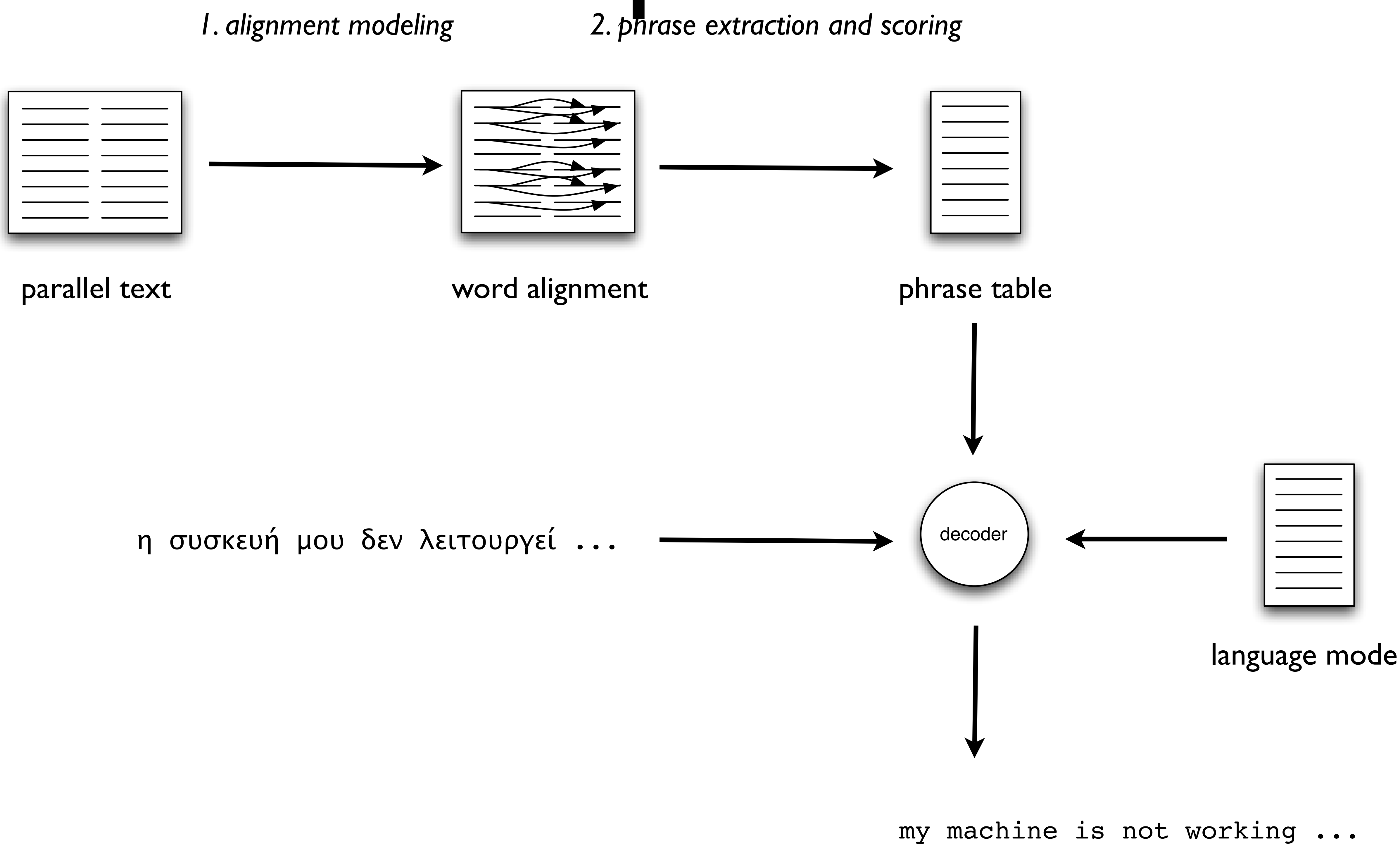
The Phrase-Based SMT Pipeline



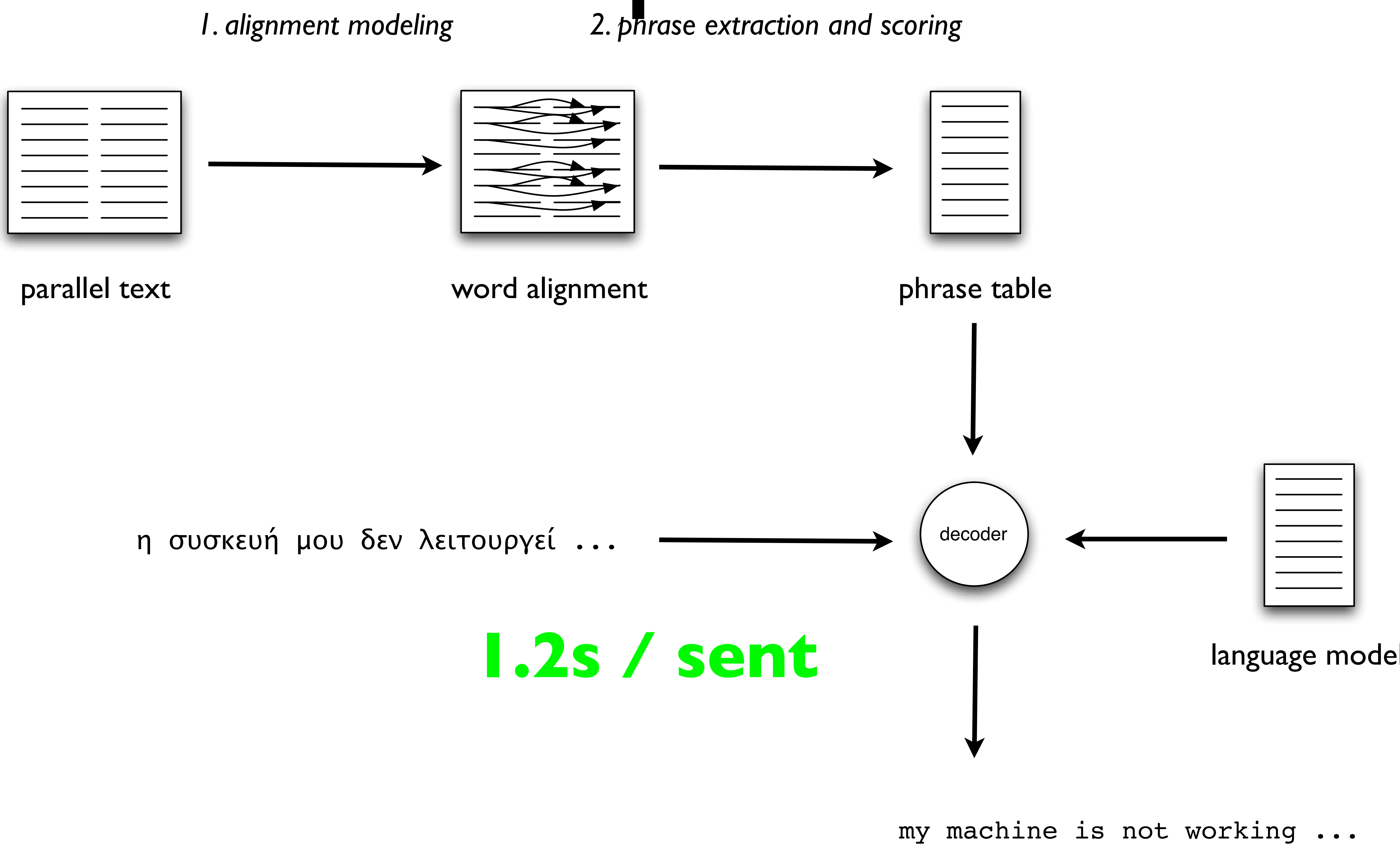
The Phrase-Based SMT Pipeline



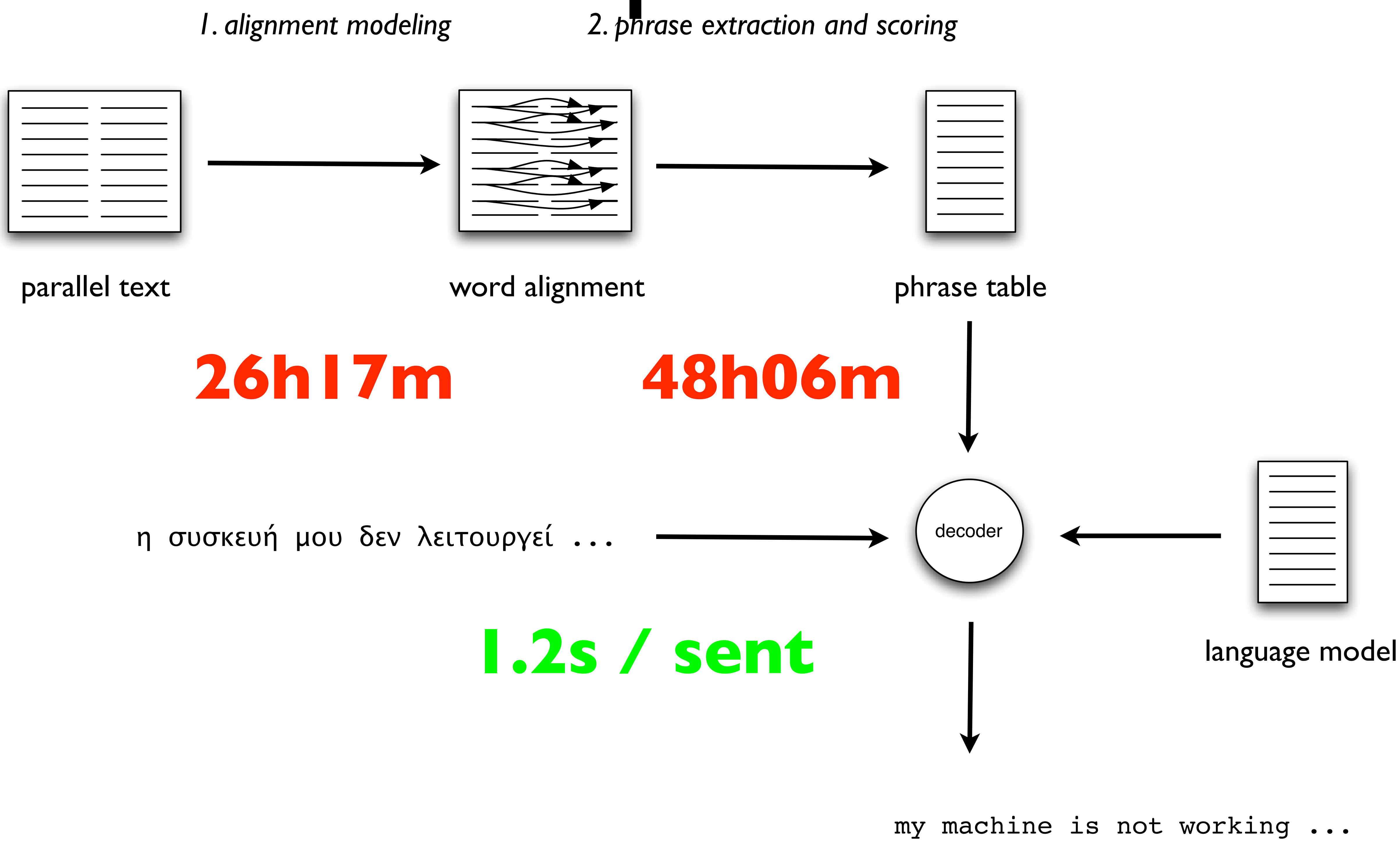
The Phrase-Based SMT Pipeline



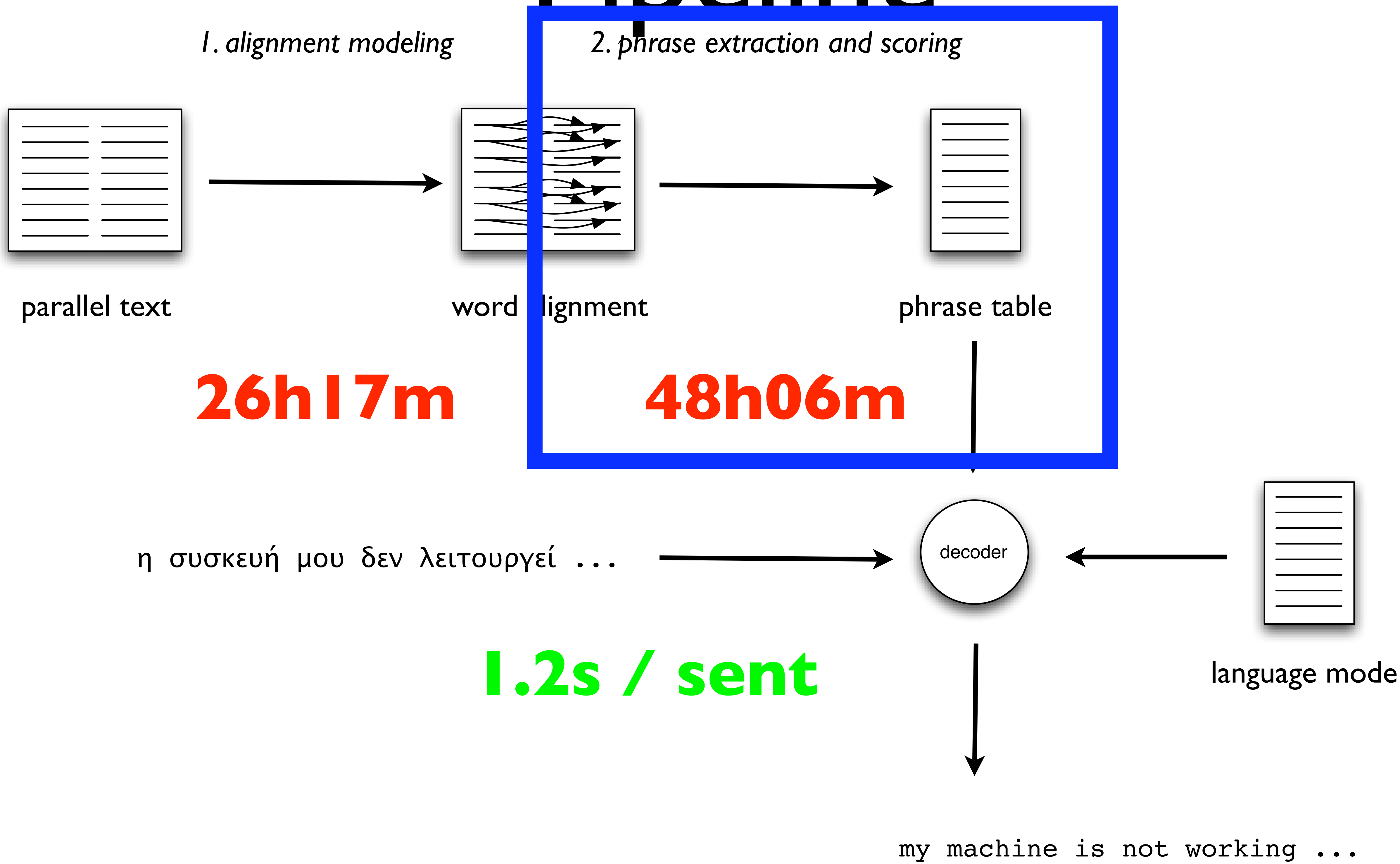
The Phrase-Based SMT Pipeline



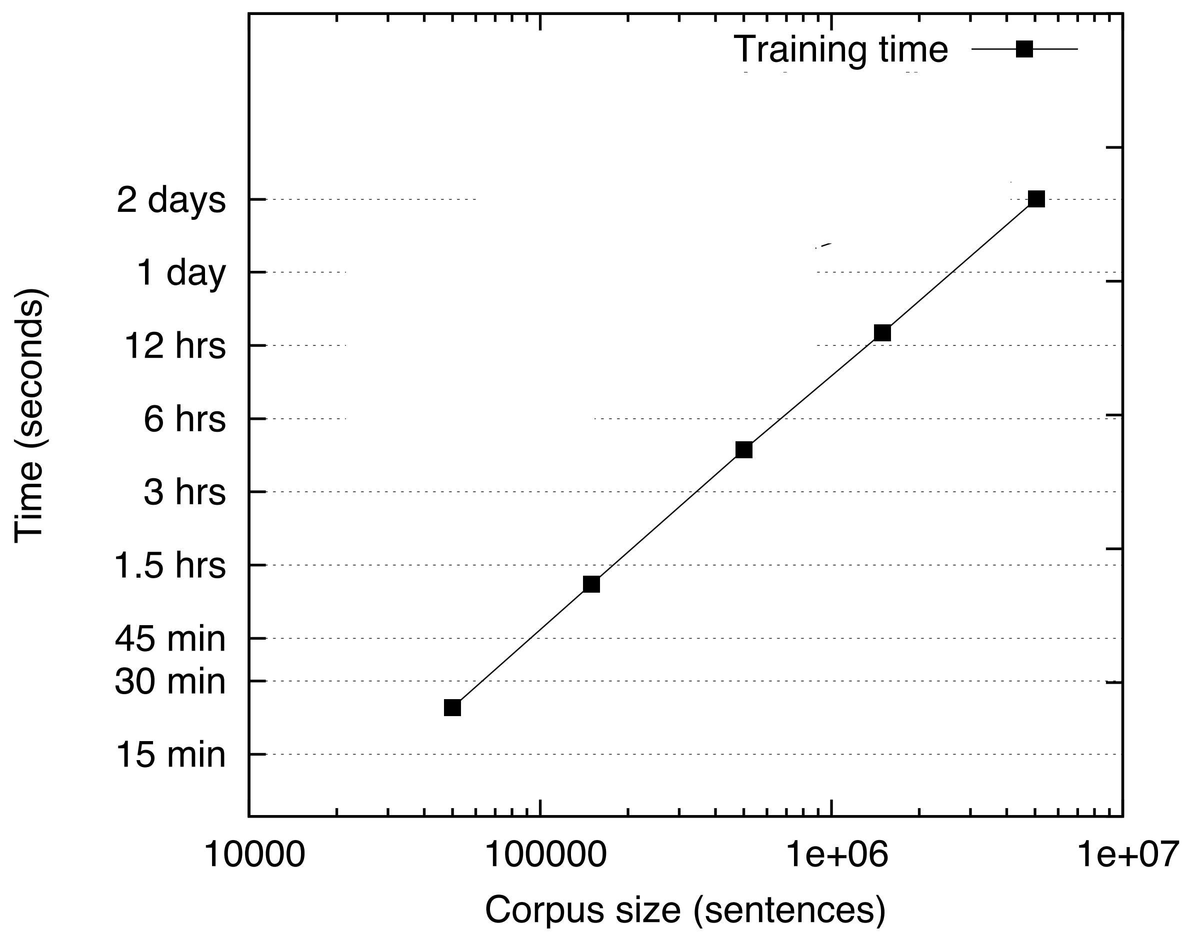
The Phrase-Based SMT Pipeline



The Phrase-Based SMT Pipeline



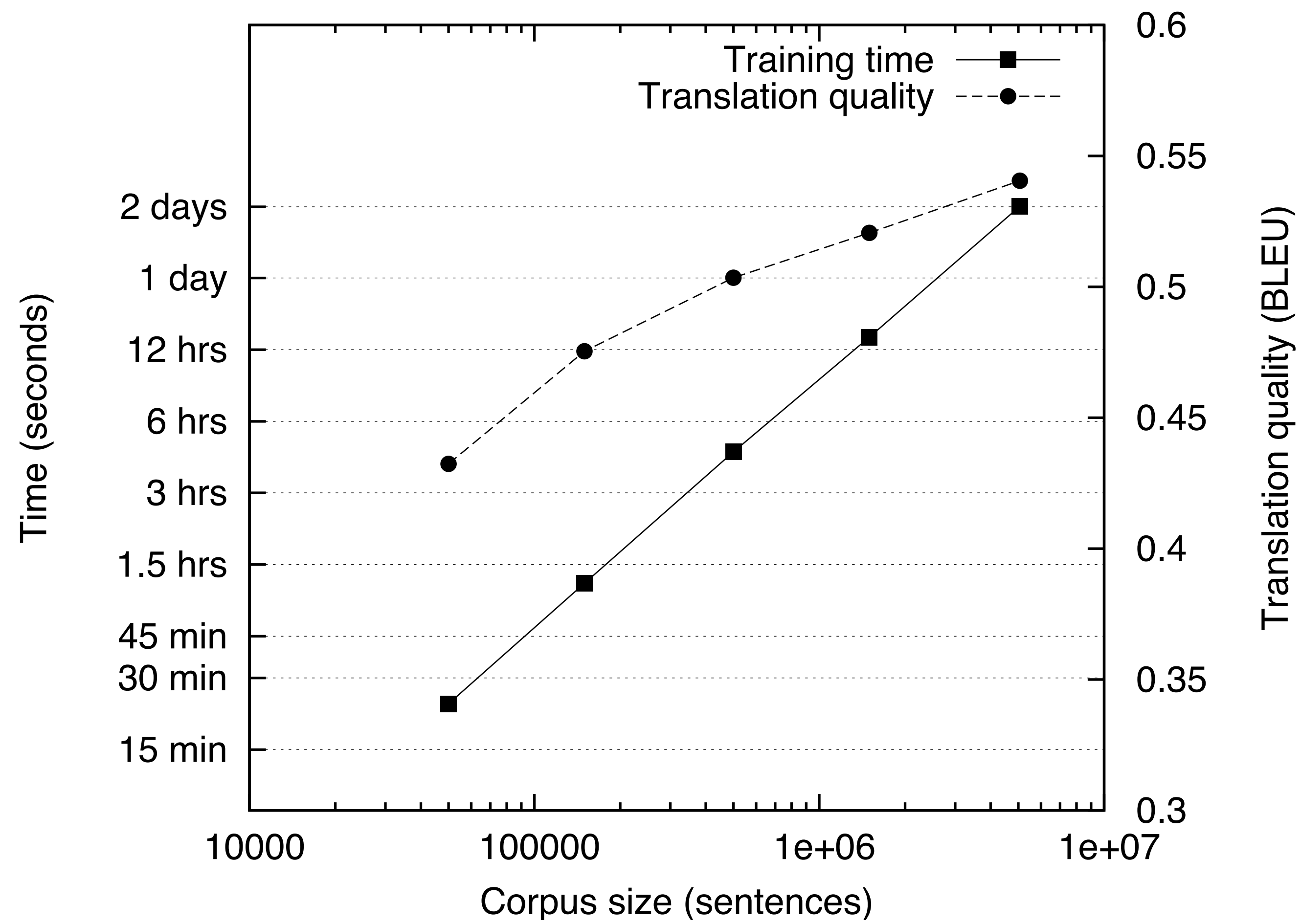
Is less data the answer?



Timing experiments conducted on Arabic-English training corpora publicly available from the LDC.
Test set is the NIST MT03 evaluation set.

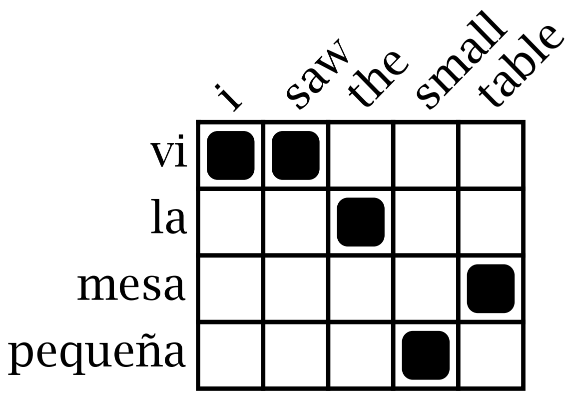
Is less data the answer?

Probably not...



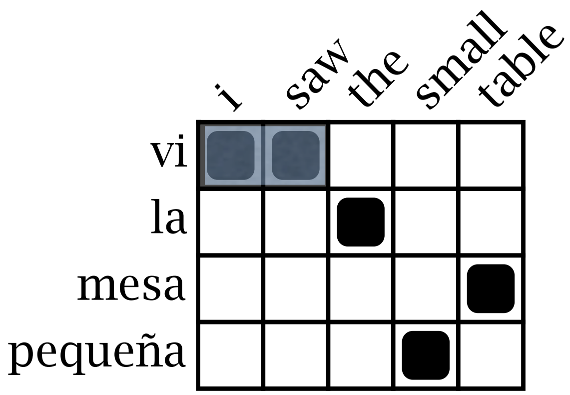
Timing experiments conducted on Arabic-English training corpora publicly available from the LDC.
Test set is the NIST MT03 evaluation set.

Building a phrase table



Step 1: extract phrases from a word aligned parallel text

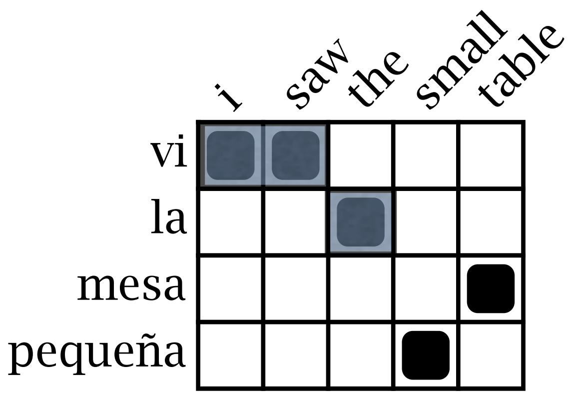
Building a phrase table



<i saw, vi>

Step 1: extract phrases from a word aligned parallel text

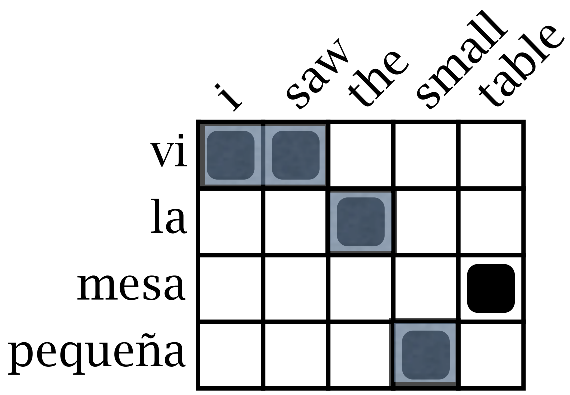
Building a phrase table



<i saw, vi> <the, la>

Step 1: extract phrases from a word aligned parallel text

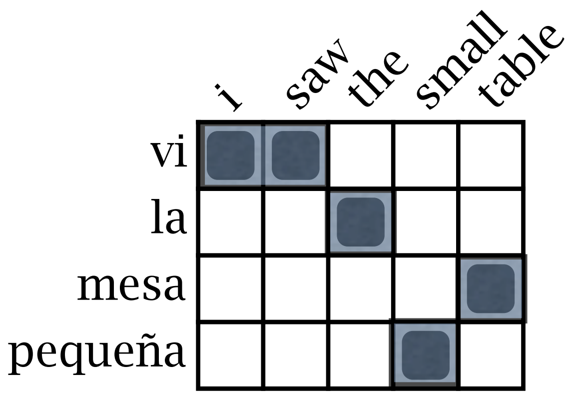
Building a phrase table



<i saw, vi> <the, la> <small, pequeña>

Step 1: extract phrases from a word aligned parallel text

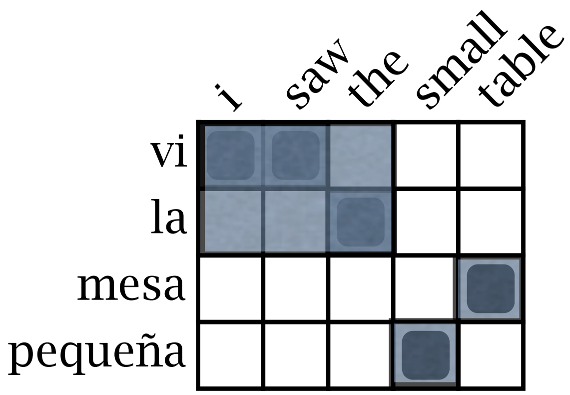
Building a phrase table



<i saw, vi> <the, la> <small, pequeña>
<table, mesa>

Step 1: extract phrases from a word aligned parallel text

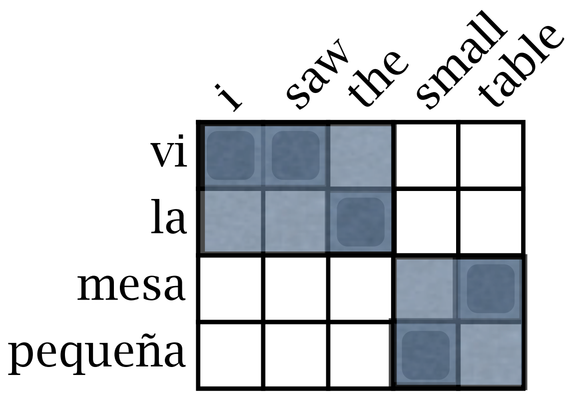
Building a phrase table



<i saw, vi> <the, la> <small, pequeña>
<table, mesa> <i saw the, vi la>

Step 1: extract phrases from a word aligned parallel text

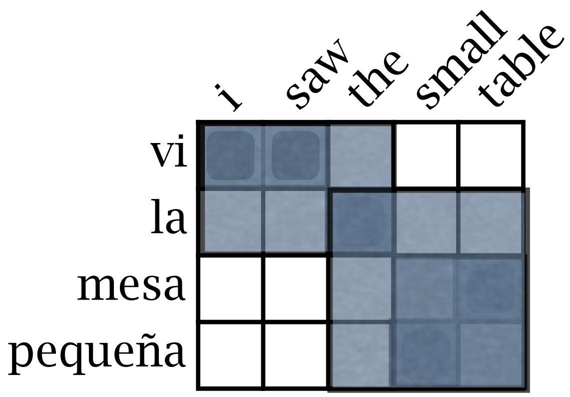
Building a phrase table



<i saw, vi> <the, la> <small, pequeña>
<table, mesa> <i saw the, vi la>
<small table, mesa pequeña>

Step 1: extract phrases from a word aligned parallel text

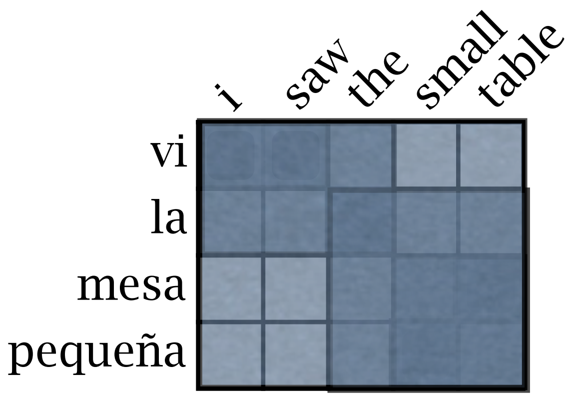
Building a phrase table



<i saw, vi> <the, la> <small, pequeña>
<table, mesa> <i saw the, vi la>
<small table, mesa pequeña>
<the small table, la mesa pequeña>

Step 1: extract phrases from a word aligned parallel text

Building a phrase table



<i saw, vi> <the, la> <small, pequeña>
<table, mesa> <i saw the, vi la>
<small table, mesa pequeña>
<the small table, la mesa pequeña>
<i saw the small table, vi la mesa pequeña>

Step 1: extract phrases from a word aligned parallel text

Building a phrase table

<i saw, vi> <the, la> <small, pequeña>
<table, mesa> <i saw the, vi la>
<small table, mesa pequeña>
<the small table, la mesa pequeña>
<i saw the small table, vi la mesa pequeña>
...

<i saw, vi>	15
<i saw the, vi la>	5
<small, pequeña>	72
<the, la>	5434
<the, el>	6218
<table, mesa>	2
...	

Step 2: compute joint counts

Building a phrase table

<i saw, vi>	15
<i saw the, vi la>	5
<small, pequeña>	72
<the, la>	5434
<the, el>	6218
<table, mesa>	2
...	

<i saw, *>	15
<i saw the, *>	5
<small, *>	72
<the, *>	11652
<table, *>	2
...	

Step 2: compute marginal counts

Building a phrase table

<i saw, vi>	15
<i saw the, vi la>	5
<small, pequeña>	72
<the, la>	5434
<the, el>	5434
<table, mesa>	2
...	

+

<i saw, *>	15
<i saw the, *>	5
<small, *>	72
<the, *>	11652
<table, *>	2
...	

→

<i saw, vi>	1.0
<i saw the, vi la>	1.0
<small, pequeña>	1.0
<the, la>	0.47
<the, el>	0.53
<table, mesa>	1.0
...	

Step 2: join and normalize

MapReduce

- Phrase translation probabilities are just relative frequencies f(e|f)
- Relative frequencies can be estimated using MapReduce.
- Why MapReduce?
  - Easy parallelization across many machines
  - No expensive infrastructure required

Computing Relative Frequencies

$$P_{MLE}(B|A) = \frac{c(A,B)}{c(A)} = \frac{c(A,B)}{\sum_{B'} c(A,B')}$$

Method 1

Map_1	$\langle A, B\rangle \rightarrow \langle\langle A, B\rangle, 1\rangle$
Reduce_1	$\langle\langle A, B\rangle, c(A, B)\rangle$
Map_2	$\langle\langle A, B\rangle, c(A, B)\rangle \rightarrow \langle\langle A,^* \rangle, c(A, B)\rangle$
Reduce_2	$\langle\langle A,^* \rangle, c(A)\rangle$
Map_3	$\langle\langle A, B\rangle, c(A, B)\rangle \rightarrow \langle A, \langle B, c(A, B)\rangle\rangle$
Reduce_3	$\langle A, \langle B, \frac{c(A,B)}{c(A)}\rangle\rangle$

Method 2

Map_1	$\langle A, B\rangle \rightarrow \langle\langle A, B\rangle, 1\rangle; \langle\langle A,^* \rangle, 1\rangle$
Reduce_1	$\langle\langle A, B\rangle, \frac{c(A,B)}{c(A)}\rangle$

Method 3

Map_1	$\langle A, B_i\rangle \rightarrow \langle A, \langle B_i : 1\rangle\rangle$
Reduce_1	$\langle A, \langle B_1 : \frac{c(A,B_1)}{c(A)}\rangle, \langle B_2 : \frac{c(A,B_2)}{c(A)}\rangle \cdots\rangle$

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a					
b					

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	1				
b					

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	1	1			
b					

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	1	1			
b	1				

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	I	II			
b	I				

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	I	II	I		
b	I				

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	I	II	I		
b	I			I	

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	I	III	I		
b	I			I	

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	I	III	I		
b	II			I	

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	I	IIII	I		
b	II			I	

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	I	IIII	I		
b	III			I	

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	II	IIII	I		
b	III			I	

Mapper counts joint events.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	2	IIII	I		
b	III			I	

Reducer computes counts.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	2	4	I		
b	I I I			I	

Reducer computes counts.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	2	4	1		
b	III			I	

Reducer computes counts.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	2	4	1		
b	3			1	

Reducer computes counts.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	2	4	1		
b	3			1	

Reducer computes counts.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	2	4	1		
b	3			1	

A second reducer computes marginals.

Method 1

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	2	4	1		**7**
b	3			1	

A second reducer computes marginals.

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	2	4	1		**7**
b	3			1	**4**

A second reducer computes marginals.

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	2	4	1		**7**
b	3			1	**4**

A second reducer computes marginals.

Alternative: mappers emits marginal counts too for each event, a single reducer computes

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	4	1		**7**
b	3			1	**4**

Reducer can sort marginal before all other sums and normalize, one cell at a time.

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	0.6	1		**7**
b	3			1	**4**

Reducer can sort marginal before all other sums and normalize, one cell at a time.

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	0.6	0.1		**7**
b	3			1	**4**

Reducer can sort marginal before all other sums and normalize, one cell at a time.

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	0.6	0.1		**7**
b	0.8			1	**4**

Reducer can sort marginal before all other sums and normalize, one cell at a time.

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	0.6	0.1		**7**
b	0.8			0.2	**4**

Reducer can sort marginal before all other sums and normalize, one cell at a time.

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	0.6	0.1		**7**
b	0.8			0.2	**4**

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	0.6	0.1		7
b	0.8			0.2	4

The join is a very large, expensive sort. Can we do better?

Methods 1/2

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	0.6	0.1		7
b	0.8			0.2	4

The join is a very large, expensive sort. Can we do better?

Yes - if the CPDs we are estimating have few parameters...

Method 3

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	II	IIII	I		
b	III			I	

Method 3

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	I I	IIII	I		
b	I I I			I	

If memory allows, each reducer job counts, marginalizes, **and** normalizes.

Method 3

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	0.6	0.1		7
b	I I I			I	

If memory allows, one reducer counts, marginalizes, **and** normalizes.

Method 3

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

	1	2	3	4	Σ
a	0.3	0.6	0.1		7
b	0.8			0.2	4

If memory allows, one reducer counts, marginalizes, **and** normalizes.

Method 3

Corpus: <a,1> <a,2> <b,1> <a,2> <a,3> <b,4> <a,2> <b,1> <a,2> <b,1> <a,1>

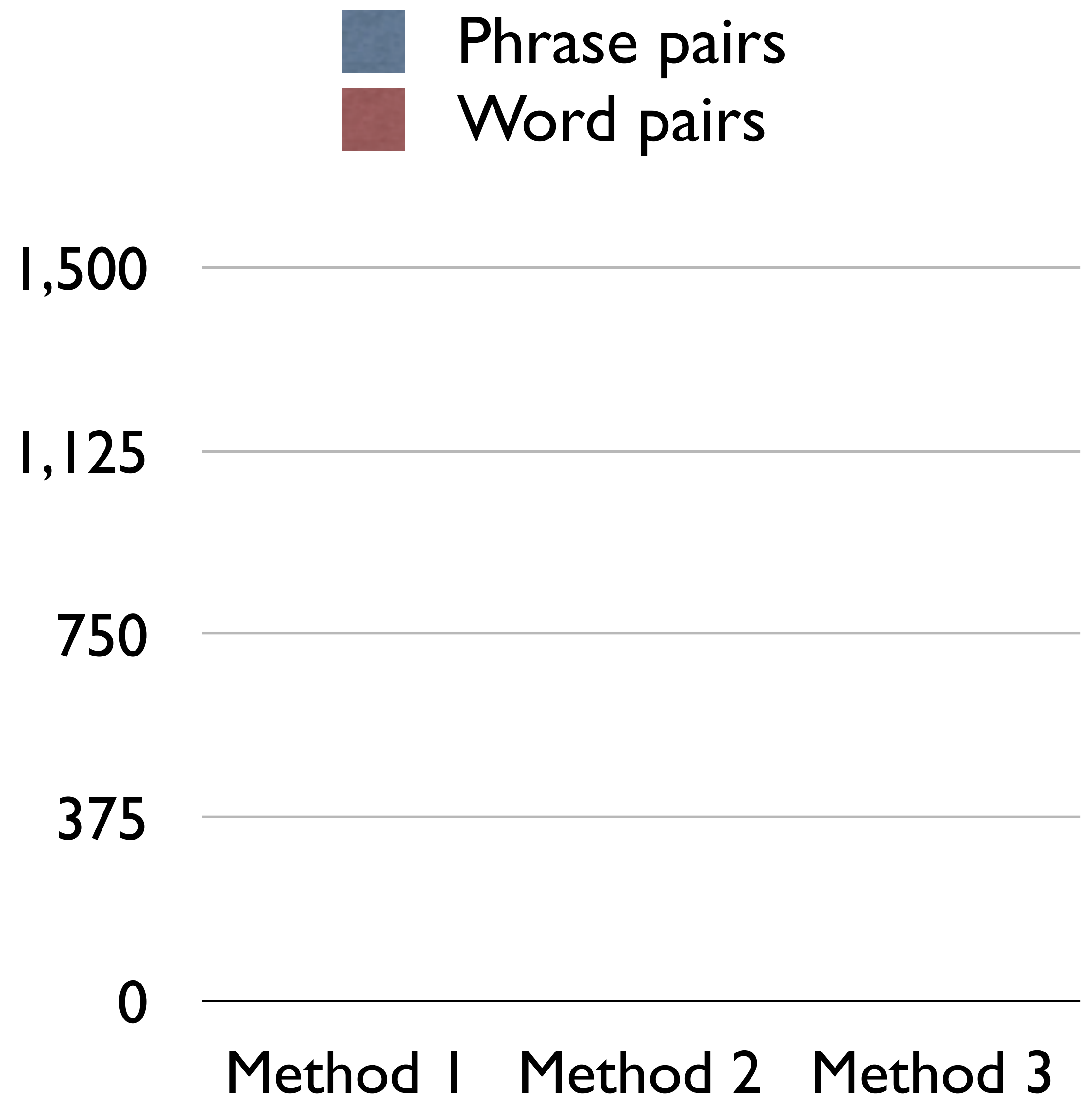
	1	2	3	4	Σ
a	0.3	0.6	0.1		7
b	0.8			0.2	4

Rather than sorting keys from $V_1 x V_2$,
we just sort over item into bins from V_2

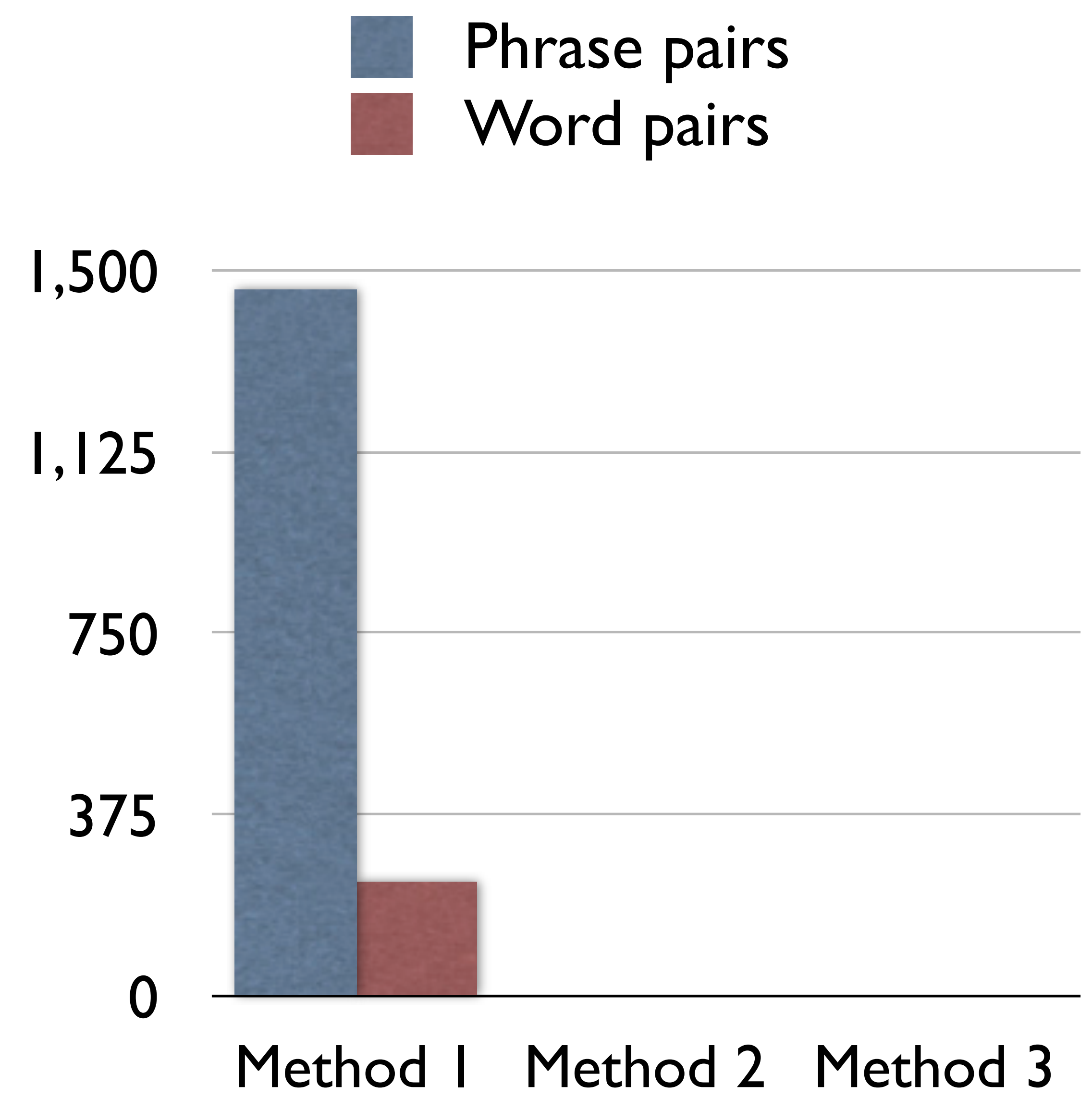
Computing Relative Frequency

- Phrase pairs
- Word pairs

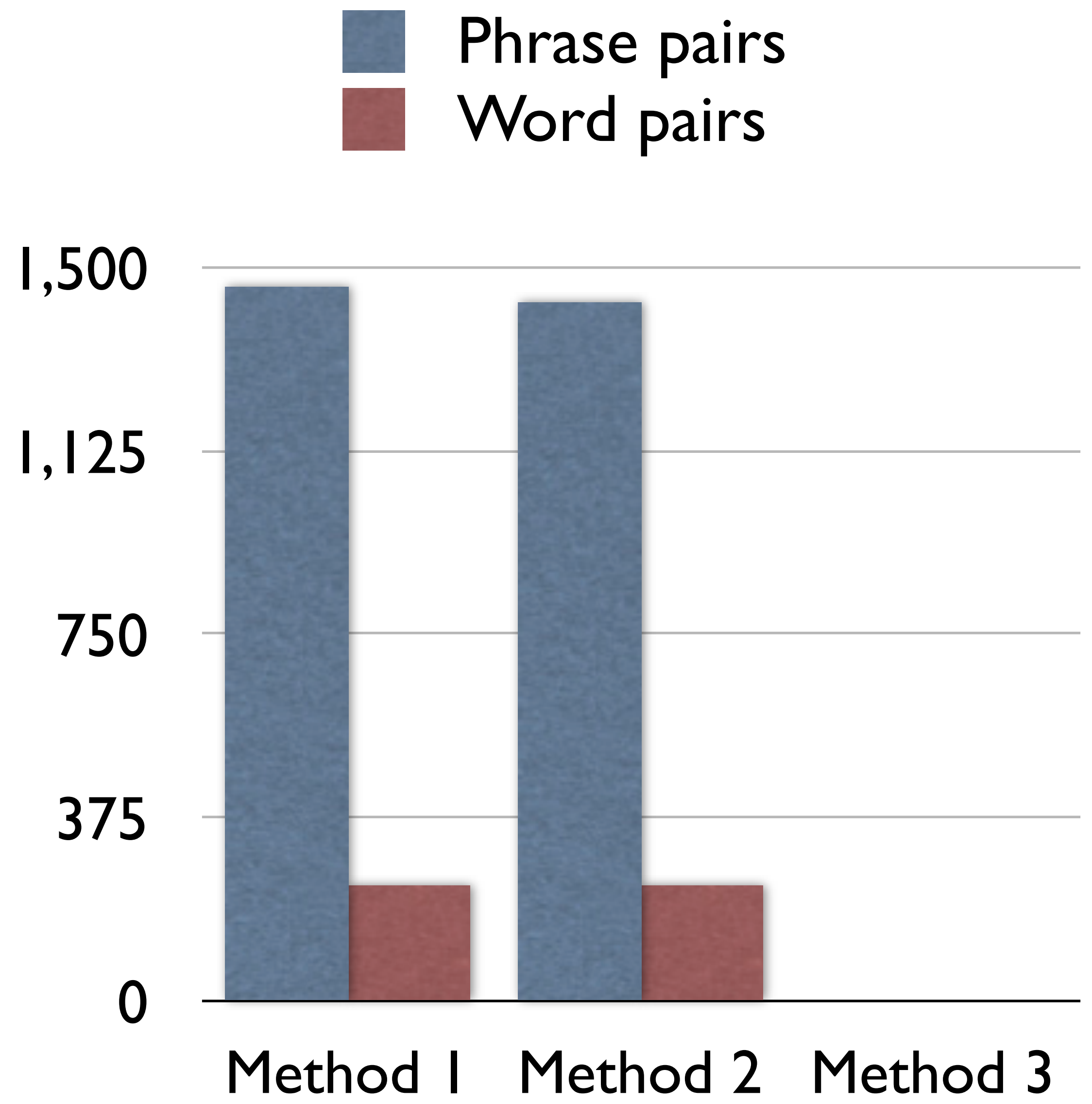
Computing Relative Frequency



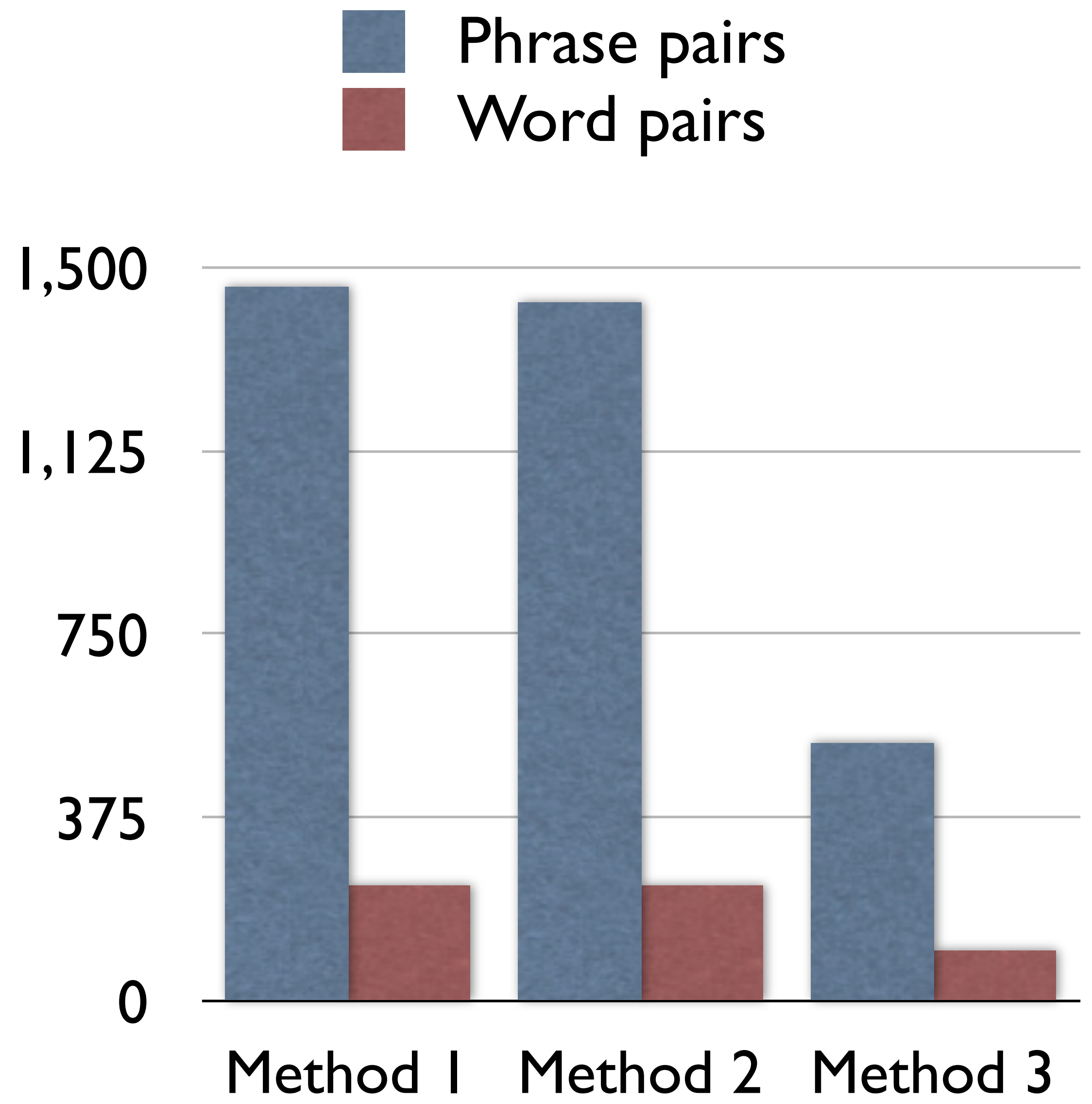
Computing Relative Frequency



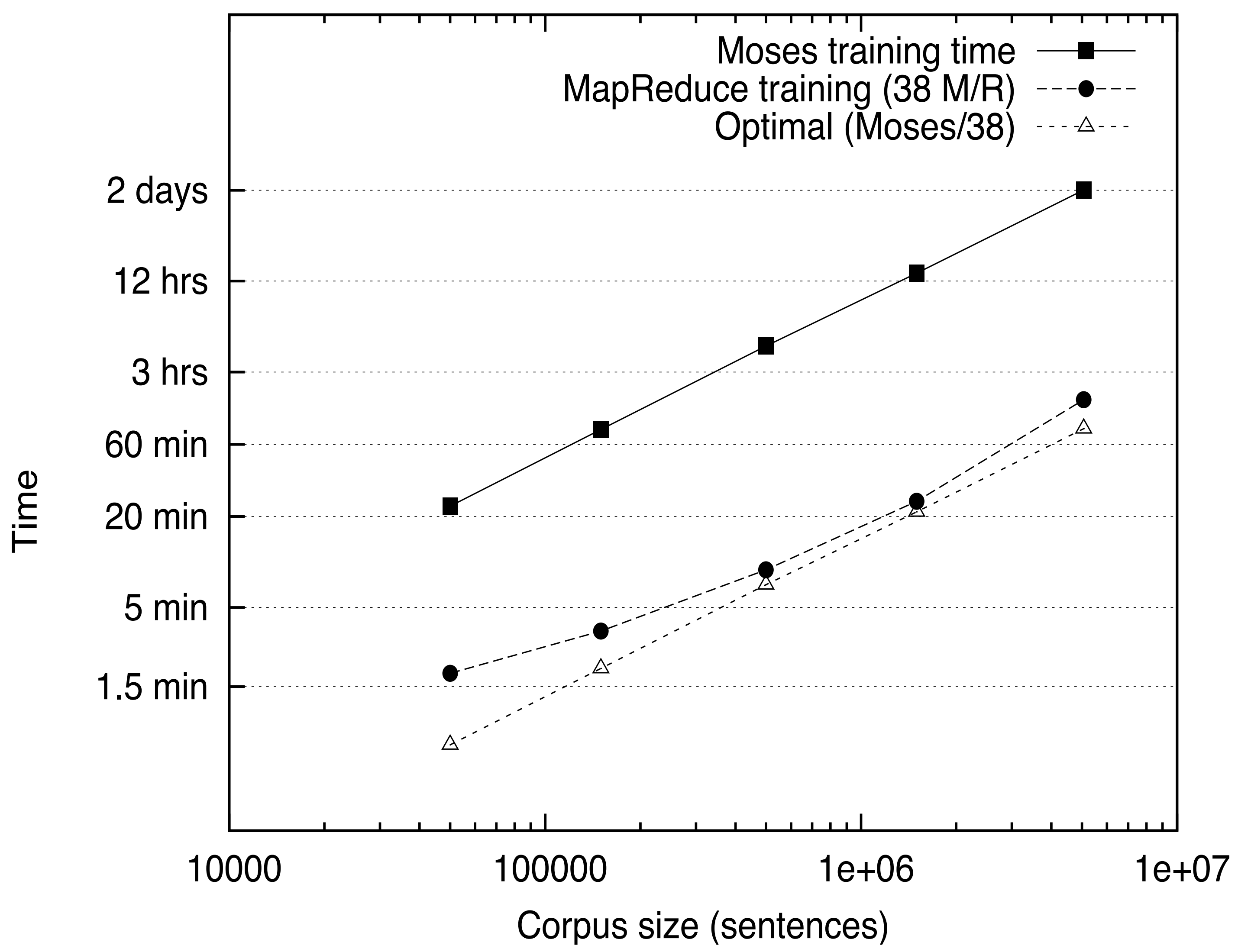
Computing Relative Frequency



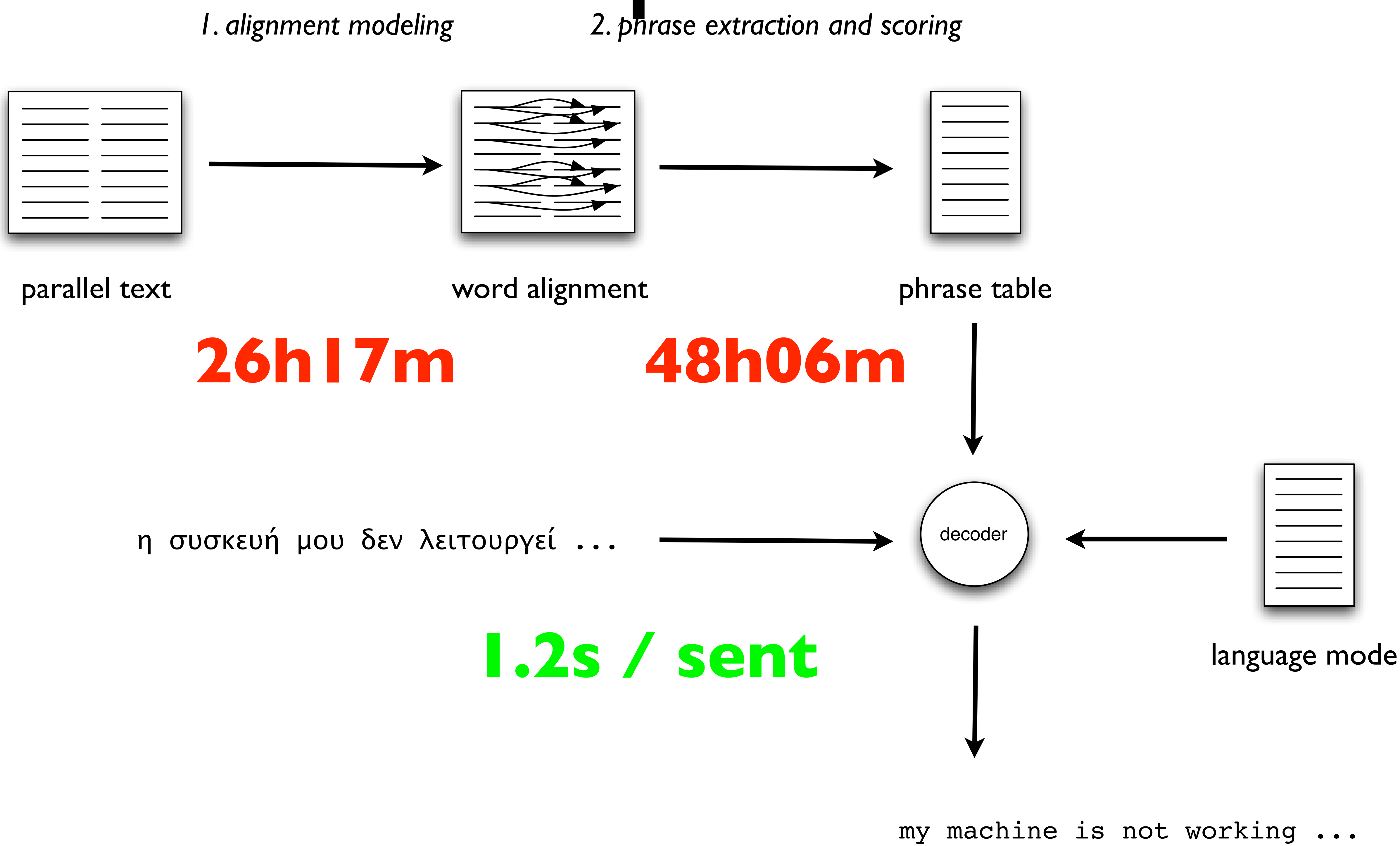
Computing Relative Frequency



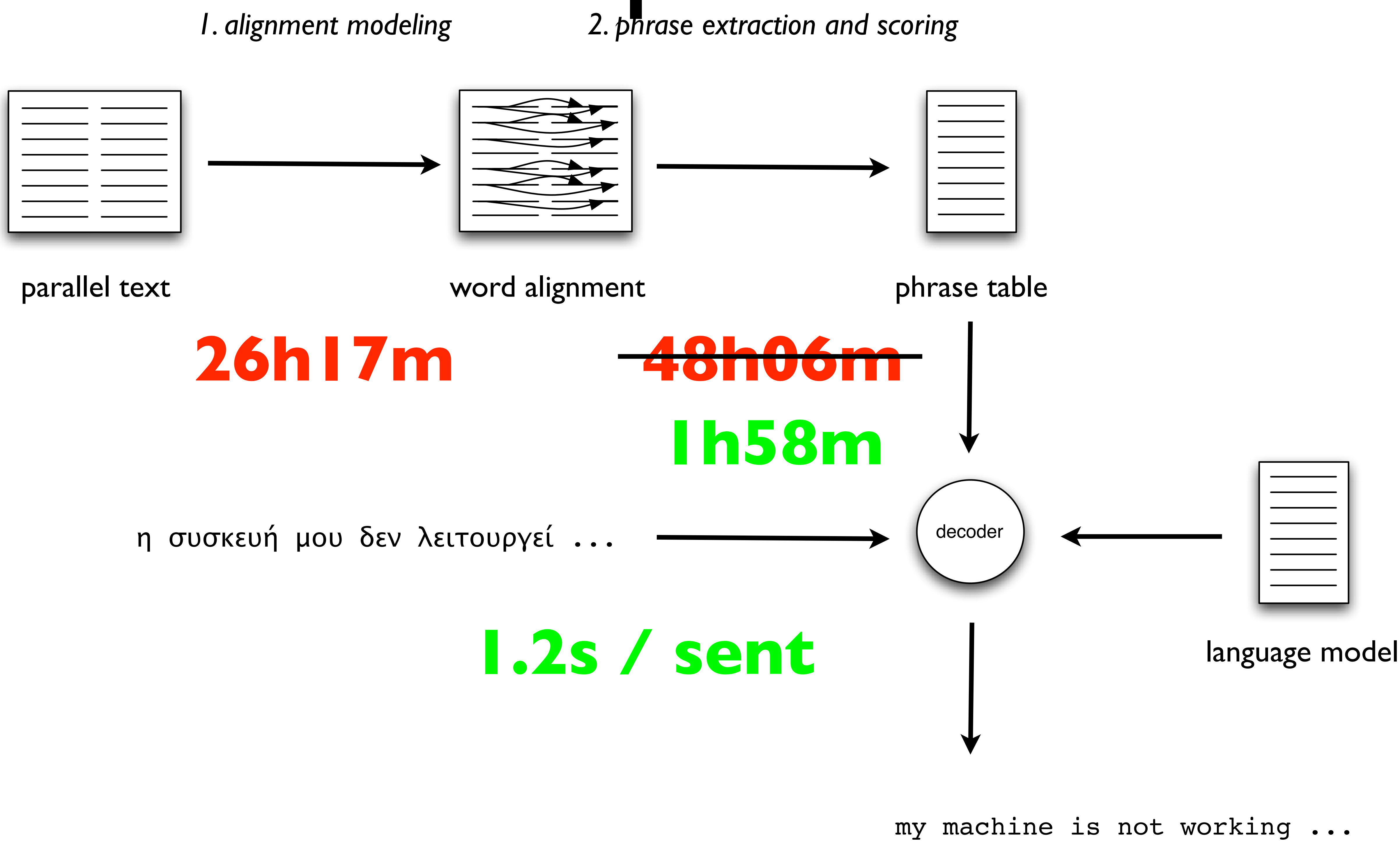
MapReduce Phrase-table building



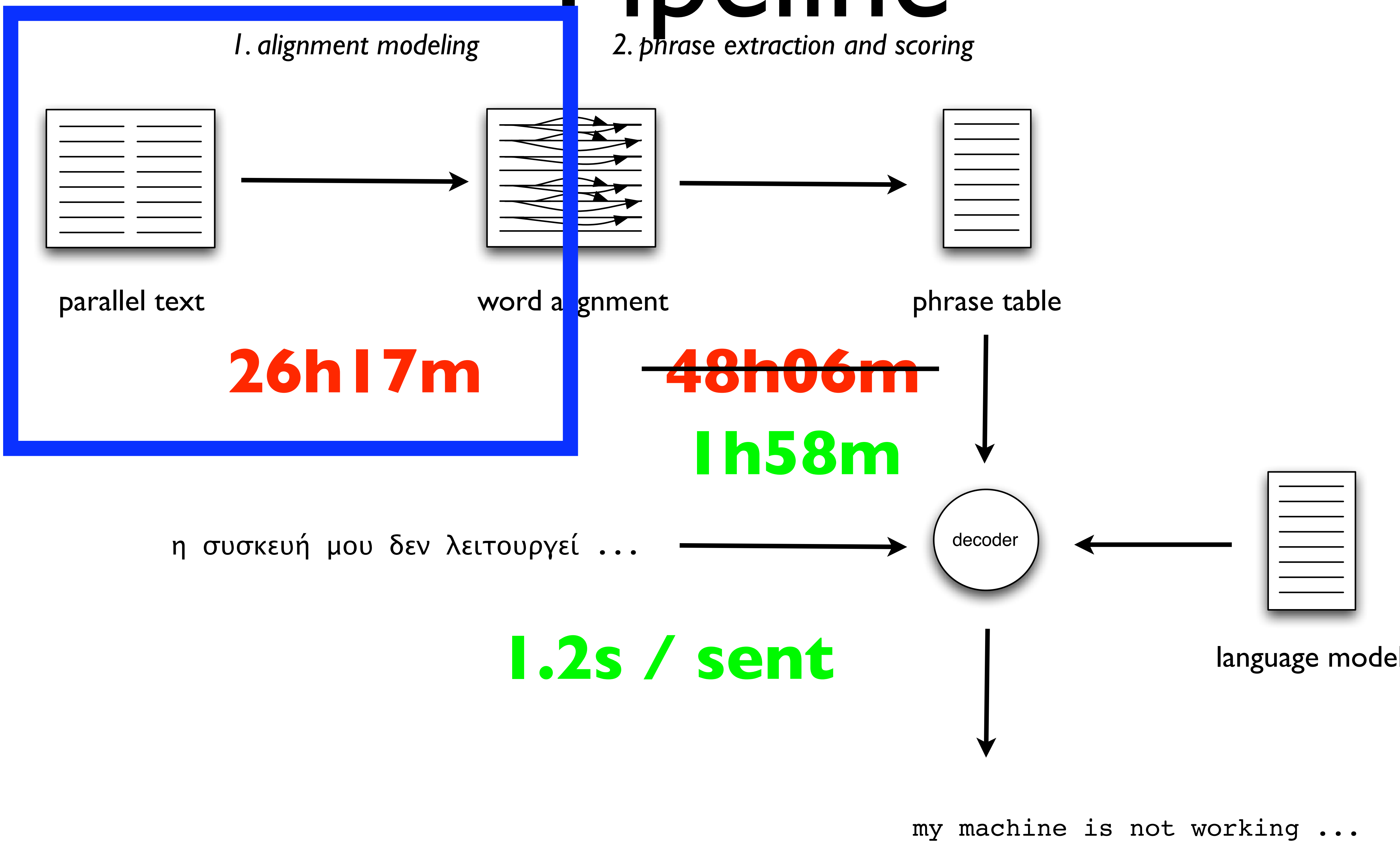
The Phrase-Based SMT Pipeline



The Phrase-Based SMT Pipeline

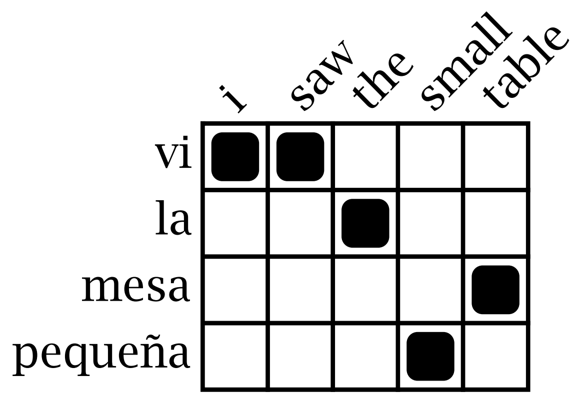


The Phrase-Based SMT Pipeline



Word alignment

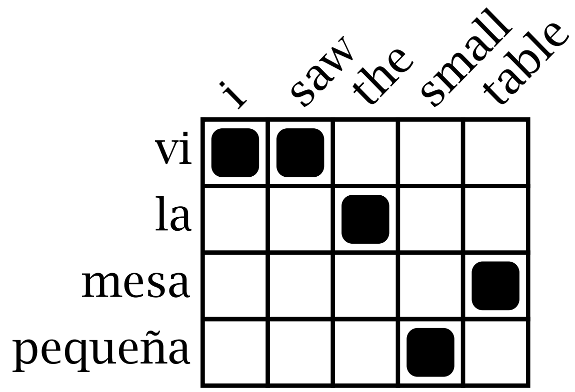
To build our models, we need this:



But, the alignment points aren't given...

Word alignment

To build our models, we need this:



But, the alignment points aren't given...

EM to the rescue!

Generative alignment models: a brief intro

$$P(f_1^m|e_1^l) \;=\; \sum_{a_1^m} P(f_1^m, a_1^m|e_1^l)$$

Generative alignment models: a brief intro

$$
\begin{aligned}
P(f_1^m|e_1^l) &= \sum_{a_1^m} P(f_1^m, a_1^m|e_1^l) \\
&= \sum_{a_1^m} P(a_1^m|e_1^l, f_1^m) \prod_{j=1}^{m} P(f_j|e_{a_j})
\end{aligned}
$$

Assume a *lexical* model!

Generative alignment models: a brief intro

$$
\begin{aligned}
P(f_1^m|e_1^l) &= \sum_{a_1^m} P(f_1^m, a_1^m|e_1^l) \\
&= \sum_{a_1^m} P(a_1^m|e_1^l, f_1^m) \prod_{j=1}^{m} P(f_j|e_{a_j})
\end{aligned}
$$

Still too complicated, so we make one of two further assumptions:

(IBM Model 1) $P(a_1^m|e_1^l, f_1^m) =$ *uniform*

(HMM) $P(a_1^m|e_1^l, f_1^m) = \prod_{j=1}^{m} P(a_j|a_{j-1})$

Generative alignment models: a brief intro

$$
\begin{aligned}
P(f_1^m|e_1^l) &= \sum_{a_1^m} P(f_1^m, a_1^m|e_1^l) \\
&= \sum_{a_1^m} P(a_1^m|e_1^l, f_1^m) \prod_{j=1}^{m} P(f_j|e_{a_j})
\end{aligned}
$$

Still too complicated, so we make one of two further assumptions:

(IBM Model 1) $P(a_1^m|e_1^l, f_1^m) =$ *uniform*

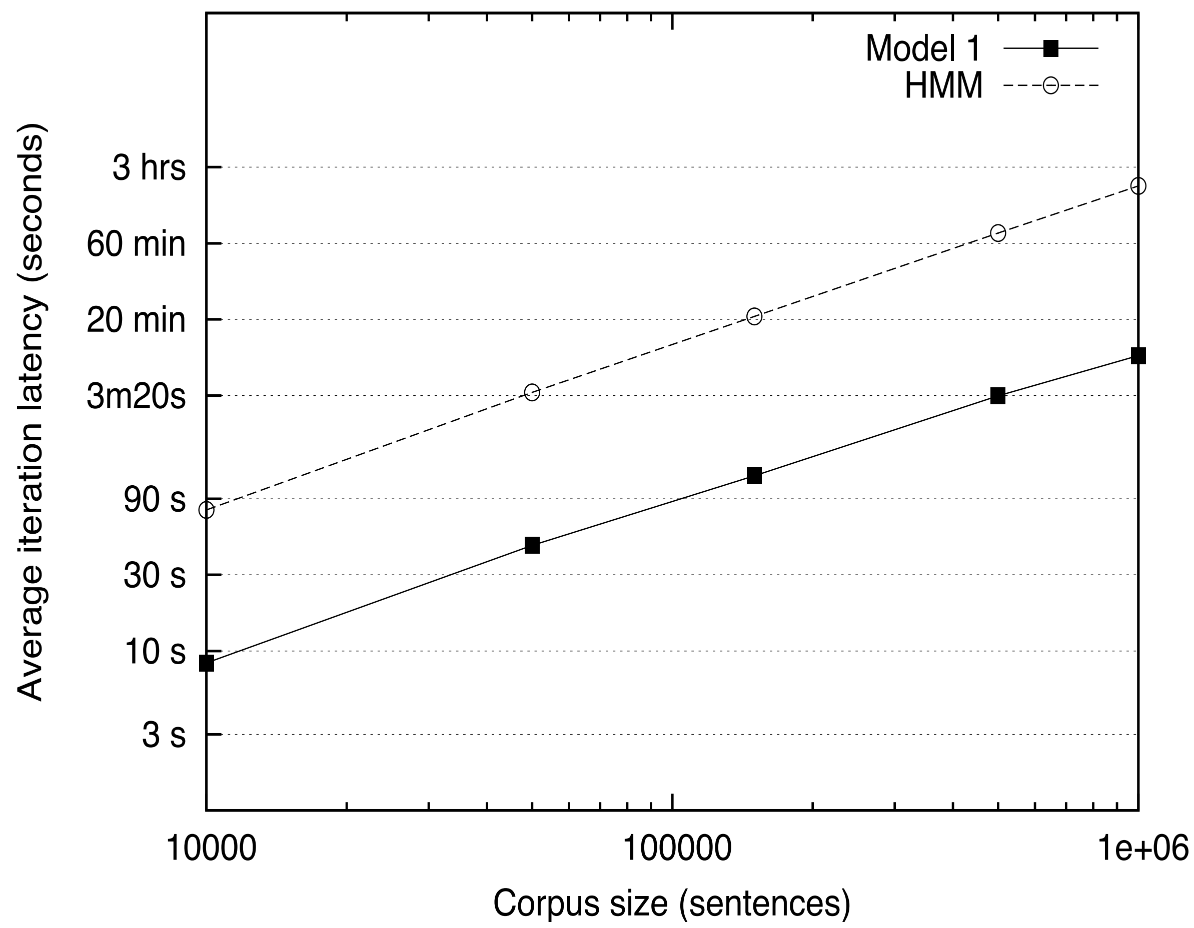
(HMM) $P(a_1^m|e_1^l, f_1^m) = \prod_{j=1}^{m} P(a_j|a_{j-1})$

Once we have such a model, computing the Viterbi alignment is simply:

$$
\hat{a}_1^m = \arg\max_{a_1^m} P(a_1^m|e_1^l, f_1^m) \prod_{j=1}^{m} P(f_j|e_{a_j})
$$

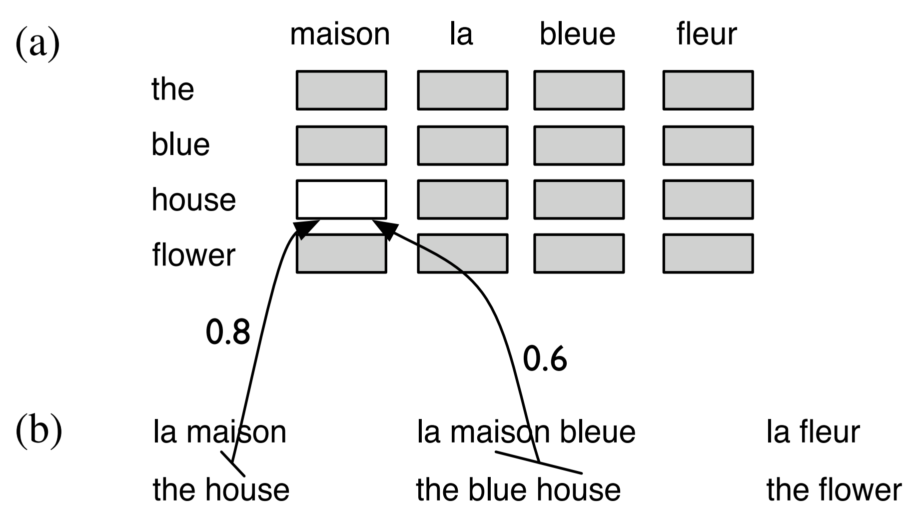
Word alignment

A familiar problem with the conventional tools:



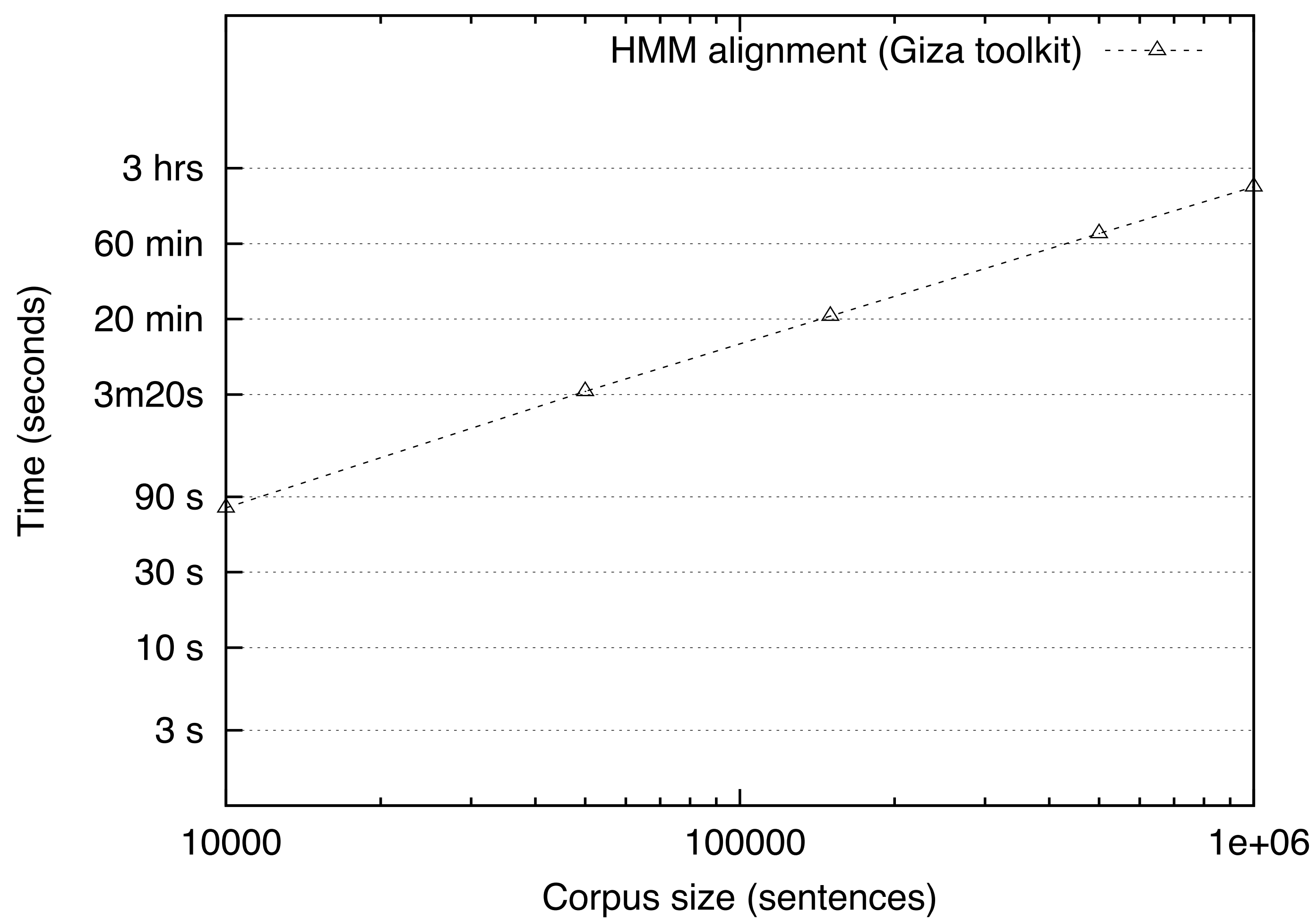
EM for MapReduce

- EM relies on MLE, but counts are fractional rather than whole

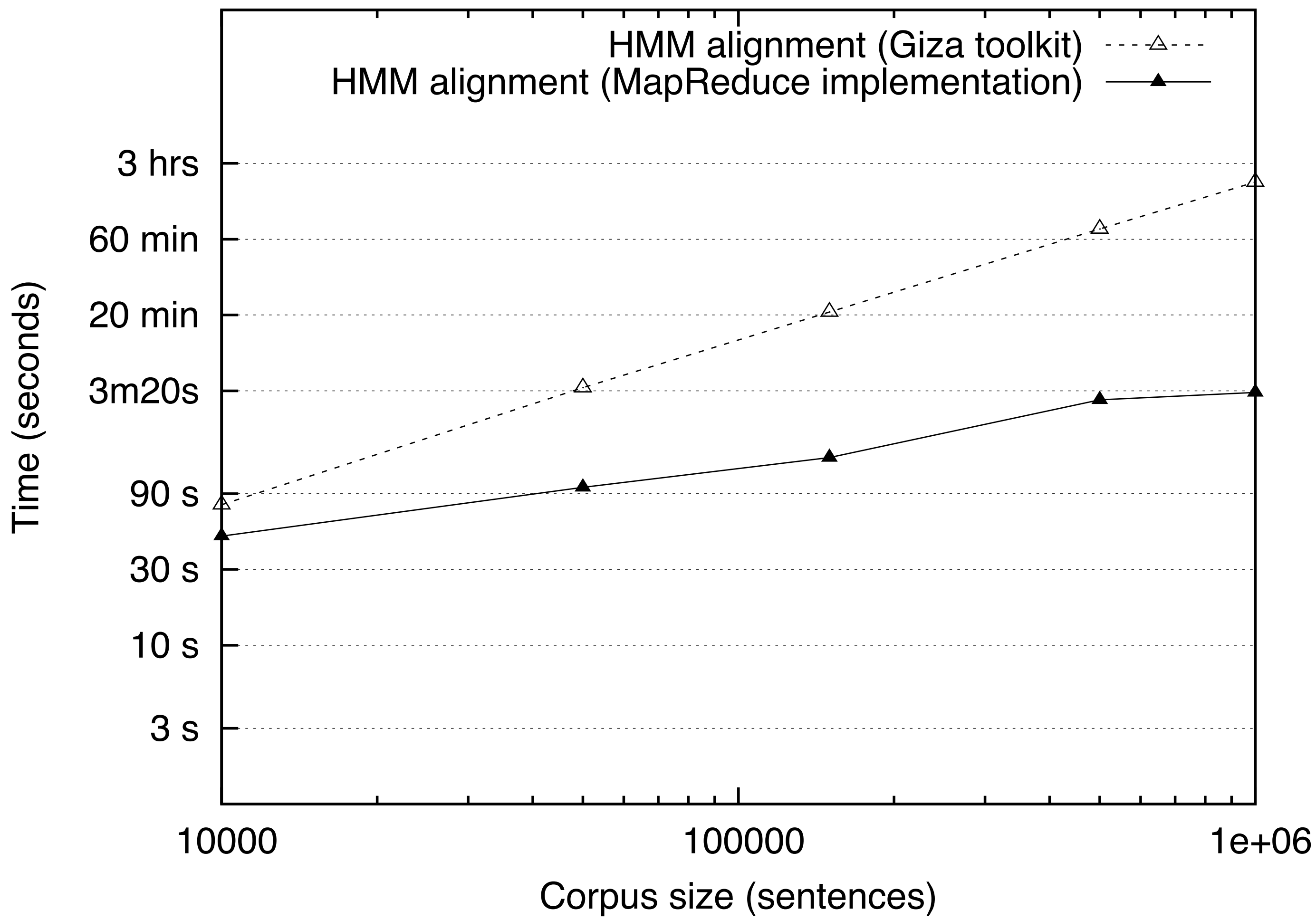


- Same MR strategies are available (and same optimizations!)

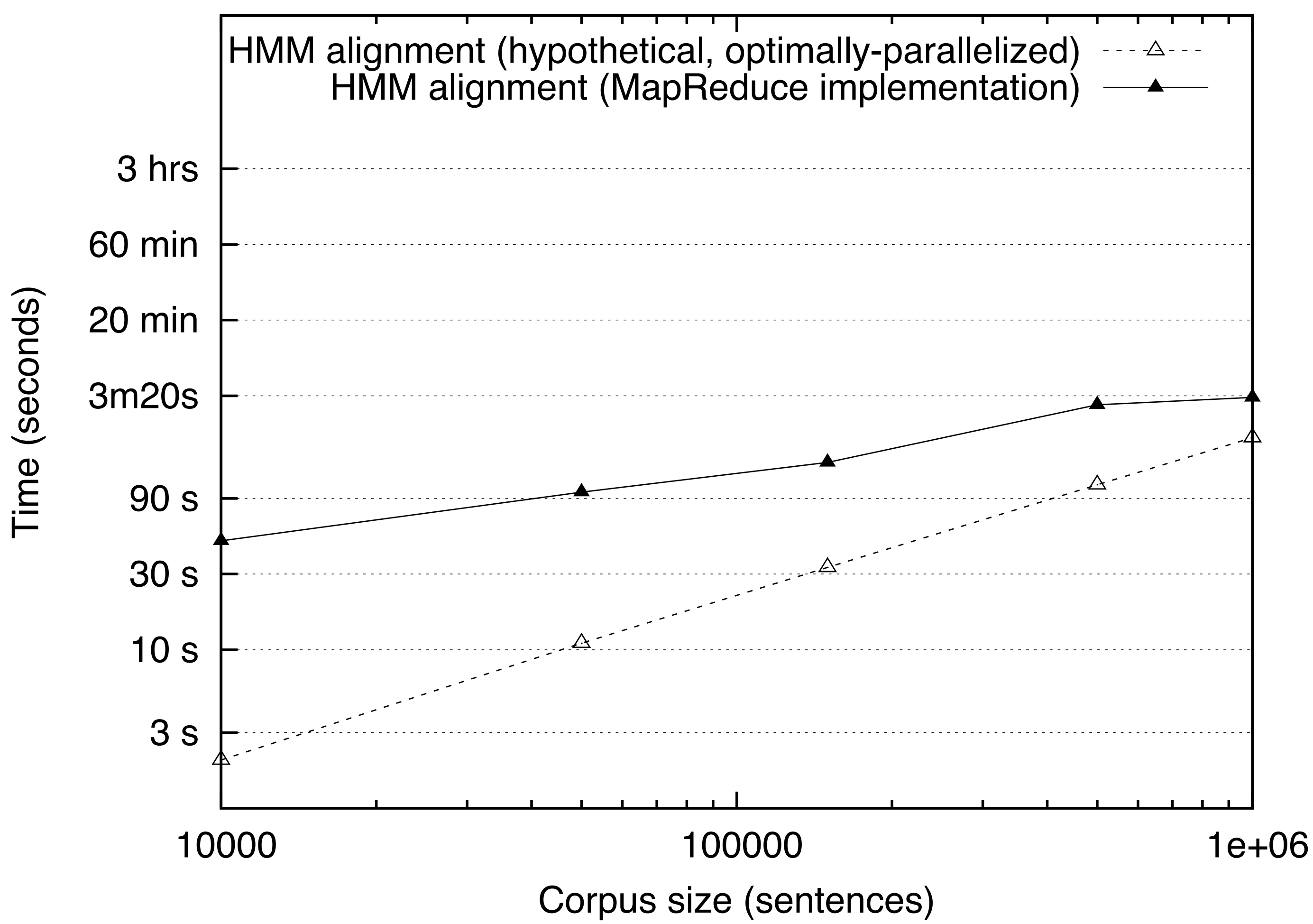
MapReduce word alignment



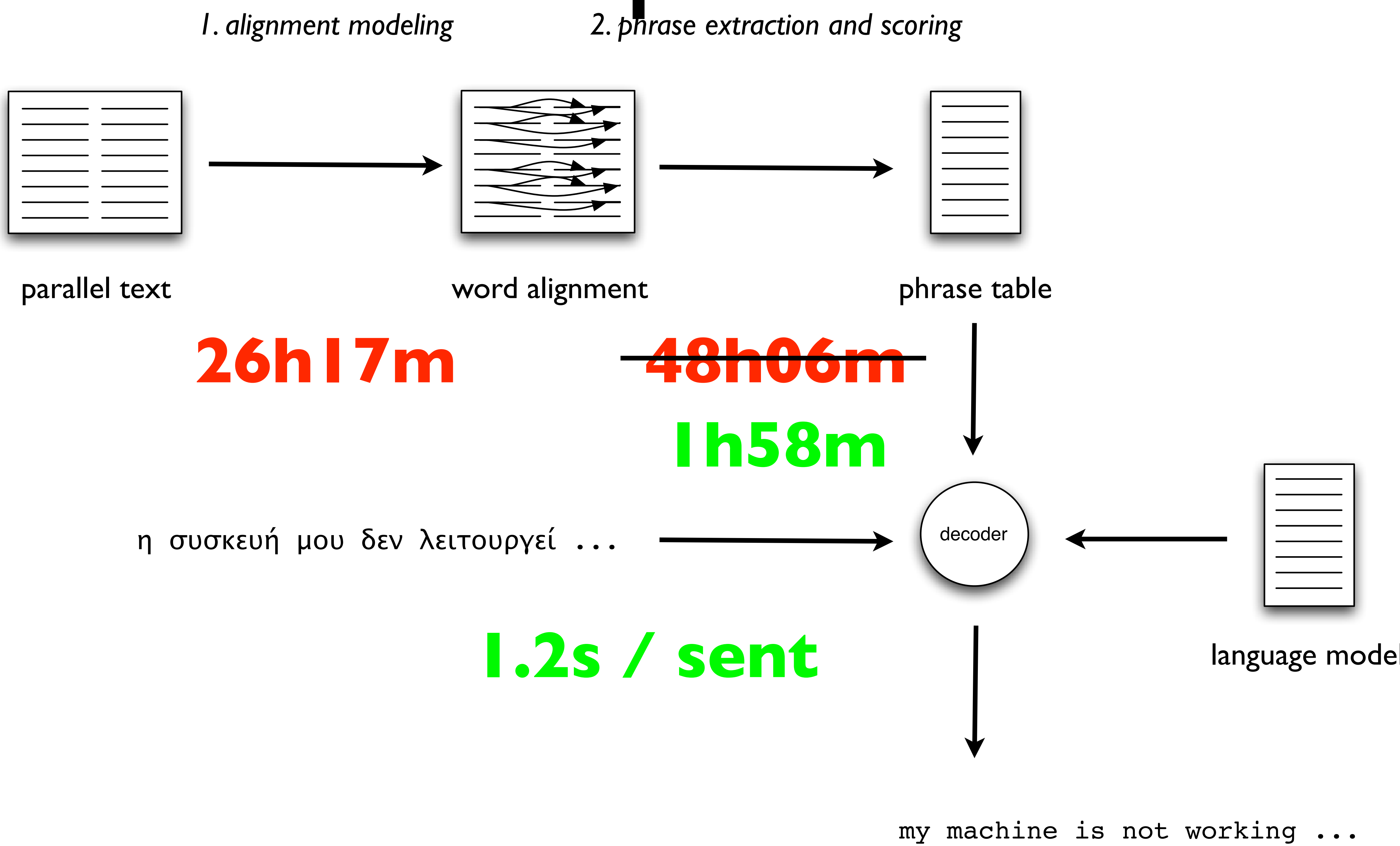
MapReduce word alignment



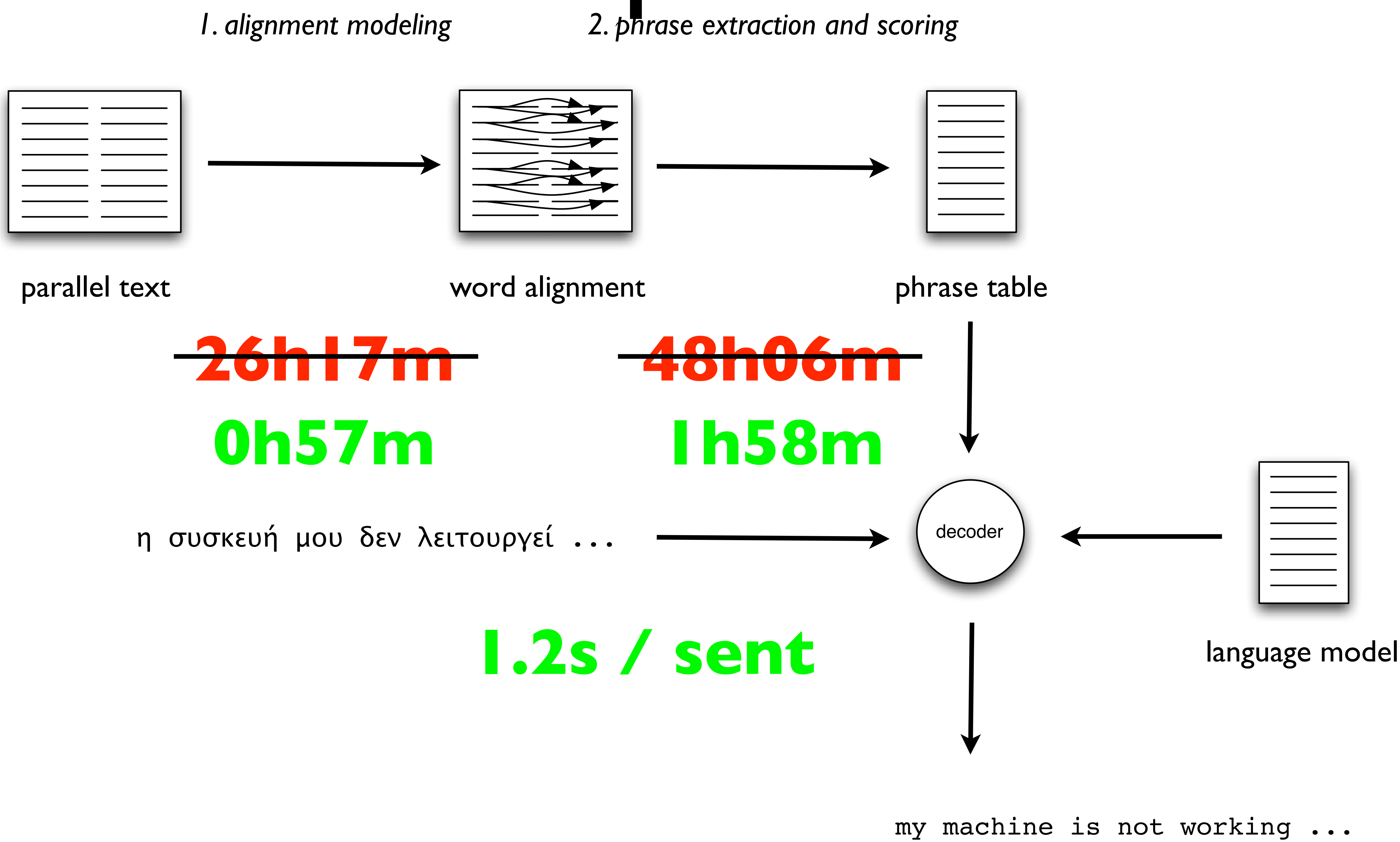
MapReduce word alignment



The Phrase-Based SMT Pipeline



The Phrase-Based SMT Pipeline



Future Work

- Word alignment
  - How to access/distribute the prior model?
  - Is EM really a good choice?
    - Good results in a Bayesian framework
    - Ongoing work using a CRF-based model
    - Are exact solutions really necessary?
  - How can we improve data locality?

Thank You!

Jimmy Lin	Chris Manning
Eugene Hung	CBCB@UMD
Philip Resnik	IBM
Miles Osborne	Google

*This research was supported by the GALE program of the Defense Advanced Projects Agency, Contract No. HR0011-06-2-0001