Algorithms for NLP

Machine Translation III

Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley

Announcements

- P4 has been released
- Due Dec 4 at 11:59pm

Phrase-Based Translation Overview

Input: lo haré | rápidamente | .

Translations: I'll do it | quickly | .

quickly | I'll do it | .

The decoder...

tries different segmentations,

translates phrase by phrase,

and considers reorderings.

Objective:

$$\arg\max_{\mathbf{e}} \left[P(\mathbf{f}|\mathbf{e}) \cdot P(\mathbf{e}) \right]$$

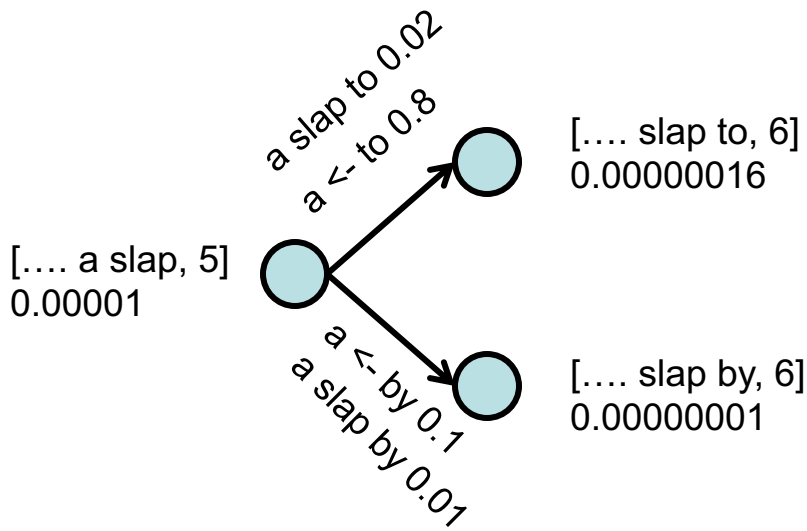
$$\arg\max_{\mathbf{e}} \left[\prod_{\langle \bar{e}, \bar{f} \rangle} P(\bar{f}|\bar{e}) \cdot \prod_{i=1}^{|\mathbf{e}|} P(e_i | e_{i-1}, e_{i-2}) \right]$$

Phrase-Based Decoding

Monotonic Word Translation

Maria	no	dio	una	bofetada	a	la	bruja	verde
Mary	not did not no	give	a	slap	to by	the	witch	green

- Cost is LM * TM
- It's an HMM?
  - $P(e|e_{-1},e_{-2})$
  - $P(f|e)$
- State includes
  - Exposed English
  - Position in foreign
- Dynamic program loop?




```
for (fPosition in 1…|f|)
  for (eContext in allEContexts)
    for (eOption in translations[fPosition])
      score = scores[fPosition-1][eContext] * LM(eContext+eOption) * TM(eOption, fWord[fPosition])
      scores[fPosition][eContext[2]+eOption] =max score
```

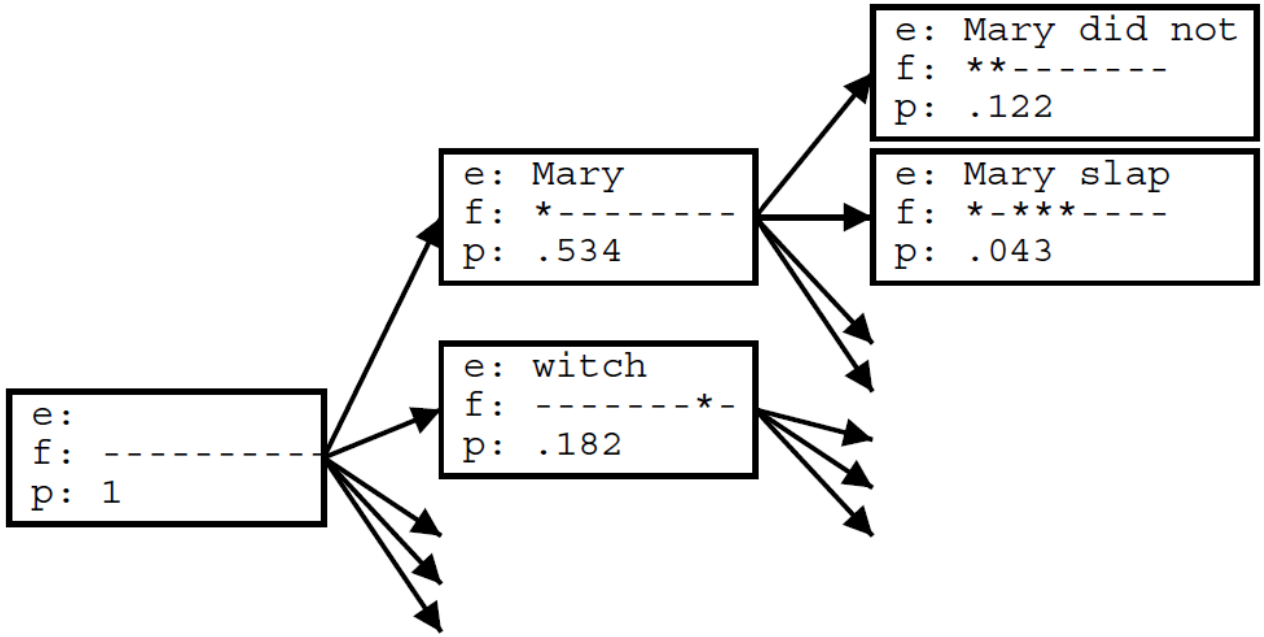
Phrase Translation

- If monotonic, almost an HMM; technically a semi-HMM

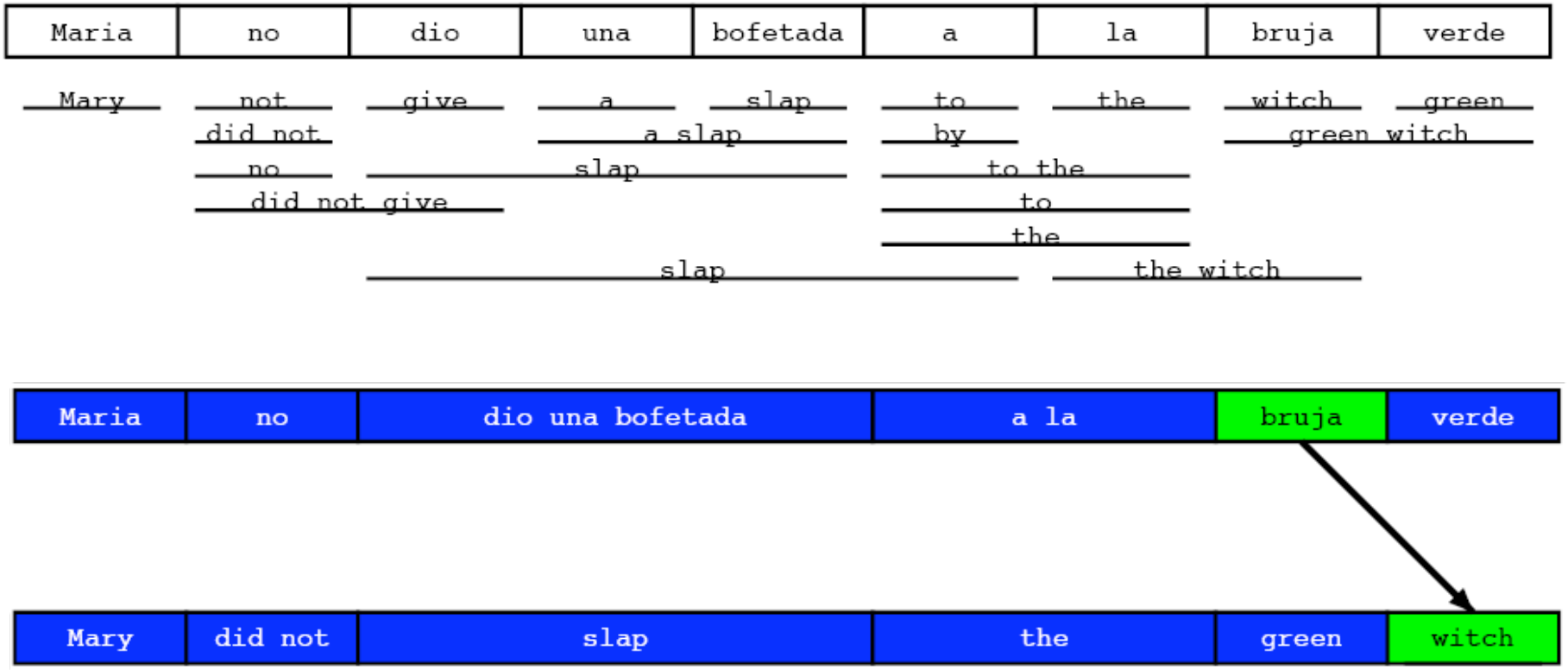
```
for (fPosition in 1…|f|)
  for (lastPosition < fPosition)
    for (eContext in eContexts)
      for (eOption in translations[fPosition])
        … combine hypothesis for (lastPosition ending in eContext) with eOption
```

- If distortion… now what?

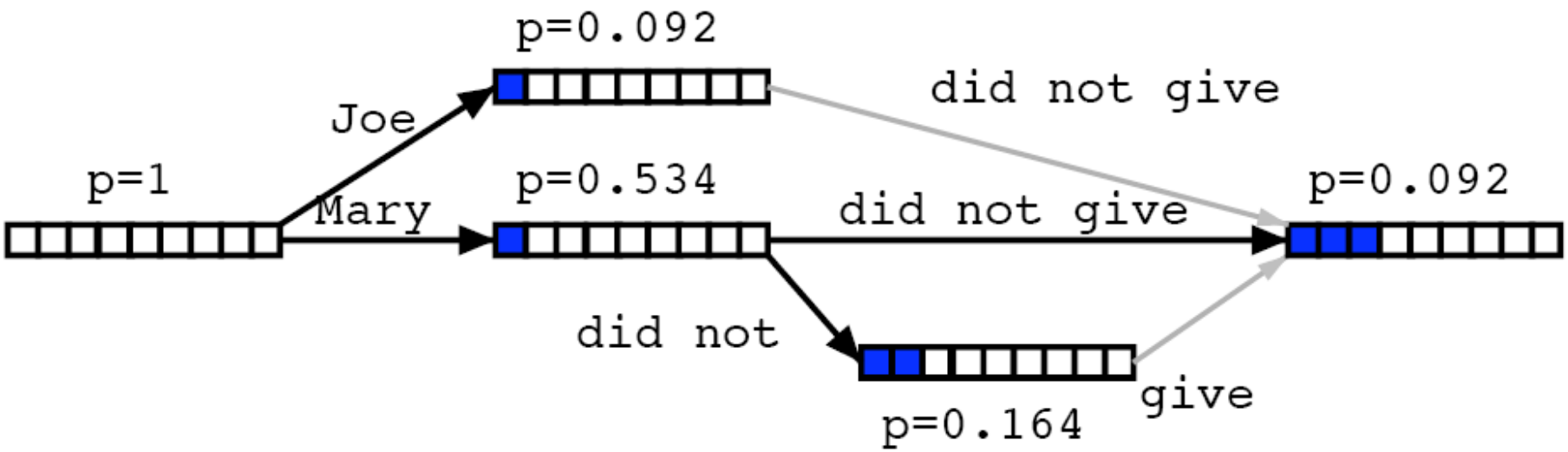
Non-Monotonic Phrasal MT



The Pharaoh Decoder



Hypotheis Lattices



Parameter Tuning

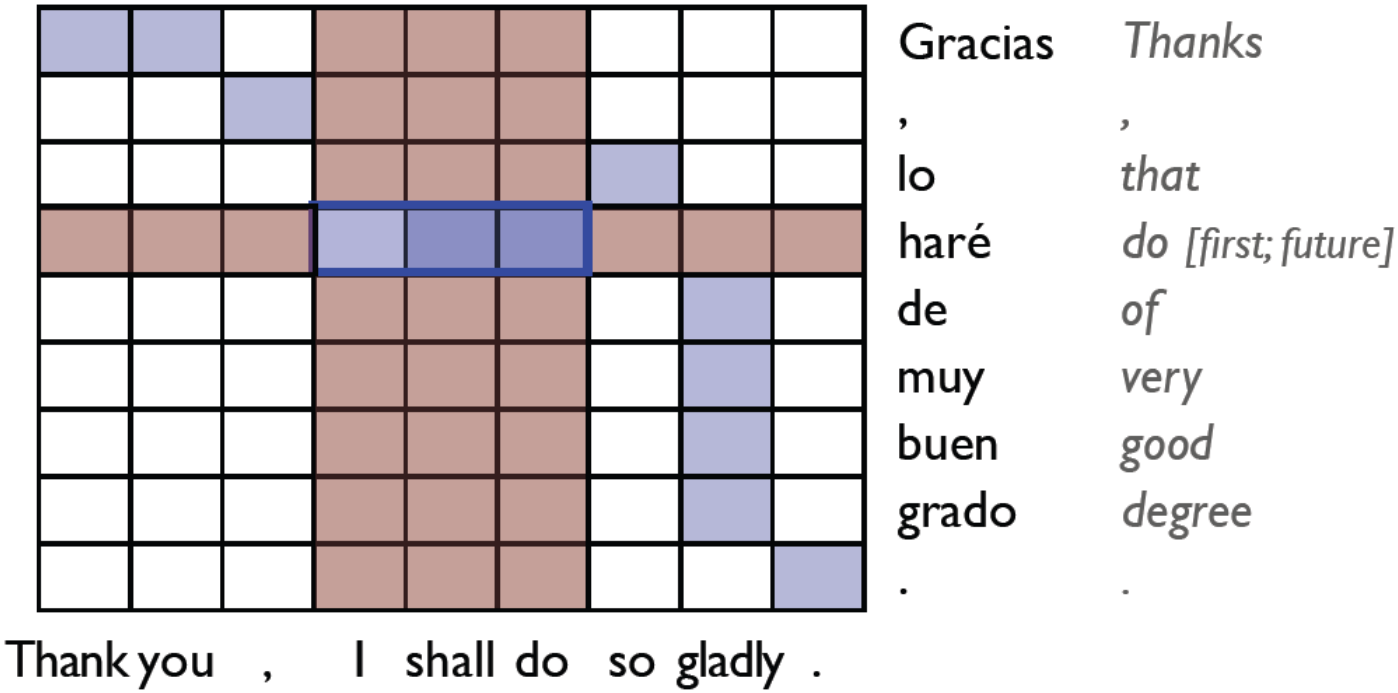
Counting Phrase Pairs

Input:

Gracias , lo haré de muy buen grado .
Thank you , I shall do so gladly .

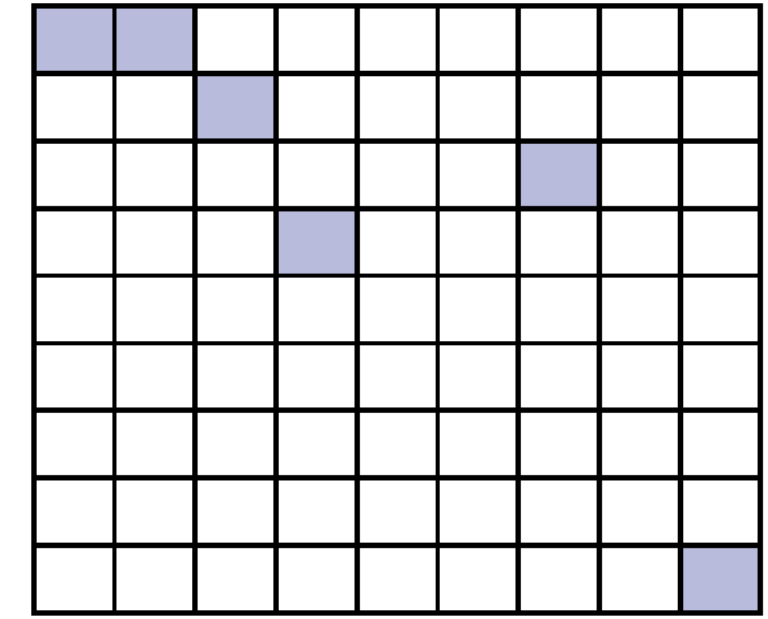
First, we learn word alignments,

then we infer aligned phrases.



What Happens in Practice

A real word alignment
(GIZA++ Model 4 with
grow-diag-final combination)



	Gloss
Gracias	*Thanks*
,	*,*
lo	*that*
haré	*do [first; future]*
de	*of*
muy	*very*
buen	*good*
grado	*degree*
.	*.*

Thank you , I shall do so gladly .

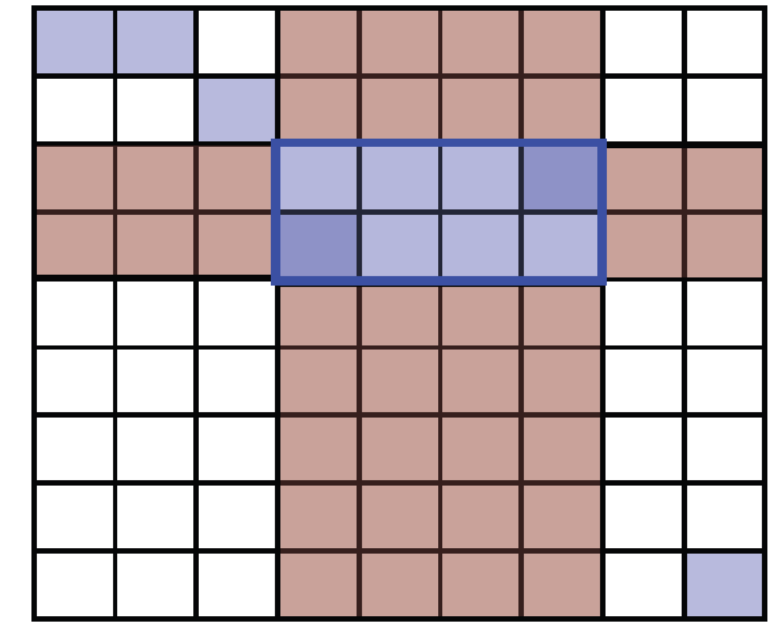
Grow-diag-final?

```
GROW-DIAG-FINAL(e2f,f2e):
 neighboring = ((-1,0),(0,-1),(1,0),(0,1),(-1,-1),(-1,1),(1,-1),(1,1))
 alignment = intersect(e2f,f2e);
 GROW-DIAG(); FINAL(e2f); FINAL(f2e);

GROW-DIAG():
 iterate until no new points added
   for english word e = 0 ... en
     for foreign word f = 0 ... fn
       if ( e aligned with f )
         for each neighboring point ( e-new, f-new ):
           if ( ( e-new not aligned or f-new not aligned ) and
                ( e-new, f-new ) in union( e2f, f2e ) )
             add alignment point ( e-new, f-new )
FINAL(a):
 for english word e-new = 0 ... en
   for foreign word f-new = 0 ... fn
     if ( ( e-new not aligned or f-new not aligned ) and
          ( e-new, f-new ) in alignment a )
       add alignment point ( e-new, f-new )
```

What Happens in Practice

A real word alignment
(GIZA++ Model 4 with
grow-diag-final combination)

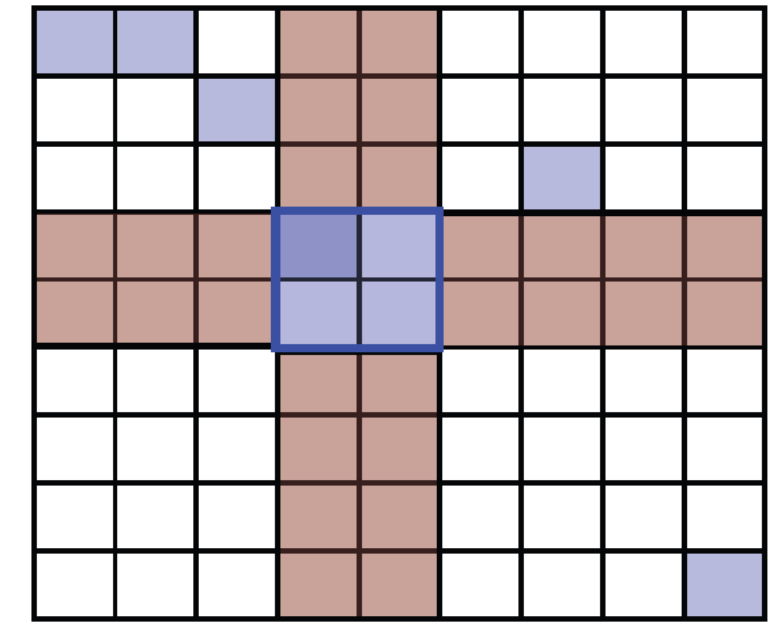


	Gloss
Gracias	*Thanks*
,	*,*
lo	*that*
haré	*do [first; future]*
de	*of*
muy	*very*
buen	*good*
grado	*degree*
.	*.*

Thank you , I shall do so gladly .

What Happens in Practice

A real word alignment
(GIZA++ Model 4 with
grow-diag-final combination)

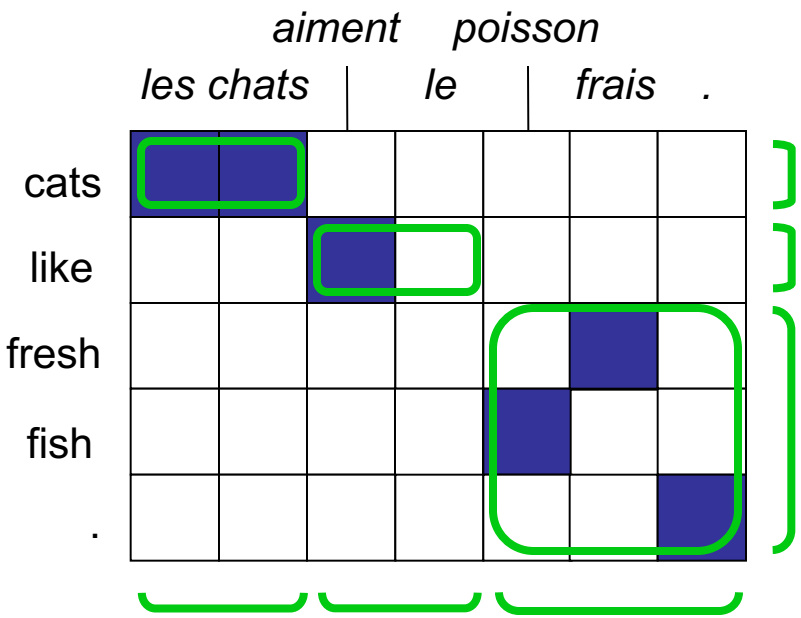


	Gloss
Gracias	*Thanks*
,	*,*
lo	*that*
haré	*do [first; future]*
de	*of*
muy	*very*
buen	*good*
grado	*degree*
.	*.*

Thank you , I shall do so gladly .

Phrase Scoring

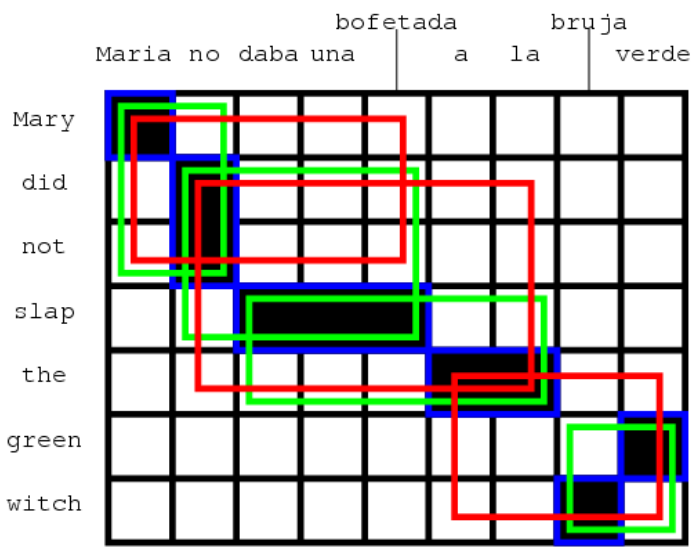
$$\phi_{new}(\bar{e}_j|\bar{f}_i) = \frac{c(\bar{f}_i, \bar{e}_j)}{c(\bar{f}_i)}$$

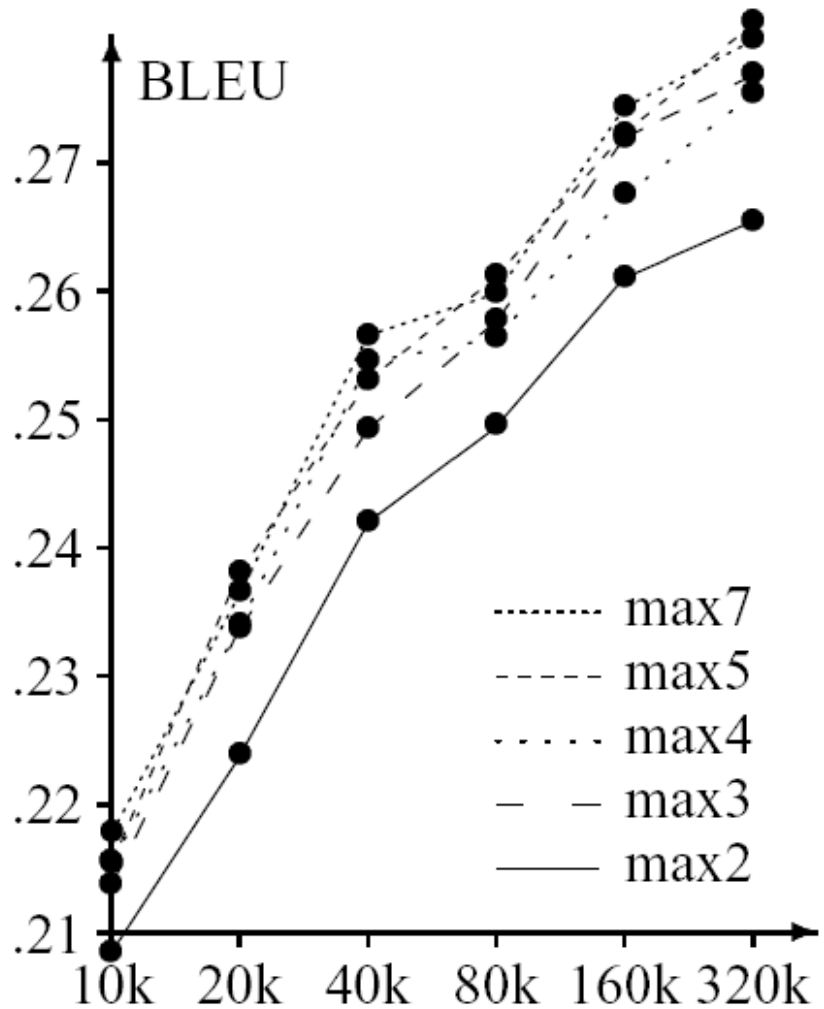


- Learning weights has been tried, several times:
  - [Marcu and Wong, 02]
  - [DeNero et al, 06]
  - … and others
- Seems not to work well, for a variety of partially understood reasons

Phrase Size

- Phrases do help
  - But they don't need to be long



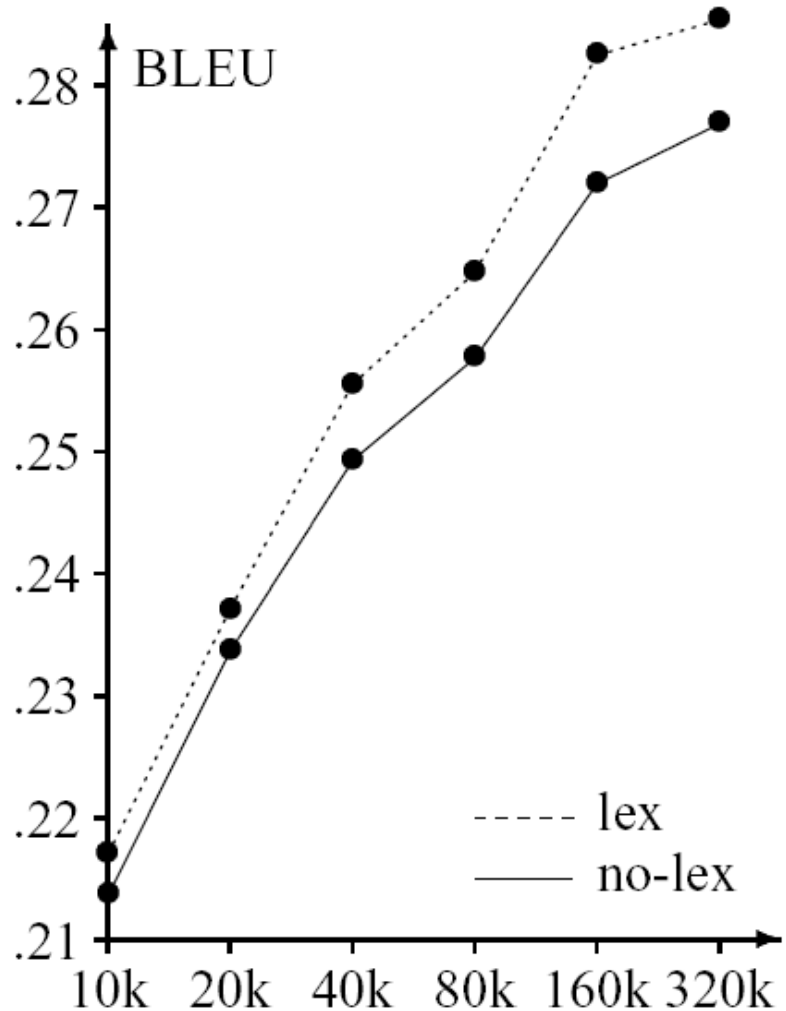


Lexical Weighting

$$\phi(\bar{f}_i|\bar{e}_i) = \frac{count(\bar{f}_i, \bar{e}_i)}{count(\bar{e}_i)} p_w(\bar{f}_i|\bar{e}_i)$$

```
        f1 f2 f3
NULL    -- -- ##
  e1    ## -- --
  e2    -- ## --
  e3    -- ## --
```

$$\begin{aligned} p_w(\bar{f}|\bar{e}, a) &= p_w(f_1 f_2 f_3 | e_1 e_2 e_3, a) \\ &= w(f_1|e_1) \\ &\quad \times \frac{1}{2}(w(f_2|e_2) + w(f_2|e_3)) \\ &\quad \times w(f_3|\text{NULL}) \end{aligned}$$

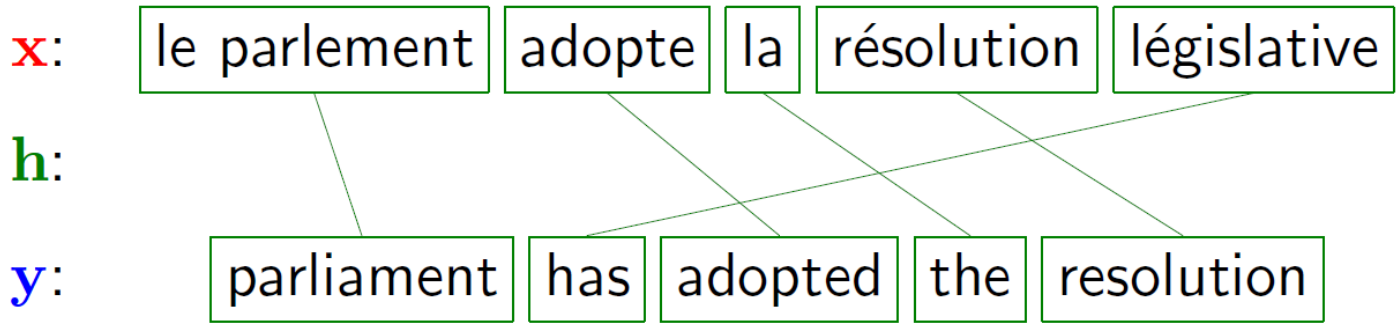


Tuning for MT

- Features encapsulate lots of information
  - Basic MT systems have around 6 features
  - P(e|f), P(f|e), lexical weighting, language model
- How to tune feature weights?
- Idea 1: Use your favorite classifier

Why Tuning is Hard

- Problem: There are latent variables
  - Alignments and segementations

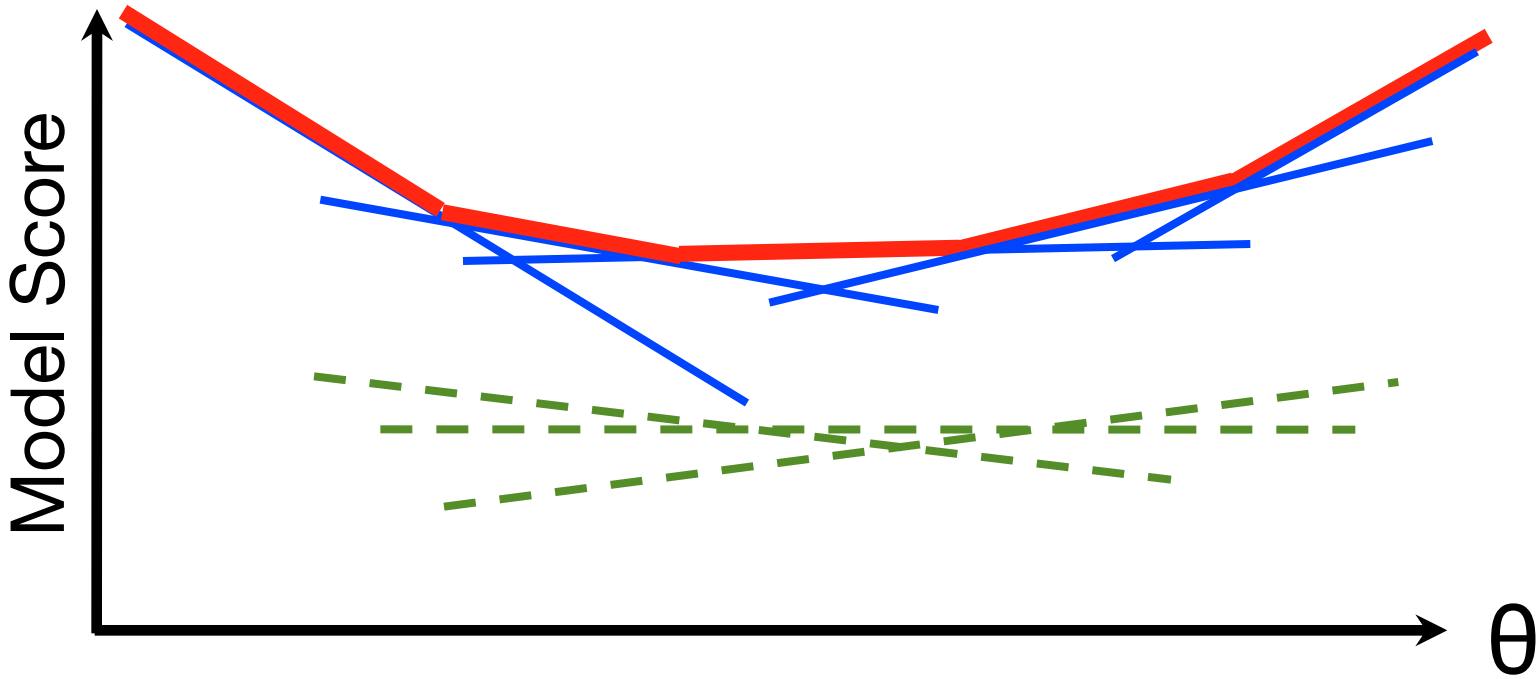


Why Tuning is Hard

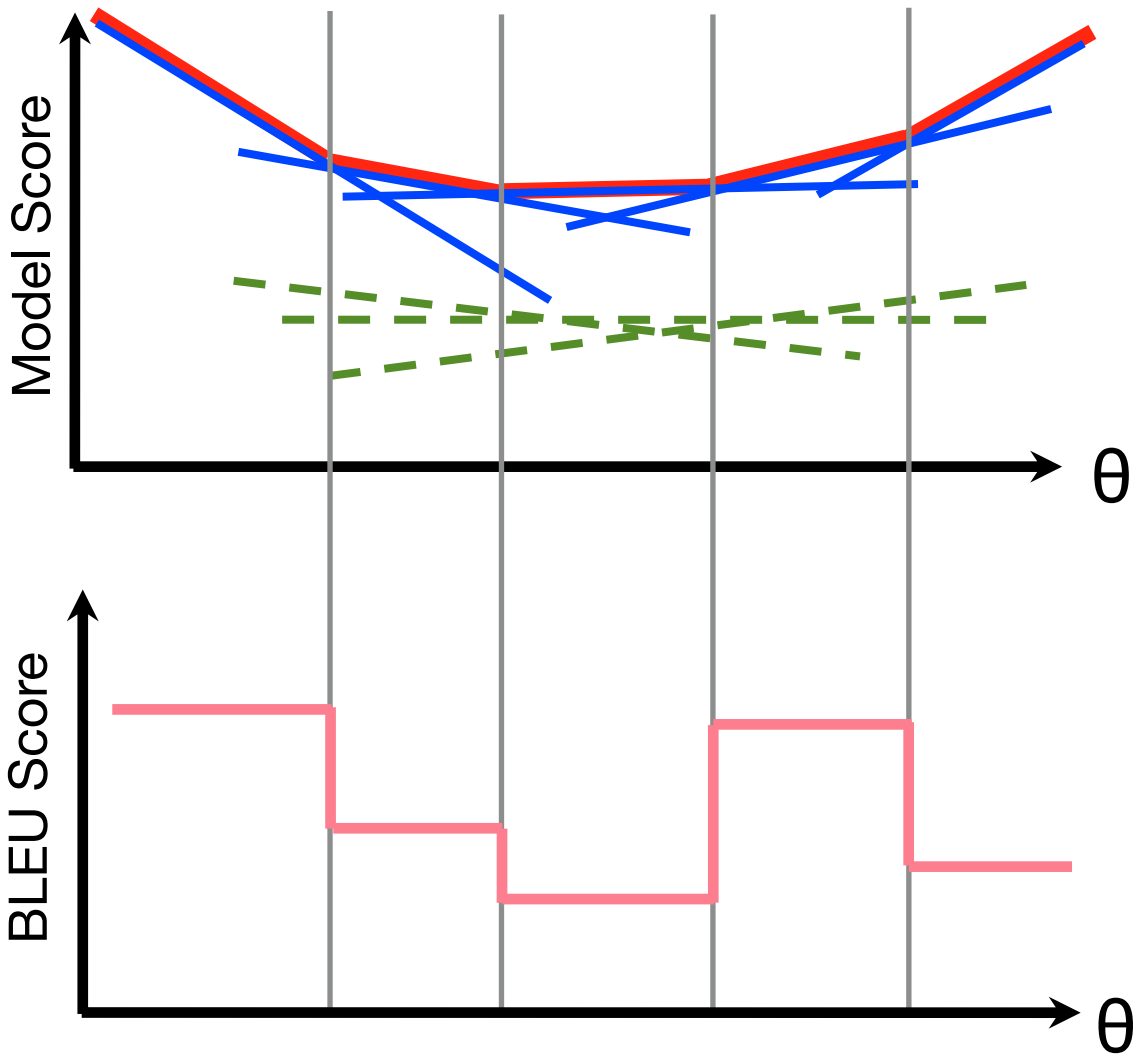
- Problem: Computational constraints
  - Discriminative training involves repeated decoding
  - Very slow! So people tune on sets much smaller than those used to build phrase tables

Minimum Error Rate Training

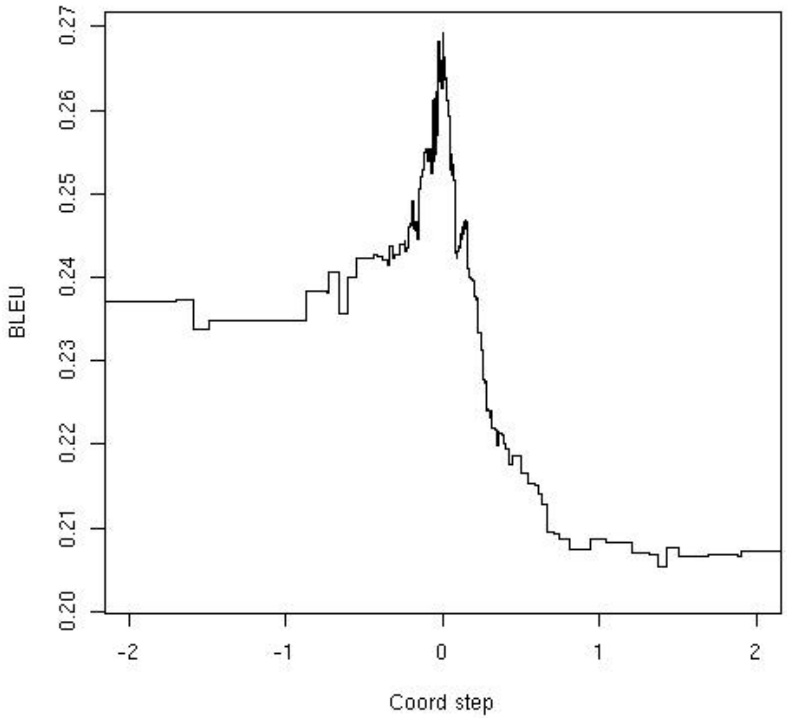
- Standard method: minimize BLEU directly (Och 03)
  - MERT is a discontinuous objective
  - Only works for max ~10 features, but works very well then
  - Here: k-best lists, but forest methods exist (Machery et al 08)
  - Lots of alternatives have been explored for more features

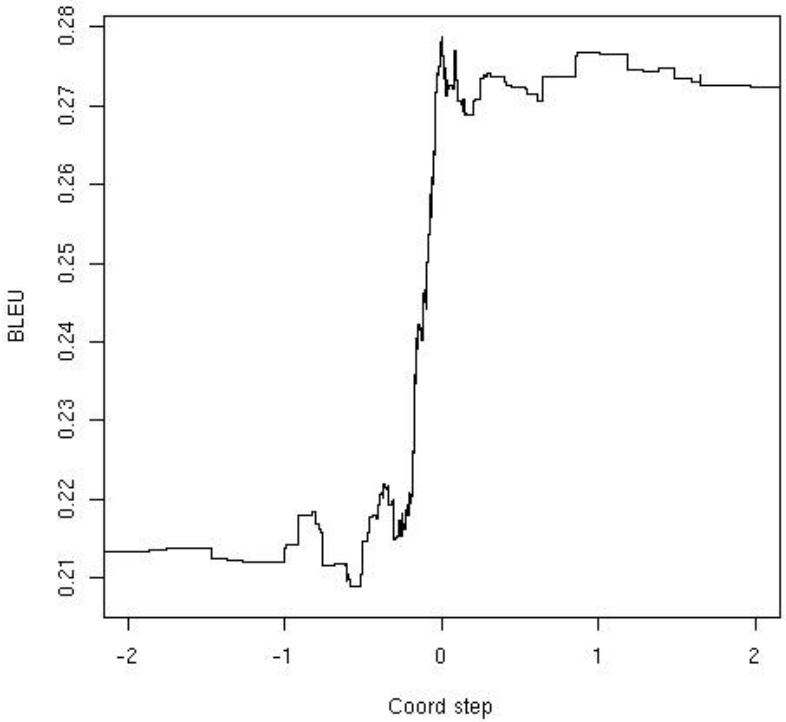


MERT



MERT


BLEU
0.20
0.21
0.22
0.23
0.24
0.25
0.26
0.27
-2
-1
0
1
2
Coord step


BLEU
0.21
0.22
0.23
0.24
0.25
0.26
0.27
0.28
-2
-1
0
1
2
Coord step

Syntactic Models

Translating with Tree Transducers

Input **Output**

lo haré de muy buen grado .

Grammar

Translating with Tree Transducers

Input **Output**

lo haré de muy buen grado .

Grammar

ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input **Output**

ADV

lo haré de muy buen grado .

ADV
|
gladly

Grammar

ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input **Output**

ADV

lo haré de muy buen grado .

ADV
|
gladly

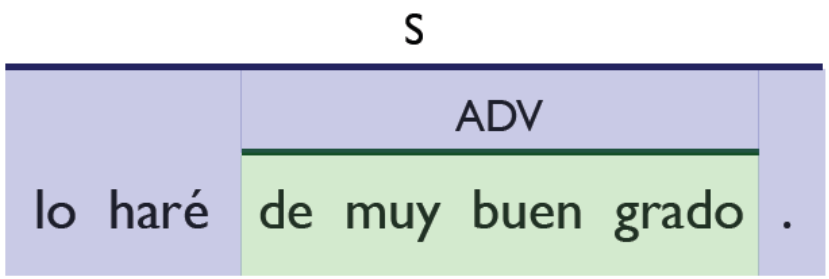
Grammar

S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

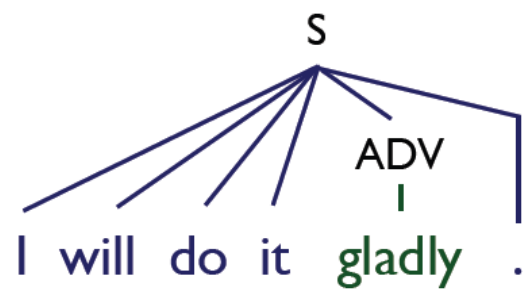
ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input



Output



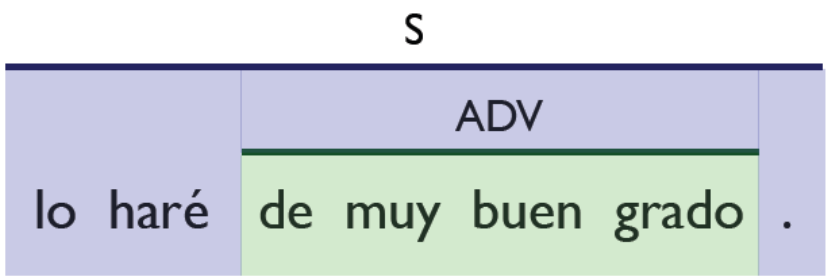
Grammar

S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

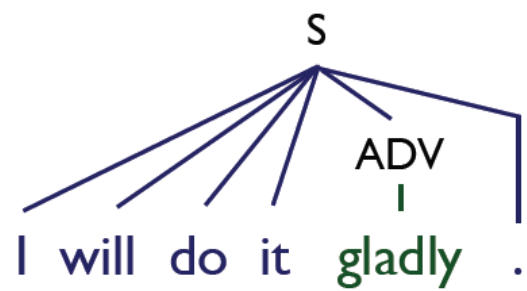
ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

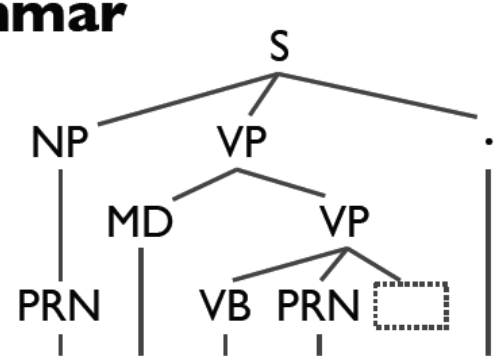
Input



Output



Grammar



S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input **Output**

ADV

lo haré de muy buen grado .

ADV
|
gladly

Grammar

S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input **Output**

ADV

lo haré de muy buen grado .

ADV
|
gladly

Grammar

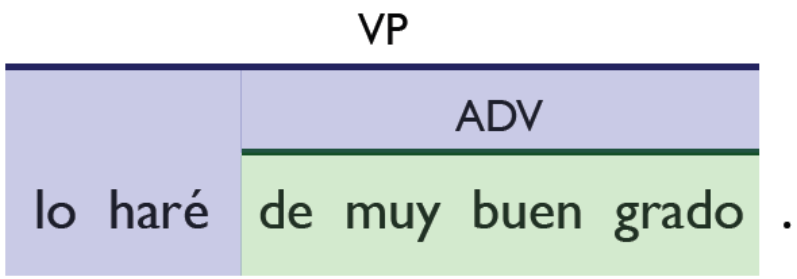
VP → ⟨ lo haré ADV ; will do it ADV ⟩

S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

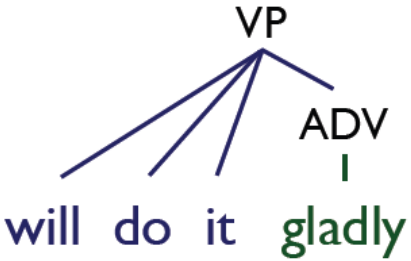
ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input



Output



Grammar

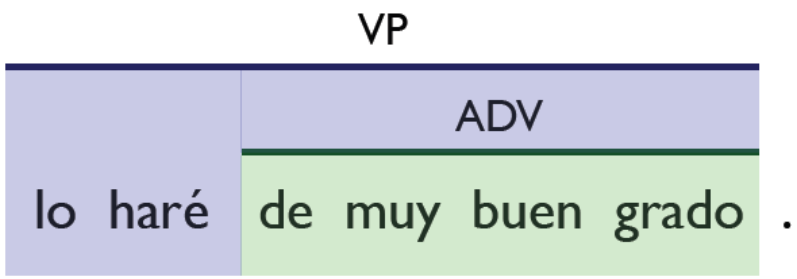
VP → ⟨ lo haré ADV ; will do it ADV ⟩

S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

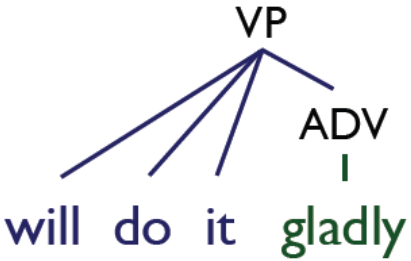
ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input



Output



Grammar

S → ⟨ VP . ; I VP . ⟩

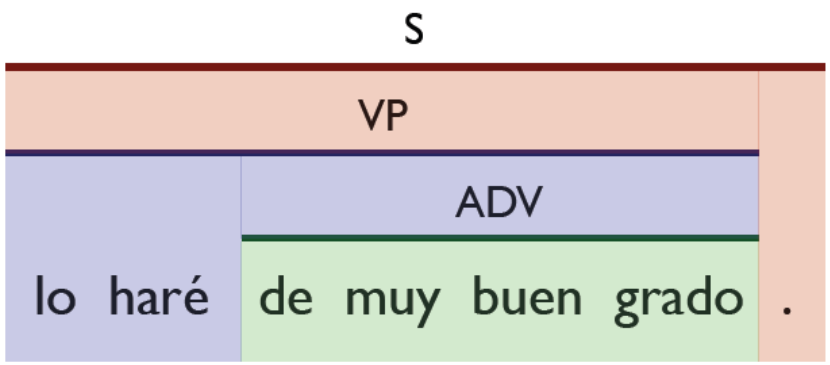
VP → ⟨ lo haré ADV ; will do it ADV ⟩

S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

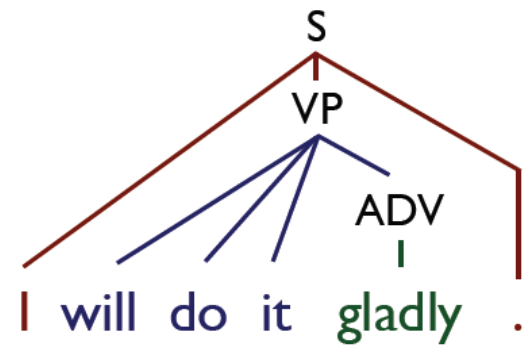
ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input



Output



Grammar

S → ⟨ VP . ; I VP . ⟩

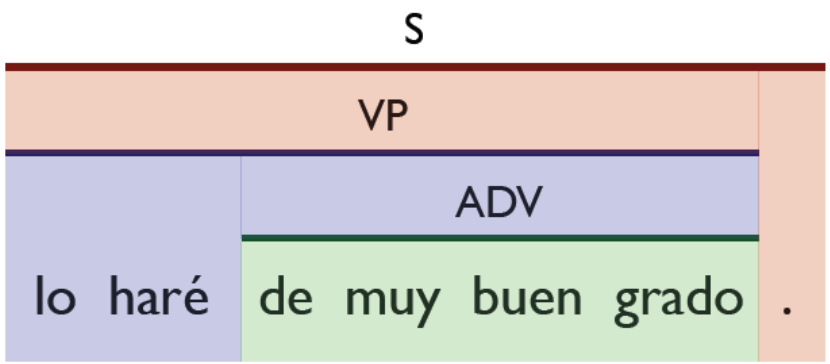
VP → ⟨ lo haré ADV ; will do it ADV ⟩

S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

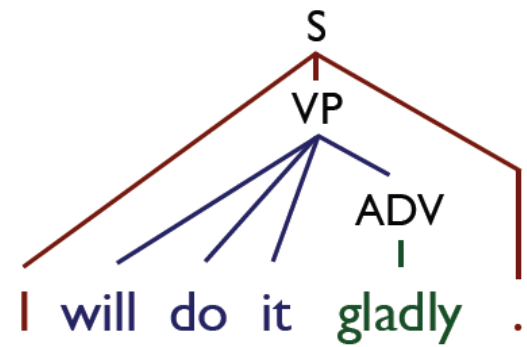
ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input



Output



Grammar

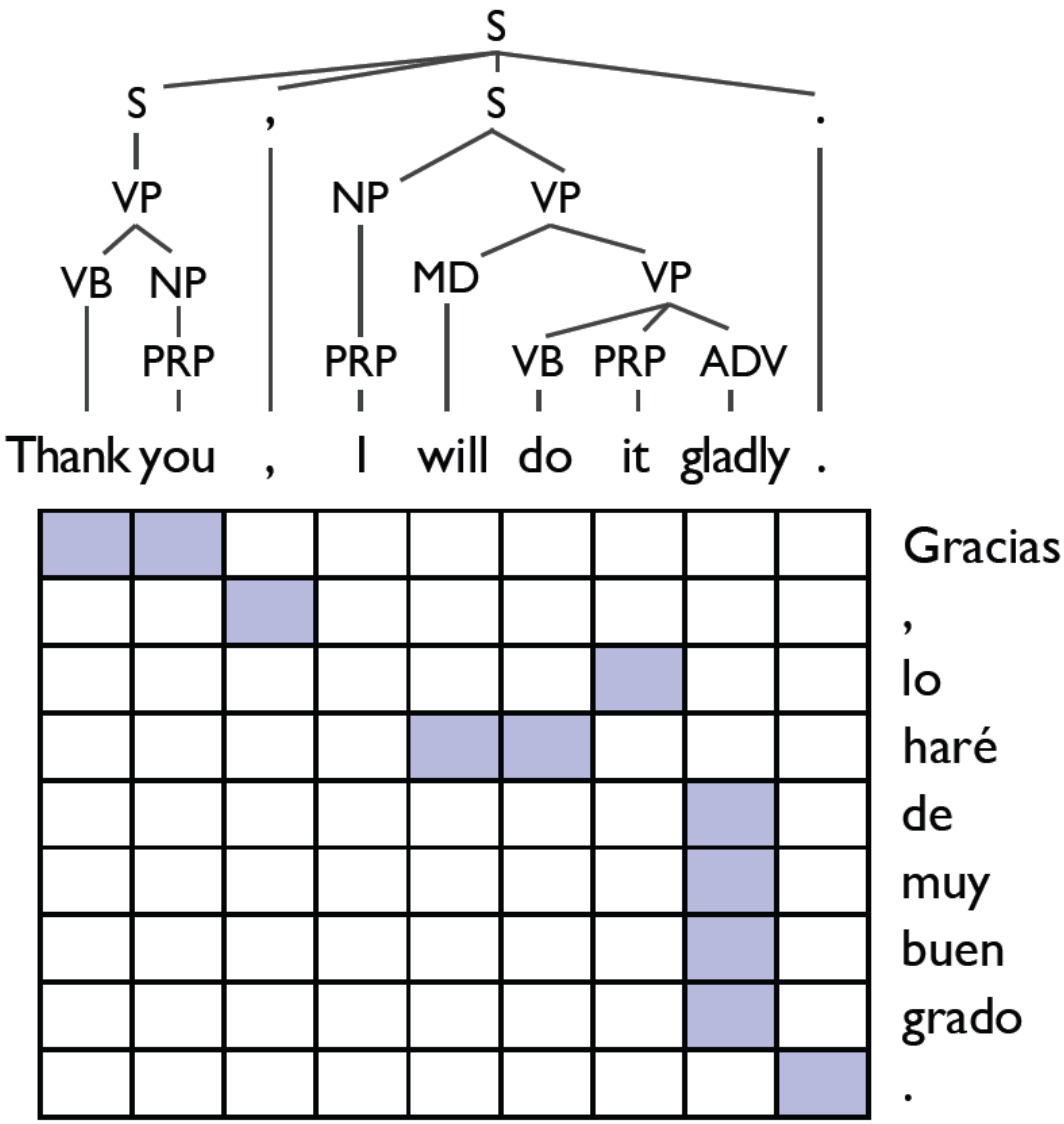
S → ⟨ VP . ; I VP . ⟩ **OR** S → ⟨ VP . ; you VP . ⟩

VP → ⟨ lo haré ADV ; will do it ADV ⟩

S → ⟨ lo haré ADV . ; I will do it ADV . ⟩

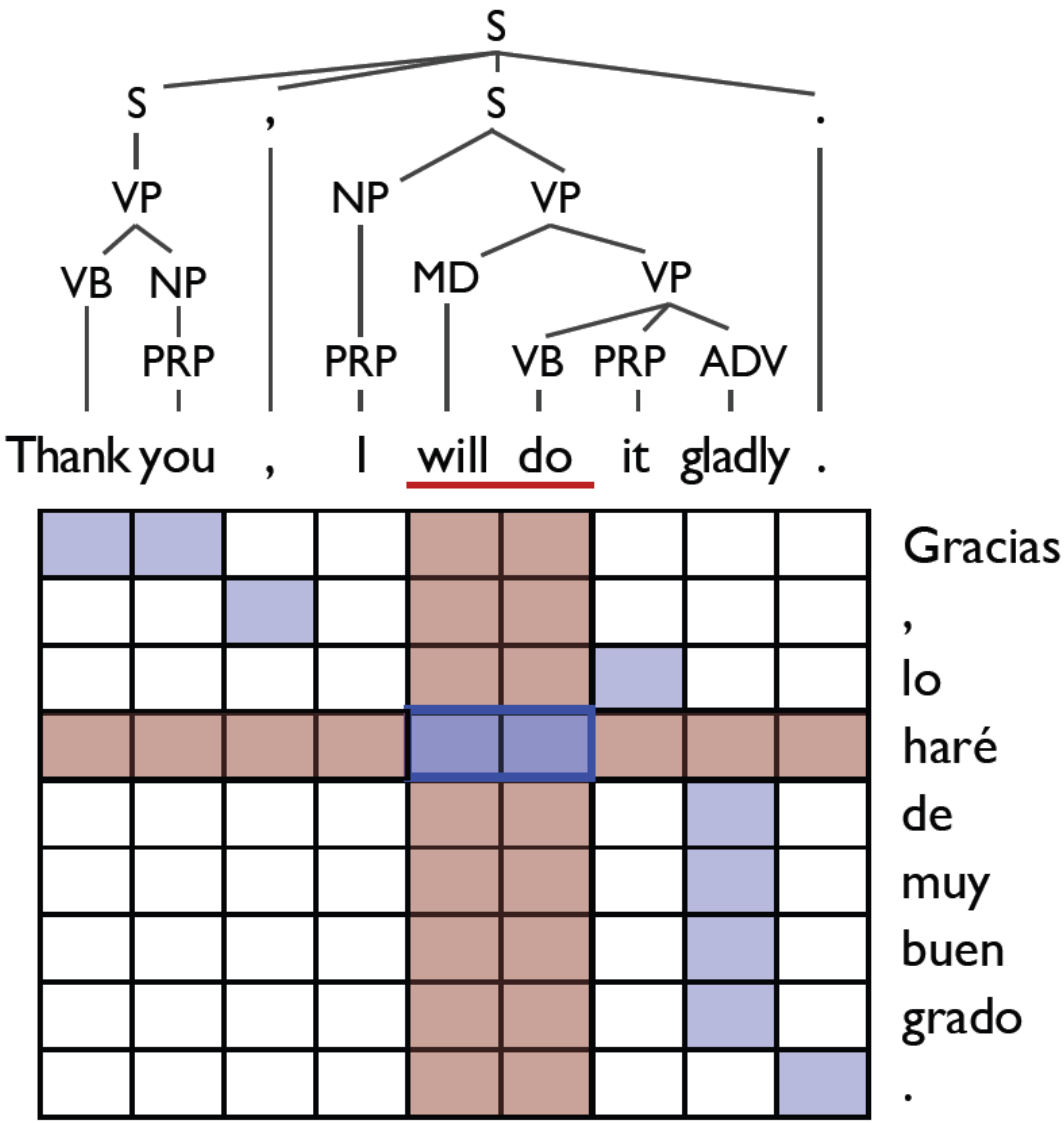
ADV → ⟨ de muy buen grado ; gladly ⟩

Learning Grammars for Translation



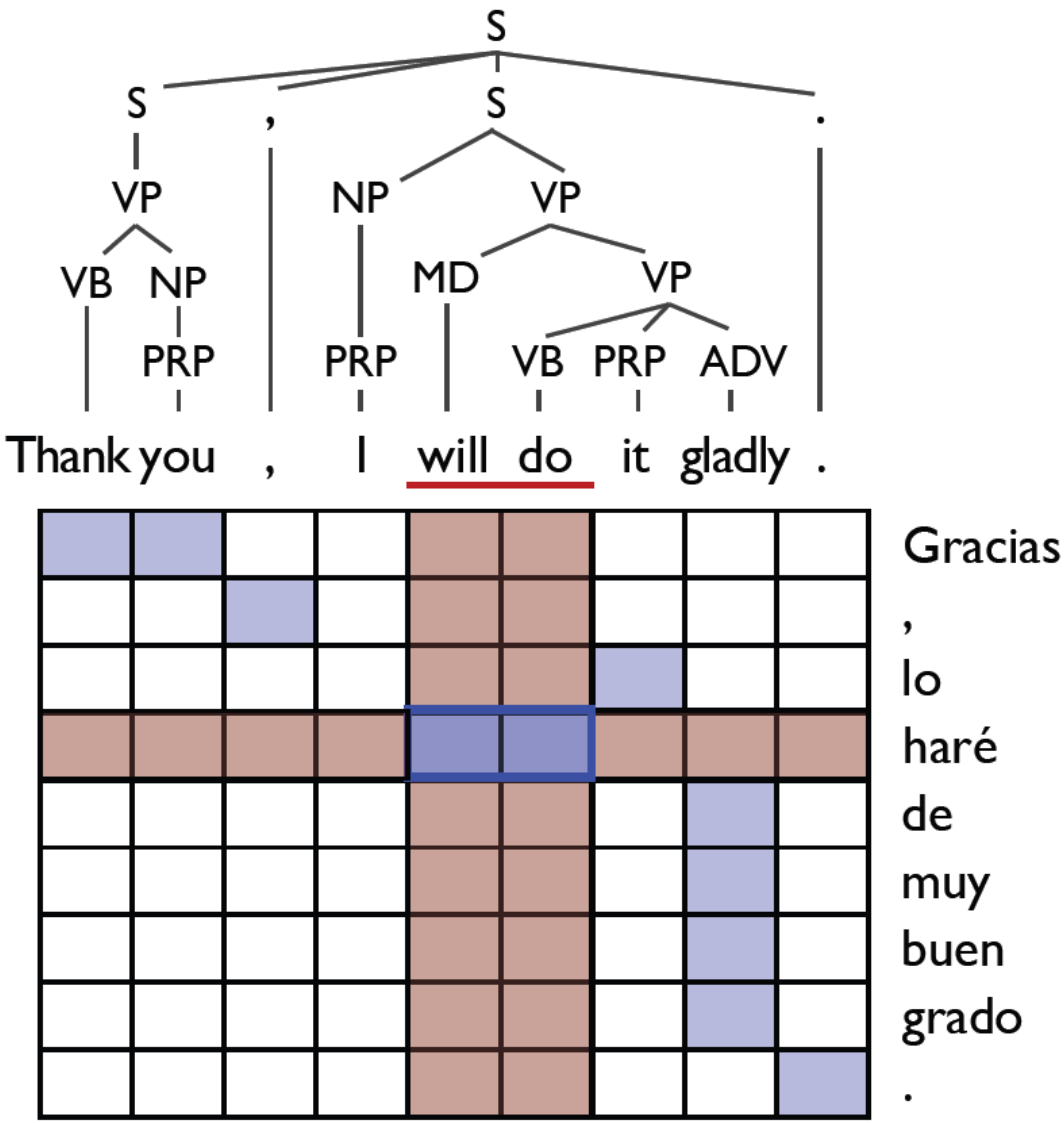
Grammar Rules

Learning Grammars for Translation



Grammar Rules

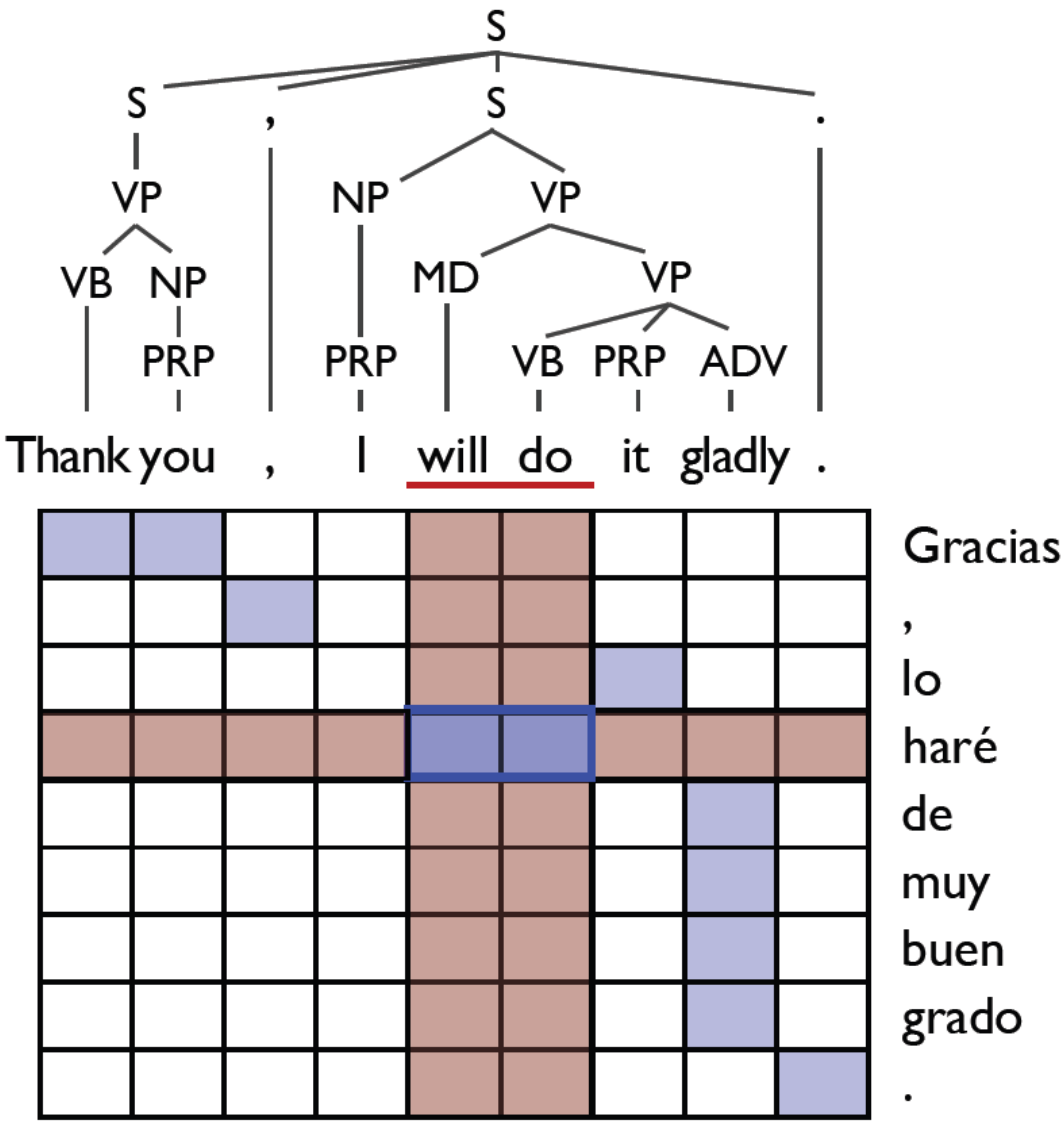
Learning Grammars for Translation



Grammar Rules

⟨haré ; will do⟩

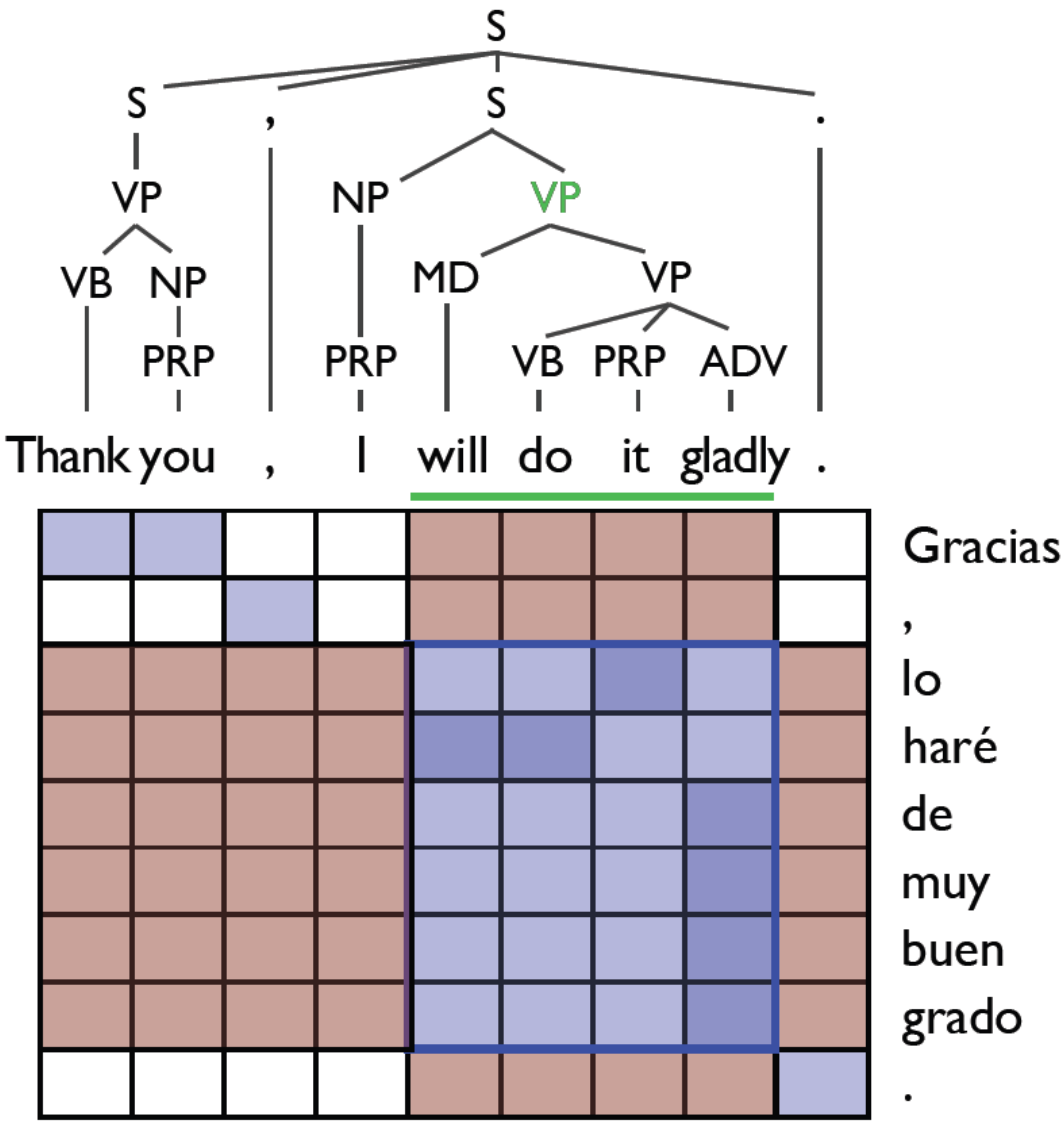
Learning Grammars for Translation



Grammar Rules

~~⟨haré ; will do⟩~~

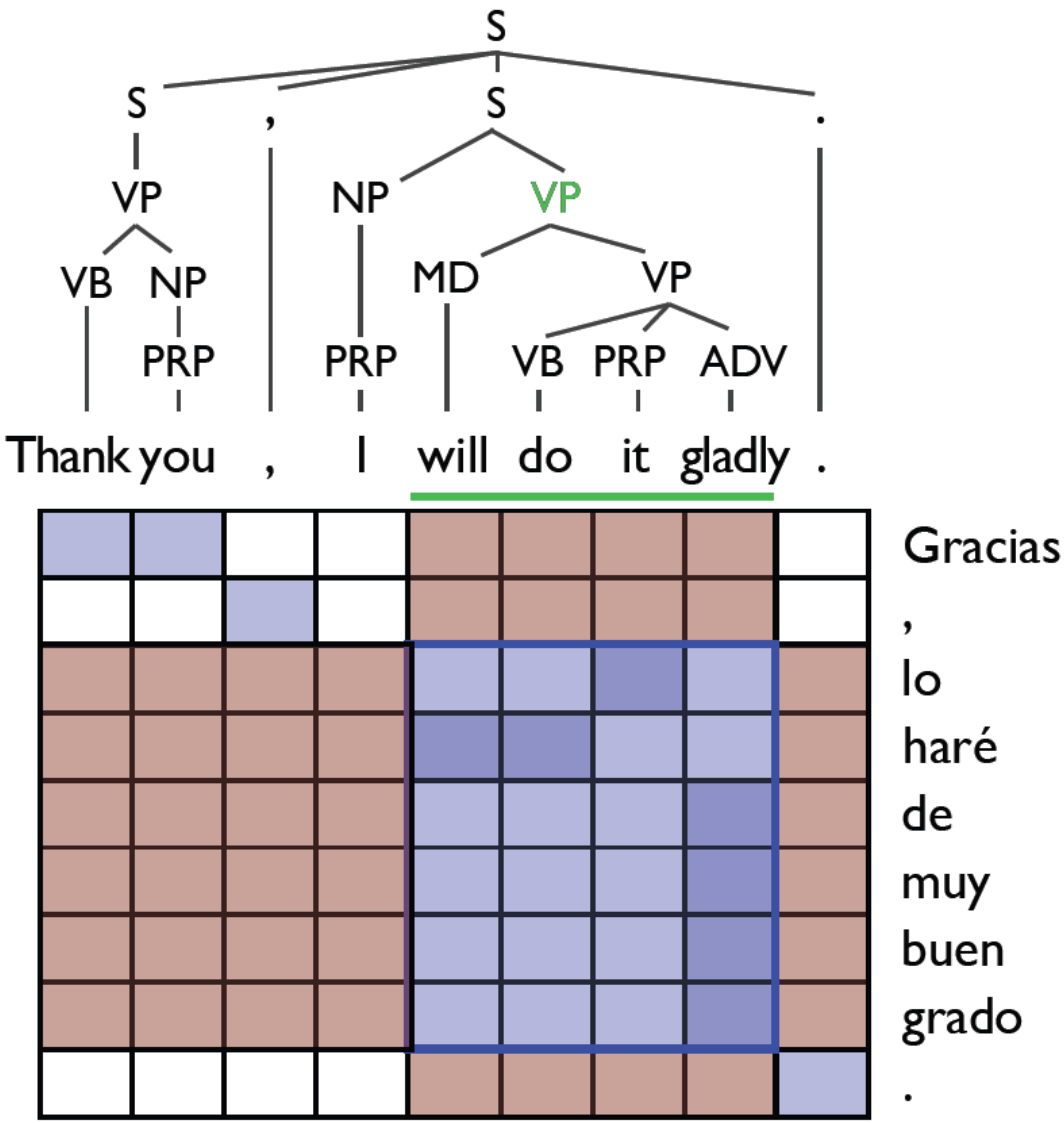
Learning Grammars for Translation



Grammar Rules

~~⟨haré ; will do⟩~~

Learning Grammars for Translation



Grammar Rules

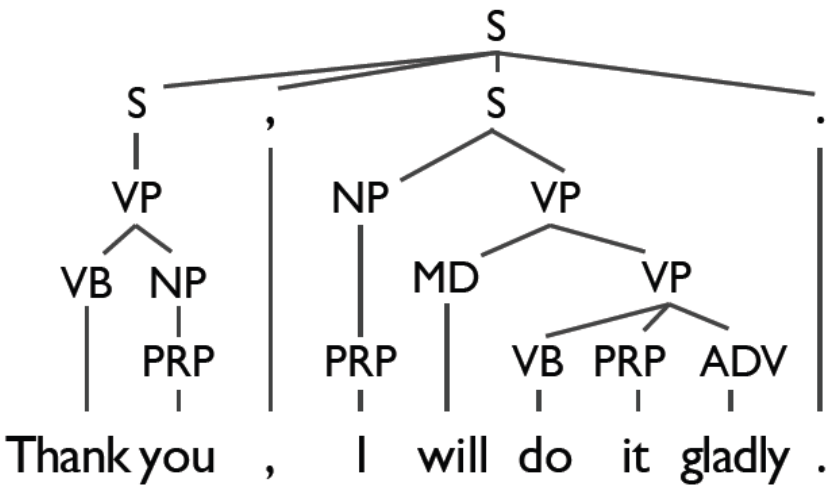
~~⟨haré ; will do⟩~~

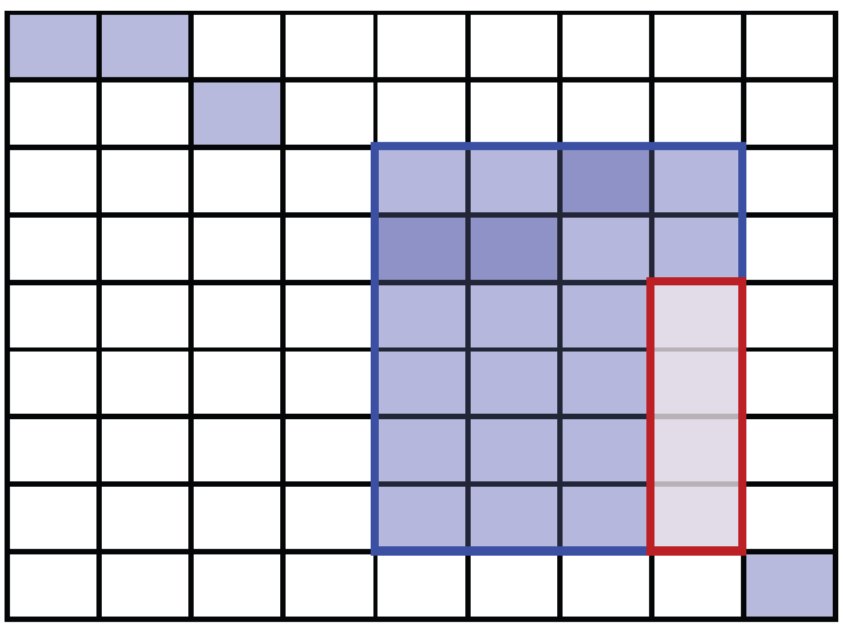
VP →

⟨lo haré de ... grado ;

will do it gladly⟩

Learning Grammars for Translation





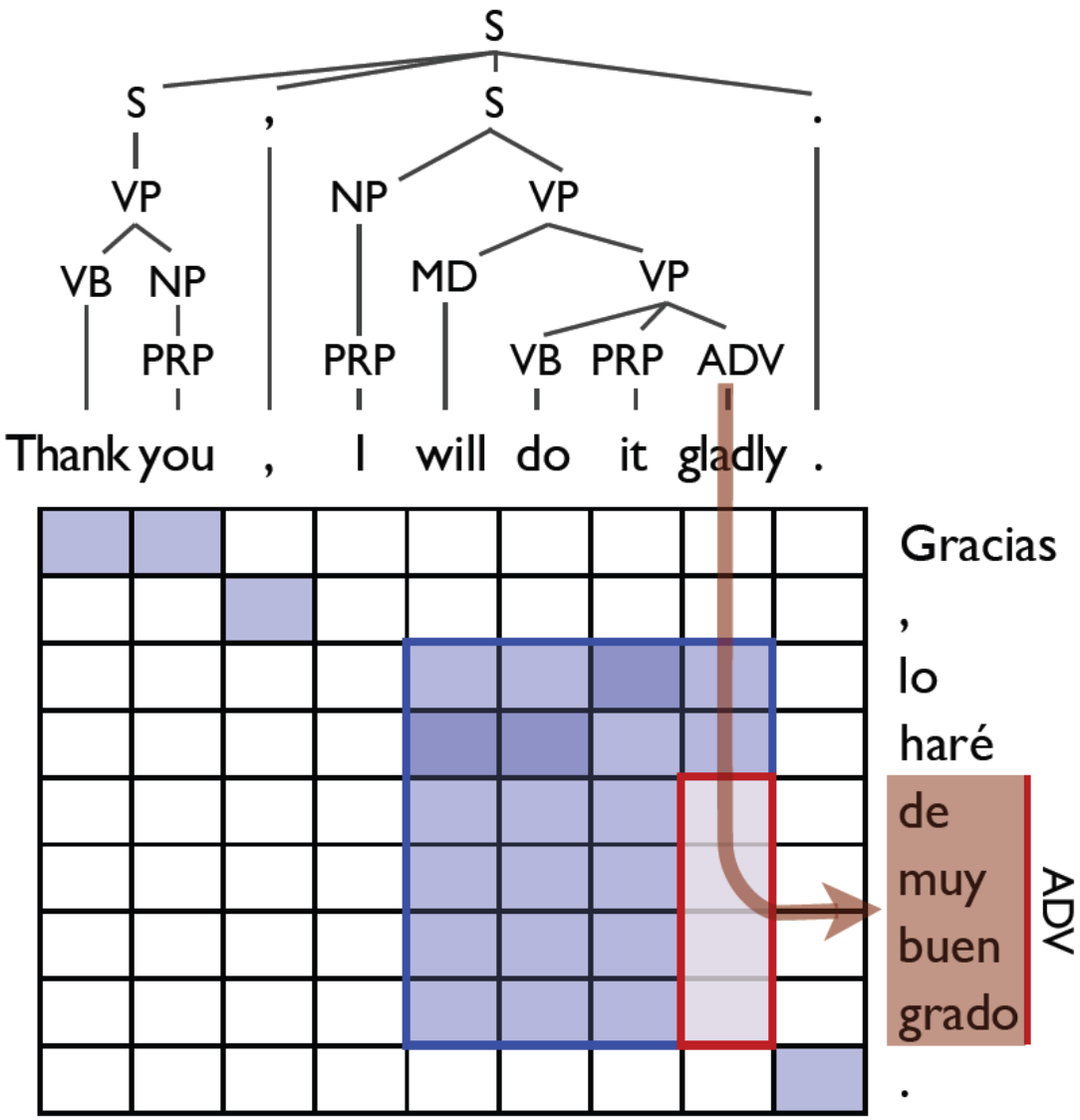
Gracias
,
lo
haré
de
muy
buen
grado
.

Grammar Rules

~~⟨haré ; will do⟩~~

VP →
⟨lo haré de ... grado ;
will do it gladly⟩

Learning Grammars for Translation



Grammar Rules

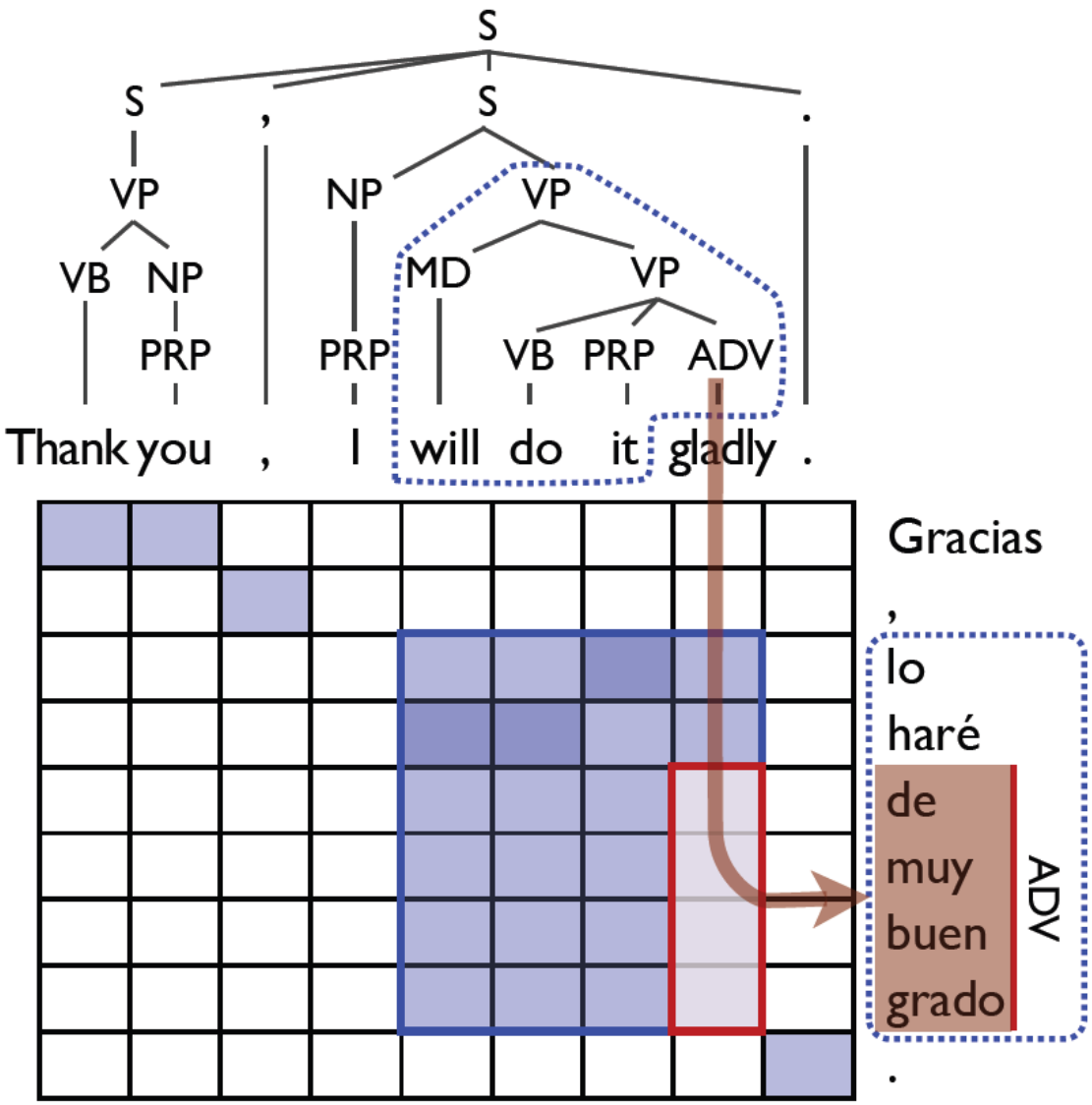
~~⟨haré ; will do⟩~~

VP →

⟨lo haré de ... grado ;

will do it gladly⟩

Learning Grammars for Translation



Grammar Rules

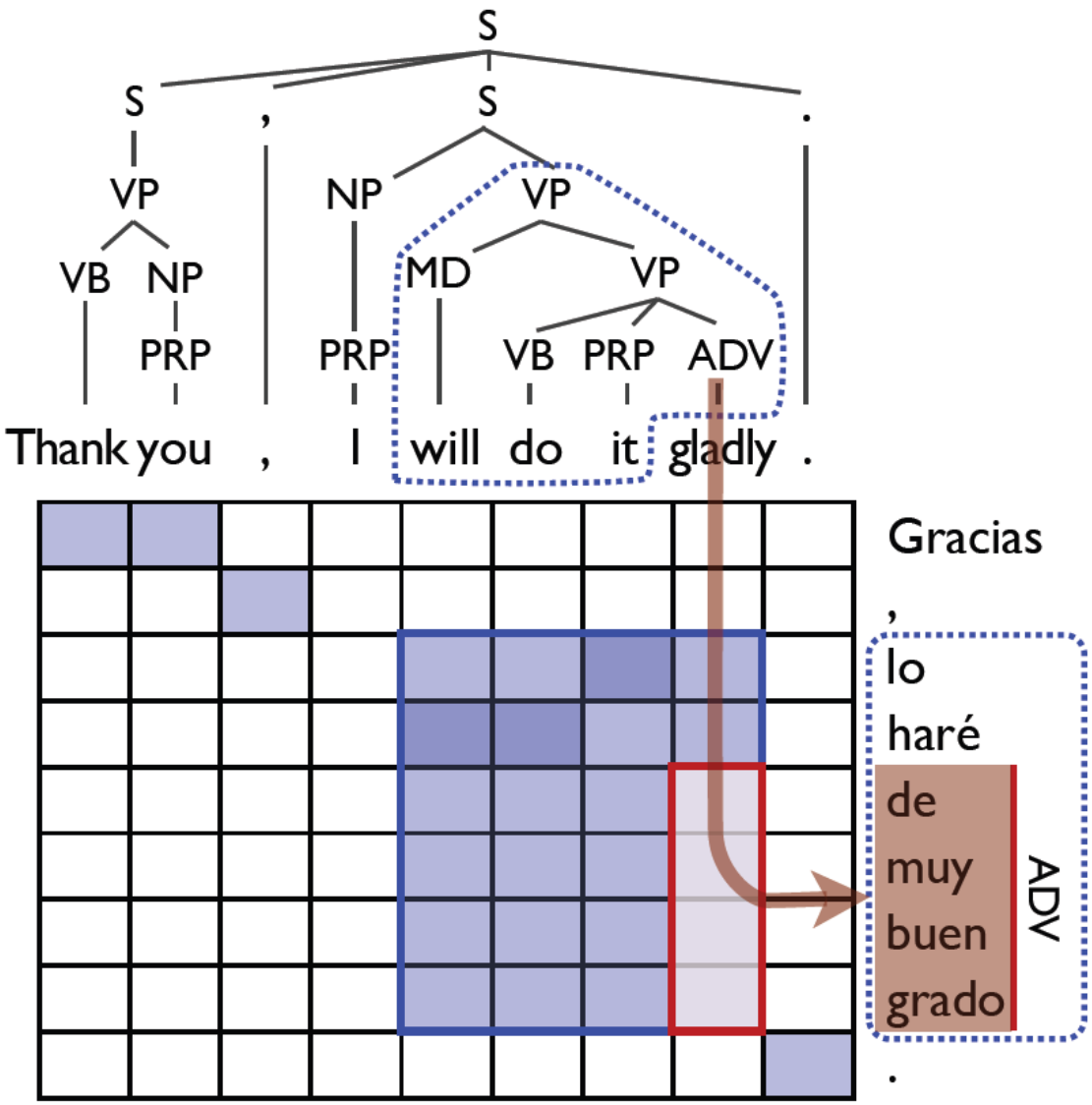
~~⟨haré ; will do⟩~~

VP →

⟨lo haré de ... grado ;

will do it gladly⟩

Learning Grammars for Translation



Grammar Rules

~~⟨haré ; will do⟩~~

VP →
⟨lo haré de ... grado ;
will do it gladly⟩

VP →
⟨lo haré ADV ;
will do it ADV⟩

The Size of Tree Transducer Grammars

Extracted a transducer grammar from a 220 million word bitext

Relativized the grammar to each test sentence

Kept all rules with at most 6 non-terminals

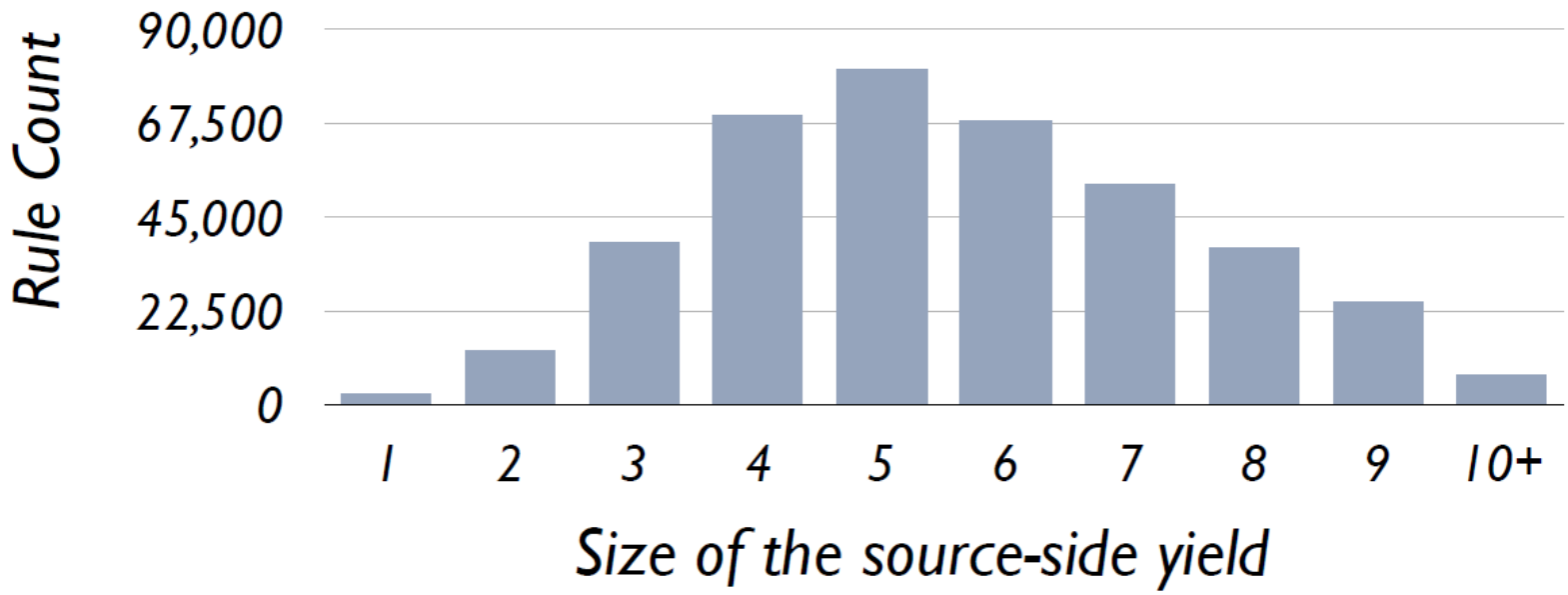
The Size of Tree Transducer Grammars

Extracted a transducer grammar from a 220 million word bitext

Relativized the grammar to each test sentence

Kept all rules with at most 6 non-terminals

Rules matching an example 40-word sentence



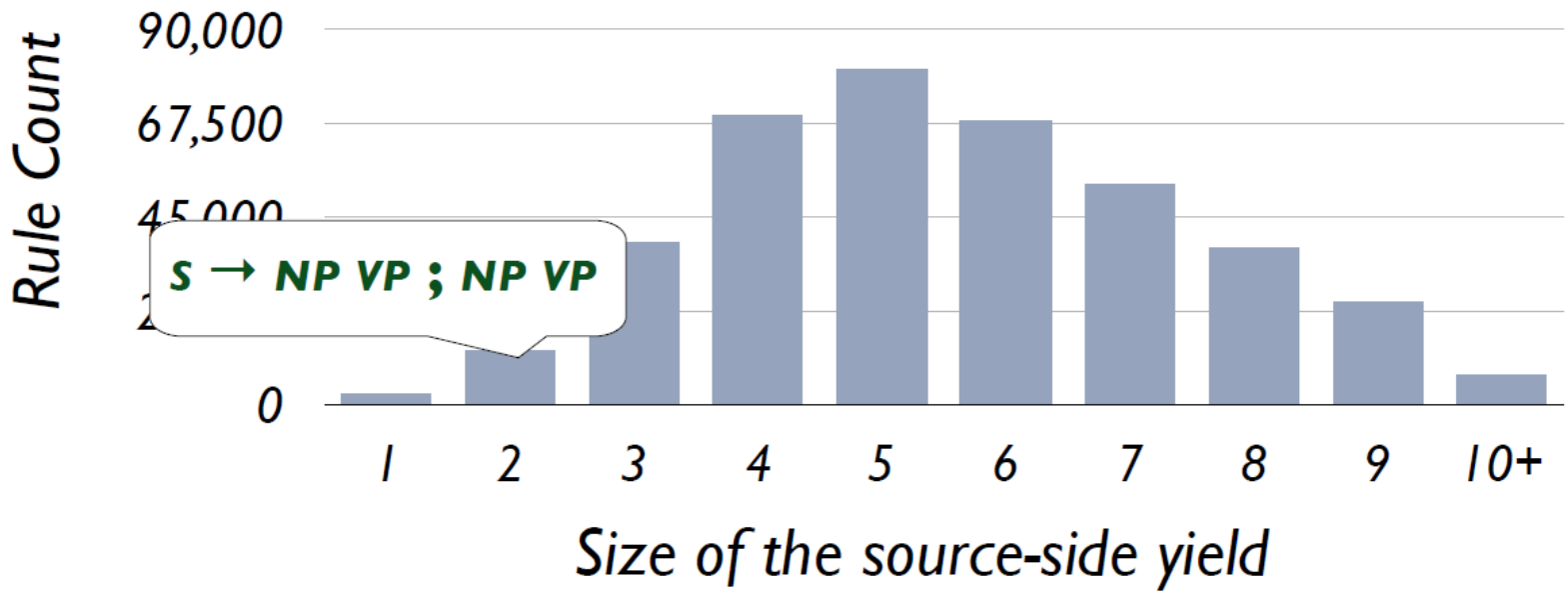
The Size of Tree Transducer Grammars

Extracted a transducer grammar from a 220 million word bitext

Relativized the grammar to each test sentence

Kept all rules with at most 6 non-terminals

Rules matching an example 40-word sentence



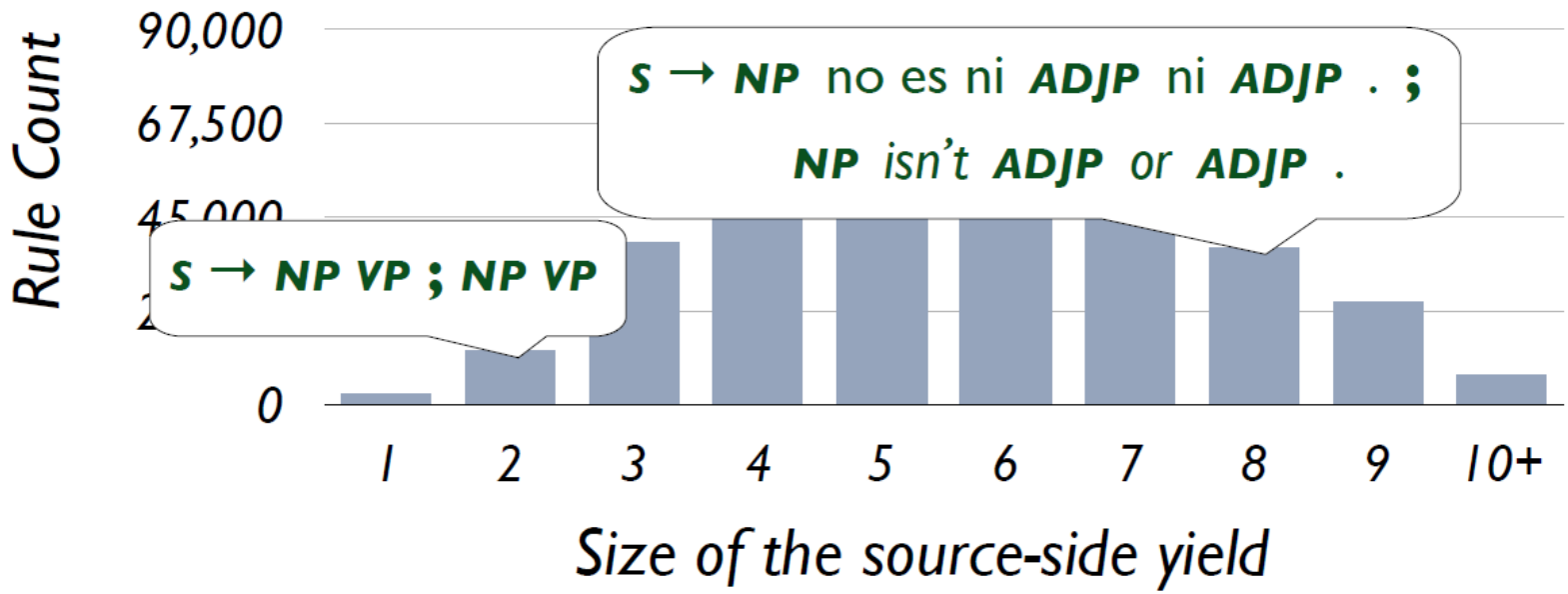
The Size of Tree Transducer Grammars

Extracted a transducer grammar from a 220 million word bitext

Relativized the grammar to each test sentence

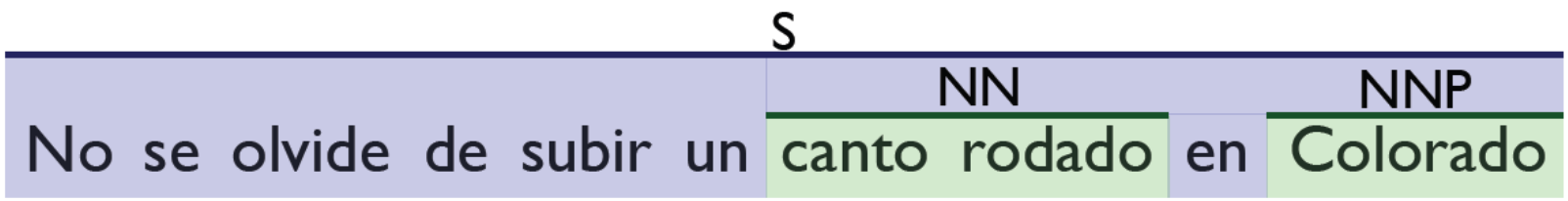
Kept all rules with at most 6 non-terminals

Rules matching an example 40-word sentence



Syntactic Decoding

Tree Transducer Grammars



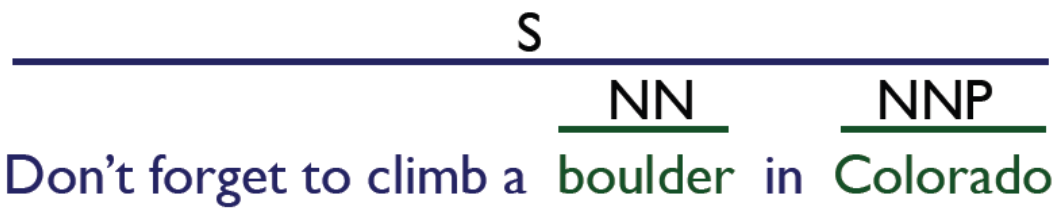
Synchronous Grammar

NNP → Colorado ; *Colorado*

NN → canto rodado ; *boulder*

S → No se olvide de subir un ***NN*** en ***NNP*** ; *Don't forget to climb a **NN** in **NNP***

Output



CKY-style Bottom-up Parsing

For each
span length:

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

Binary rule: $X \rightarrow Y\ Z$

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

Binary rule:	$X \rightarrow Y\ Z$
Split points:	$i < k < j$
Operations:	$O(j - i)$
Time scales with:	Grammar constant

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

$_i$ No se olvide de subir un canto rodado en Colorado $_j$

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

S → No se ***VB*** de subir un ***NN*** en ***NNP***

$_i$ No se olvide de subir un canto rodado en Colorado $_j$

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

S → No se ***VB*** de subir un ***NN*** en ***NNP***

$_i$ No se olvide de subir un canto rodado en Colorado $_j$

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

S → No se ***VB*** de subir un ***NN*** en ***NNP***

$_i$ No se olvide de subir un canto rodado en Colorado $_j$

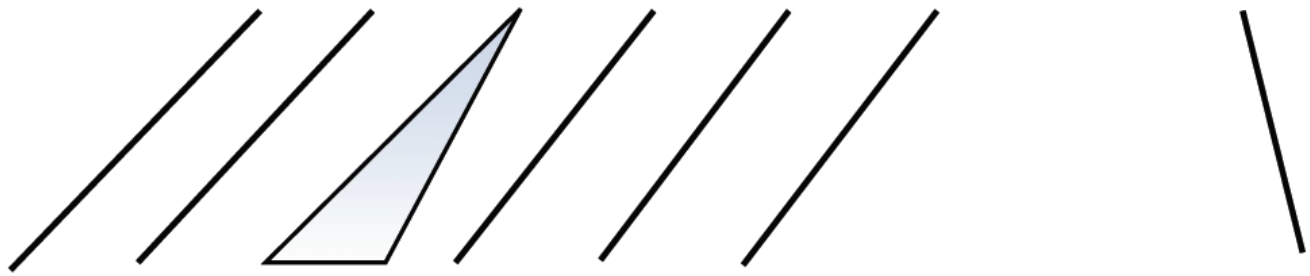
CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

S → No se ***VB*** de subir un ***NN*** en ***NNP***



$_i$ No se olvide de subir un canto rodado en Colorado $_j$

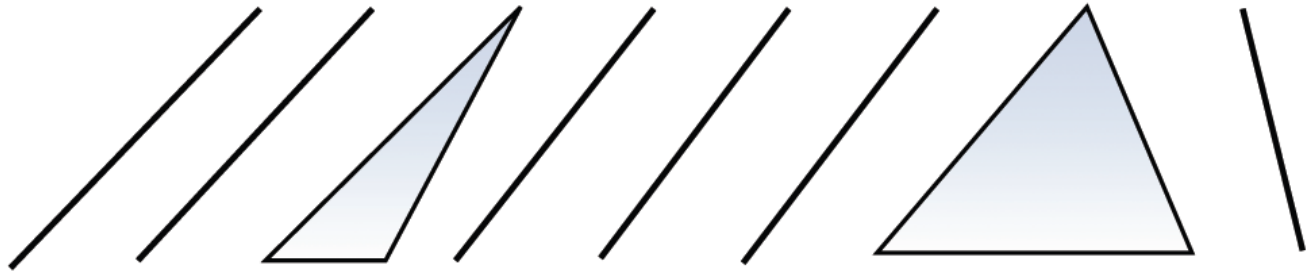
CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

S → No se ***VB*** de subir un ***NN*** en ***NNP***



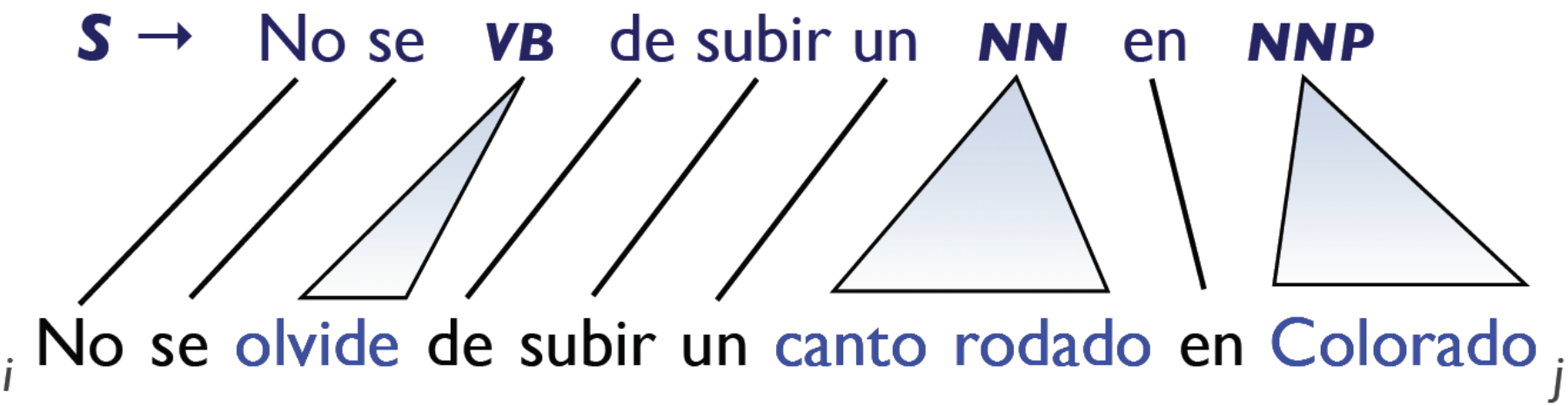
$_i$ No se olvide de subir un canto rodado en Colorado $_j$

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

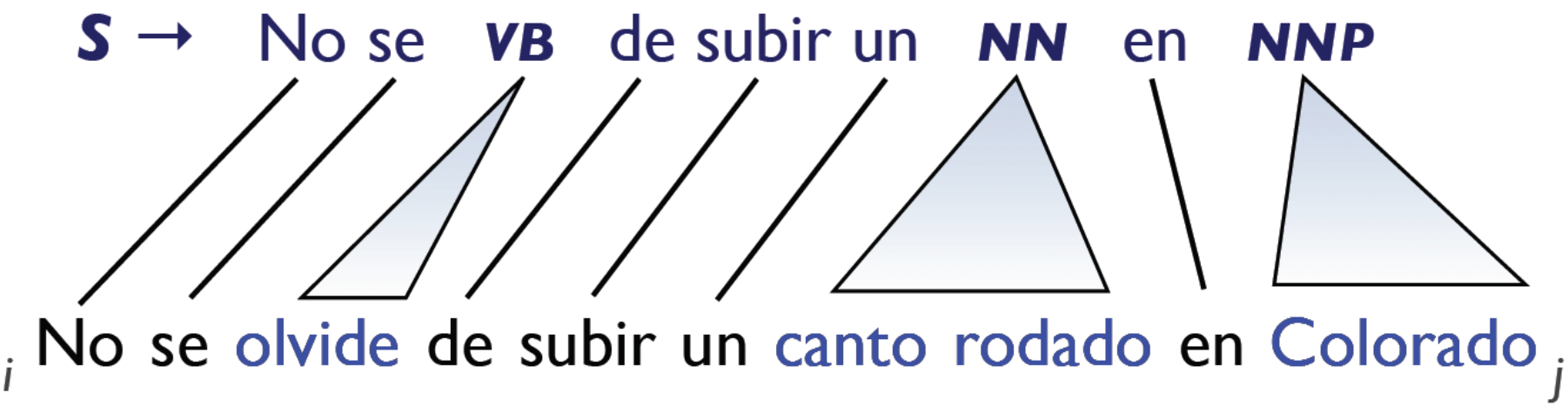


CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]



Many untransformed lexical rules can be applied in linear time

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

S → No se ***VP*** ***NP*** ***PP***

$_i$ No se olvide de subir un canto rodado en Colorado $_j$

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

S → No se ***VP*** ***NP*** ***PP***

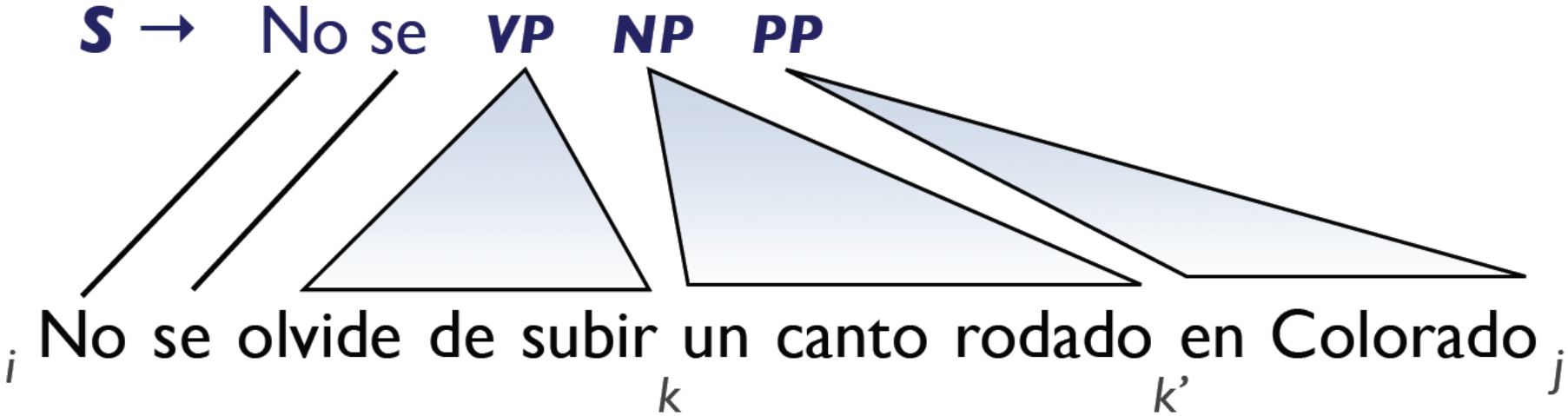
$_i$ No se olvide de subir un canto rodado en Colorado $_j$

CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]

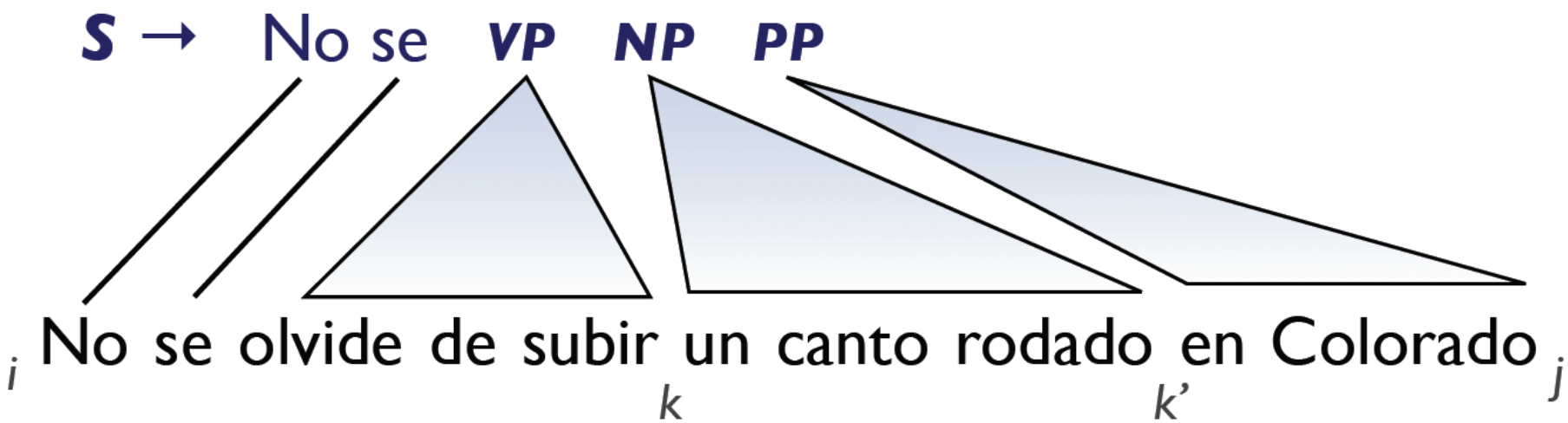


CKY-style Bottom-up Parsing

For each span length:

For each span [i,j]:

Apply all grammar rules to [i,j]



Problem: *Applying adjacent non-terminals is slow*

Eliminating Non-terminal Sequences

Lexical Normal Form (LNF)

(a) lexical rules have at most one adjacent non-terminal
(b) all unlexicalized rules are binary.

Original rule: ***S*** → No se ***VB*** ***VB*** un ***NN*** ***PP***

Transformed rules: ***S*** → No se ***VB~VB*** un ***NN~PP***

VB~VB → ***VB*** ***VB***

NN~PP → ***NN*** ***PP***

Parsing stages:

- Lexical rules are applied by matching
- Unlexicalized rules are applied by iterating over split points

Speeding up Lexical Rule Application

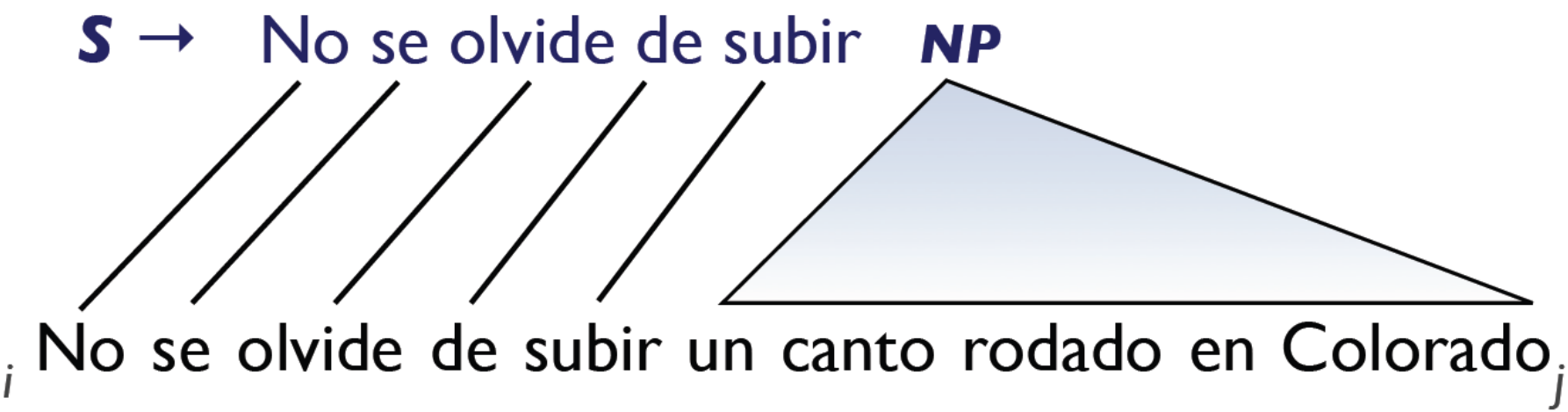
Problem: *Lexical rules can apply to many spans*

S → No se olvide de subir ***NP***

$_i$ No se olvide de subir un canto rodado en Colorado $_j$

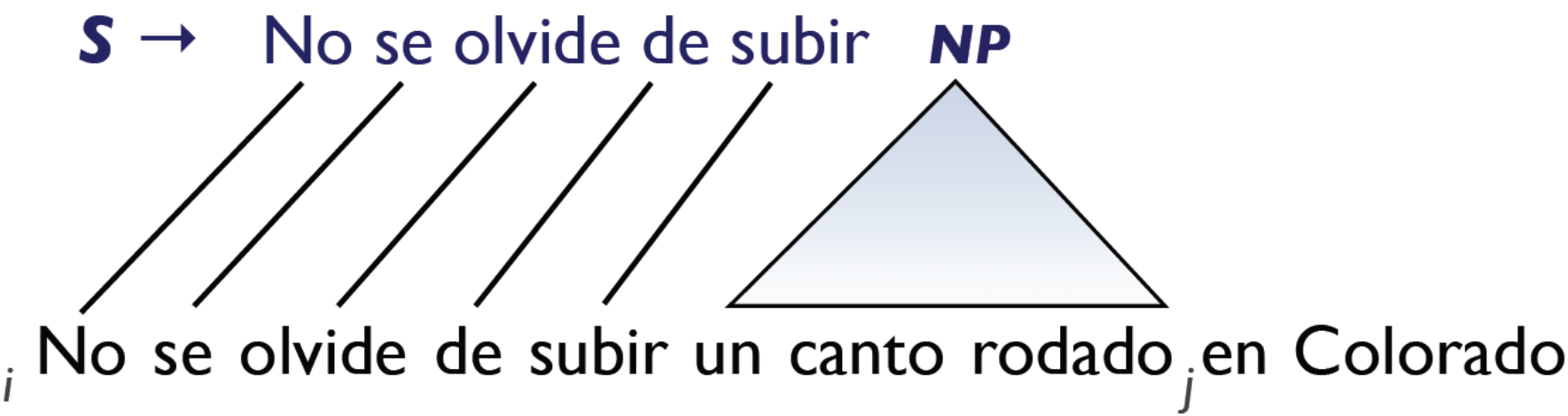
Speeding up Lexical Rule Application

Problem: *Lexical rules can apply to many spans*



Speeding up Lexical Rule Application

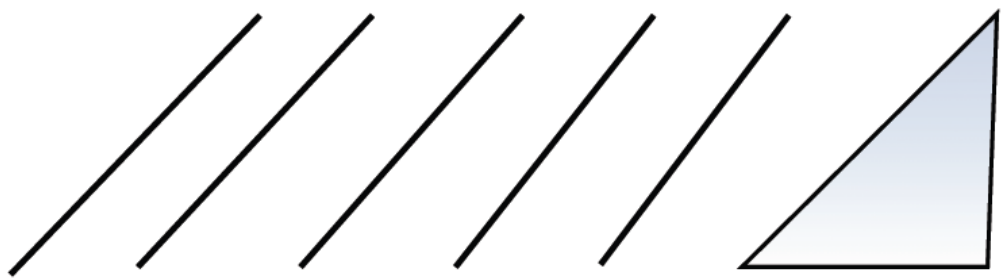
Problem: *Lexical rules can apply to many spans*



Speeding up Lexical Rule Application

Problem: *Lexical rules can apply to many spans*

S → No se olvide de subir ***NP***



$_i$ No se olvide de subir un canto $_j$ rodado en Colorado

Flexible Syntax

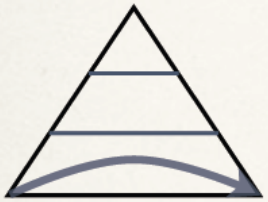
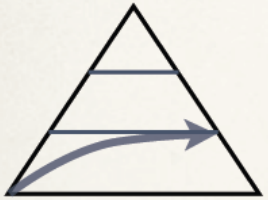
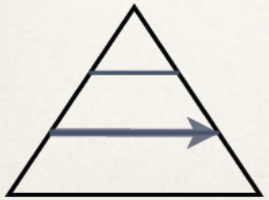
Soft Syntactic MT: From Chiang 2010

reference: An official from Japan ’s science and technology ministry said , " We are highly encouraged by Abraham ’s comment .

Hiero: Officials of the Japanese ministry of education and science , " said Abraham speeches , we are deeply encouraged by .

string-to-tree: Japan ’s ministry of education , culture , sports , science and technology , " Abraham ’s statement , which is most encouraging , " the official said .

Previous work

	string-to-string	ITG (Wu 1997)	Hiero (Chiang 2005)
	string-to-tree	Yamada & Knight 2001	Galley et al 2004/2006
	tree-to-string		Huang et al 2006 Y Liu et al 2006
	tree-to-tree	DOT (Poutsma 2000) Eisner 2003	Stat-XFER (Lavie et al 2008) M Zhang et al. 2008 Y Liu et al., 2009

Hiero Rules

$$S \rightarrow \langle S_{\boxed{1}} X_{\boxed{2}}, S_{\boxed{1}} X_{\boxed{2}} \rangle$$

$$S \rightarrow \langle X_{\boxed{1}}, X_{\boxed{1}} \rangle$$

$$X \rightarrow \langle \text{yu } X_{\boxed{1}} \text{ you } X_{\boxed{2}}, \text{have } X_{\boxed{2}} \text{ with } X_{\boxed{1}} \rangle$$

$$X \rightarrow \langle X_{\boxed{1}} \text{ de } X_{\boxed{2}}, \text{the } X_{\boxed{2}} \text{ that } X_{\boxed{1}} \rangle$$

$$X \rightarrow \langle X_{\boxed{1}} \text{ zhiyi, one of } X_{\boxed{1}} \rangle$$

$$X \rightarrow \langle \text{Aozhou, Australia} \rangle$$

$$X \rightarrow \langle \text{shi, is} \rangle$$

$$X \rightarrow \langle \text{shaoshu guojia, few countries} \rangle$$

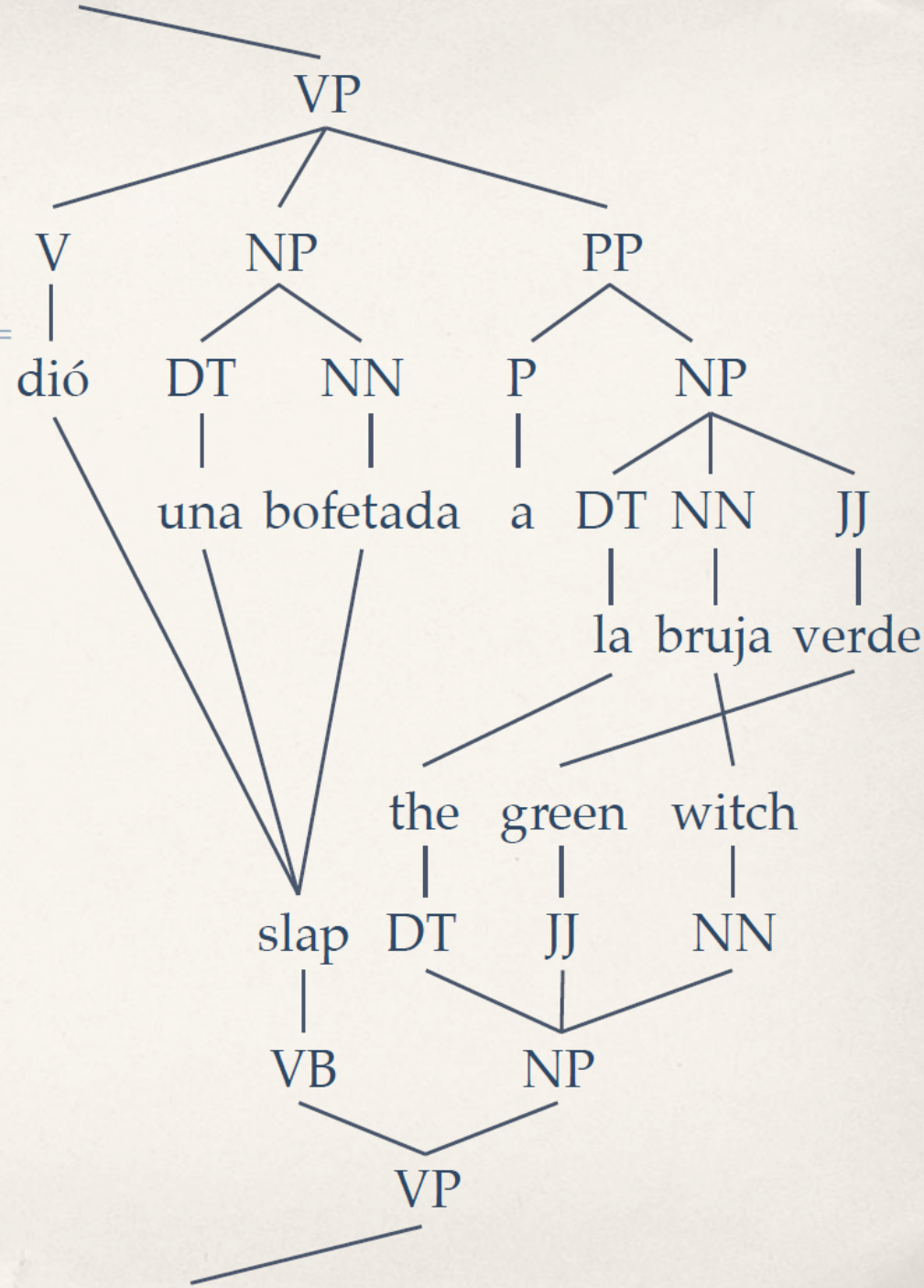
$$X \rightarrow \langle \text{bangjiao, diplomatic relations} \rangle$$

$$X \rightarrow \langle \text{Bei Han, North Korea} \rangle$$

From [Chiang et al, 2005]

STSG extraction

1. Phrases
   - respect word alignments
   - are syntactic constituents on *both* sides
2. Phrase pairs form rules
3. Subtract phrases to form rules

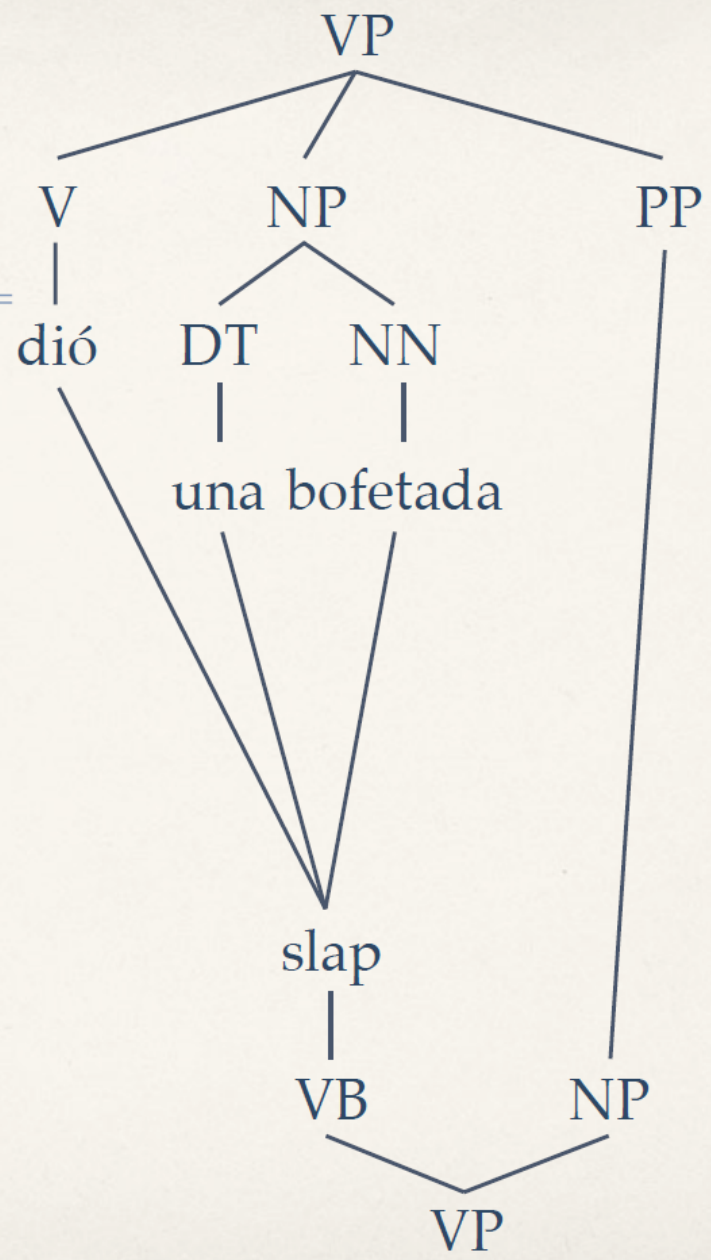


STSG extraction

1. Phrases
   - respect word alignments
   - are syntactic constituents on *both* sides
2. Phrase pairs form rules
3. Subtract phrases to form rules

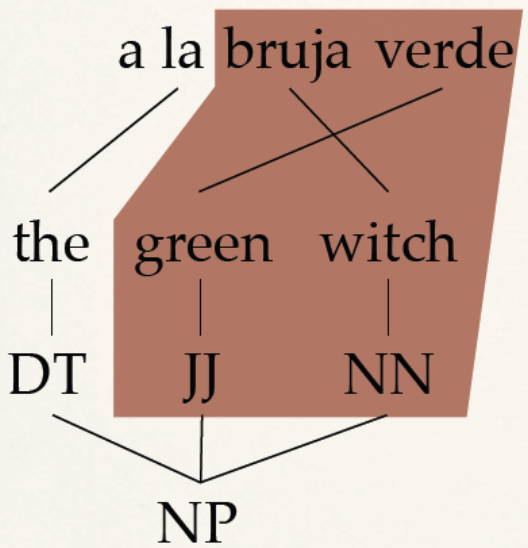
STSG extraction

1. Phrases
   - respect word alignments
   - are syntactic constituents on *both* sides
2. Phrase pairs form rules
3. Subtract phrases to form rules

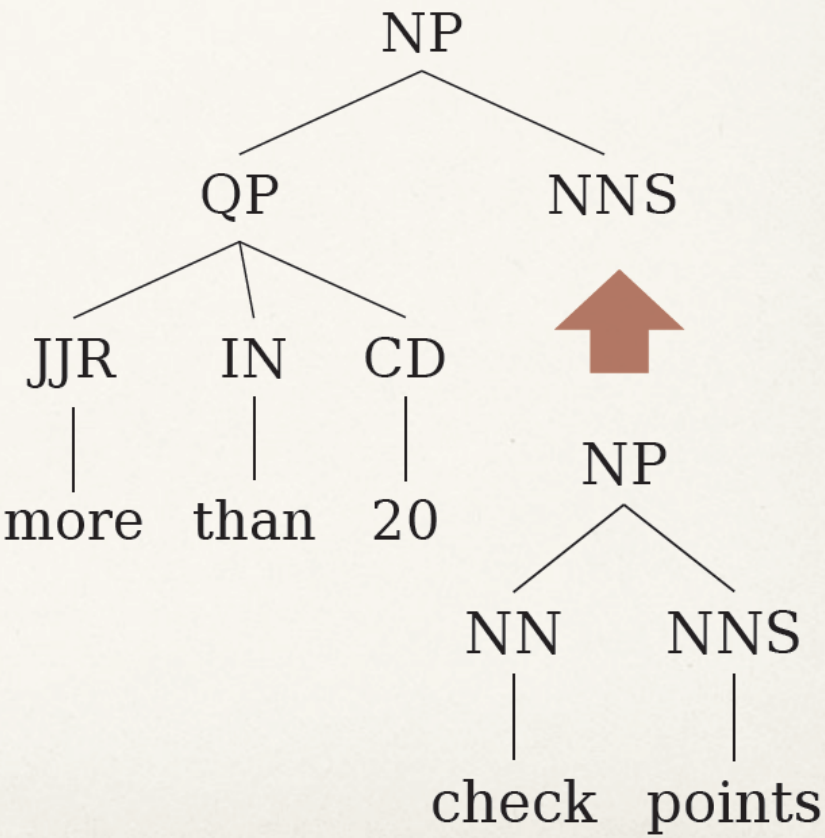


Why is tree-to-tree hard?

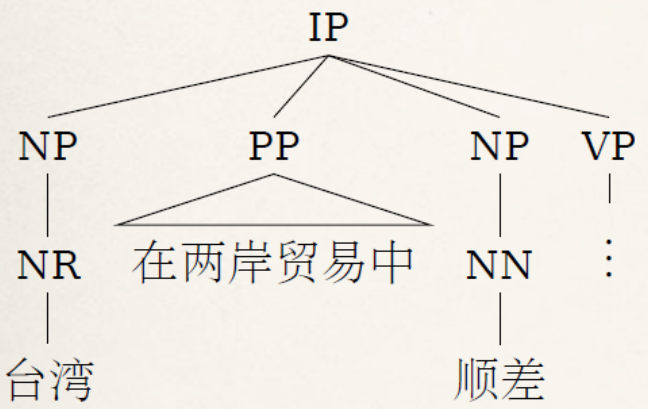
too few rules



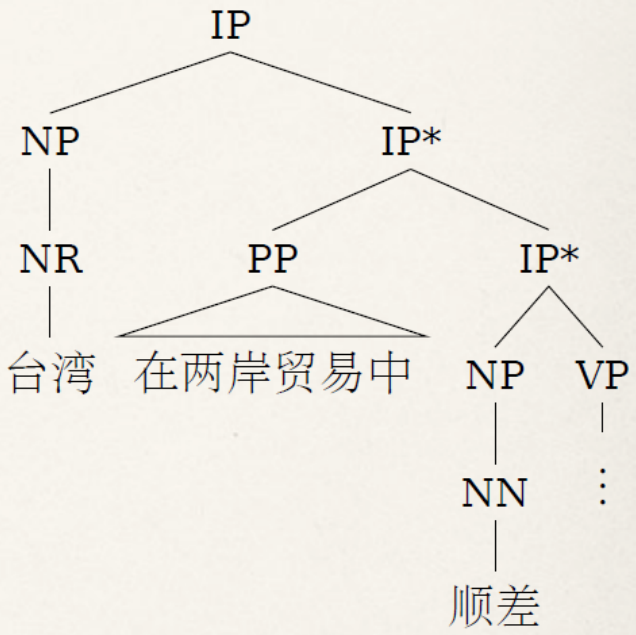
too few derivations



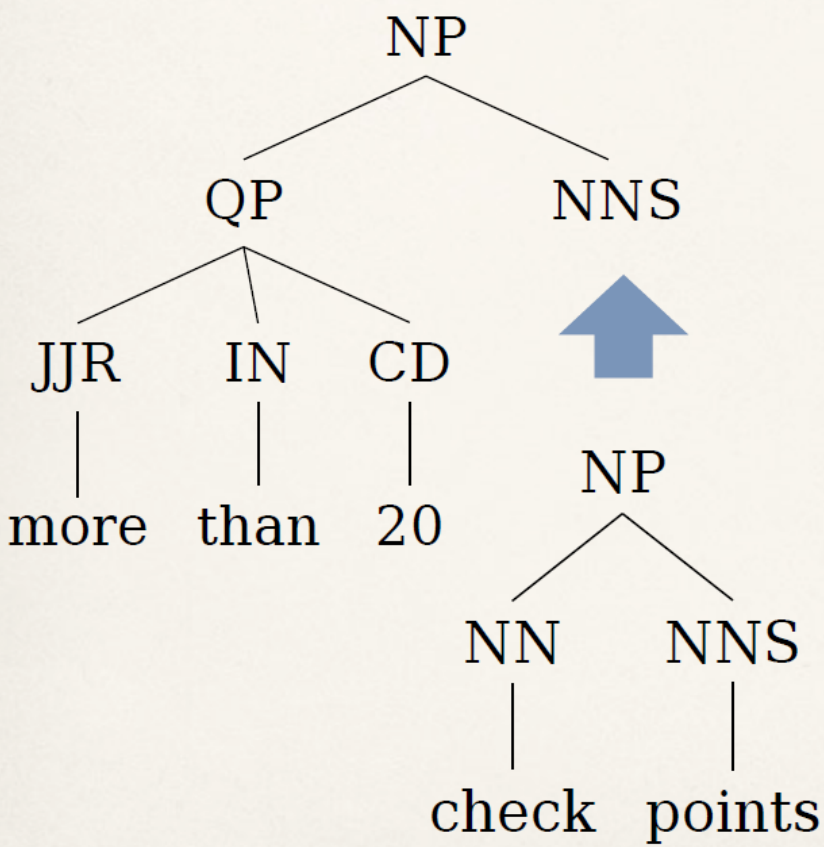
Extracting more rules



binarize head-out

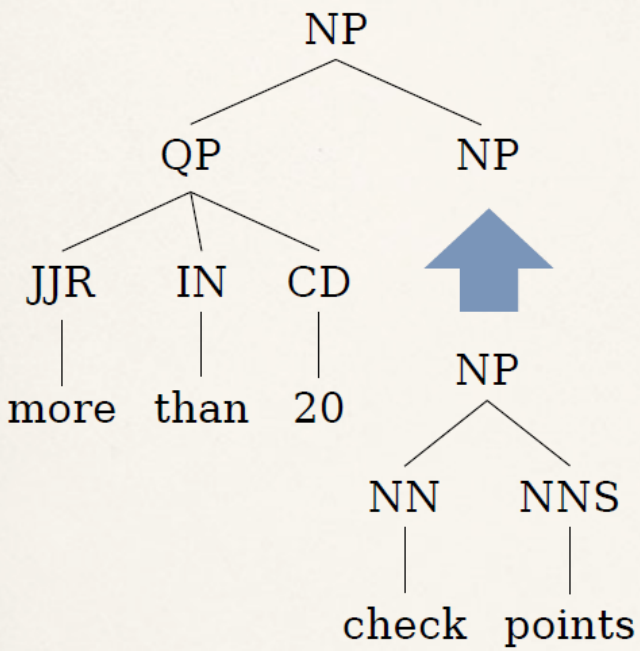


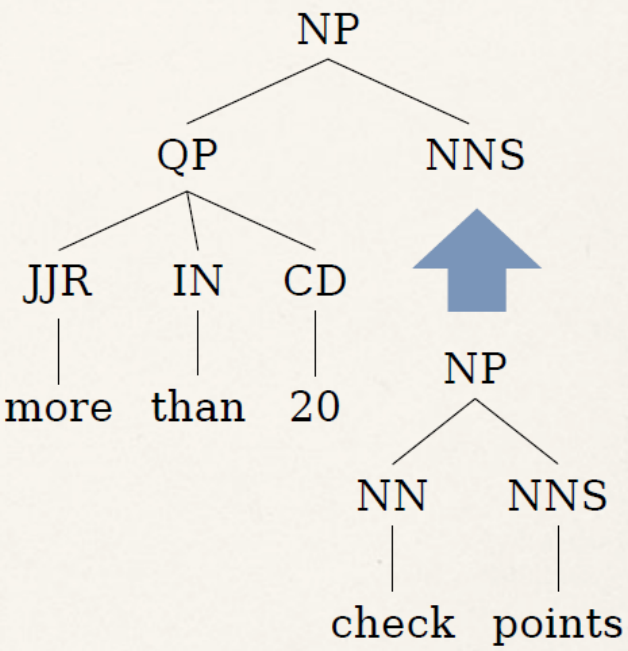
Allow more derivations



- STSG: allow only matching substitutions
- Hiero-like: allow any substitutions
- Let the model learn to choose:
  - matching substitutions
  - mismatching substitutions
  - monotone phrase-based

Allow more derivations





fire *subst*:NP→NP
fire *subst:match*

fire *subst*:NNS→NP
fire *subst:unmatch*

Allow more derivations

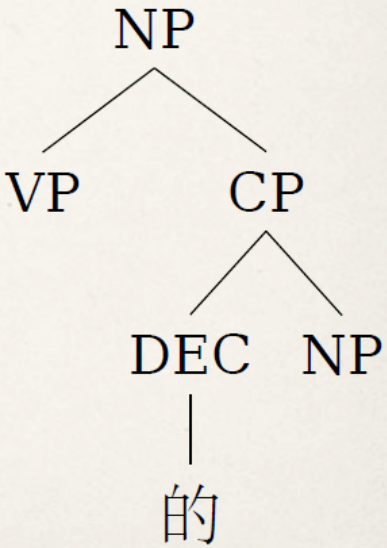
Hiero-like decoding

$$\frac{[X,i,j] \qquad [X,j+1,k]}{[X,i,k]}$$

X → X 的 X

STSG decoding

$$\frac{[VP,i,j] \qquad [NP,j+1,k]}{[NP,i,k]}$$



fuzzy STSG decoding

$$\frac{[A,i,j] \qquad [B,j+1,k]}{[NP,i,k]}$$

Results

	Chinese-English			Arabic-English		
extraction	rules	feats	BLEU	rules	feats	BLEU
Hiero	440M	1k	23.7	790M	1k	48.9
fuzzy STSG	50M	5k	23.9	38M	5k	47.5
fuzzy STSG +binarize	64M	5k	24.3	40M	6k	48.1
fuzzy STSG +SAMT	440M	160k	24.3	790M	130k	49.7

Example tree-to-tree translation

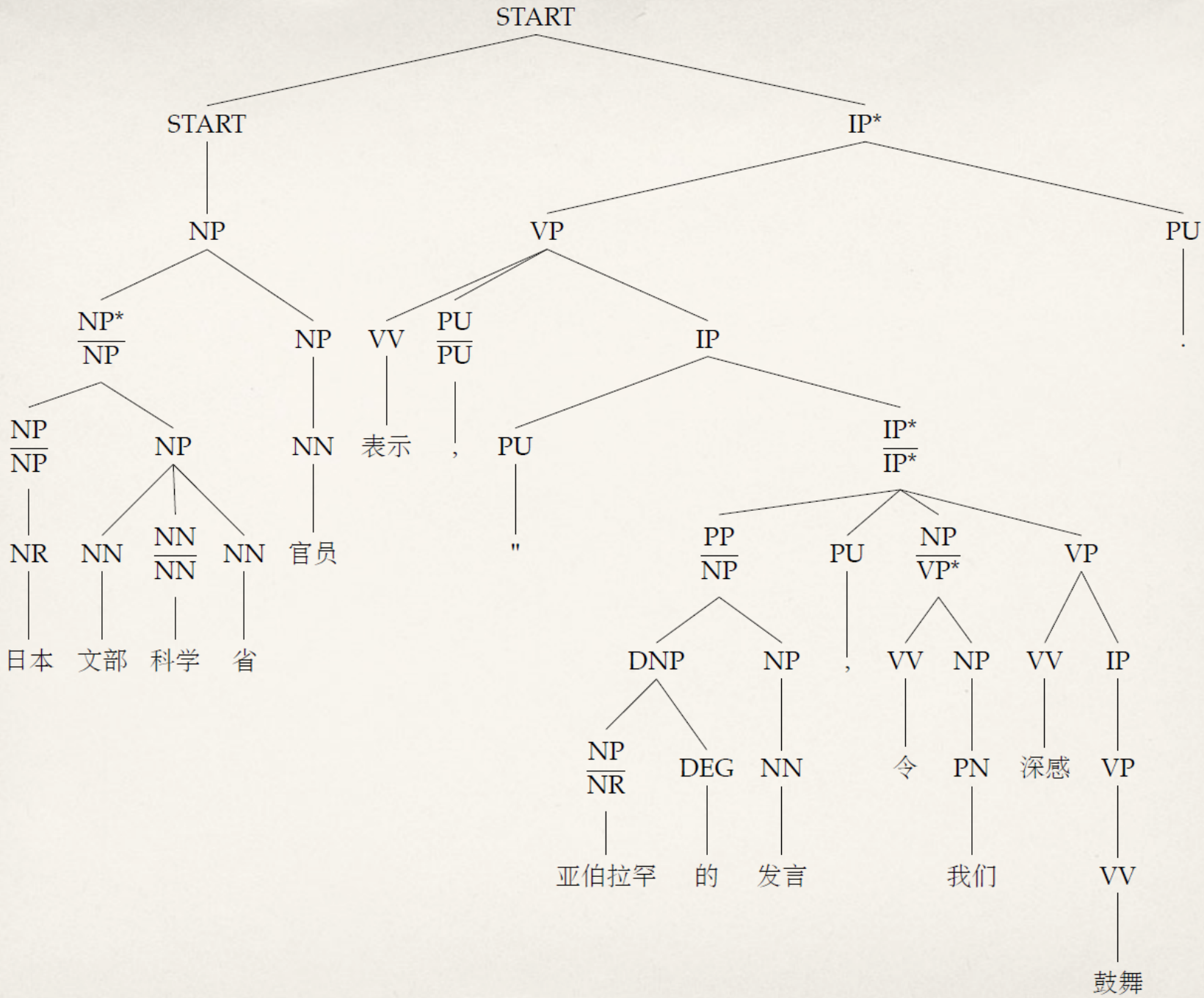
日本 文部 科学 省 官员 表示 , " 亚 伯拉罕 的 发言 , 令 我们 深感 鼓舞
Japan MEXT official said , “ Abraham ’s comment make us deeply-feel courage

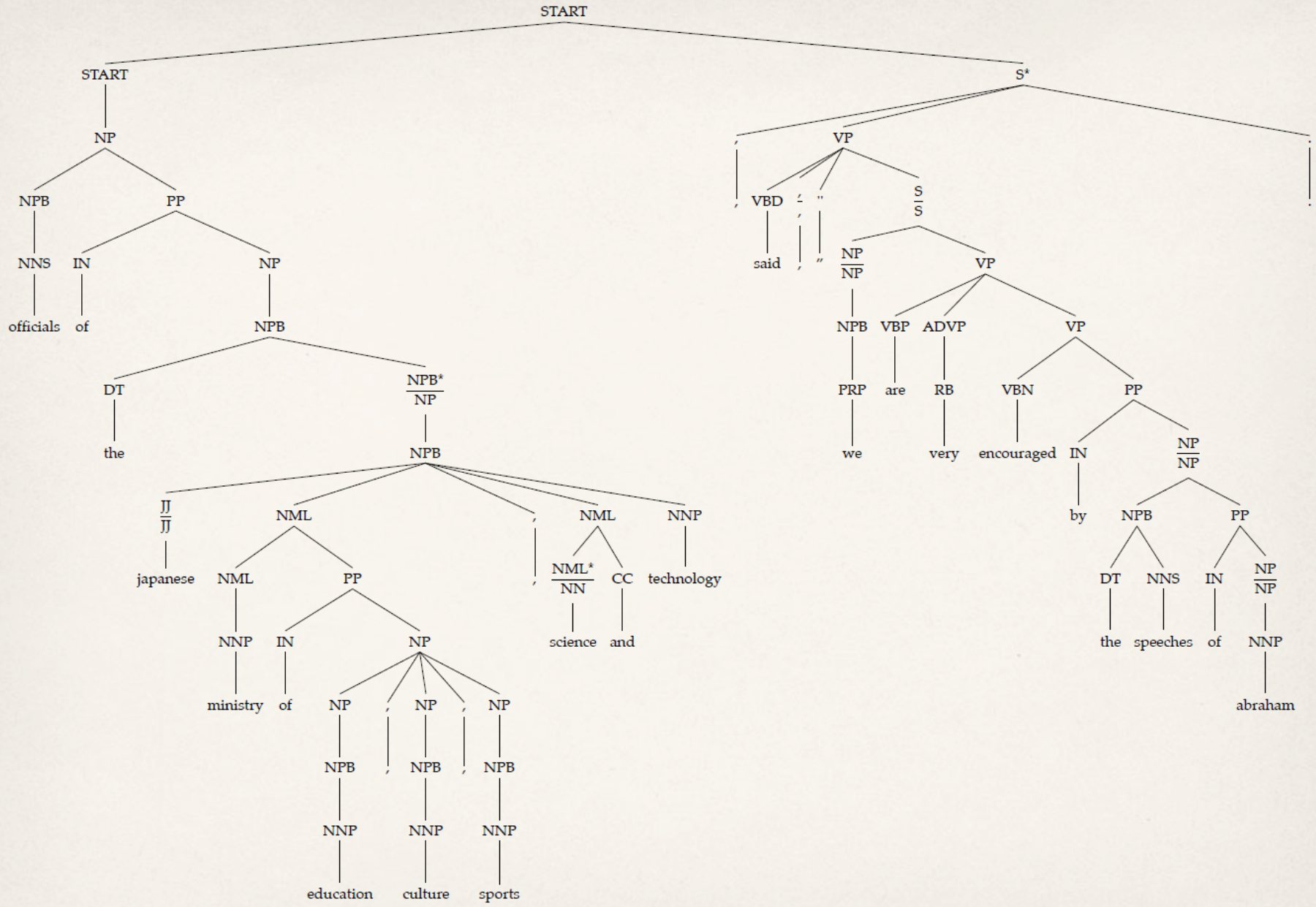
reference: An official from Japan ’s science and technology ministry said , " We are highly encouraged by Abraham ’s comment .

Hiero: Officials of the Japanese ministry of education and science , " said Abraham speeches , we are deeply encouraged by .

string-to-tree: Japan ’s ministry of education , culture , sports , science and technology , " Abraham ’s statement , which is most encouraging , " the official said .

Fuzzy STSG, binarize: Officials of the Japanese ministry of education , culture , sports , science and technology , said , " we are very encouraged by the speeches of Abraham .


START
START
IP*
NP
VP
PU
NP*/NP
NP
VV
PU/PU
IP
.
NP/NP
NP
NN
表示
,
PU
IP*/IP*
NR
NN
NN/NN
NN
官员
"
PP/NP
PU
NP/VP*
VP
日本
文部
科学
省
DNP
NP
,
VV
NP
VV
IP
NP/NR
DEG
NN
令
PN
深感
VP
亚伯拉罕
的
发言
我们
VV
鼓舞


START
START
S*
NP
NPB
PP
NNS
IN
NP
officials
of
NPB
DT
NPB*
NP
the
NPB
JJ
JJ
japanese
NML
NML
PP
NNP
IN
NP
ministry
of
NP
NPB
NNP
education
,
NP
NPB
NNP
culture
,
NP
NPB
NNP
sports
,
NML
NML*
NN
science
CC
and
NNP
technology
VP
VBD
said
S
S
NP
NP
NPB
PRP
we
VP
VBP
are
ADVP
RB
very
VP
VBN
encouraged
PP
IN
by
NP
NP
NPB
DT
the
NNS
speeches
PP
IN
of
NP
NP
NNP
abraham

Speeding up Lexical Rule Application

Anchored Lexical Normal Form (ALNF)

(a) lexical rules match a constant number of spans
(b) all unlexicalized rules are binary.

Original rule: **S** → No se **VB** **VB** un **NN** **PP**

LNF rule: **S** → No se **VB~VB** un **NN~PP**

Transformed rules: **S** → **S/NN~PP** **NN~PP**

S/NN~PP → No se **VB~VB** un

All lexical rule yields begin and end with a lexical item

Binarizing Sequences of Non-Terminals

We must select a binary derivation for each non-terminal sequence

Original:

S → VB NP NP PP

Binarization options:

S → VB~NP~NP PP

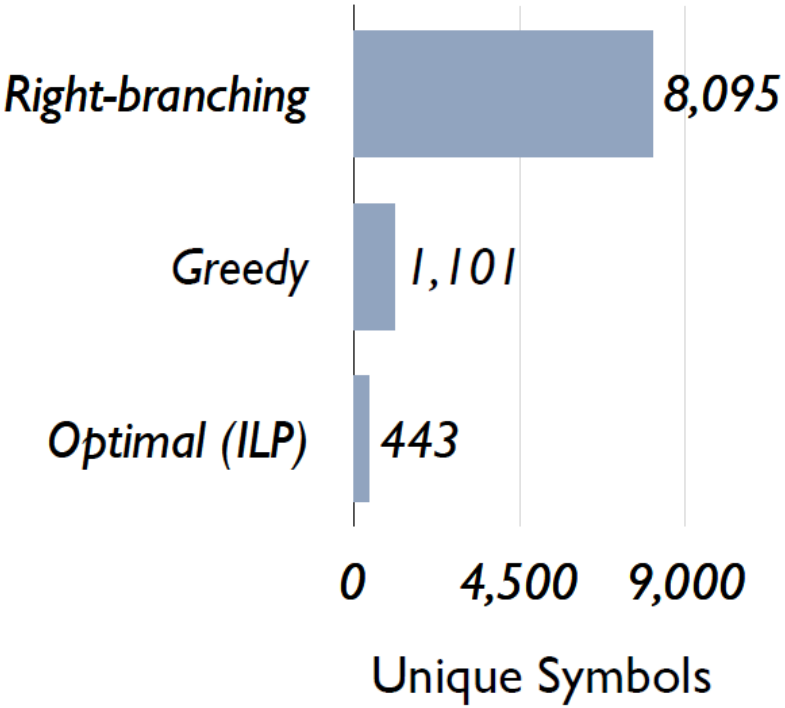
S → VB NP~NP~PP

S → VB~NP NP~PP

Objective function:

The minimum number of grammar symbols, such that all non-terminal sequences have binary derivations

Binarization of an example sentence-specific grammar



Speeding up Lexical Rule Application

Anchored Lexical Normal Form (ALNF)

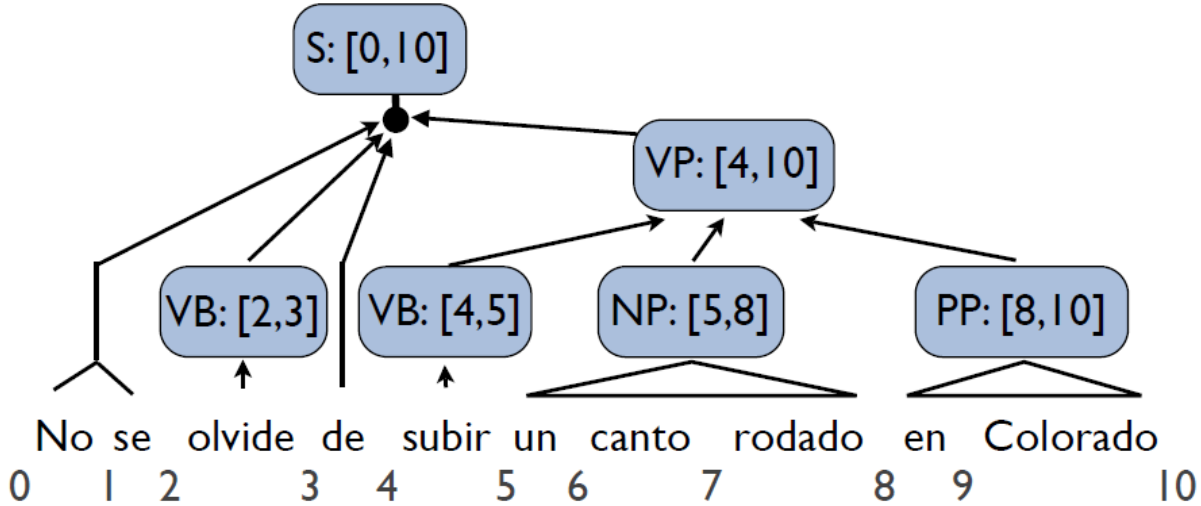
(a) lexical rules match a constant number of spans
(b) all unlexicalized rules are binary.

Original rule: ***S*** → No se ***VB*** ***VB*** un ***NN*** ***PP***

LNF rule: ***S*** → No se ***VB~VB*** un ***NN~PP***

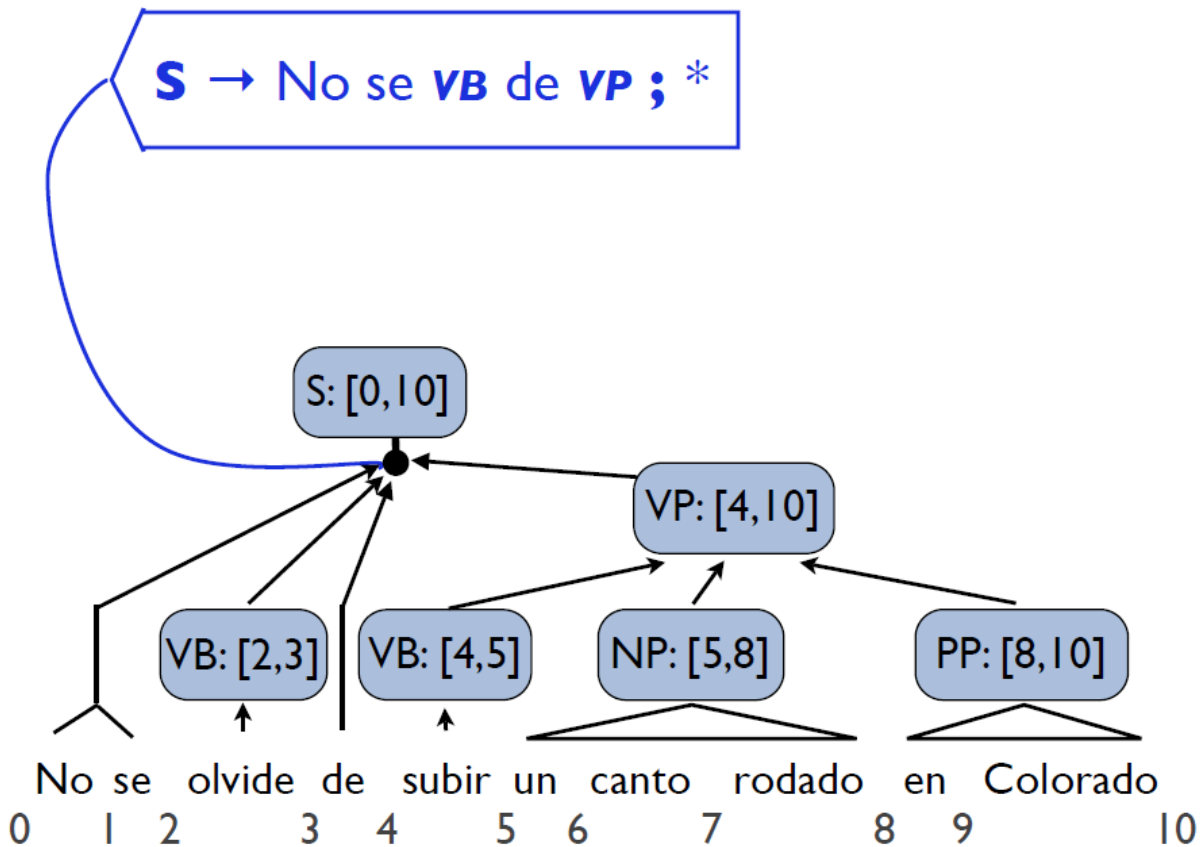
Integrating a Language Model

Approach: *Top-down lazy forest reranking with priority queues (a.k.a., cube growing)*



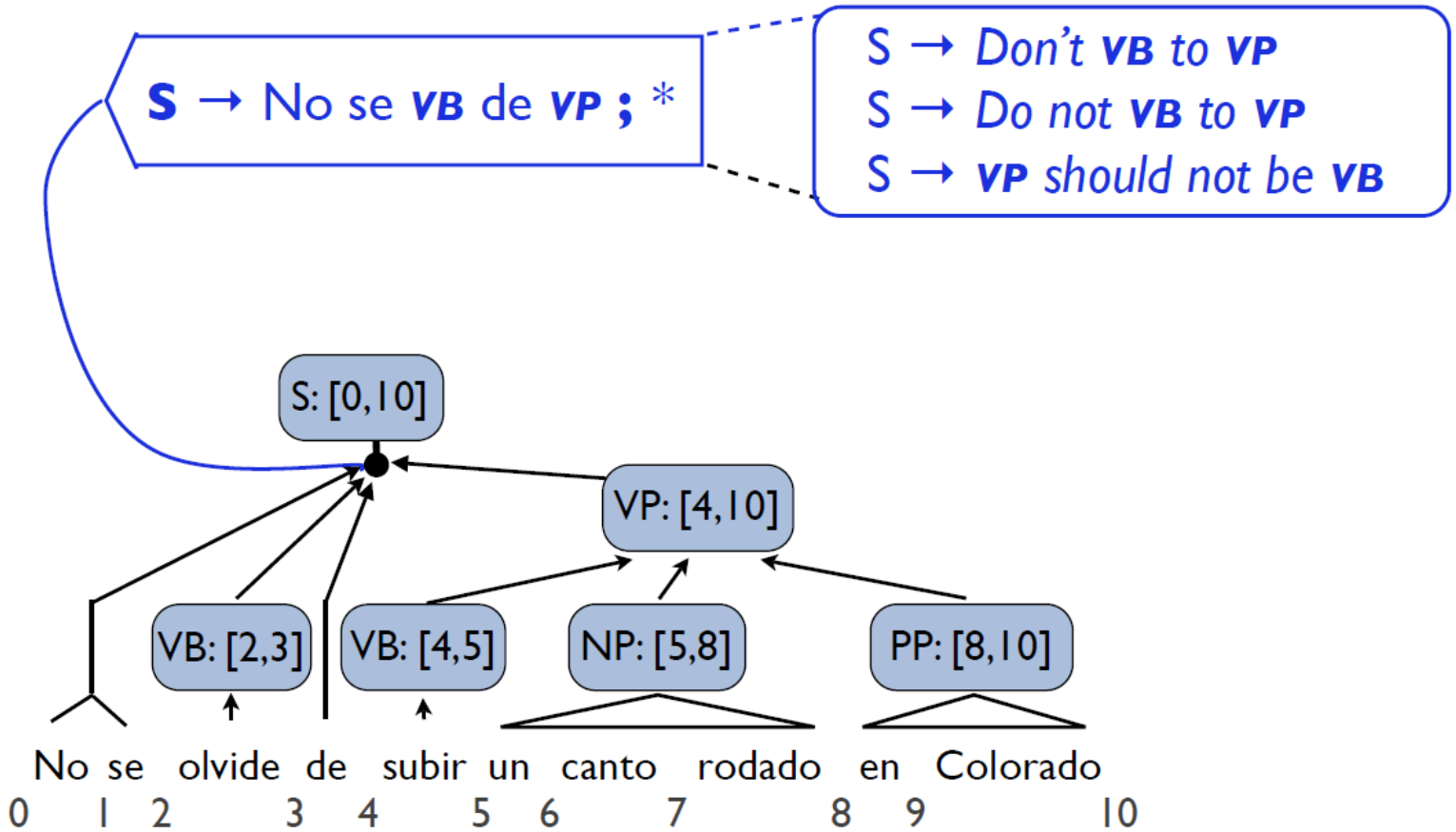
Integrating a Language Model

Approach: *Top-down lazy forest reranking with priority queues (a.k.a., cube growing)*



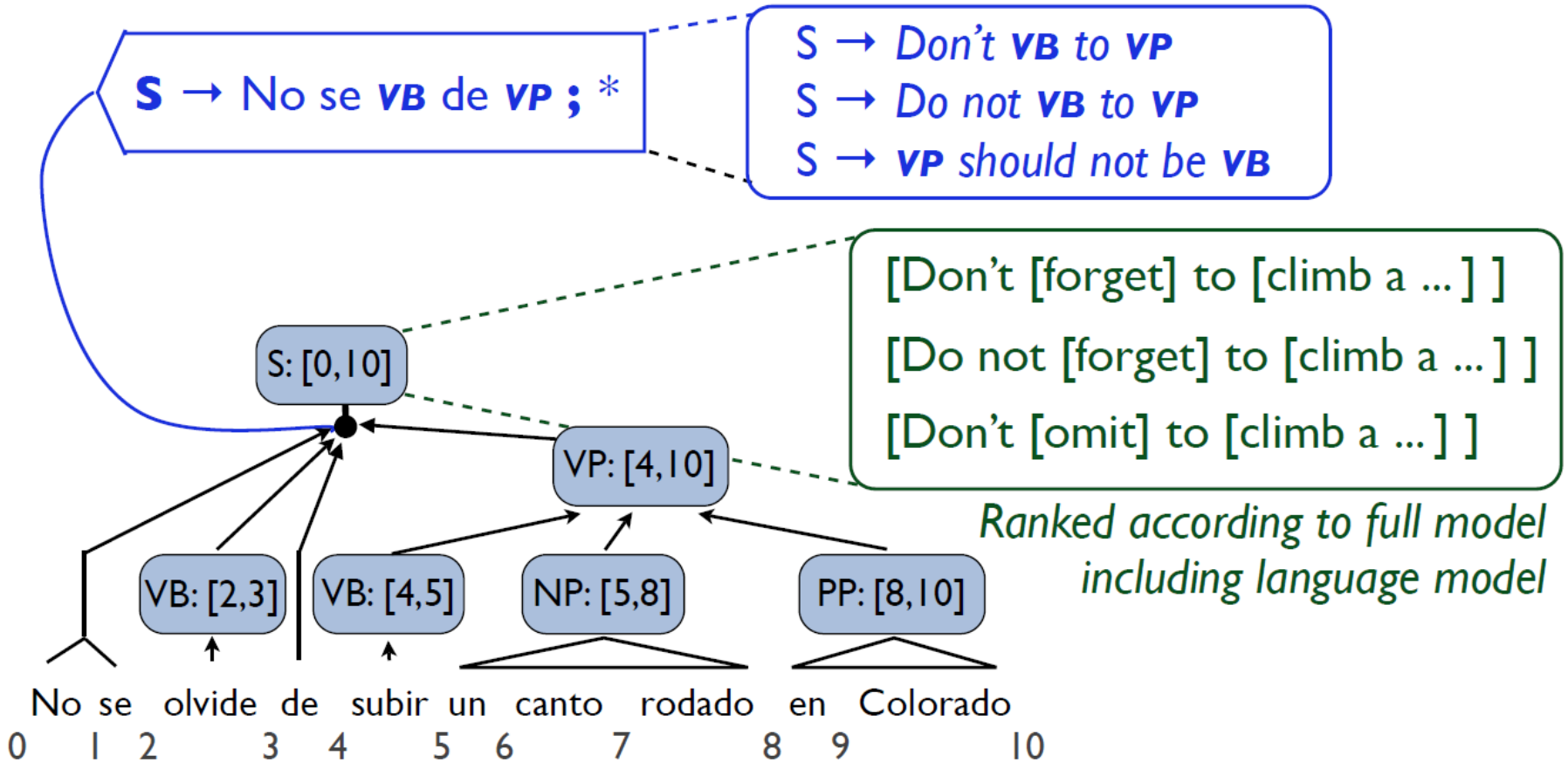
Integrating a Language Model

Approach: *Top-down lazy forest reranking with priority queues (a.k.a., cube growing)*



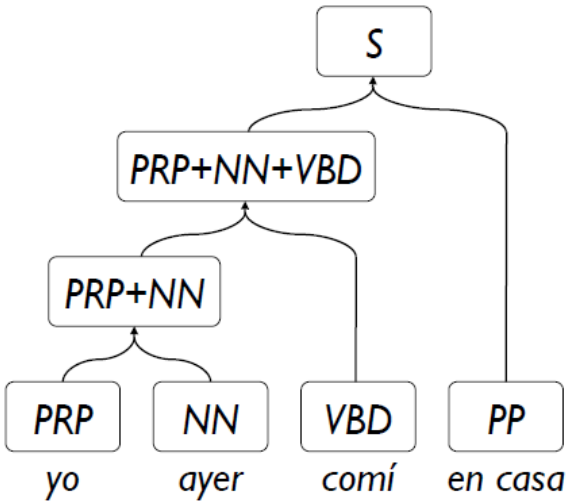
Integrating a Language Model

Approach: *Top-down lazy forest reranking with priority queues (a.k.a., cube growing)*



Rebinarizing for LM Integration (ACL '09)

Parse with ALNF grammar



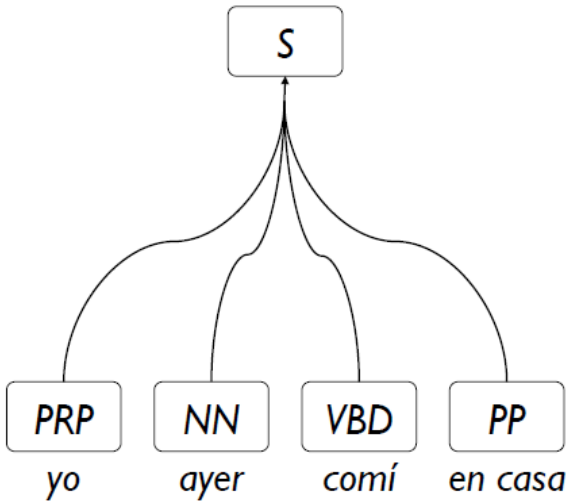
$S \rightarrow$ PRP_1 VBD_3 NN_2 PP_4

$S \rightarrow$ $[[PRP_1$ $NN_2]$ $VBD_3]$ PP_4

$S \rightarrow$ PRP_1 VBD_3 PP_4 NN_2

$S \rightarrow$ $[[PRP_1$ $NN_2]$ $VBD_3]$ PP_4

Collapse out binarization



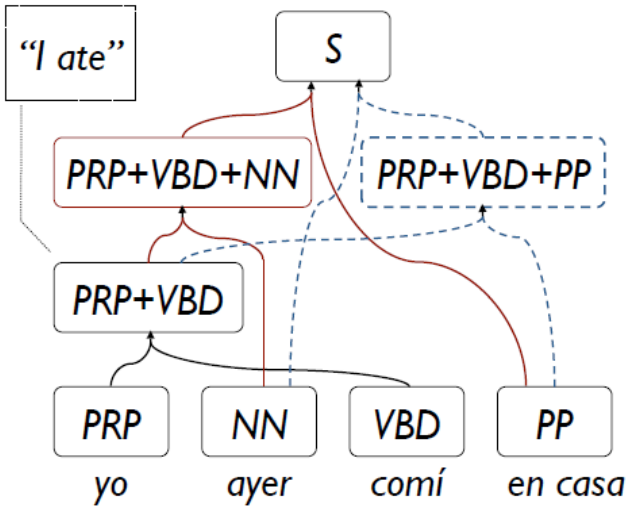
$S \rightarrow$ PRP_1 VBD_3 NN_2 PP_4

PRP_1 NN_2 VBD_3 PP_4

$S \rightarrow$ PRP_1 VBD_3 PP_4 NN_2

PRP_1 NN_2 VBD_3 PP_4

Rebinarize for LM integration



$S \rightarrow$ $[[PRP_1$ $VBD_3]$ $NN_2]$ PP_4

PRP_1 NN_2 VBD_3 PP_4

$S \rightarrow$ $[[PRP_1$ $VBD_3]$ $PP_4]$ NN_2

PRP_1 NN_2 VBD_3 PP_4

Coarse-to-Fine LM Integration

Observation: *The best translations almost always have a translation model score close to the Viterbi parse score*

Coarse-to-fine beaming: *A forest node's beam size is proportional to its posterior under the translation model*