Syntax-based Statistical Machine Translation

Philip Williams and Philipp Koehn

29 October 2014
Part I  -  Introduction
Part II  -  Rule Extraction
Part III  -  Decoding
Part IV  -  Extensions
What Do We Mean by Syntax-based SMT?

- “Syntax-based” is a very inclusive term. It refers to a large family of approaches:
  - Hiero, syntax-directed MT, syntax-augmented MT, syntactified phrase-based MT, tree-to-string, string-to-dependency, dependency treelet-based, soft syntax, fuzzy tree-to-tree, tree-based, . . .

- We mean that the translation model uses a tree-based representation of language.
  - We don’t count syntax-based preordering or syntactic LMs.

- We will focus on four widely-used approaches:
  1. Hierarchical phrase-based
  2. Tree-to-string
  3. String-to-tree
  4. Tree-to-tree
Why Use Syntax?

- Many translation problems can be best explained by pointing to syntax
  - reordering, e.g., verb movement in German–English translation
  - long distance agreement (e.g., subject-verb) in output

- Encourage grammatically coherent output

- Important step towards more linguistically motivated models (semantics)

- State-of-the art for some language pairs
  - Chinese-English (NIST 2008)
  - English-German (WMT 2012)
  - German-English (WMT 2013)
Statistical Machine Translation

Given a source string, $s$, find the target string, $t^*$, with the highest probability according to a distribution $p(t|s)$:

$$t^* = \arg \max_t p(t|s)$$

1. Model a probability distribution $p(t|s)$
2. Learn the parameters for the model
3. Find or approximate the highest probability string $t^*$
Statistical Machine Translation

1. Model a probability distribution $p(t|s)$
   - How is syntax used in modelling?

2. Learn the parameters for the model
   - What are the parameters of a syntax-based model?

3. Find or approximate the highest probability string $t^*$
   - How do we decode with a syntax-based model?
Modelling $p(t|s)$

- Most SMT models use Och and Ney’s (2002) log-linear formulation:

$$p(t|s) = \frac{\exp \left( \sum_{m=1}^{M} \lambda_m h_m(t,s) \right)}{\sum_{t'} \exp \left( \sum_{m=1}^{M} \lambda_m h_m(t',s) \right)}$$

$h_1, \ldots, h_M$ are real-valued functions and $\lambda_1, \ldots, \lambda_M$ are real-valued constants

- Denominator can be ignored during search:

$$t^* = \arg \max_{t} p(t|s)$$

$$= \arg \max_{t} \sum_{m=1}^{M} \lambda_m h_m(t, s)$$
Modelling $p(t|s)$

$$t^* = \arg \max_t \sum_{m=1}^{M} \lambda_m h_m(t, s)$$ (1)

- In word-based models, $s$ and $t$ are modelled as sequences of words.
- In phrase-based models, $s$ and $t$ are modelled as sequences of phrases.
- So what about syntax-based models?
Hierarchical Phrase-based MT

Like phrase pairs. . .

Für britische Skandale ist dieser nicht besonders schlüpfrig.
As British political scandals go, this one is not particularly juicy.

But with nesting:

Für britische Skandale ist dieser nicht besonders schlüpfrig.
As British political scandals go, this one is not particularly juicy.
Hierarchical Phrase-based MT

Hierarchical phrase pairs:

- ist dieser
- nicht besonders
- schlüpfrig
- , this one is
- not particularly
- juicy

are modelled using Synchronous Context-Free Grammar (SCFG):

\[
X \rightarrow \text{ist dieser } X_1 \mid \text{, this one is } X_1 \\
X \rightarrow \text{nicht besonders } X_1 \mid \text{not particularly } X_1 \\
X \rightarrow \text{schlüpfrig} \mid \text{juicy}
\]
Hierarchical Phrase-based MT

Rules can include up to two non-terminals:

\[ x \rightarrow deshalb \ x_1 \ die \ x_2 \ | \ therefore \ the \ x_2 \ x_1 \]
\[ x \rightarrow x_1 \ und \ x_2 \ | \ x_1 \ and \ x_2 \]

Glue rules concatenate hierarchical phrases:

\[ S \rightarrow X_1 \ | \ X_1 \]
\[ S \rightarrow S_1 \ X_2 \ | \ S_1 \ X_2 \]
Hierarchical Phrase-based MT

• Synchronous Context-Free Grammar:
  – Rewrite rules of the form $\langle A, B \rangle \rightarrow \langle \alpha, \beta, \sim \rangle$
  – $A$ and $B$ are source and target non-terminals, respectively
  – $\alpha$ and $\beta$ are strings of terminals and non-terminals for the source and target sides, respectively.
  – $\sim$ is a one-to-one correspondence between source and target non-terminals.

• Hiero grammars are a special case of SCFG:
  – One non-terminal type, $x$, on source side
  – Two non-terminal types, $x$ and $s$, on target side
  – Various restrictions on rule form (see Chiang (2007))
SCFG Derivation

- Derivation starts with pair of linked $s$ symbols.
SCFG Derivation

\[ S_1 \mid S_1 \]

\[ \Rightarrow S_2 \, X_3 \mid S_2 \, X_3 \]

- \( S \rightarrow S_1 \, X_2 \mid S_1 \, X_2 \) (glue rule)
SCFG Derivation

\[ s_1 \mid s_1 \]
\[ \Rightarrow s_2 \ x_3 \mid s_2 \ x_3 \]
\[ \Rightarrow s_2 \ x_4 \ und \ x_5 \mid s_2 \ x_4 \ and \ x_5 \]

- \( x \rightarrow x_1 \ und \ x_2 \mid x_1 \ and \ x_2 \)
SCFG Derivation

\[ S_1 \ | \ S_1 \]
\[ \Rightarrow S_2 \ x_3 \ | \ S_2 \ x_3 \]
\[ \Rightarrow S_2 \ x_4 \ und \ x_5 \ | \ S_2 \ x_4 \ and \ x_5 \]
\[ \Rightarrow S_2 \ unfounded \ and \ x_5 \ | \ S_2 \ unfounded \ and \ x_5 \]

- \[ X \rightarrow unfounded \ | \ unfounded \]
SCFG Derivation

\[ S_1 \quad | \quad S_1 \]
\[ \Rightarrow \quad S_2 \; X_3 \quad | \quad S_2 \; X_3 \]
\[ \Rightarrow \quad S_2 \; X_4 \; \text{und} \; X_5 \quad | \quad S_2 \; X_4 \; \text{and} \; X_5 \]
\[ \Rightarrow \quad S_2 \; \text{unzutreffend und} \; X_5 \quad | \quad S_2 \; \text{unfounded and} \; X_5 \]
\[ \Rightarrow \quad S_2 \; \text{unzutreffend und} \; \text{irreführend} \quad | \quad S_2 \; \text{unfounded and} \; \text{misleading} \]

- \( X \rightarrow \text{irreführend} \quad | \quad \text{misleading} \)
SCFG Derivation

\[ s_1 \mid s_1 \]
\[ \Rightarrow s_2 \, x_3 \mid s_2 \, x_3 \]
\[ \Rightarrow s_2 \, x_4 \, \text{und} \, x_5 \mid s_2 \, x_4 \, \text{and} \, x_5 \]
\[ \Rightarrow s_2 \, \text{unzutreffend und} \, x_5 \mid s_2 \, \text{unfounded and} \, x_5 \]
\[ \Rightarrow s_2 \, \text{unzutreffend und irreführend} \mid s_2 \, \text{unfounded and misleading} \]
\[ \Rightarrow x_6 \, \text{unzutreffend und irreführend} \mid x_6 \, \text{unfounded and misleading} \]

- \( S \rightarrow X_1 \mid X_1 \) (glue rule)
SCFG Derivation

\[ S_1 \mid S_1 \]
\[ \Rightarrow S_2 X_3 \mid S_2 X_3 \]
\[ \Rightarrow S_2 X_4 \text{ und } X_5 \mid S_2 X_4 \text{ and } X_5 \]
\[ \Rightarrow S_2 \text{ unzutreffend und } X_5 \mid S_2 \text{ unfounded and } X_5 \]
\[ \Rightarrow S_2 \text{ unzutreffend und irreführend } \mid S_2 \text{ unfounded and misleading} \]
\[ \Rightarrow X_6 \text{ unzutreffend und irreführend } \mid X_6 \text{ unfounded and misleading} \]
\[ \Rightarrow \text{deshalb } X_7 \text{ die } X_8 \text{ unzutreffend und irreführend } \]
\[ \quad \mid \text{therefore the } X_8 X_7 \text{ unfounded and misleading} \]

- \( X \rightarrow \text{deshalb } X_1 \text{ die } X_2 \mid \text{therefore the } X_2 X_1 \) (non-terminal reordering)
SCFG Derivation

\[ \begin{align*}
S_1 & \quad | \quad S_1 \\
\Rightarrow & \quad S_2 \ S_3 \quad | \quad S_2 \ S_3 \\
\Rightarrow & \quad S_2 \ S_4 \text{ und } X_5 \quad | \quad S_2 \ S_4 \text{ and } X_5 \\
\Rightarrow & \quad S_2 \text{ unzutreffend und } X_5 \quad | \quad S_2 \text{ unfounded and } X_5 \\
\Rightarrow & \quad S_2 \text{ unzutreffend und irreführend} \quad | \quad S_2 \text{ unfounded and misleading} \\
\Rightarrow & \quad S_6 \text{ unzutreffend und irreführend} \quad | \quad S_6 \text{ unfounded and misleading} \\
\Rightarrow & \quad \text{deshalb } \ X_7 \text{ die } X_8 \text{ unzutreffend und irreführend} \\
& \quad \quad | \quad \text{therefore the } X_8 \ X_7 \text{ unfounded and misleading} \\
\Rightarrow & \quad \text{deshalb } \text{sei } X_8 \text{ unzutreffend und irreführend} \\
& \quad \quad | \quad \text{therefore the } X_8 \text{ was unfounded and misleading}
\end{align*} \]

- \( X \rightarrow \text{sei} \quad | \quad \text{was} \)
SCFG Derivation

\[ S_1 \mid S_1 \]

\[ \Rightarrow S_2 X_3 \mid S_2 X_3 \]

\[ \Rightarrow S_2 X_4 \text{ und } X_5 \mid S_2 X_4 \text{ and } X_5 \]

\[ \Rightarrow S_2 \text{ unzutreffend und } X_5 \mid S_2 \text{ unfounded and } X_5 \]

\[ \Rightarrow S_2 \text{ unzutreffend und irreführend } \mid S_2 \text{ unfounded and misleading} \]

\[ \Rightarrow X_6 \text{ unzutreffend und irreführend } \mid X_6 \text{ unfounded and misleading} \]

\[ \Rightarrow \text{deshalb } X_7 \text{ die } X_8 \text{ unzutreffend und irreführend} \]

\[ \mid \text{therefore the } X_8 \text{ X}_7 \text{ unfounded and misleading} \]

\[ \Rightarrow \text{deshalb sei die } X_8 \text{ unzutreffend und irreführend} \]

\[ \mid \text{therefore the } X_8 \text{ was unfounded and misleading} \]

\[ \Rightarrow \text{deshalb sei die } \text{Werbung} \text{ unzutreffend und irreführend} \]

\[ \mid \text{therefore the advertisement was unfounded and misleading} \]

\[ \bullet x \rightarrow \text{Werbung } \mid \text{advertisement} \]
Hierarchical Phrase-based MT

- We can now define the search in terms of SCFG derivations

\[
    t^* = \arg \max_t \sum_{m=1}^{M} \lambda_m h_m(t, s) 
\]  

\[
= \arg \max_t \sum_d \sum_{m=1}^{M} \lambda_m h_m(t, s, d) 
\]  

\[d \in D\], the set of synchronous derivations with source \(s\) and yield \(t\).

- In practice, approximated with search for single-best derivation:

\[
d^* = \arg \max_d \sum_{m=1}^{M} \lambda_m h_m(t, s, d) 
\]  

Hierarchical Phrase-based MT

- Search for single-best derivation:

\[ d^* = \arg \max_d \sum_{m=1}^{M} \lambda_m h_m(t, s, d) \]  \hspace{1cm} (3)

- Rule-local feature functions allow decomposition of derivation scores:

\[ h_m(d) = \sum_{r_i} h_m(r_i) \]

- But n-gram language model can’t be decomposed this way. . .

\[ d^* = \arg \max_d \left( \lambda_1 \log p_{LM}(d) + \sum_{r_i} \sum_{m=2}^{M} \lambda_m h_m(r_i) \right) \]  \hspace{1cm} (4)
Hierarchical Phrase-based MT

- Summary so far:
  - Generalizes concept of phrase pair to allow nested phrases
  - Formalized using SCFG
  - No use of linguistic annotation: syntactic in a purely formal sense
  - Model uses standard SMT log-linear formulation
  - Search over derivations

- Later:
  - Rule extraction and scoring
  - Decoding (search for best derivation)
  - $k$-best extraction
Hierarchical phrase pairs but with embedded tree fragments on the source side:

Each source subphrase is a complete subtree.
Tree-to-String

Formalized using Synchronous Tree-Substitution Grammar (STSG):

As British political scandals go

Für britische Skandale

Für britische Skandale

for as British X1 go

scandals

scandals
Tree-to-String

• Synchronous Tree Substitution Grammar (STSG):
  – Grammar rules have the form \( \langle \pi, \gamma, \sim \rangle \)
  – \( \pi \) is a tree with source terminal and non-terminal leaves
  – \( \gamma \) is a string\(^1\) of target terminals and non-terminals
  – \( \sim \) is a one-to-one correspondence between source and target non-terminals.

• Unlike Hiero:
  – Linguistic-annotation (on source-side)
  – No limit to number of substitution sites (non-terminals)
  – No reordering limit during decoding

\(^1\)Technically, a 1-level tree formed by adding \( X \) as the root and the symbols from \( \gamma \) as children.
Tree-to-String

- Derivation involves synchronous rewrites (like SCFG)
- Tree fragments required to match input parse tree.
  - Motivation: tree provides context for rule selection ("syntax-directed")
- Efficient decoding algorithms available: source tree constrains rule options
- Search for single-best derivation:

\[ d^* = \arg \max_d \left( \lambda_1 \log p_{LM}(d) + \sum_{r_i} \sum_{m=2}^M \lambda_m h_m(r_i) \right) \]

where source-side of \( d \) must match input tree
**String-to-Tree**

Hierarchical phrase pairs but with embedded tree fragments on the target side:

```
Für britische Skandale ist dieser nicht besonders schlüpfrig.
```

Each target subphrase is a complete subtree.
String-to-Tree

Formalized using STSG:

```
NP1
SBAR
S
VP für X1
IN as
Für britische Skandale
```

Or SCFG:

```
SBAR → für X1 | as NP1 go
NP → britische Skandale | British political scandals
```
String-to-Tree

- Derivation is a rewriting process, like hierachical phrase-based and tree-to-string
  - Rewrites only allowed if target labels match at substitution sites
  - Internal tree structure not used in derivation (hence frequent use of SCFG)
  - Motivation: constraints provided by target syntax lead to more fluent output

- Later:
  - Rule extraction and scoring
  - Decoding (Hiero will be special case of S2T)
  - $k$-best extraction (likewise)
Tree-to-Tree

Hierarchical phrase pairs but with embedded tree fragments on both sides:

Formalized using STSG
Tree-to-Tree

Differences in source and target syntactic structure increasingly important

Can be differences in treebank annotation style or simply differences in language choice
Summary So Far

- We have introduced four models:

<table>
<thead>
<tr>
<th>Model</th>
<th>Formalism</th>
<th>Source Syntax</th>
<th>Target Syntax</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hiero</td>
<td>SCFG</td>
<td>N</td>
<td>N</td>
<td>string</td>
</tr>
<tr>
<td>T2S</td>
<td>STSG</td>
<td>Y</td>
<td>N</td>
<td>tree</td>
</tr>
<tr>
<td>S2T</td>
<td>STSG or SCFG</td>
<td>N</td>
<td>Y</td>
<td>string</td>
</tr>
<tr>
<td>T2T</td>
<td>STSG</td>
<td>Y</td>
<td>Y</td>
<td>tree</td>
</tr>
</tbody>
</table>

- Next:
  - Rule extraction
Part I - Introduction

Part II - Rule Extraction

Part III - Decoding

Part IV - Extensions
Learning Synchronous Grammars

• Extracting rules from a word-aligned parallel corpus

• First: Hierarchical phrase-based model
  – only one non-terminal symbol $x$
  – no linguistic syntax, just a formally syntactic model

• Then: Synchronous phrase structure model
  – non-terminals for words and phrases: NP, VP, PP, ADJ, ...
  – corpus must also be parsed with syntactic parser
Extracting Phrase Translation Rules

| I shall be passing some comments to you | Ich werde Ihnen die entsprechenden Anmerkungen aushändigen |

shall be = werde
Extracting Phrase Translation Rules

I shall be passing some comments on to you.

Ich werde Ihnen die entsprechenden Anmerkungen aushändigen.

Some comments = die entsprechenden Anmerkungen.
Extracting Phrase Translation Rules

I shall be passing on to you some comments.

Ich werde Ihnen die entsprechenden Anmerkungen aushändigen.

= shall be passing on to you some comments.
Extracting Hierarchical Phrase Translation Rules

Ich werde Ihnen die entsprechenden Anmerkungen aushändigen

werde X aushändigen

= shall be passing on X

subtracting subphrase
Formal Definition

- Recall: consistent phrase pairs

\[(\bar{e}, \bar{f}) \text{ consistent with } A \iff \]
\[\forall e_i \in \bar{e} : (e_i, f_j) \in A \rightarrow f_j \in \bar{f} \]
AND \[\forall f_j \in \bar{f} : (e_i, f_j) \in A \rightarrow e_i \in \bar{e} \]
AND \[\exists e_i \in \bar{e}, f_j \in \bar{f} : (e_i, f_j) \in A \]

- Let \( P \) be the set of all extracted phrase pairs \((\bar{e}, \bar{f})\)
Formal Definition

- Extend recursively:

\[
\text{if } (\bar{e}, \bar{f}) \in P \text{ AND } (\bar{e}_{\text{SUB}}, \bar{f}_{\text{SUB}}) \in P \\
\text{AND } \bar{e} = \bar{e}_{\text{PRE}} + \bar{e}_{\text{SUB}} + \bar{e}_{\text{POST}} \\
\text{AND } \bar{f} = \bar{f}_{\text{PRE}} + \bar{f}_{\text{SUB}} + \bar{f}_{\text{POST}} \\
\text{AND } \bar{e} \neq \bar{e}_{\text{SUB}} \text{ AND } \bar{f} \neq \bar{f}_{\text{SUB}} \\
\text{add } (e_{\text{PRE}} + x + e_{\text{POST}}, f_{\text{PRE}} + x + f_{\text{POST}}) \text{ to } P
\]

(note: any of \(e_{\text{PRE}}, e_{\text{POST}}, f_{\text{PRE}}, \) or \(f_{\text{POST}}\) may be empty)

- Set of hierarchical phrase pairs is the closure under this extension mechanism
Comments

• Removal of multiple sub-phrases leads to rules with multiple non-terminals, such as:

\[ Y \rightarrow X_1 X_2 \mid X_2 \text{ of } X_1 \]

• Typical restrictions to limit complexity [Chiang, 2005]
  - at most 2 nonterminal symbols
  - at least 1 but at most 5 words per language
  - span at most 15 words (counting gaps)
Learning Syntactic Translation Rules
Constraints on Syntactic Rules

- Same word alignment constraints as hierarchical models

- Hierarchical: rule can cover any span
  ⇔ syntactic rules must cover constituents in the tree

- Hierarchical: gaps may cover any span
  ⇔ gaps must cover constituents in the tree

- Much fewer rules are extracted (all things being equal)
Impossible Rules

English span not a constituent
no rule extracted
Rules with Context

Rule with this phrase pair requires syntactic context

Syntax-based Statistical Machine Translation
Too Many Rules Extractable

• Huge number of rules can be extracted
  (every alignable node may or may not be part of a rule → exponential number of rules)

• Need to limit which rules to extract

• Option 1: similar restriction as for hierarchical model
  (maximum span size, maximum number of terminals and non-terminals, etc.)

• Option 2: only extract minimal rules ("GHKM" rules)
I shall be passing on to you some comments.

Ich werde Ihnen die entsprechenden Anmerkungen aushändigen.

Extract: set of smallest rules required to explain the sentence pair.
I shall be passing on to you some comments.

Extracted rule: \text{PRP} \rightarrow \text{Ich} \mid \text{I}
Ich werde Ihnen die entsprechenden Anmerkungen aushändigen.

Extracted rule: \textit{PRP} $\rightarrow$ \textit{Ihnen} | \textit{you}
I shall be passing on to you some comments.

Extracted rule: \( DT \rightarrow \text{die} | \text{some} \)
Lexical Rule

Extracted rule: NNS → Anmerkungen | comments
I shall be passing on to you some comments.

Extracted rule: PP → X | to PRP
I shall be passing on to you some comments

Ich werde Ihnen die entsprechenden Anmerkungen aushändigen

Extracted rule: \( NP \rightarrow X_1 \ X_2 \ | \ DT_1 \ NNS_2 \)
I shall be passing on to you some comments.

Extracted rule: $VP \rightarrow X_1 \ X_2 \ aushändigen \mid passing \ on \ PP_1 \ NP_2$
I shall be passing on to you some comments.

Ich werde Ihnen die entsprechenden Anmerkungen aushändigen.

Extracted rule: $\text{VP} \rightarrow \text{werde X} \mid \text{shall be VP}$ (ignoring internal structure)
I shall be passing on to you some comments

Extracted rule: \( S \rightarrow X_1 X_2 \mid PRP_1 VP_2 \)

DONE — note: one rule per alignable constituent
I shall be passing on to you some comments.

Ich werde Ihnen die entsprechenden Anmerkungen aushändigen.

Attach to neighboring words or higher nodes → additional rules.

Syntax-based Statistical Machine Translation
Too Few Phrasal Rules?

• Lexical rules will be 1-to-1 mappings (unless word alignment requires otherwise)

• But: phrasal rules very beneficial in phrase-based models

• Solutions
  – combine rules that contain a maximum number of symbols
    (as in hierarchical models, recall: ”Option 1”)
  – compose minimal rules to cover a maximum number of non-leaf nodes
Composed Rules

- Current rules

\[ X_1 X_2 = \underbrace{\text{NP}}_{\text{DT}_1 \text{ NNS}_1} \]

\[ \text{die} = \underbrace{\text{DT}}_{\text{some}} \quad \text{entsprechenden Anmerkungen} = \underbrace{\text{NNS}}_{\text{comments}} \]

- Composed rule

\[ \text{die entsprechenden Anmerkungen} = \underbrace{\text{NP}}_{\text{DT} \text{ NNS}} \quad \text{some comments} \]

(1 non-leaf node: NP)
Composed Rules

• Minimal rule:

\[ x_1 \times x_2 \text{ aushändigen} = \text{VP} \]

3 non-leaf nodes:
VP, PP, NP

• Composed rule:

\[ \text{Ihnen } x_1 \text{ aushändigen} = \text{VP} \]

3 non-leaf nodes:
VP, PP and NP
Relaxing Tree Constraints

• Impossible rule

\[
\begin{align*}
X &= MD \quad VB \\
\text{werde} &= \text{shall} \quad \text{be}
\end{align*}
\]

• Create new non-terminal label: MD+VB

⇒ New rule

\[
\begin{align*}
X &= MD+VB \\
\text{werde} &= \underline{MD} \quad \underline{VB} \\
\text{shall} &= \text{be}
\end{align*}
\]
Zollmann Venugopal Relaxation

• If span consists of two constituents, join them: $X+Y$

• If span consists of three constituents, join them: $X+Y+Z$

• If span covers constituents with the same parent $x$ and include
  - every but the first child $y$, label as $X\backslash Y$
  - every but the last child $y$, label as $X/Y$

• For all other cases, label as FAIL

⇒ More rules can be extracted, but number of non-terminals blows up
Special Problem: Flat Structures

- Flat structures severely limit rule extraction.

- Can only extract rules for individual words or entire phrase.
Relaxation by Tree Binarization

More rules can be extracted
Left-binarization or right-binarization?
Scoring Translation Rules

- Extract all rules from corpus

- Score based on counts
  - joint rule probability: $p(\text{LHS}, \text{RHS}_f, \text{RHS}_e)$
  - rule application probability: $p(\text{RHS}_f, \text{RHS}_e | \text{LHS})$
  - direct translation probability: $p(\text{RHS}_e | \text{RHS}_f, \text{LHS})$
  - noisy channel translation probability: $p(\text{RHS}_f | \text{RHS}_e, \text{LHS})$
  - lexical translation probability: $\prod_{e_i \in \text{RHS}_e} p(e_i | \text{RHS}_f, a)$

- Edinburgh’s WMT System:
  - $p(\text{RHS}_e, \text{LHS} | \text{RHS}_f)$ and $p(\text{RHS}_f | \text{RHS}_e, \text{LHS})$
  - lexical translation probability: $\prod_{e_i \in \text{RHS}_e} p(e_i | \text{RHS}_f, a)$
  - PCFG probability of tree fragment: $p_{pcfg}(\pi)$
  - rule rareness and rule count penalties: $\exp(-1/\text{count}(r))$ and $\exp(1)$
Outline

1. Hiero/S2T decoding (SCFG with string input)
   - Viterbi decoding with local features (-LM)
   - \(k\)-best extraction
   - LM integration (cube pruning)
   - The S2T algorithm, as implemented in Moses

2. T2S decoding (STSG with tree input)
   - Vanilla T2S: non-directional, cube pruning

3. T2T decoding (STSG with tree input)
   - Included for completeness — better alternatives explored later
Viterbi S2T Decoding (-LM)

**Objective**  Find the highest-scoring synchronous derivation $d^*$

**Input**  $s_1 s_2 \ldots s_n$

<table>
<thead>
<tr>
<th>Grammar</th>
<th>$r_1$</th>
<th>$C_1 \rightarrow \alpha_1 \mid \beta_1$</th>
<th>$w_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_2$</td>
<td>$C_2 \rightarrow \alpha_2 \mid \beta_2$</td>
<td>$w_2$</td>
<td></td>
</tr>
<tr>
<td>$r_3$</td>
<td>$C_3 \rightarrow \alpha_3 \mid \beta_3$</td>
<td>$w_3$</td>
<td></td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$C_{</td>
<td>G</td>
<td>} \rightarrow \alpha_{</td>
</tr>
</tbody>
</table>

- $C_i$, $\alpha_i$ and $\beta_i$ are LHS, source RHS, target RHS of rule $r_i$, respectively.
- $w_i$ is weight of rule $r_i$ (weighted product of rule-local feature functions).
- $|G|$ is the number of rules in the grammar $G$. 
Viterbi S2T Decoding (-LM)

Objective  Find the highest-scoring synchronous derivation $d^*$

Solution

1. Project grammar
   Project weighted SCFG to weighted CFG
   $f : G \rightarrow G'$ (many-to-one rule mapping)

2. Parse
   Find Viterbi parse of sentence wrt $G'$

3. Translate
   Produce synchronous tree pair by applying inverse projection $f'$
Example

**Input**

jemand mußte Josef K. verleumdet haben
someone must Josef K. slandered have

**Grammar**

\[
\begin{align*}
\text{r}_1 : & \text{ NP } \rightarrow Josef K. | Josef K. & 0.90 \\
\text{r}_2 : & \text{ VBN } \rightarrow verleumdet | slandered & 0.40 \\
\text{r}_3 : & \text{ VBN } \rightarrow verleumdet | defamed & 0.20 \\
\text{r}_4 : & \text{ VP } \rightarrow mußte x_1 x_2 haben | must have VBN_2 NP_1 & 0.10 \\
\text{r}_5 : & \text{ S } \rightarrow jemand x_1 | someone VP_1 & 0.60 \\
\text{r}_6 : & \text{ S } \rightarrow jemand mußte x_1 x_2 haben | someone must have VBN_2 NP_1 & 0.80 \\
\text{r}_7 : & \text{ S } \rightarrow jemand mußte x_1 x_2 haben | NP_1 must have been VBN_1 by someone & 0.05 \\
\end{align*}
\]

(Six derivations in total)
Example

Input

jemand mußte Josef K. verleumdet haben
someone must Josef K. slandered have

⇒ $r_1$: NP → Josef K. | Josef K. 0.90
⇒ $r_2$: VBN → verleumdet | slandered 0.40
⇒ $r_3$: VBN → verleumdet | defamed 0.20
⇒ $r_4$: VP → mußte X₁ X₂ haben | must have VBN₂ NP₁ 0.10
⇒ $r_5$: S → jemand X₁ | someone VP₁ 0.60
⇒ $r_6$: S → jemand mußte X₁ X₂ haben | someone must have VBN₂ NP₁ 0.80
⇒ $r_7$: S → jemand mußte X₁ X₂ haben | NP₁ must have been VBN₁ by someone 0.05

Derivation 1

[Diagram showing the derivation process]
Example

Input
jemand mußte Josef K. verleumdet haben
someone must Josef K. slandered have

⇒ \( r_1 \): NP \( \rightarrow \) Josef K. | Josef K.
⇒ \( r_2 \): VBN \( \rightarrow \) verleumdet | slandered
⇒ \( r_3 \): VBN \( \rightarrow \) verleumdet | defamed
⇒ \( r_4 \): VP \( \rightarrow \) mußte \( X_1 \) \( X_2 \) haben | must have \( VBN_2 \) \( NP_1 \)
⇒ \( r_5 \): S \( \rightarrow \) jemand \( X_1 \) | someone \( VP_1 \)
⇒ \( r_6 \): S \( \rightarrow \) jemand mußte \( X_1 \) \( X_2 \) haben | someone must have \( VBN_2 \) \( NP_1 \)
⇒ \( r_7 \): S \( \rightarrow \) jemand mußte \( X_1 \) \( X_2 \) haben | \( NP_1 \) must have been \( VBN_1 \) by someone

Grammar

Derivation 2
Example

**Input** jemand mußte Josef K. verleumdet haben
someone must Josef K. slandered have

⇒ $r_1$: NP $\rightarrow$ Josef K. | Josef K. 0.90
⇒ $r_2$: VBN $\rightarrow$ verleumdet | slandered 0.40
$r_3$: VBN $\rightarrow$ verleumdet | defamed 0.20
⇒ $r_4$: VP $\rightarrow$ mußte $X_1 X_2$ haben | must have VBN$_2$ NP$_1$ 0.10
$r_5$: S $\rightarrow$ jemand $X_1$ | someone VP$_1$ 0.60
⇒ $r_6$: S $\rightarrow$ jemand mußte $X_1 X_2$ haben | someone must have VBN$_2$ NP$_1$ 0.80
$r_7$: S $\rightarrow$ jemand mußte $X_1 X_2$ haben | NP$_1$ must have been VBN$_1$ by someone 0.05

**Grammar**

**Derivation 3**

Syntax-based Statistical Machine Translation 74
Example

Input  jemand mußte Josef K. verleumdet haben
someone must Josef K. slandered have

⇒ r₁: NP → Josef K. | Josef K. 0.90
⇒ r₂: VBN → verleumdet | slandered 0.40
⇒ r₃: VBN → verleumdet | defamed 0.20
⇒ r₄: VP → müßte X₁ X₂ haben | must have VBN₂ NP₁ 0.10
⇒ r₅: S → jemand X₁ | someone VP₁ 0.60
⇒ r₆: S → jemand müßte X₁ X₂ haben | someone must have VBN₂ NP₁ 0.80
⇒ r₇: S → jemand müßte X₁ X₂ haben | NP₁ must have been VBN₁ by someone 0.05

Grammar

Derivation 4

Syntax-based Statistical Machine Translation 75
Example

Input  jemand mußte Josef K. verleumdet haben
someone must Josef K. slandered have

⇒ $r_1$: NP → Josef K. | Josef K. 0.90
⇒ $r_2$: VBN → verleumdet | slandered 0.40
$r_3$: VBN → verleumdet | defamed 0.20
$r_4$: VP → mußte $X_1$ $X_2$ haben | must have VBN$_2$ NP$_1$ 0.10
$r_5$: S → jemand $X_1$ | someone VP$_1$ 0.60
$r_6$: S → jemand mußte $X_1$ $X_2$ haben | someone must have VBN$_2$ NP$_1$ 0.80
⇒ $r_7$: S → jemand mußte $X_1$ $X_2$ haben | NP$_1$ must have been VBN$_1$ by someone 0.05

Derivation 5

Source  Target

Syntax-based Statistical Machine Translation 76
Example

**Input**  jemand mußte Josef K. verleumdet haben

someone must Josef K. slandered have

$$\Rightarrow r_1: \text{NP} \rightarrow \text{Josef K. | Josef K.} \quad 0.90$$

$$\Rightarrow r_2: \text{VBN} \rightarrow \text{verleumdet | slandered} \quad 0.40$$

$$\Rightarrow r_3: \text{VBN} \rightarrow \text{verleumdet | defamed} \quad 0.20$$

**Grammar**

$$\Rightarrow r_4: \text{VP} \rightarrow \text{mußte } X_1 X_2 \text{ haben | must have VBN}_2 \text{ NP}_1 \quad 0.10$$

$$\Rightarrow r_5: \text{S} \rightarrow \text{jemand } X_1 | \text{someone VP}_1 \quad 0.60$$

$$\Rightarrow r_6: \text{S} \rightarrow \text{jemand mußte } X_1 X_2 \text{ haben | someone must have VBN}_2 \text{ NP}_1 \quad 0.80$$

$$\Rightarrow r_7: \text{S} \rightarrow \text{jemand mußte } X_1 X_2 \text{ haben | NP}_1 \text{ must have been VBN}_1 \text{ by someone} \quad 0.05$$

**Derivation 6**
Step 1: Project Grammar to CFG

\[
\begin{align*}
{r_1}: & \quad \text{NP} \rightarrow \text{Josef K.} \mid \text{Josef K.} & 0.90 \\
{r_2}: & \quad \text{VBN} \rightarrow \text{verleumdet} \mid \text{slandered} & 0.40 \\
{r_3}: & \quad \text{VBN} \rightarrow \text{verleumdet} \mid \text{defamed} & 0.20 \\
{r_4}: & \quad \text{VP} \rightarrow \text{mußte X_1 X_2 haben} \mid \text{must have VBN_2 NP_1} & 0.10 \\
{r_5}: & \quad \text{S} \rightarrow \text{jemand X_1} \mid \text{someone VP_1} & 0.60 \\
{r_6}: & \quad \text{S} \rightarrow \text{jemand mußte X_1 X_2 haben} \mid \text{someone must have VBN_2 NP_1} & 0.80 \\
{r_7}: & \quad \text{S} \rightarrow \text{jemand mußte X_1 X_2 haben} \mid \text{NP_1 must have been VBN_1 by someone} & 0.05 \\
\end{align*}
\]

\[
\begin{align*}
{q_1}: & \quad \text{NP} \rightarrow \text{Josef K.} & 0.90 \\
{q_2}: & \quad \text{VBN} \rightarrow \text{verleumdet} & 0.40 \\
{q_3}: & \quad \text{VP} \rightarrow \text{mußte NP VBN haben} & 0.10 \\
{q_4}: & \quad \text{S} \rightarrow \text{jemand VP} & 0.60 \\
{q_5}: & \quad \text{S} \rightarrow \text{jemand mußte NP VBN haben} & 0.80 \\
\end{align*}
\]

• \( G \) is original synchronous grammar, \( G' \) is monolingual projection
Step 1: Project Grammar to CFG

$\Rightarrow r_1$: NP $\rightarrow$ Josef K. | Josef K. 0.90
$r_2$: VBN $\rightarrow$ verleumdet | slandered 0.40
$r_3$: VBN $\rightarrow$ verleumdet | defamed 0.20
$r_4$: VP $\rightarrow$ mußte $x_1$ $x_2$ haben | must have VBN $vbn$ np $np$ 0.10
$r_5$: S $\rightarrow$ jemand $x_1$ | someone VP $vp$ 0.60
$r_6$: S $\rightarrow$ jemand mußte $x_1$ $x_2$ haben | someone must have VBN $vbn$ $vbn$ np $np_1$ 0.80
$r_7$: S $\rightarrow$ jemand mußte $x_1$ $x_2$ haben | NP $np_1$ must have been VBN $vbn$ by someone 0.05

$\Rightarrow q_1$: NP $\rightarrow$ Josef K. 0.90
$q_2$: VBN $\rightarrow$ verleumdet 0.40
$q_3$: VP $\rightarrow$ mußte NP VBN haben 0.10
$q_4$: S $\rightarrow$ jemand VP 0.60
$q_5$: S $\rightarrow$ jemand mußte NP VBN haben 0.80

- Projected rule gets LHS and source RHS (but with target non-terminal labels)
Step 1: Project Grammar to CFG

\[
\begin{align*}
G & \quad r_1: \quad NP \rightarrow Josef K. \mid Josef K. \quad 0.90 \\
\Rightarrow & \quad r_2: \quad VBN \rightarrow verleumdet \mid slandered \quad 0.40 \\
\Rightarrow & \quad r_3: \quad VBN \rightarrow verleumdet \mid defamed \quad 0.20 \\
& \quad r_4: \quad VP \rightarrow müßte X_1 X_2 haben \mid must have VBN_2 NP_1 \quad 0.10 \\
& \quad r_5: \quad S \rightarrow jemand X_1 \mid someone VP_1 \quad 0.60 \\
& \quad r_6: \quad S \rightarrow jemand müßte X_1 X_2 haben \mid someone must have VBN_2 NP_1 \quad 0.80 \\
& \quad r_7: \quad S \rightarrow jemand müßte X_1 X_2 haben \mid NP_1 must have been VBN_1 by someone \quad 0.05 \\
\end{align*}
\]

\[
\begin{align*}
G' & \quad q_1: \quad NP \rightarrow Josef K. \quad 0.90 \\
\Rightarrow & \quad q_2: \quad VBN \rightarrow verleumdet \quad 0.40 \\
& \quad q_3: \quad VP \rightarrow müßte NP VBN haben \quad 0.10 \\
& \quad q_4: \quad S \rightarrow jemand VP \quad 0.60 \\
& \quad q_5: \quad S \rightarrow jemand müßte NP VBN haben \quad 0.80 \\
\end{align*}
\]

- Many-to-one: weight of projected rule is the best from set of projecting rules
Step 1: Project Grammar to CFG

\[ G \]

\begin{align*}
    r_1 &: NP \rightarrow Josef K. \mid Josef K. & 0.90 \\
    r_2 &: VBN \rightarrow verleumdet \mid slandered & 0.40 \\
    r_3 &: VBN \rightarrow verleumdet \mid defamed & 0.20 \\
    G \quad \Rightarrow r_4 &: VP \rightarrow mußte x_1 x_2 haben \mid must have VBN_2 NP_1 & 0.10 \\
    r_5 &: S \rightarrow jemand x_1 \mid someone VP_1 & 0.60 \\
    r_6 &: S \rightarrow jemand mußte x_1 x_2 haben \mid someone must have VBN_2 NP_1 & 0.80 \\
    r_7 &: S \rightarrow jemand mußte x_1 x_2 haben \mid NP_1 must have been VBN_1 by someone & 0.05
\end{align*}

\[ G' \]

\begin{align*}
    q_1 &: NP \rightarrow Josef K. & 0.90 \\
    q_2 &: VBN \rightarrow verleumdet & 0.40 \\
    G' \quad \Rightarrow q_3 &: VP \rightarrow mußte NP VBN haben & 0.10 \\
    q_4 &: S \rightarrow jemand VP & 0.60 \\
    q_5 &: S \rightarrow jemand mußte NP VBN haben & 0.80
\end{align*}

- Target non-terminal labels projected to monolingual rule (in source order)
Step 1: Project Grammar to CFG

\[
\begin{align*}
  r_1 : & \quad \text{NP} \rightarrow \text{Josef K.} \mid \text{Josef K.} & 0.90 \\
  r_2 : & \quad \text{VBN} \rightarrow \text{verleumdet} \mid \text{slandered} & 0.40 \\
  r_3 : & \quad \text{VBN} \rightarrow \text{verleumdet} \mid \text{defamed} & 0.20 \\
  r_4 : & \quad \text{VP} \rightarrow \text{mußte } x_1 x_2 \text{ haben} \mid \text{must have VBN2 NP1} & 0.10 \\
  r_5 : & \quad \text{S} \rightarrow \text{jemand } x_1 \mid \text{someone VP1} & 0.60 \\
  r_6 : & \quad \text{S} \rightarrow \text{jemand mußte } x_1 x_2 \text{ haben} \mid \text{someone must have VBN2 NP1} & 0.80 \\
  r_7 : & \quad \text{S} \rightarrow \text{jemand mußte } x_1 x_2 \text{ haben} \mid \text{NP1 must have been VBN1 by someone} & 0.05 \\
\end{align*}
\]

\[
\begin{align*}
  q_1 : & \quad \text{NP} \rightarrow \text{Josef K.} & 0.90 \\
  q_2 : & \quad \text{VBN} \rightarrow \text{verleumdet} & 0.40 \\
  q_3 : & \quad \text{VP} \rightarrow \text{mußte NP VBN haben} & 0.10 \\
  q_4 : & \quad \text{S} \rightarrow \text{jemand VP} & 0.60 \\
  q_5 : & \quad \text{S} \rightarrow \text{jemand mußte NP VBN haben} & 0.80 \\
\end{align*}
\]

- And so on...
Step 1: Project Grammar to CFG

\[ r_1: \text{NP} \rightarrow \text{Josef K. | Josef K.} \quad 0.90 \]
\[ r_2: \text{VBN} \rightarrow \text{verleumdet | slandered} \quad 0.40 \]
\[ r_3: \text{VBN} \rightarrow \text{verleumdet | defamed} \quad 0.20 \]

\[ G \]
\[ r_4: \text{VP} \rightarrow \text{mußte \( X_1 \) \( X_2 \) haben | must have VBN\(_2\) NP\(_1\)} \quad 0.10 \]
\[ r_5: \text{S} \rightarrow \text{jemand \( X_1 \) | someone VP\(_1\)} \quad 0.60 \]
\[ \Rightarrow r_6: \text{S} \rightarrow \text{jemand mußte \( X_1 \) \( X_2 \) haben | someone must have VBN\(_2\) NP\(_1\)} \quad 0.80 \]
\[ \Rightarrow r_7: \text{S} \rightarrow \text{jemand mußte \( X_1 \) \( X_2 \) haben | NP\(_1\) must have been VBN\(_1\) by someone} \quad 0.05 \]

\[ G' \]
\[ q_1: \text{NP} \rightarrow \text{Josef K.} \quad 0.90 \]
\[ q_2: \text{VBN} \rightarrow \text{verleumdet} \quad 0.40 \]
\[ q_3: \text{VP} \rightarrow \text{mußte NP VBN haben} \quad 0.10 \]
\[ q_4: \text{S} \rightarrow \text{jemand VP} \quad 0.60 \]
\[ \Rightarrow q_5: \text{S} \rightarrow \text{jemand mußte NP VBN haben} \quad 0.80 \]

- And so on.
Step 2: Find Viterbi Parse

- Standard weighted parsing algorithms.
- Binarization can be explicit (like CYK) or implicit (like Earley / CYK+).
Step 3: Reconstruct Synchronous Derivation

1-best parse tree

S

jemand mußte NP VBN haben

Josef K. verleumdet

Source-side parse tree
Step 3: Reconstruct Synchronous Derivation

1-best parse tree

Source-side parse tree

- Source-side: replace non-terminals with Xs
Step 3: Reconstruct Synchronous Derivation

- Target-side: invert grammar projection
Step 3: Reconstruct Synchronous Derivation

- Target-side: invert grammar projection

NP \rightarrow Josef K. \mid Josef K.
Step 3: Reconstruct Synchronous Derivation

1-best parse tree

S
- jemand mußte
  - jemand: NP
  - mußte: VBN

VBN haben

Source-side parse tree

NP
- Josef
- K.
VBN verleumdet

Target-side: invert grammar projection (multiple rules? pick highest-scoring)

\[
\begin{align*}
VBN & \rightarrow \mathit{verleumdet} \mid \mathit{slandered} \quad 0.4 \\
VBN & \rightarrow \mathit{verleumdet} \mid \mathit{defamed} \quad 0.2
\end{align*}
\]
Step 3: Reconstruct Synchronous Derivation

1-best parse tree

Source-side parse tree

• Target-side: invert grammar projection (multiple rules? pick highest-scoring)

\[
S \rightarrow \text{jemand mußte } x_1 \ x_2 \ \text{haben} \quad | \quad \text{someone must have VBN}_2 \ \text{NP}_1 \\
S \rightarrow \text{jemand mußte } x_1 \ x_2 \ \text{haben} \quad | \quad \text{NP}_1 \ \text{must have been VBN}_2 \ \text{by someone} \\
\]

0.80

0.05

Syntax-based Statistical Machine Translation
**$k$-best Extraction**

**Objective**  Find the $k$-best synchronous derivations $d_1, d_2, \ldots d_k$

Well. . .

1. 1-best derivation is 1-best monolingual parse tree with best set of translations

2. 2-best is one of
   (a) 1-best monolingual parse tree with second best set of translations, and
   (b) 2-best monolingual parse tree with best translations

3. 3-best derivation is ‘the other one’ or one of
   (a) 1-best monolingual parse tree with third best set of translations, and
   (b) 2-best monolingual parse tree with second best translations, and
   (c) 3-best monolingual parse tree with best translations

4. 4-best derivation is ‘one of what’s left’ or . . .
**$k$-best Extraction**

**Objective** Find the $k$-best synchronous derivations $d_1, d_2, \ldots d_k$

Well. . .

1. 1-best derivation is 1-best monolingual parse tree with best set of translations
2. 2-best is one of
   (a) 1-best monolingual parse tree with second best set of translations, and
   (b) 2-best monolingual parse tree with best translations
3. . . .

We know part of the solution: how to get the $k$-best monolingual derivations (Huang and Chiang, 2005)
Digression: Parsing and Hypergraphs

Syntax-based Statistical Machine Translation
Digression: Parsing and Hypergraphs

- Generalization of a graph: hyperedges connect two sets of vertices
- Terminology: vertices and hyperedges (nodes and arcs)
- A parse forest can be represented by a rooted, connected, labelled, directed, acyclic hypergraph (Klein and Manning, 2001)
- Vertices represent parsing states; hyperedges represent rule applications
Monolingual $k$-best Extraction

Huang and Chiang (2005) provide efficient algorithms for $k$-best extraction.

**Objective**
Extract the $k$-best monolingual derivations $d_1, d_2, \ldots, d_k$ from a weighted parse forest

**Outline**
(Alg. 3)

1. The 1-best subderivation for every vertex (and its incoming hyperedges) is known from the outset

2. Given the $i$-best derivation, the next best candidate along the same hyperedge is identical except for a substitution at a single incoming vertex

3. At the top vertex, generates candidates by recursively asking predecessors for next best subderivations.

4. Maintain priority queue of candidates at each vertex
Synchronous $k$-best Extraction

Replace hyperedges according to $f'$ (invert grammar projection)

- The standard $k$-best extraction algorithm now gives the $k$-best synchronous derivations.
- The second hypergraph is sometimes called a “translation hypergraph”.
- We’ll call the first the “parse forest hypergraph” or the “parse hypergraph.”
S2T Decoding (LM-) Summary

Objective
Find the $k$-best synchronous derivations $d_1, d_2, \ldots d_k$

Solution
1. **Project grammar**
   Project weighted SCFG to unweighted CFG
   \[ f : G \to G' \] (many-to-one)

2. **Parse**
   Build parse hypergraph wrt $G'$

3. **Invert projection**
   Expand hypergraph by replacing hyperedges according to $f'$

4. **Extract derivations**
   Extract $k$-best derivations using Huang and Chiang’s (2005) algorithm
**LM Integration**

**Without LM**  
$k$-best derivation is $k$-best path through translation hypergraph

**Optimal substructure**  
If global best path includes $VBN_{4,4}$ then best path must include hyperedge labelled $r_2$
LM Integration

Consider the two paths that include the hyperedge labelled $r_6$:

What's the best path through this hypergraph? For bi-gram LM we need to compute:

\[
p(\text{have | } \langle s \rangle) \times p(\text{slandered | have}) \times p(\text{Josef | slandered}) \times \ldots
\]

\[
p(\text{have | } \langle s \rangle) \times p(\text{defamed | have}) \times p(\text{Josef | defamed}) \times \ldots
\]
State Splitting?

Restore optimal substructure property by splitting states:

- Vertex labels include first and last words of translation.
- Hyperedges labelled with weights that incorporate LM costs.
- $k$-best derivation is $k$-best path.
State Splitting?

**Objective**
Find the $k$-best synchronous derivations $d_1, d_2, \ldots d_k$

**Potential Solution**
1. **Project grammar**
   Project weighted SCFG to weighted CFG $f : G \rightarrow G'$

2. **Parse**
   Build parse hypergraph wrt $G'$

3. **Invert projection + split states**
   Expand hypergraph by replacing hyperedges according to $f'$. *During replacement, split states and add LM costs*

4. **Extract derivations**
   Extract $k$-best derivations (Huang and Chiang, 2005)
State Splitting?

- Pick a search vertex for \( \text{NP}_{3,4} \) from the set \{ \( \text{NP}_{3,4}, \text{Josef K.} \) \}
- Pick a search vertex for \( \text{VBN}_{5,5} \) from the set \{ \( \text{VBN}_{5,5}, \text{slandered} \), \( \text{VBN}_{5,5}, \text{defamed} \) \}
- Pick a synchronous rule from the set \( f'(q_5) = \{ r_6, r_7 \} \) (i.e. pick a target-side)

The full set is generated by taking the Cartesian product of these three sets.
The Search Hypergraph is Too Large.

The parse hypergraph has $O(n^3)$ space constraints (assuming certain grammar properties.)

With a $m$-gram LM the search hypergraph is much larger:

<table>
<thead>
<tr>
<th></th>
<th>Vertices</th>
<th>Hyperedges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>$O(n^2</td>
<td>C</td>
</tr>
<tr>
<td>Search</td>
<td>$O(n^2</td>
<td>C</td>
</tr>
</tbody>
</table>

$C$ is the set of target non-terminals  
$T$ is the set of target-side terminals  
$n$ is the input sentence length  
$m$ is the order of the LM  
$A$ is the maximum rule arity
Heuristic Search

- In practice, only part of the search hypergraph can be explored.
- During search, a partial search hypergraph is generated in topological order.
- Three main strategies for reducing search space:

  **Parse forest pruning** Avoid splitting some parse forest hyperedges by pre-pruning the forest (methods can be exact or inexact).

  **Heuristic best-first splitting** e.g. cube pruning. Use a splitting algorithm that finds expanded hyperedges in approximately best-first order.

  **Beam search** Bin vertices according to source word span and category. Keep only the highest-scoring vertices for use later in the search.
Strategy 1: Parse Forest Pruning

- If parse forest is constructed in full prior to search then dead-ends can be pruned away.

- State splitting can be restricted to a small subset of promising hyperedges.
  - Moses ranks hyperedges according to -LM rule cost plus sums of incoming +LM vertex costs.

- Monolingual forest pruning methods (Inside-outside estimates, see e.g. Charniak and Johnson (2005)).

(Forest pruning methods haven’t been widely explored in the MT literature.)
Strategy 2: Heuristic Best-First State Splitting

- For every hyperedge in the parse hypergraph, there can be very many corresponding hyperedges in the search hypergraph.

- Cube pruning (Chiang, 2007) is most widely-used approximate algorithm but see Heafield et al. (2013) for a faster alternative.
Cube Pruning

<table>
<thead>
<tr>
<th>Slandered</th>
<th>Defamed</th>
<th>Maligned</th>
<th>Libelled</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.3</td>
<td>2.2</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Arrange all the choices in a “cube”

(here: a square, generally an orthotope, also called a hyperrectangle)
Create the First Hyperedge

<table>
<thead>
<tr>
<th>Hyperedges created in cube: (0,0)</th>
</tr>
</thead>
</table>

**Hyperedges**

- slandered 1.0
- defamed 1.3
- maligned 2.2
- libelled 2.6

**Dual Edges**

- Josef K.
- K.
- our protagonist

**Diagram**

```
+--------+--------+--------+
|        | Josef K.|        |
|        | 1.5    | 1.7    |
|        | Josef K.|        |
| slandered| 1.0    |        |
| defamed  | 1.3    |        |
| maligned | 2.2    |        |
| libelled | 2.6    |        |
+--------+--------+--------+
```
"Pop" Hyperedge

<table>
<thead>
<tr>
<th></th>
<th>1.5 Josef K.</th>
<th>1.7 K.</th>
<th>2.6 Josef</th>
<th>3.2 our protagonist</th>
</tr>
</thead>
<tbody>
<tr>
<td>slandered</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>defamed</td>
<td>2.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maligned</td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libelled</td>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libelled</td>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Hyperedges created in cube: ε
- Hyperedges popped: (0,0)
Create Neighboring Hyperedges

- Hyperedges created in cube: (0,1), (1,0)

- Hyperedges popped: (0,0)
Pop Best Hyperedge

- Hyperedges created in cube: (0,1)
- Hyperedges popped: (0,0), (1,0)
Create Neighboring Hyperedges

- Hyperedges created in cube: (0,1), (1,1), (2,0)
- Hyperedges popped: (0,0), (1,0)
More of the Same

- Hyperedges created in cube: (0,1), (1,2), (2,1), (2,0)
- Hyperedges popped: (0,0), (1,0), (1,1)
Queue of Cubes

- Many parse hyperedges for any given span
- Each of them will have a cube
- We can create a queue of cubes

⇒ Always pop off the most promising hyperedge, regardless of cube

- May have separate queues for different target constituent labels
Strategy 3: Beam search

- Bin vertices according to source word span and category.

- Keep only the highest-scoring vertices for use later in the search.
Putting it All Together: The S2T Decoding Algorithm in Moses

Objective
Find the $k$-best synchronous derivations $d_1, d_2, \ldots d_k$

Outline
1. Project grammar
   Project weighted SCFG to weighted CFG $f : G \rightarrow G'$

2. Interleaved parse + search
   Span-by-span, build parse hypergraph wrt $G'$ and build partial search hypergraph

3. Extract derivations
   Extract $k$-best derivations (Huang and Chiang, 2005)
Decoding: Components

- Vertices of the parse hypergraph are stored in a chart (includes input sentence)
- Hyperedges are enumerated but not stored in chart
- Terminology: PChart, PVertex, PHyperedge
Decoding: Components

- Parser generates PHyperedges for given span of PChart
- Parser has access to partially-completed PChart
- For now, the parser is a black-box component but we’ll return to parsing. . .
Decoding: Components

- Vertices of the search hypergraph are stored in a chart (includes input sentence)
- Vertices are stored in stacks (one per span + category), which are sorted
- Hyperedges are stored (unlike in PChart)
- Terminology: SChart, SVertex, SHyperedge
Decoding: Components

- Cube pruning algorithm (or similar) produces SHyperedges from PHyperedges
- A single SVertex can be produced multiple times so must check for this (‘recombination’)

Syntax-based Statistical Machine Translation 120
The Moses S2T Decoding Algorithm

1: initialize PChart and SChart by adding vertices for input words
2: for each span (in parser-defined order) do
3:   p-hyperedges = ForestPrune(parser.EnumerateHyperedges(span, p-chart), s-chart)
4:   for all p-hyperedges do
5:     create a cube for it
6:     create first s-hyperedge in cube
7:     place cube in queue
8:   end for
9: for specified number of pops do
10:    pop off best s-hyperedge of any cube in queue
11:    add it to a category-specific buffer
12:    create its neighbors
13: end for
14: for category do
15:   recombine s-hyperedges from buffer and move into s-chart stack
16:   sort stack
17: end for
18: end for
 Parsing for S2T Decoding

- Parser’s job is to enumerate PHyperedges, span-by-span.
- Parser has access to partially-filled PChart.
Parsing for S2T Decoding

• Can we just use CYK / CYK+ / Earley?
  – All require binarization (implicit or explicit).

• **Idea 1** Binarize $G'$
  – Binary normal forms exist for monolingual CFG grammars.
  – *But* we still need to know the synchronous rules for $+$LM search.

• **Idea 2** Binarize $G$ before projection to CFG
  – Binarization impossible for some SCFG rules with rank $\geq 4$
  – Not necessarily a problem: non-binarizable cases are rare in word-aligned translation data (Zhang et al., 2006)
  – But tricky in practice: how do we weight rules? And what about grammar inflation?
How to Avoid Binarization

• Hopkins and Langmead (2010) define a grammar property called scope:

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Scope</th>
<th>Pattern</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>a b c d e</td>
<td>0</td>
<td>a ◊ ◊ ◊ e</td>
<td>2</td>
</tr>
<tr>
<td>a ◊ c ◊ e</td>
<td>0</td>
<td>◊ b c d ◊</td>
<td>2</td>
</tr>
<tr>
<td>a ◊ ◊ d e</td>
<td>1</td>
<td>◊ ◊ c d ◊</td>
<td>3</td>
</tr>
<tr>
<td>◊ b c d e</td>
<td>1</td>
<td>◊ ◊ ◊ ◊ ◊</td>
<td>6</td>
</tr>
</tbody>
</table>

• They prove that a sentence of length $n$ can be parsed with a scope $k$ grammar in $O(nk)$ chart updates without binarization.

• They demonstrate empirically that reducing a GHKM grammar to scope-3 by pruning does not harm translation quality compared to synchronous binarization (and pruning is much simpler).

• Chung et al. (2011) perform similar comparison and achieve same result.
Specialized Parsing Algorithms

- CYK+ and Earley are popular choices for S2T decoding.

- But storing large numbers of dotted rules is problematic in practice (Chung et al. 2011 find scope-3 slower than binarized grammar with Earley parser, which they attribute to dotted rule storage).

- Several parsing algorithms have been designed specifically for synchronous translation grammars: DeNero et al. (2009), Hopkins and Langmead (2010), Sennrich (2014).

- We use Sennrich (2014)'s recursive variant of CYK+:
  - Good performance on WMT-scale task: fast, low-memory overhead
  - Simpler than CYK+ and alternatives
  - No dotted rule storage
Parsing for S2T Decoding (Moses-style)

\[ q_1: \ NP \rightarrow \ Josef \ K. \]
\[ q_2: \ VBN \rightarrow \ verleumdet \]
\[ q_3: \ VP \rightarrow \ mußte \ NP \ VBN \ haben \]
\[ q_4: \ S \rightarrow \ jemand \ VP \]
\[ q_5: \ S \rightarrow \ jemand \ mußte \ NP \ VBN \ haben \]

- Projected grammar \( G' \) is represented as a trie (sometimes called a prefix tree)
- Edges are labelled with terminals and non-terminals
- Labels along path (from root) represent prefix of rule RHS
- Vertices in black are associated with group of rules from \( G \) (sub-grouped by rule LHS)
• Sennrich (2014)’s parsing algorithm visits cells in right-to-left, depth-first order.
• We consider situation where all of PChart filled except for left-most diagonal.
• Recall that PVertices are stored, but PHyperedges are not.
Parsing for S2T Decoding - Example

- Tail prefix: []
- Recursion level: 0
• Tail prefix: []
• Recursion level: 0
• Look for edge labelled ‘jemand’ at root node
Parsing for S2T Decoding - Example

- Tail prefix: [jemand\textsubscript{1,1}]
- Recursion level: 0
- Look for edge labelled ‘jemand’ at root node - found
Parsing for S2T Decoding - Example

• Tail prefix: [jemand_{1,1}]
• Recursion level: 0
• Check for rules at current node - none
Parsing for S2T Decoding - Example

- Tail prefix: [jemand₁,₁]
- Recursion level: 0
- Now visit each cell along previous diagonal (recursive step)
Parsing for S2T Decoding - Example

- Tail prefix: \([\text{jemand}_{1,1}]\)
- Recursion level: 1
- Look for edge labelled ‘müßte’ at current node
Parsing for S2T Decoding - Example

- Tail prefix: [jemand_{1,1}, müßte_{2,2}]
- Recursion level: 1
- Look for edge labelled ‘müßte’ at current node - found
Parsing for S2T Decoding - Example

- Tail prefix: [jemand<sub>1,1</sub>, mußte<sub>2,2</sub>]
- Recursion level: 1
- Now visit each cell along previous diagonal
Parsing for S2T Decoding - Example

- Tail prefix: \([\text{jemand}_{1,1}, \text{mußte}_{2,2}]\)
- Recursion level: 2
- Look for edge labelled ‘Josef’ at current node
Parsing for S2T Decoding - Example

- Tail prefix: \([\text{jemand}_1,1, \text{mußte}_2,2]\)
- Recursion level: 2
- Look for edge labelled ‘Josef’ at current node - not found
• Tail prefix: [jemand₁,₁, müßte₂,₂]

• Recursion level: 2

• Look for edge labelled ‘NP’ at current node
Parsing for S2T Decoding - Example

- Tail prefix: \([\text{jemand}_{1,1}, \text{mußte}_{2,2}, \text{NP}_{3,4}]\)
- Recursion level: 2
- Look for edge labelled ‘NP’ at current node - found
• Tail prefix: [jemand$_1,1$, mußte$_2,2$, NP$_3,4$]
• Recursion level: 3
• And so on...
• Tail prefix: [jemand₁,₁, müßte₂,₂, NP₃,₄, VBN₅,₅]
• Recursion level: 3
• And so on...
• Tail prefix: \[jemand_{1,1}, \text{mu\ss}te_{2,2}, \text{NP}_{3,4}, \text{VBN}_{5,5}, \text{haben}_{6,6}\]

• Recursion level: 4

• And so on...
• Tail prefix: [jemand\(_{1,1}\), mußte\(_{2,2}\), NP\(_{3,4}\), VBN\(_{5,5}\), haben\(_{6,6}\)]

• Recursion level: 4

• At this point we add a PVertex for each LHS from trie node’s rule group
Parsing for S2T Decoding - Example

- Tail prefix: [jemand_{1,1}, müßte_{2,2}, NP_{3,4}, VBN_{5,5}, haben_{6,6}]
- Recursion level: 4
- At this point we add a PVertex for each LHS from trie node’s rule group
• Tail prefix: \([\text{jemand}_{1,1}, \text{mußte}_{2,2}, \text{NP}_{3,4}, \text{VBN}_{5,5}, \text{haben}_{6,6}]\)

• Recursion level: 4

• Together the PVertex and tail prefix constitute a complete PHyperedge.
• Tail prefix: \([\text{jemand}_{1,1}, \text{müßte}_{2,2}, \text{NP}_{3,4}, \text{VBN}_{5,5}, \text{haben}_{6,6}]\)

• Recursion level: 4

• Reached end of sentence, so now the recursion stack unwinds
• Tail prefix: \([\text{jemand}_{1,1}, \text{mußte}_{2,2}, \text{NP}_{3,4}, \text{VBN}_{5,5}]\)

• Recursion level: 3

• The recursion stack unwinds...
• Tail prefix: \([\text{jemand}_{1,1}, \text{mußte}_{2,2}, \text{NP}_{3,4}]\)

• Recursion level: 2

• The recursion stack unwinds...
• Tail prefix: [jemand_{1,1}, mußte_{2,2}]

• Recursion level: 1

• The parser continues trying to extend the tail...
• Tail prefix: [jemand\textsubscript{1,1}]

• Recursion level: 1

• The parser continues trying to extend the tail. . .
Parsing for S2T Decoding - Example

- Tail prefix: [jemand$_{1,1}$, VP$_{2,6}$]
- Recursion level: 1
- PVertex $S_{1,6}$ has already been added, but new tail means new PHyperedge
Decoding Performance in Practice

- S2T Moses system trained using all English-German data from WMT14

- Span limit can be used to reduce decoding time (limit is typically 10-15 for Hiero; can be higher or unlimited for S2T)
String-to-Tree Decoding - Summary

- Input sentence is a string.
- Decoding algorithm based on monolingual parsing.
- Hiero decoding is special-case of S2T decoding.
- To integrate a $m$-gram LM, the parse forest hypergraph is expanded to a (much-larger) search hypergraph.
- Heavy pruning is required in practice.
Tree-to-String Decoding
Reminder

- Translation rules are STSG rules with source-side syntax

```
PP-MP
  APPR ADJA [NN1]
    für britische
```

- Input is parse tree

```
TOP
  S-TOP
    PP-MO APPR ADJA NN ist PDS
      für britische Skandale
dieser
  AP-PD
    nicht besonders schlüpfing
```

Syntax-based Statistical Machine Translation 155
Outline

Objective
Find the $k$-best synchronous derivations $d_1, d_2, \ldots, d_k$

Outline
1. Project grammar
   Project weighted STSG to unweighted TSG $f : G \rightarrow G'$
2. Match rules
   Find rules from $G'$ that match input tree, record in match hypergraph
3. Search
   In post-order traversal of match hypergraph, build partial search hypergraph
4. Extract derivations
   Extract $k$-best derivations (Huang and Chiang, 2005)
Step 1: Project Grammar

- Take source-side of rule, ignore weights.
Step 2: Match Rules, Build Match Hypergraph

- Look for rules that match input tree
Step 2: Match Rules, Build Match Hypergraph

- For each matching rule, add hyperedge to match hypergraph
Step 2: Match Rules, Build Match Hypergraph

- Match hypergraph encodes forest of possible derivation trees from $G'$
Step 3: Build Partial Search Hypergraph

- Cube pruning algorithm produces SHyperedges from MHyperedges
- Translations not necessarily constituents (unlike S2T)
Step 3: Build Partial Search Hypergraph

• Vertices are stored in stacks, one per input tree node
The T2S Decoding Algorithm

1: build match hypergraph by matching grammar rules to input tree
2: for each m-vertex (post-order) do
3: for all incoming m-hyperedges do
4: create a cube for it
5: create first s-hyperedge in cube
6: place cube in queue
7: end for
8: for specified number of pops do
9: pop off best s-hyperedge of any cube in queue
10: add it to a buffer
11: create its neighbors
12: end for
13: recombine s-hyperedges from buffer and move into stack
14: sort and prune stack
15: end for
Rule Matching by DFA Intersection

- Rules are encoded as DFAs. Scheme here is from Matthews et al. (2014)
- Input tree encoded in same way.
- Standard DFA intersection algorithm produces rule match hypergraph.
Tree-to-String Decoding - Summary

- Input sentence is a parse tree.
- Tree constrains rule choice: much smaller search space than S2T.
- Decoding algorithm based on rule matching with LM integration.
- LM integration identical to S2T.
A Sketch of Tree-to-Tree Decoding

- STSG with tree input.

- T2T decoding is combination of S2T and T2S:
  - Search state expanded to include target-side category
  - Rule matching used to select rules; further constrained by target categories
  - Multiple category-specific stacks per input tree node
  - LM integration identical to S2T / T2S.

- Exact T2T not widely used in practice due to syntactic divergence.
Part I - Introduction
Part II - Rule Extraction
Part III - Decoding
Part IV - Extensions
“Fuzzy” Syntax

• In a nutshell: move syntax out of grammar and into feature functions
  – Syntax becomes a soft constraint
  – Motivated by syntactic divergence problem in tree-to-tree model

• “Learning to Translate with Source and Target Syntax” (Chiang, 2010)
  – Zhang et al (2011) use fuzzy syntax on source-side of string-to-tree model and explore alternative feature functions
“Fuzzy” Syntax

- Parse trees on both sides of training data
- Uses Hiero rule extraction but with SAMT-style labelling

- Only most frequent labelling kept (one-to-one correspondence with Hiero rules)

Syntax-based Statistical Machine Translation 169
“Fuzzy” Syntax

- Rule labels not used during parsing but retrieved for search

- Feature functions score substitutions
  - e.g. if a NP is rewritten as a ADJA+NN on source side then the feature subst\textsubscript{NP→ADJA+NN} fires

- Tens of thousands of features

- Outperforms exact tree-to-tree (0.4 BLEU on Zh-En; 1.5 BLEU on Ar-En)
Forest-to-String

- Translation quality of T2S model depends on accuracy of 1-best (or k-best) parse tree(s) for input sentences
- Forest-to-string extends T2S by using (pruned) parse forest as input

- Algorithm is identical to T2S except for rule matching step
- “Forest-based Translation” (Mi et al., 2008)
Forest-to-String

- Using forest gives better speed-quality trade-off than using $k$-best trees

(Figure taken from Mi et al., 2008)
Tree Transformation

- Adapting training data for syntax-based MT is an active area of research (tree binarization, label coarsening / refinement, word alignment edits).

- “Transforming Trees to Improve Syntactic Convergence” (Burkett and Klein, 2012) proposes a tree restructuring method to improve rule extraction.

(Figure taken from Burkett and Klein, 2012)
Tree Transformation

- Defines six classes of transformation

\[
\begin{align*}
\text{Type: ARTICULATE} & \quad \text{Args: A: PARENT, B: LEFT, C: RIGHT} \\
\text{Type: FLATTEN} & \quad \text{Args: A: PARENT, B: TARGET}
\end{align*}
\]

- Error-based learning method using GHKM frontier node count as metric

- Sequence of transformations learned from subset of training data then applied to full corpus

- Gain of 0.9 BLEU over baseline on Chinese to English; outperforms simple left and right binarization
Dependency

A different view on syntax

SCFG phrase structure

```
S
  NP
    DT NN
    the dog
  VP
    V
    chews
    DT NN
    a bone
```

Syntactic dependency grammar

```
DET DET
SUBJ
OBJ
the dog chews a bone
```

Syntax-based Statistical Machine Translation 175
Phrase Structure is not Enough

syntactically well-formed

semantically implausible
Dependency in SCFG

• Add head word to constituents

```
S(chews)                  VP(chews)
|                  |
NP(bone)               NP(dog)
|      |      |      |
DT     NN     V     DT     NN
  the   bone   chews  a   dog
```

• Add mapping of head words to rules

\[ \text{VP}(w_1) \rightarrow V(w_1) \text{ NP}(w_2) \]

requires identification of head child
Semantic Plausibility

Score each lexical relationship

• Rule: $VP(\text{chews}) \rightarrow V(\text{chews}) \ NP(\text{dogs})$
  - Feature: $VP(\text{chews}) \rightarrow V-\text{HEAD}(\text{chews})$ OK
  - Feature: $VP(\text{chews}) \rightarrow NP(\text{dog})$ BAD

• Rule: $S(\text{chews}) \rightarrow NP(\text{bone}) \ VP(\text{chews})$
  - Feature: $S(\text{chews}) \rightarrow NP(\text{bone})$ BAD
  - Feature: $S(\text{chews}) \rightarrow V-\text{HEAD}(\text{chews})$ OK
Informed by Source

- Languages with case marking
  - different word order
  - same dependency relationships

- Give preference to translations that preserve dependency relationships
Verb Frames

- Check if full verb frame is properly filled
  - intransitive / transitive / ditransitive
  - not just binary relationships
  - appropriate type of subjects / objects

- However: tracking verb frame is not trivial
Towards Semantics

• Different syntax — same verb-noun semantic relationships
  – The bone is chewed by the dog.
  – The dog chews the bone.
  – The bone, the dog chews.
  – A dog chewed a bone.

• Even more abstract representations
  e.g., Abstract Meaning Representation (AMR):

  (c / chew-01
   :arg0 (d / dog)
   :arg1 (b / bone))

• Generation of these types of representation open research problem
String-to-Dependency: Shen et al. (2008)

- Hiero rules but with unlabelled dependencies on target side
- Target-side allowed one head to which floating dependencies can attach

```
  r1  X    X1  flog nach X2  |X1  flew  to X2  Fixed

  r2  X    flog nach X1  |flew  to X1  Fixed

  r3  X    nach X1  |to X2  Floating (left)

  r4  X    flog nach  |flew  to   Ill-formed
```

- “A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model” (Shen et al., 2008)
String-to-Dependency

- Decoding algorithm modified to combine dependency structures.

- Restriction to well-formed rules reduces grammar size from 140M to 26M rules (no significant effect on translation quality).

- Gains of 1.2 BLEU on Zh-En from addition of dependency LM (Markov model over dependency heads).
References

- Parsing and Hypergraphs

- What’s in a Translation Rule?
  Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. HLT-NAACL 2004.

- A Hierarchical Phrase-based Model for Statistical Machine Translation
  David Chiang. ACL 2005.

- Better k-best Parsing
  Liang Huang and David Chiang. IWPT 2005.

- Syntax Augmented Machine Translation via Chart Parsing
  Andreas Zollmann and Ashish Venugopal. WMT 2006.

- Synchronous Binarization for Machine Translation
  Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. NAACL 2006.

- Hierarchical Phrase-Based Translation
References

• A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model

• Forest-Based Translation
  Haitao Mi, Liang Huang, and Qun Liu. ACL 2008.

• Efficient Parsing for Transducer Grammars
  John DeNero, Mohit Bansal, Adam Pauls, and Dan Klein. NAACL 2009.

• SCFG Decoding Without Binarization
  Mark Hopkins and Greg Langmead. EMNLP 2010.

• Learning to Translate with Source and Target Syntax
  David Chiang, ACL 2010.

• Issues Concerning Decoding with Synchronous Context-free Grammar
  Tagyoung Chung, Licheng Fang, and Daniel Gildea. ACL 2011.

• Transforming Trees to Improve Syntactic Convergence
  David Burkett and Dan Klein. EMNLP 2012.
References

- *Grouping Language Model Boundary Words to Speed K-Best Extraction from Hypergraphs*
  Kenneth Heafield, Philipp Koehn, and Alon Lavie. NAACL 2013.

- *Tree Transduction Tools for cdec*

- *A CYK+ Variant for SCFG Decoding Without a Dot Chart*
  Rico Sennrich. SSST 2014.