Natural Language Processing

Machine Translation III

Dan Klein – UC Berkeley

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**

S
ADV
lo haré de muy buen grado .

S
ADV
I will do it gladly .

Grammar

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

ADV → ⟨ de muy buen grado ; gladly ⟩

Translating with Tree Transducers

Input **Output**

S
ADV
lo haré de muy buen grado .

S
ADV
I will do it gladly .

Grammar

S
NP VP .
MD VP
PRN VB PRN

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**

VP
ADV
lo haré de muy buen grado .

VP
ADV
will do it 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**

VP
ADV
lo haré de muy buen grado .

VP
ADV
will do it gladly

Grammar

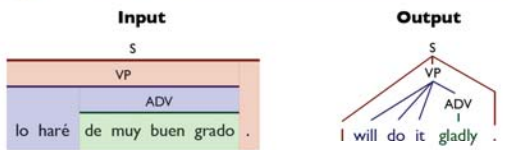
S → ⟨ VP . ; I VP . ⟩

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



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

Grammar

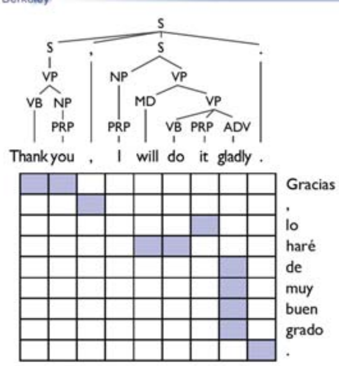
S → ⟨ VP . ; I VP . ⟩ **OR** S → ⟨ VP . ; you VP . ⟩

VP → ⟨ lo haré ADV ; will do it ADV ⟩

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

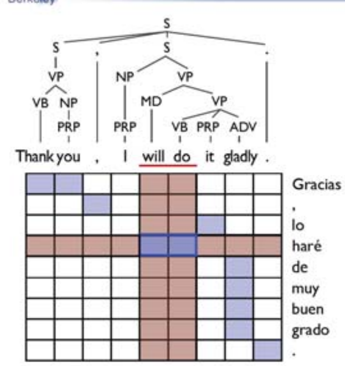
ADV → ⟨ de muy buen grado ; gladly ⟩

Learning Grammars for Translation



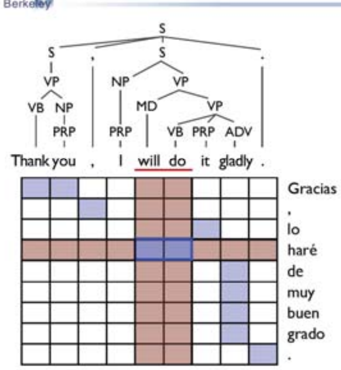
Grammar Rules

Learning Grammars for Translation



Grammar Rules

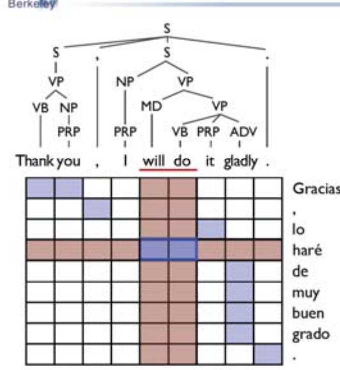
Learning Grammars for Translation



Grammar Rules

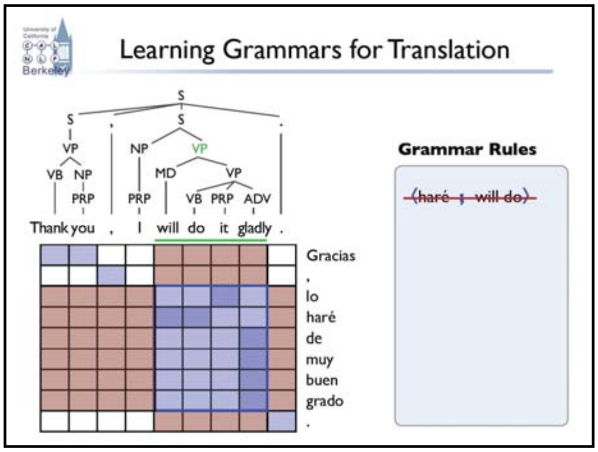
⟨haré ; will do⟩

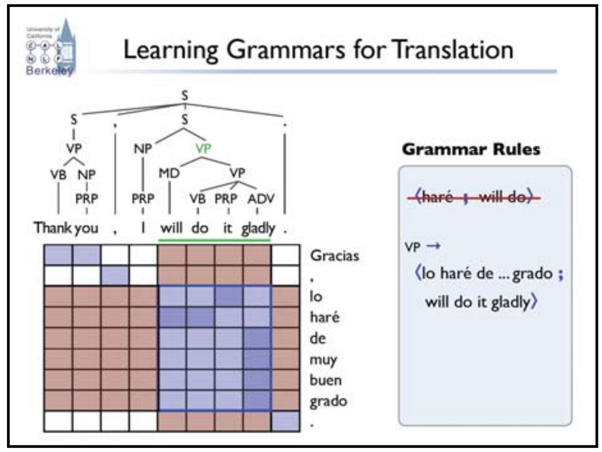
Learning Grammars for Translation

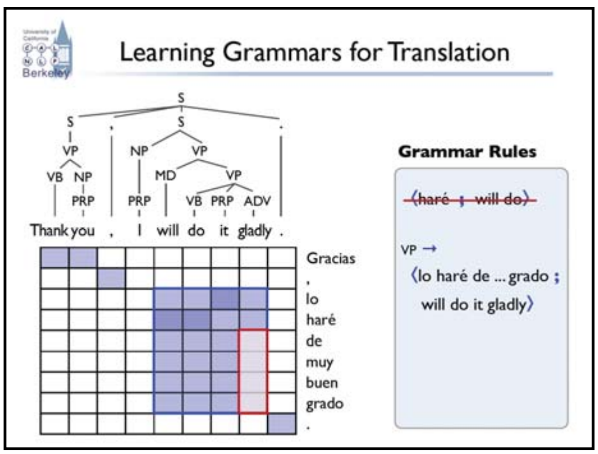


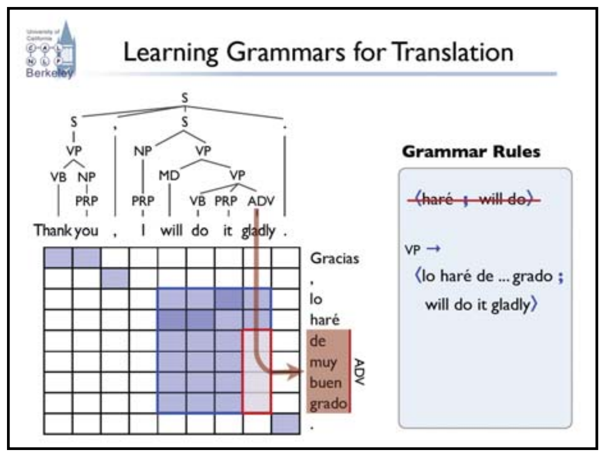
Grammar Rules

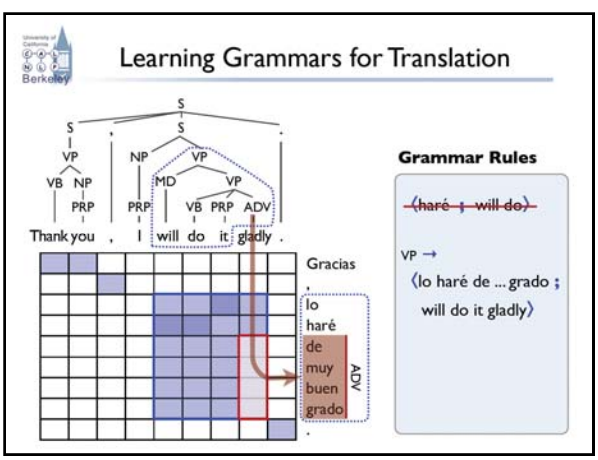
~~⟨haré ; will do⟩~~

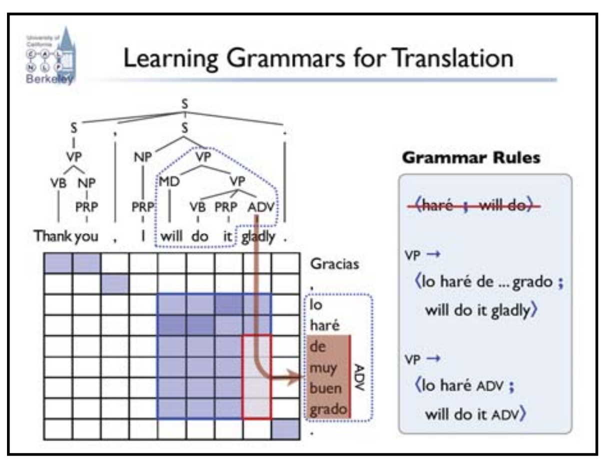

Learning Grammars for Translation
S
S , S .
VP NP VP
VB NP MD VP
PRP PRP VB PRP ADV
Thank you , I will do it gladly .
Gracias , lo haré de muy buen grado .
Grammar Rules
~~⟨haré ; will do⟩~~


Learning Grammars for Translation
S
S , S .
VP NP VP
VB NP MD VP
PRP PRP VB PRP ADV
Thank you , I will do it gladly .
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
S
S , S .
VP NP VP
VB NP MD VP
PRP PRP VB PRP ADV
Thank you , I will do it gladly .
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
S
S , S .
VP NP VP
VB NP MD VP
PRP PRP VB PRP ADV
Thank you , I will do it gladly .
Gracias , lo haré de muy buen grado .
ADV
Grammar Rules
~~⟨haré ; will do⟩~~
VP →
⟨lo haré de ... grado ;
will do it gladly⟩


Learning Grammars for Translation
S
S , S .
VP NP VP
VB NP MD VP
PRP PRP VB PRP ADV
Thank you , I will do it gladly .
Gracias , lo haré de muy buen grado .
ADV
Grammar Rules
~~⟨haré ; will do⟩~~
VP →
⟨lo haré de ... grado ;
will do it gladly⟩


Learning Grammars for Translation
S
S , S .
VP NP VP
VB NP MD VP
PRP PRP VB PRP ADV
Thank you , I will do it gladly .
Gracias , lo haré de muy buen grado .
ADV
Grammar Rules
~~⟨haré ; will do⟩~~
VP →
⟨lo haré de ... grado ;
will do it gladly⟩
VP →
⟨lo haré ADV ;
will do it ADV⟩

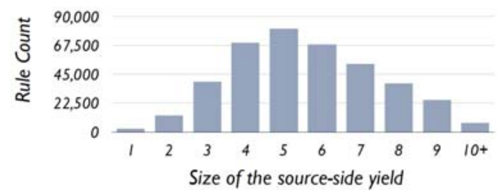
The Size of Tree Transducer Grammars

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

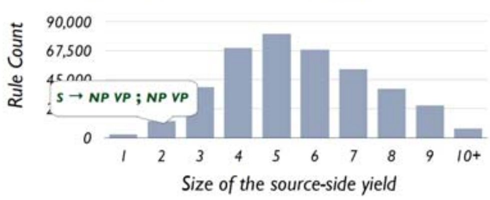
Rules matching an example 40-word sentence



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

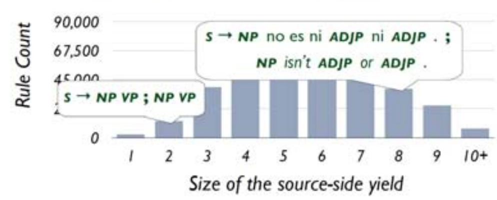
Rules matching an example 40-word sentence



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

S: No se olvide de subir un [NN: canto rodado] en [NNP: Colorado]

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

S: Don't forget to climb a [NN: boulder] in [NNP: Colorado]

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 → 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 → Y Z
Split points:	i < k < j
Operations:	O(j - i)
Time scales with:	Grammar constant

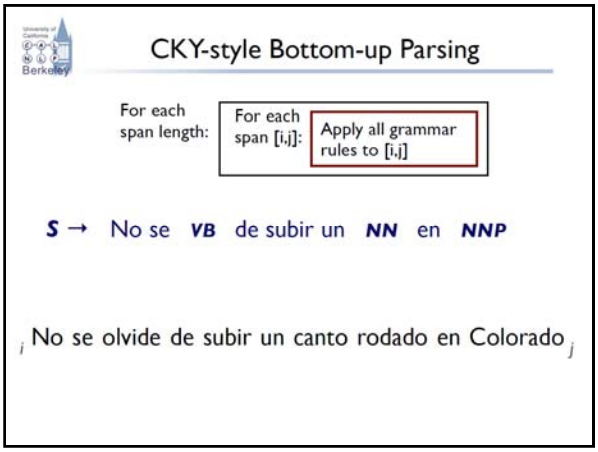
CKY-style Bottom-up Parsing

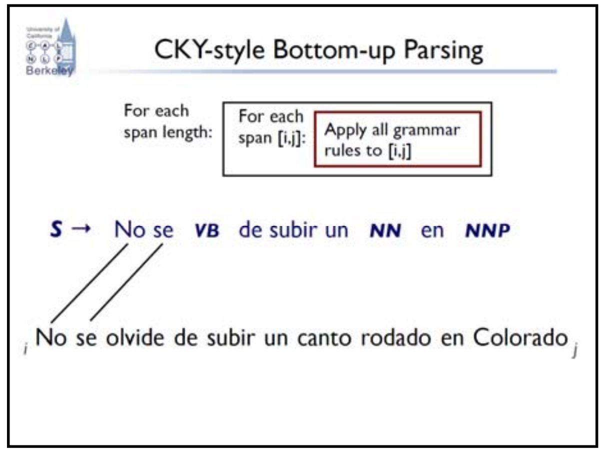
For each span length:

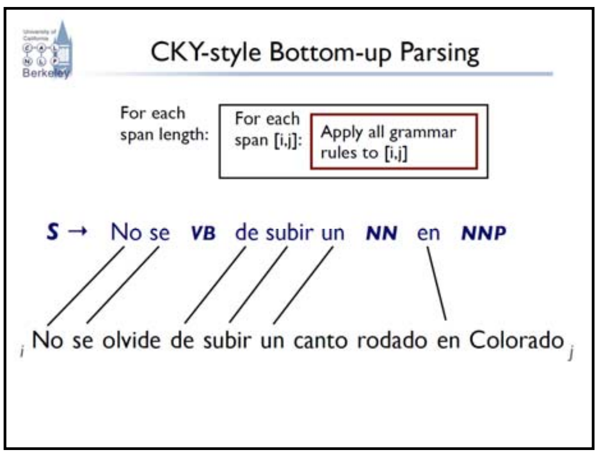
For each span [i,j]:

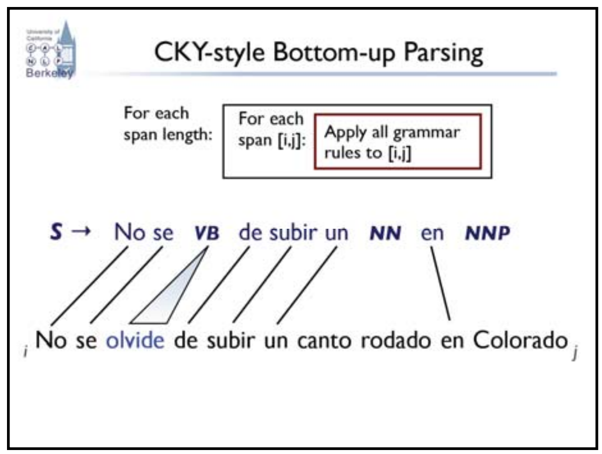
Apply all grammar rules to [i,j]

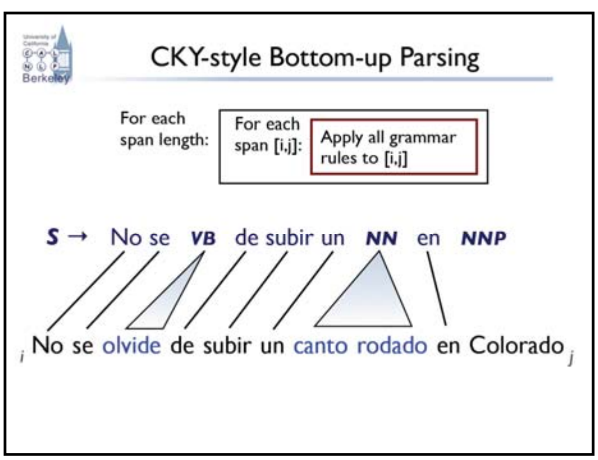
$_i$ No se olvide de subir un canto rodado en Colorado $_j$

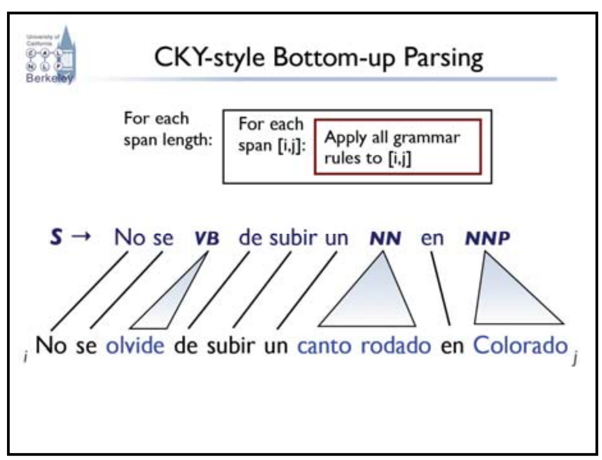

Berkeley
CKY-style Bottom-up Parsing
For each span length:
For each span [i,j]:
Apply all grammar rules to [i,j]
S → No se VB de subir un NN en NNP
i No se olvide de subir un canto rodado en Colorado j


Berkeley
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


Berkeley
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


Berkeley
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


Berkeley
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


Berkeley
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 $_j$

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]

S → No se **VP** **NP** **PP**

$_i$ No se olvide de subir $_k$ un canto rodado $_{k'}$ 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 $_k$ un canto rodado $_{k'}$ en Colorado $_j$

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

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

$S \rightarrow \langle S_{\boxed{1}} X_{\boxed{2}}, S_{\boxed{1}} X_{\boxed{2}} \rangle$
$S \rightarrow \langle X_{\boxed{1}}, X_{\boxed{1}} \rangle$
$X \rightarrow \langle \text{yu } X_{\boxed{1}} \text{ you } X_{\boxed{2}}, \text{have } X_{\boxed{2}} \text{ with } X_{\boxed{1}} \rangle$
$X \rightarrow \langle X_{\boxed{1}} \text{ de } X_{\boxed{2}}, \text{the } X_{\boxed{2}} \text{ that } X_{\boxed{1}} \rangle$
$X \rightarrow \langle X_{\boxed{1}} \text{ zhiyi, one of } X_{\boxed{1}} \rangle$
$X \rightarrow \langle \text{Aozhou, Australia} \rangle$
$X \rightarrow \langle \text{shi, is} \rangle$
$X \rightarrow \langle \text{shaoshu guojia, few countries} \rangle$
$X \rightarrow \langle \text{bangjiao, diplomatic relations} \rangle$
$X \rightarrow \langle \text{Bei Han, North Korea} \rangle$

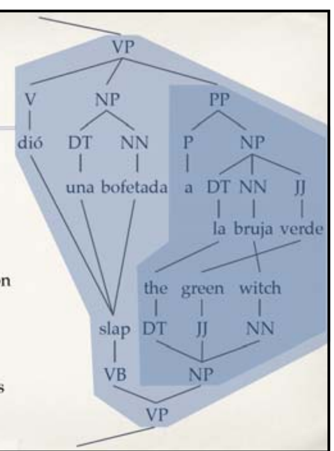
From [Chiang et al, 2005]

STSG extraction

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

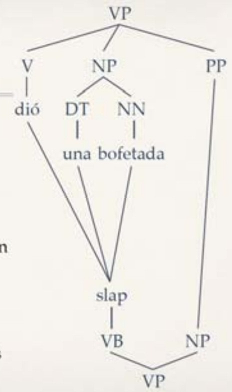
STSG extraction

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



STSG extraction

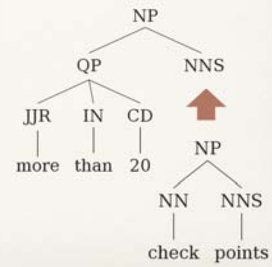
1. Phrases
   - respect word alignments
   - are syntactic constituents on *both* sides
2. Phrase pairs form rules
3. Subtract phrases to form rules



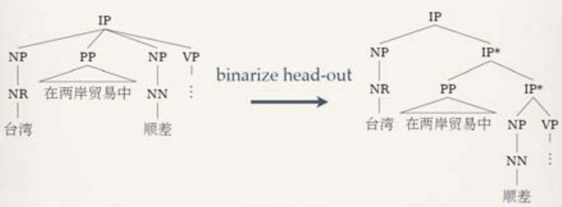
Why is tree-to-tree hard?

too few rules

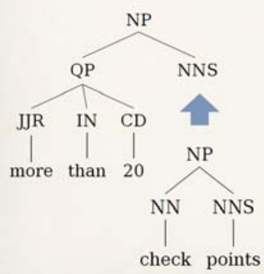
too few derivations



Extracting more rules

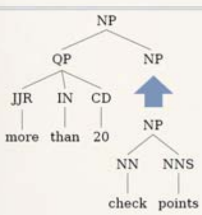


Allow more derivations

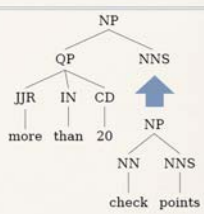


- STSG: allow only matching substitutions
- Hiero-like: allow any substitutions
- Let the model learn to choose:
  - matching substitutions
  - mismatching substitutions
  - monotone phrase-based

Allow more derivations



fire *subst*:NP→NP
fire *subst*:*match*



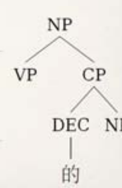
fire *subst*:NNS→NP
fire *subst*:*unmatch*

Allow more derivations

Hiero-like decoding $\dfrac{[X,i,j] \quad [X,j+1,k]}{[X,i,k]}$ X → X 的 X

STSG decoding $\dfrac{[VP,i,j] \quad [NP,j+1,k]}{[NP,i,k]}$

fuzzy STSG decoding $\dfrac{[A,i,j] \quad [B,j+1,k]}{[NP,i,k]}$



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

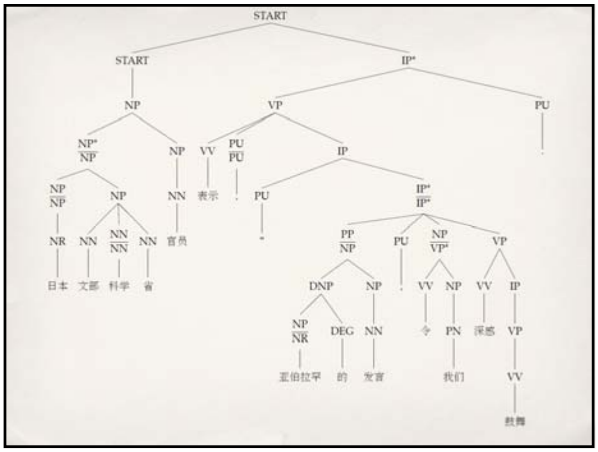
日本 文部 科学 省 官员 表示 , " 亚 伯拉罕 的 发言 , 令 我们 深感 鼓舞
Japan MEXT official said , " Abraham 's comment make us deeply-feel courage

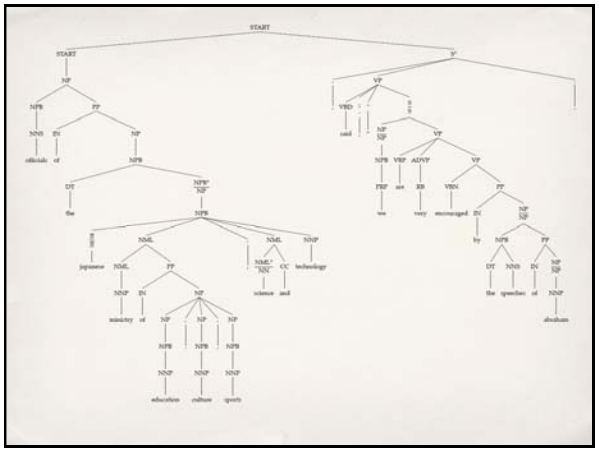
reference: An official from Japan 's science and technology ministry said , " We are highly encouraged by Abraham 's comment .

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 .





Exploiting GPUs

Lots to Parse

WIKIPEDIA
The Free Encyclopedia

≈2.6 billion words

Lots to Parse

WIKIPEDIA
The Free Encyclopedia

≈6 months (CPU)

Lots to Parse

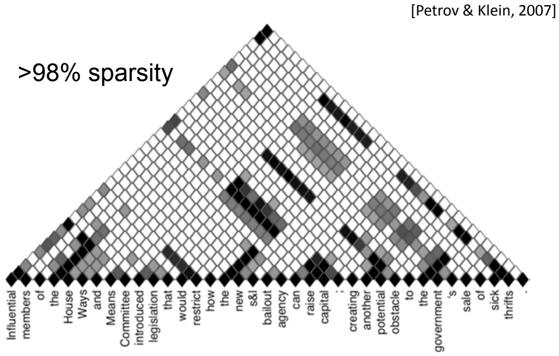
WIKIPEDIA
The Free Encyclopedia

≈3.6 days (GPU)

CPU Parsing

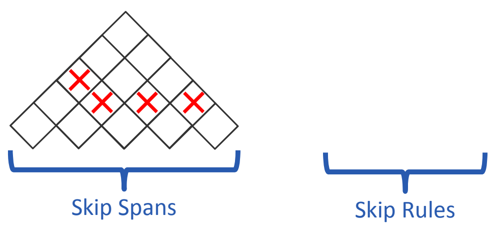
[Petrov & Klein, 2007]

>98% sparsity



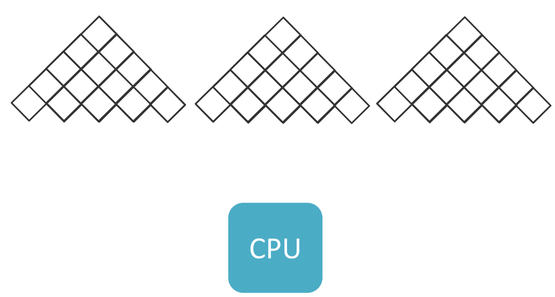
Slide credit: Slav Petrov

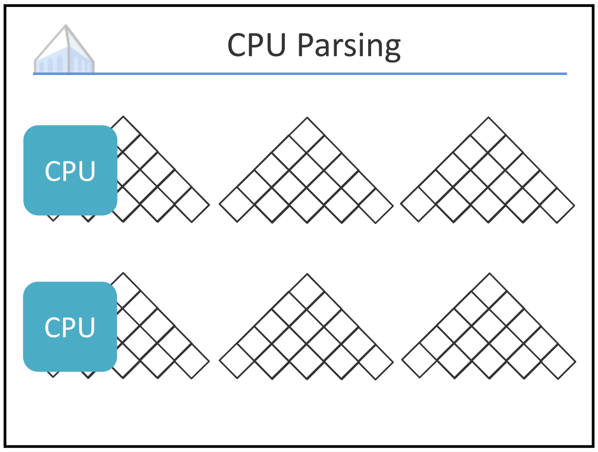
CPU Parsing

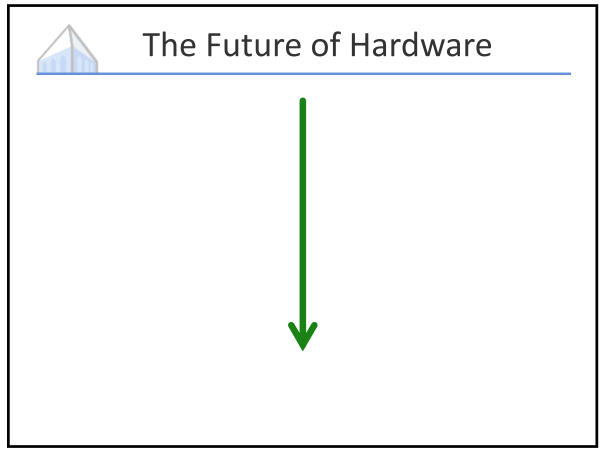


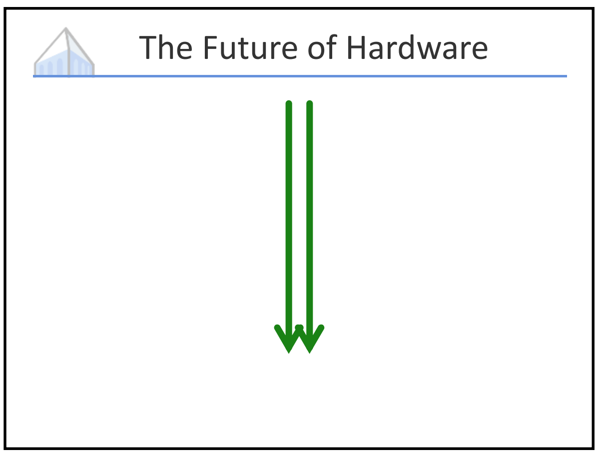
CPU Parsing

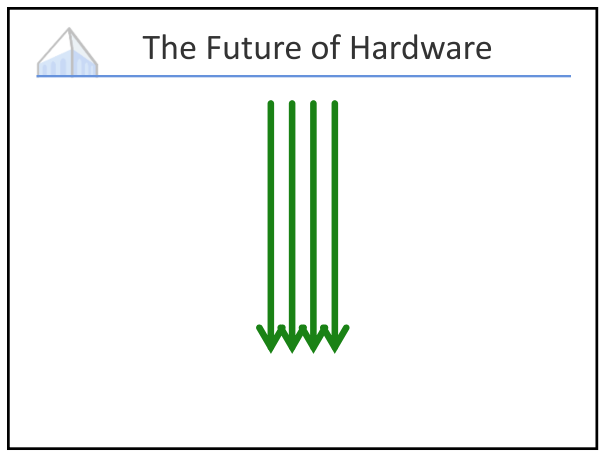
CPU Parsing

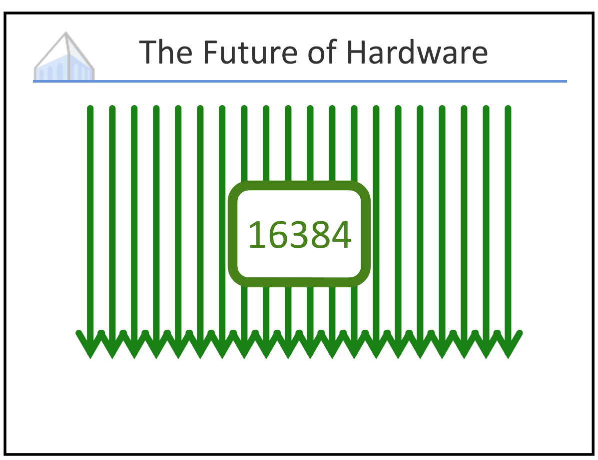


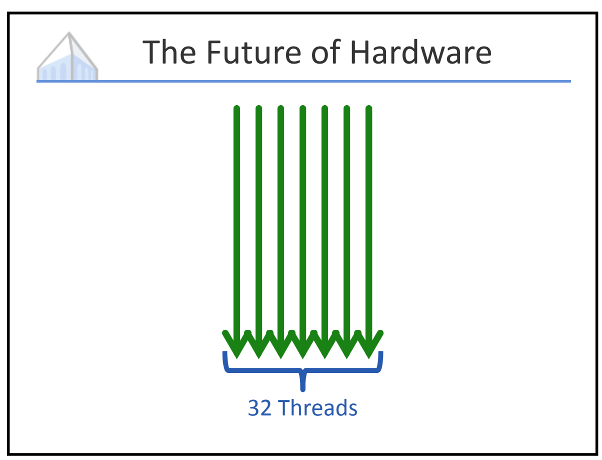

CPU Parsing
CPU
CPU


The Future of Hardware


The Future of Hardware

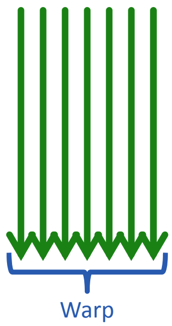

The Future of Hardware


The Future of Hardware
16384

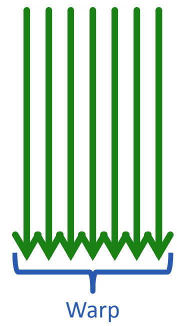

The Future of Hardware
32 Threads

The Future of Hardware

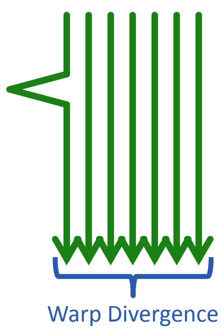
```
add.s32   %r1, %r631, %r0;
ld.global.f32   %f81, [%r1];
ld.global.f32   %f82, [%r34];
mul.ftz.f32   %f94, %f82, %f81;
mov.f32   %f95, 0f3E002E23;
mov.f32   %f96, 0f00000000;
mad.f32   %f93, %f94, %f95, %f96;
shl.b32   %r2, %r646, 8;
add.s32   %r3, %r658, %r2;
shl.b32   %r4, %r3, 2;
add.s32   %r5, %r631, %r4;
mul.lo.s32  %r6, %r646, 588;
shl.b32   %r7, %r6, 1;
add.s32   %r8, %r5, %r7;
ld.global.f32   %f83, [%r8];
mul.ftz.f32   %f98, %f82, %f83;
```



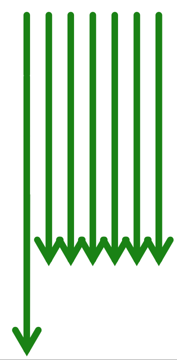
Warps



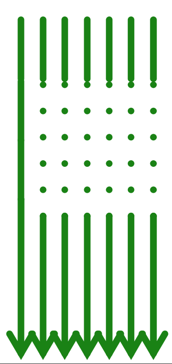
Warps



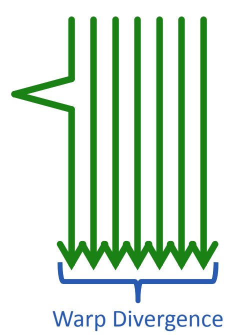
Warps



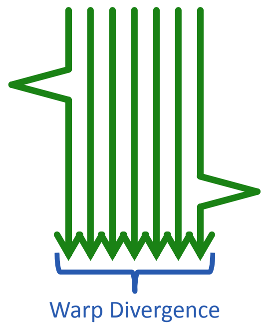
Warps



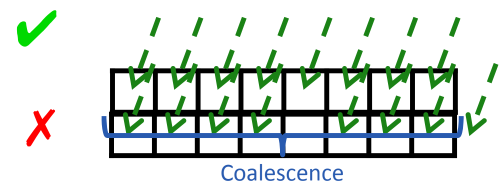
Warps



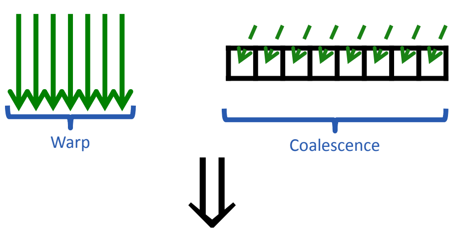
Warps



Warps

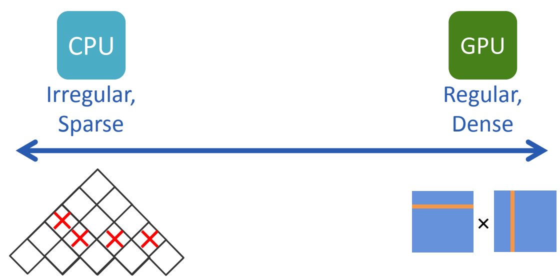


Designing GPU Algorithms

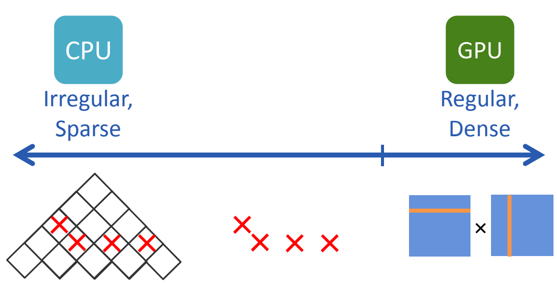


Dense, Uniform Computation

Designing GPU Algorithms

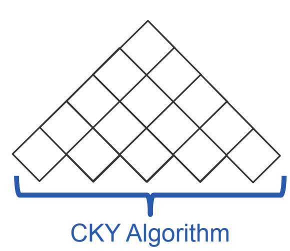


Designing GPU Algorithms



[Canny, Hall, and Klein, 2013]

Designing GPU Algorithms



CKY Parsing

```
for each sentence:
  for each span (begin, end):
    for each split:
      for each rule (P -> L R):
        score[begin, end, P]
         += ruleScore[P -> L R]
           * score[begin, split, L]
           * score[split, end, R]
```

Item Queue

Grammar Application

CKY Parsing

```
for each sentence:
  for each span (begin, end):
    for each split:

     applyGrammar(begin, split, end)
```

Item Queue

Grammar Application

CKY Parsing

```
for each parse item in sentence:

     applyGrammar(item)
```

Item Queue

Grammar Application

CKY Parsing

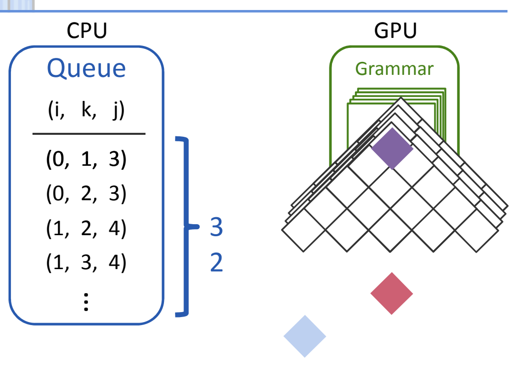
```
for each parse item in sentence:

     applyGrammar(item)
```

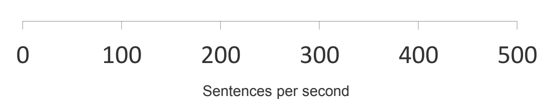
CPU

GPU

GPU Parsing Pipeline

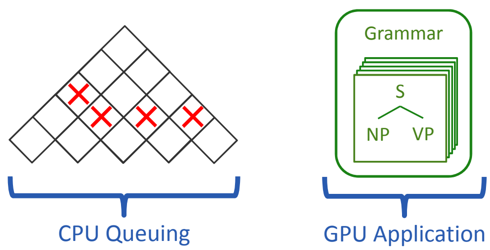


Parsing Speed

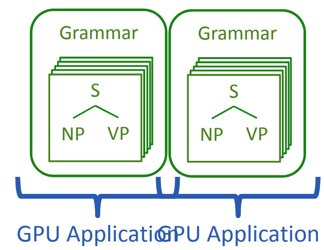


[Canny, Hall, and Klein, 2013]

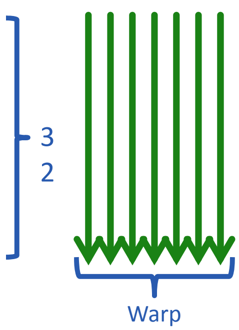
Exploiting Sparsity



Exploiting Sparsity



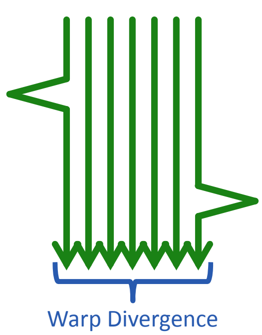
Exploiting Sparsity



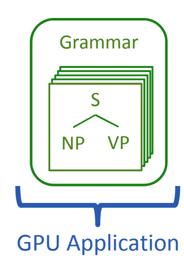
Exploiting Sparsity

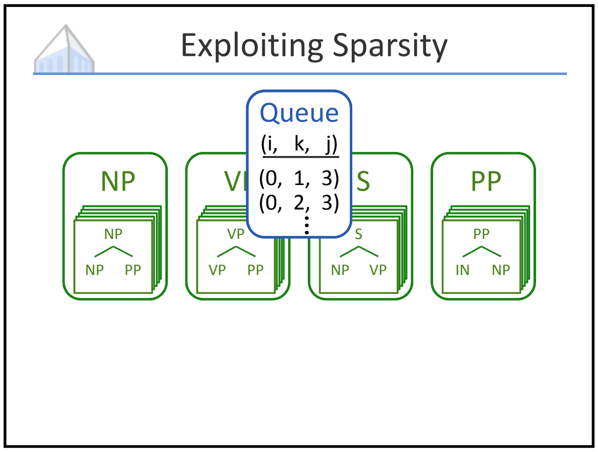
(0, 1, 3)	~~S~~	NP	~~VP~~	~~PP~~	...
(0, 2, 3)	~~S~~	~~NP~~	VP	~~PP~~	...
(1, 2, 4)	~~S~~	NP	~~VP~~	PP	...
(1, 3, 4)	~~S~~	NP	VP	~~PP~~	...
(2, 3, 5)	~~S~~	NP	VP	~~PP~~	...
(2, 4, 5)	~~S~~	NP	~~VP~~	PP	...
(3, 4, 6)	~~S~~	NP	~~VP~~	PP	...
⋮					

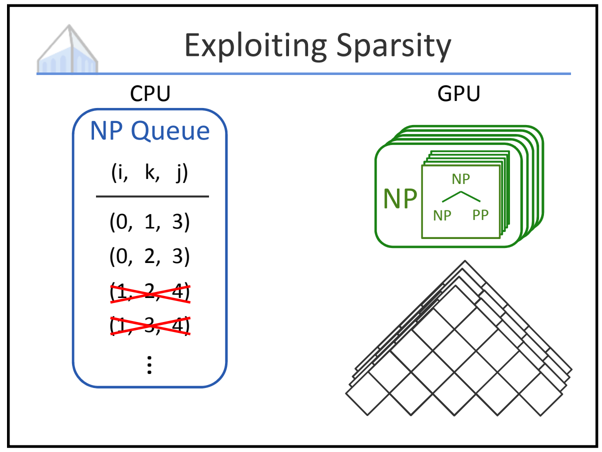
Exploiting Sparsity

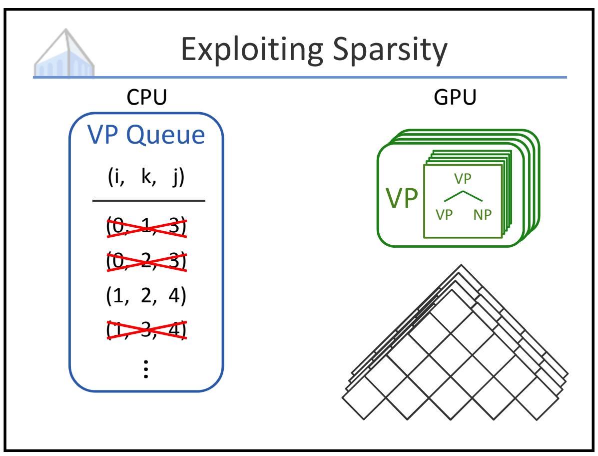


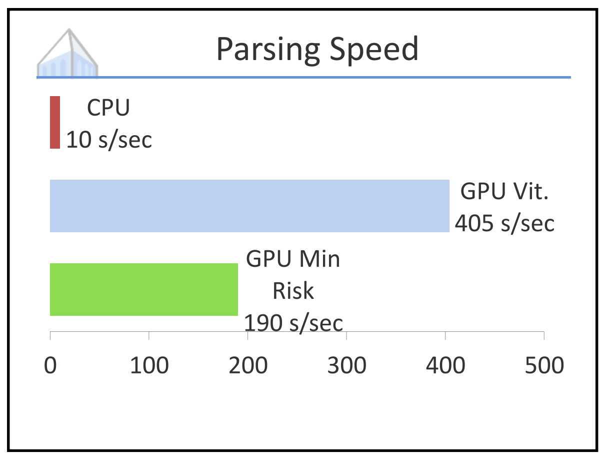
Exploiting Sparsity




Exploiting Sparsity
Queue
(i, k, j)
(0, 1, 3)
(0, 2, 3)
NP
NP
NP PP
VP
VP
VP PP
S
S
NP VP
PP
PP
IN NP


Exploiting Sparsity
CPU
NP Queue
(i, k, j)
(0, 1, 3)
(0, 2, 3)
~~(1, 2, 4)~~
~~(1, 3, 4)~~
GPU
NP
NP
NP PP


Exploiting Sparsity
CPU
VP Queue
(i, k, j)
~~(0, 1, 3)~~
~~(0, 2, 3)~~
(1, 2, 4)
~~(1, 3, 4)~~
GPU
VP
VP
VP NP


Parsing Speed
CPU
10 s/sec
GPU Vit.
405 s/sec
GPU Min
Risk
190 s/sec
0
100
200
300
400
500