



Copenhagen
Business School
HANDELSHØJSKOLEN

Breaking the barrier of context-freeness. Towards a linguistically adequate probabilistic dependency model of parallel texts

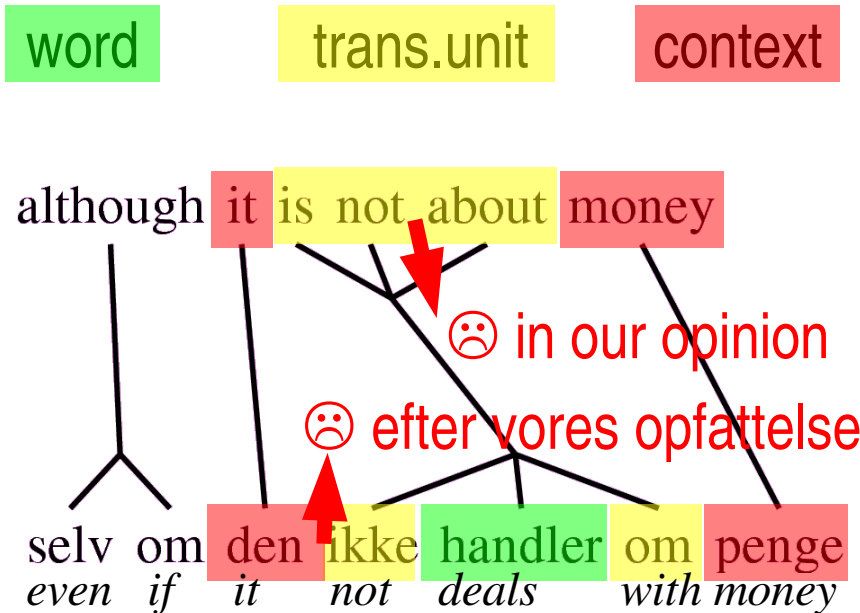
Matthias Buch-Kromann
Copenhagen Business School

Theoretical and Methodological Issues in Machine Translation
Skövde, Sept. 9, 2007



Phrase-based SMT (state-of-art)!

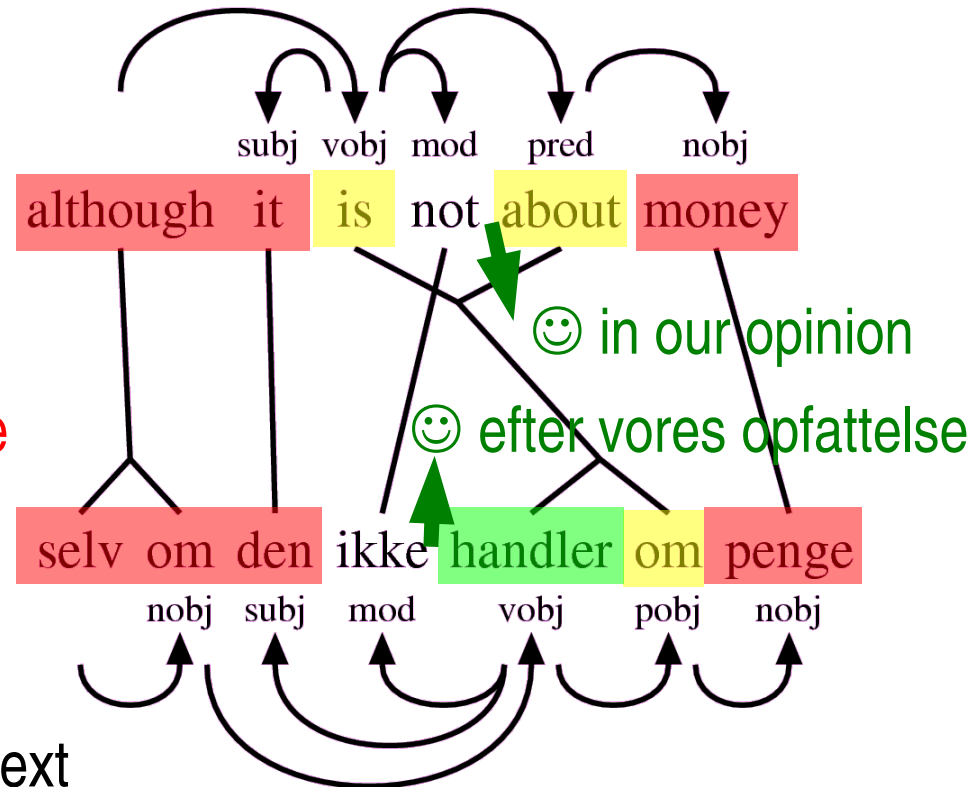
- context = aligned words, adjacent word sequences (“phrases”)



- => smaller translation units
- => more linguistically relevant context

Or syntax-based SMT?

- context = aligned words, governor, complements





1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 1.1. Phrase-based vs. syntax-based SMT
- 1.2. The purpose of this talk
- 1.3. The vision
- 1.4. The setting: our abstract SMT model

Background: Most syntax-based SMT uses context-free formalisms where sentences are always projective (no crossing branches). However, as observed by Nilsson et al (2007), 11-34% of sentences in CoNLL dep. treebanks for Slovene, Arabic, Dutch, Czech, Danish are non-projective.

Problem: Need linguistically realistic SMT models that can deal with non-projectivity, island constraints, complement-adjunct distinction, deletions and additions, translational divergences such as head-switching, etc.

This talk: Define a probabilistic dependency model that attempts to do this (as a first step, not a final solution).

Not this talk: Specify algorithms for model learning, translation and parallel parsing (ideas, see paper). Report experimental results (no implementation yet, work in progress).



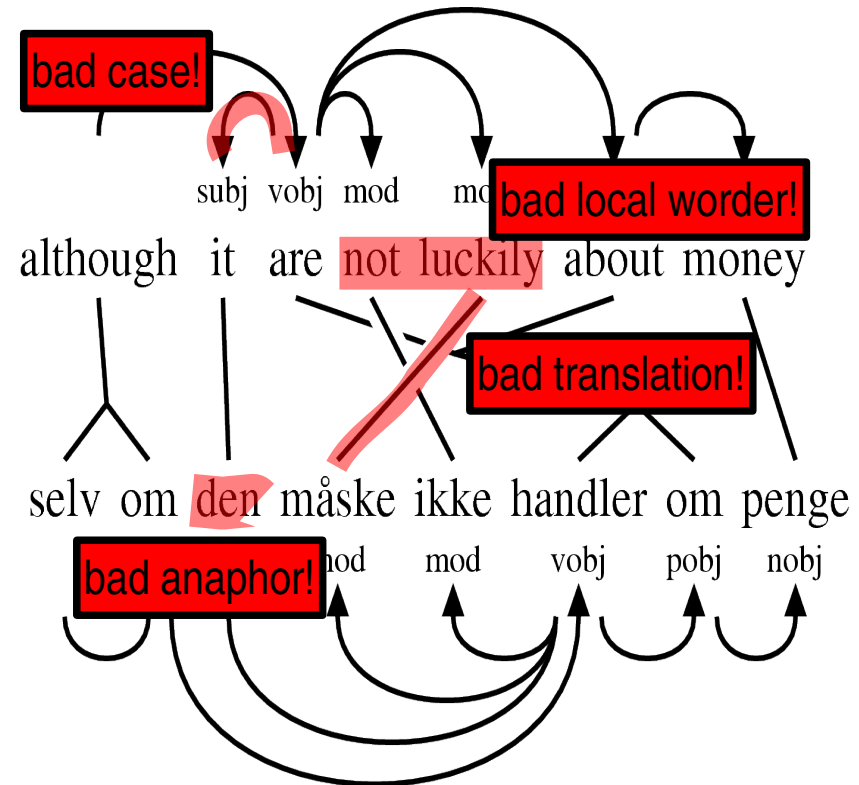
1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 1.1. Phrase-based vs. syntax-based SMT
- 1.2. The purpose of this talk
- 1.3. The vision
- 1.4. The setting: our abstract SMT model

The vision: Create a probabilistic generative dependency model that **assigns local probabilities** to each generative step in a parallel analysis.

“Unusually” low elicited local probabilities indicate **localized grammatical errors** => model makes verifiable linguistic predictions about localized errors (one evaluation criterion in future SMT).

Computation is performed by local structure-changing repair-operations that are **guided by the errors in the analysis**, as in human processing.





1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 1.1. Phrase-based vs. syntax-based SMT
- 1.2. The purpose of this talk
- 1.3. The vision
- 1.4. The setting: our abstract SMT model

Dependency model: A generative probability measure P on the space Ana of all conceivable parallel dependency analyses.

Model learning: estimate P from a given parallel dependency treebank T (and possibly a given parallel corpus C as well).

Translation: Given source text t , compute:

$$\text{Translate}(t) = \operatorname{argmax}_{A \in \text{Ana}, Y(A)=t} P(A)$$

$Y(A)$ = source text of A . $Y'(A)$ = target text of A .

Parallel parsing: Given source and target texts t, t' , compute:

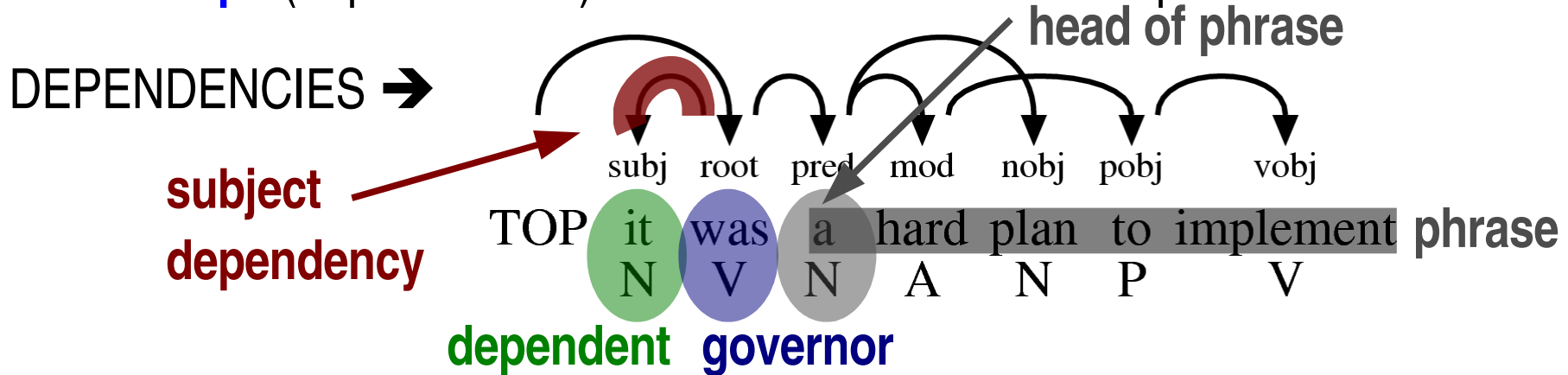
$$\text{Parse}(t, t') = \operatorname{argmax}_{A \in \text{Ana}, Y(A)=t, Y'(A)=t'} P(A)$$



1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 2.1. Deep trees (dependencies)
- 2.2. Parallel dependency analyses
- 2.3. Syntactic translation units
- 2.4. Surface trees (landing sites)
- 2.5. Word order control
- 2.6. Extraction paths and island constraints

A **dependency tree (deep tree)** encodes **complement or adjunct relationships** (dependencies). Known from CoNLL. Example:



Phrases are a derived notion: each word heads a phrase consisting of all the words that can be reached from the word by following the arrows.

dobj	direct object
iobj	indirect object
mod	modifier
nobj	nominal object

pobj	prep. object
pred	predicative
rel	relative
root	root node

subj	subject
vobj	verbal object



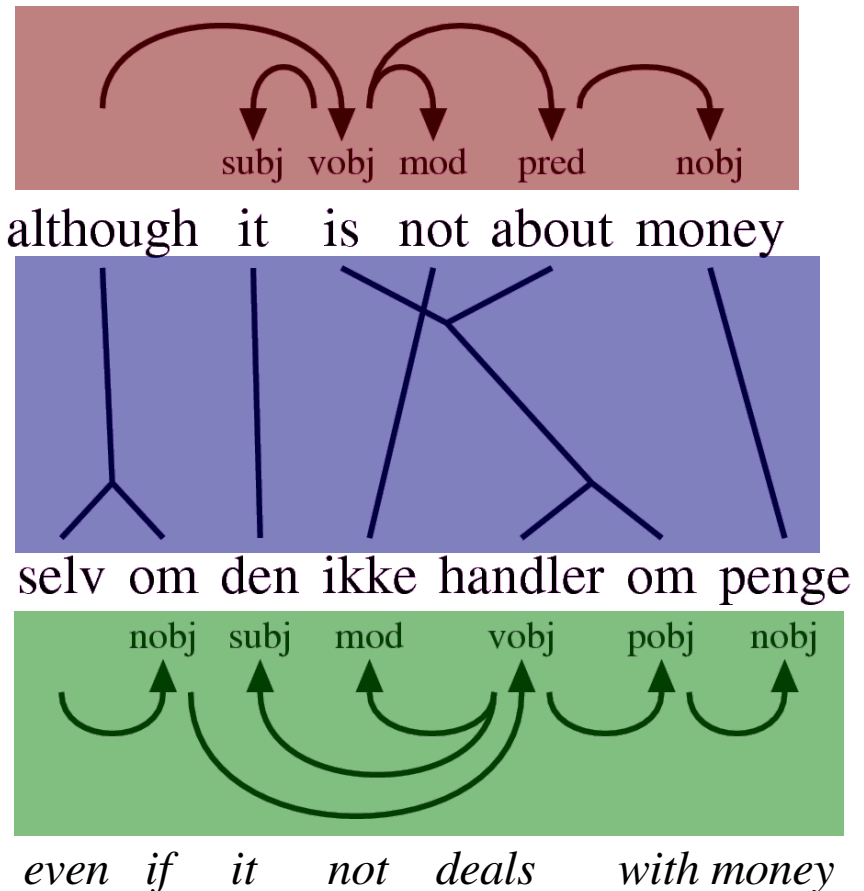
1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 2.1. Deep trees (dependencies)
- 2.2. Parallel dependency analyses
- 2.3. Syntactic translation units
- 2.4. Surface trees (landing sites)
- 2.5. Word order control
- 2.6. Extraction paths and island constraints

Parallel dependency analyses

consist of three components:

- a dependency analysis of the source text
- a dependency analysis of the target text
- a word alignment linking corresponding words in the source and target texts (lexical translation units)





1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

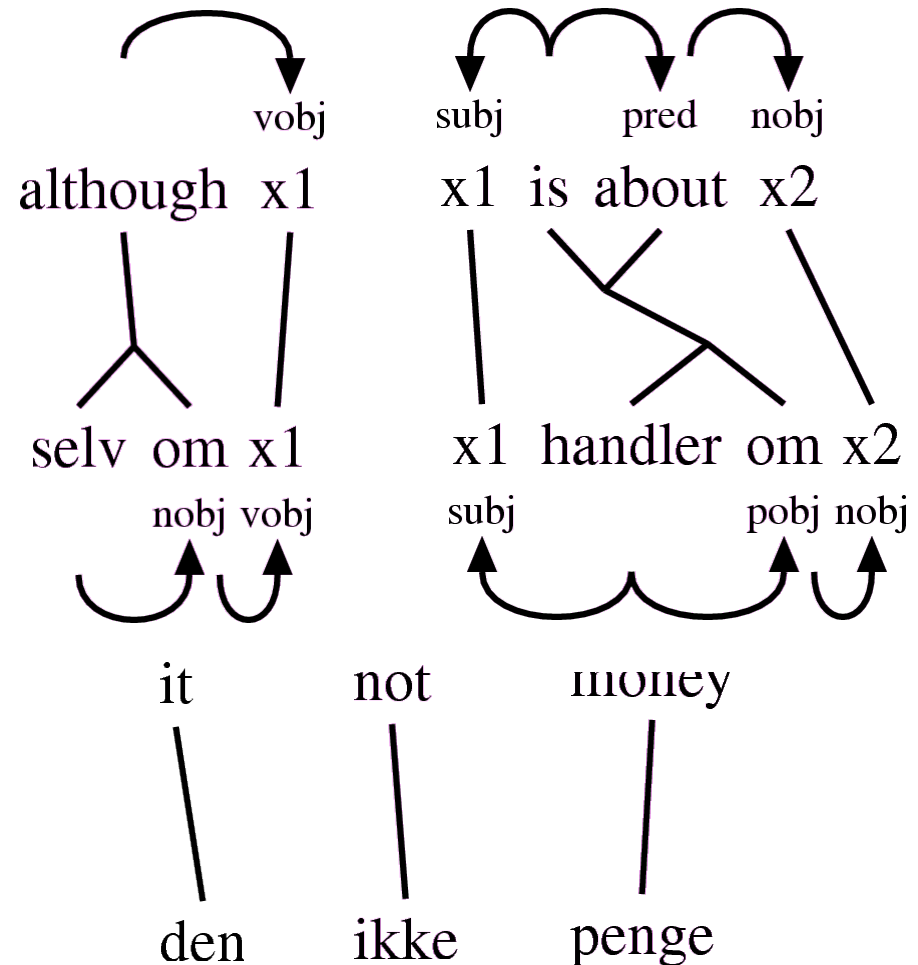
- 2.1. Deep trees (dependencies)
- 2.2. Parallel dependency analyses
- 2.3. Syntactic translation units
- 2.4. Surface trees (landing sites)
- 2.5. Word order control
- 2.6. Extraction paths and island constraints

Syntactic translation units can be computed from parallel dependency treebanks (Buch-Kromann, 2007a).

If word alignments are inconsistent with parses, **merging** can make them consistent. Eg, head-switching, relation-changing, etc.

The resulting syntactic translation units **can be much larger than original alignments** (2-50 nodes).

Small treebanks can be used to **bootstrap** large treebanks.



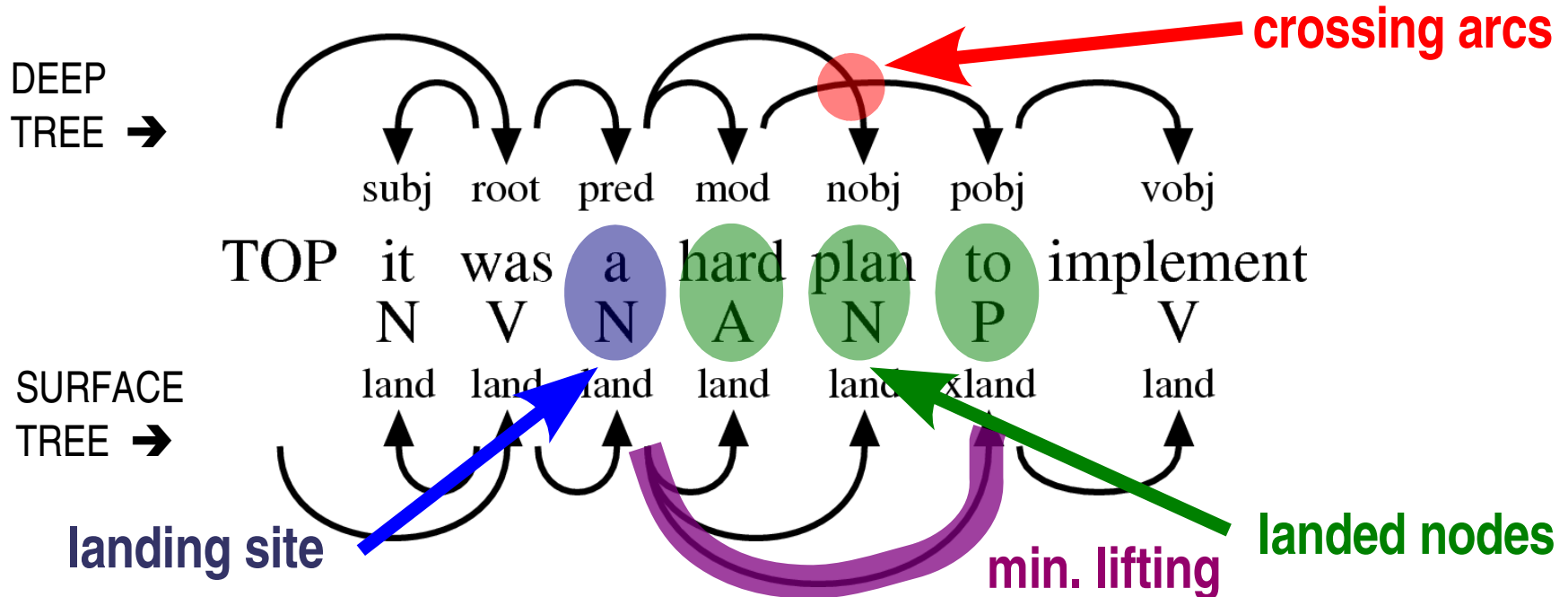
tunits are unordered!



1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 2.1. Deep trees (dependencies)
- 2.2. Parallel dependency analyses
- 2.3. Syntactic translation units
- 2.4. Surface trees (landing sites)
- 2.5. Word order control
- 2.6. Extraction paths and island constraints

Deep trees may have crossing arcs. But in order to control word order, we need a **surface tree** without crossing arcs. Example:



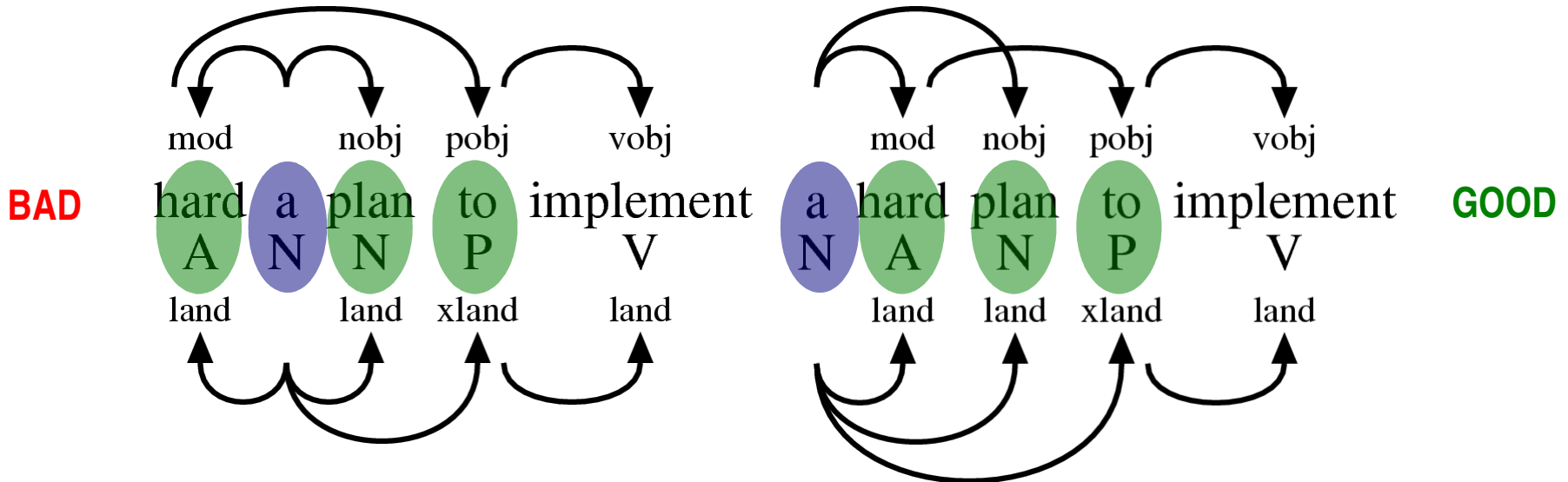
Parents in the surface tree are called **landing sites**, and children are called **landed nodes**. Landing site = lowest transitive governor that dominates all words between node and governor.



1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 2.1. Deep trees (dependencies)
- 2.2. Parallel dependency analyses
- 2.3. Syntactic translation units
- 2.4. Surface trees (landing sites)
- 2.5. Word order control
- 2.6. Extraction paths and island constraints

Landing sites allow us to **control the local word order** by looking at the relative order of the landed nodes at each landing site. Examples:



The left example is bad because the landed node "hard" precedes the landing site "a".

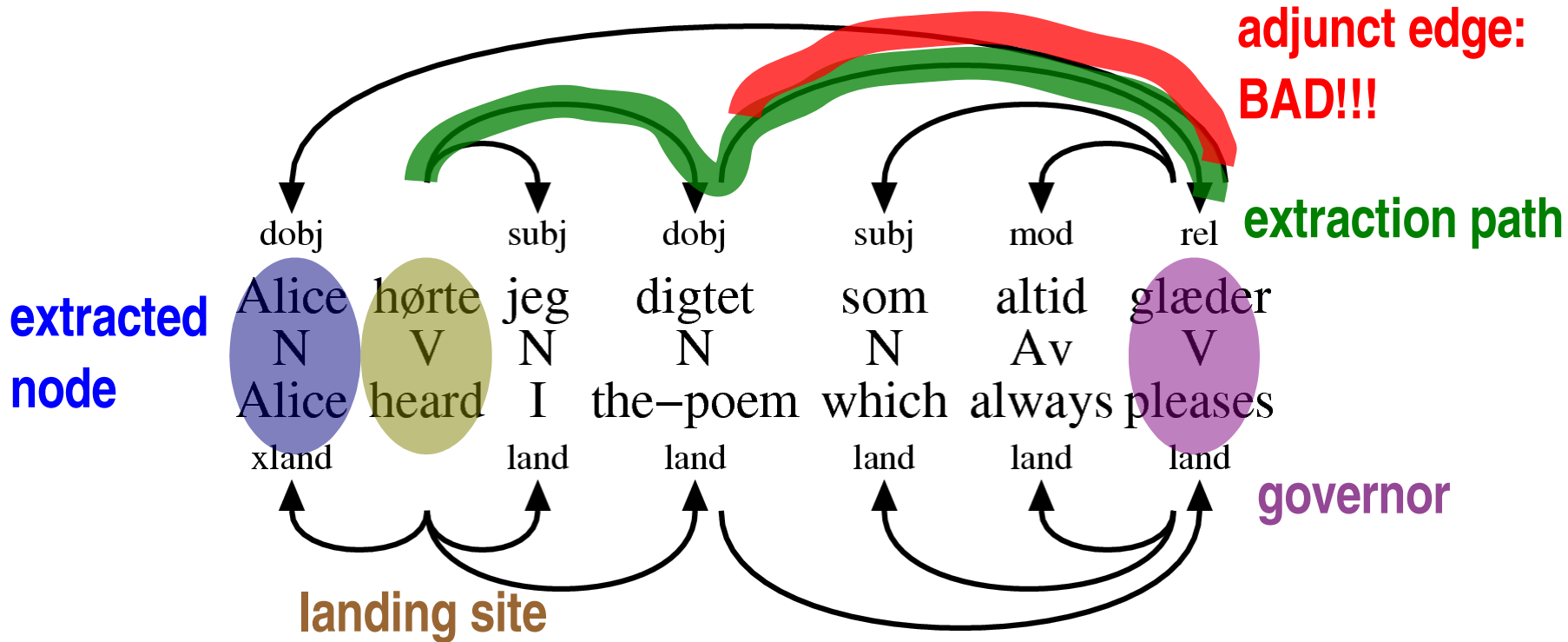
The right example is ok wrt. the order of the landed nodes at "a".



1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 2.1. Deep trees (dependencies)
- 2.2. Parallel dependency analyses
- 2.3. Syntactic translation units
- 2.4. Surface trees (landing sites)
- 2.5. Word order control
- 2.6. Extraction paths and island constraints

The **extraction path** is the path from a word's governor to its landing site. By looking at this path, we can determine whether any **island constraints** are violated. Eg, the **adjunct island constraint** is violated below.



"I heard the poem which always pleases Alice."



1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 3.1. Generative procedure
- 3.2. T1 Select landing sites and word order
- 3.3. T2 Translate arguments of translation unit
- 3.4. T3 Delete singular source adjuncts
- 3.5. T4 Translate parallel adjuncts
- 3.6. T5 Add singular target adjuncts

The dependency model is **generative**, ie, it is based on a **Markov process** (like Collins 1997).

The graph is grown in a **recursive top-down manner** – first the source analysis, then the target analysis. Each step is emitted with a certain probability.

procedure probabilistic graph generation

begin

recursively expand source root TOP (cf. Figure 7)

recursively translate source root TOP (cf. Figure 8)

return *generated graph and probability*

end

~~**Top-down expansion of source node w_i**~~

~~S1. Identify landing site and relative word order~~

~~S2. Select complement source~~

~~S3. Generate and recursively expand complements~~

~~S4. Generate and recursively expand adjuncts~~

Top-down translation of source node w_i

T1. Identify landing sites and word order in target tunit

T2. Generate and recursively expand tunit arguments

T3. Identify deleted source adjuncts

T4. Generate and recursively translate parallel adjuncts

T5. Generate added target adjuncts

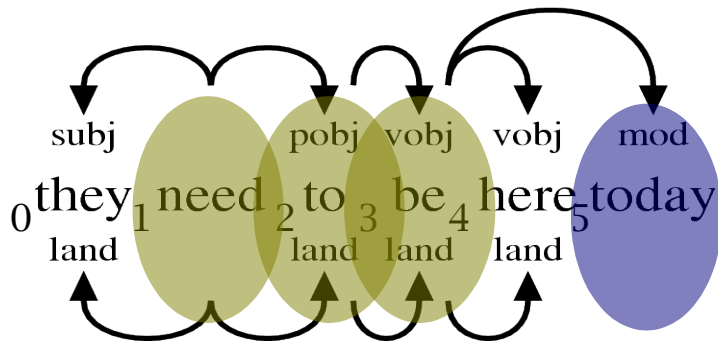


1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

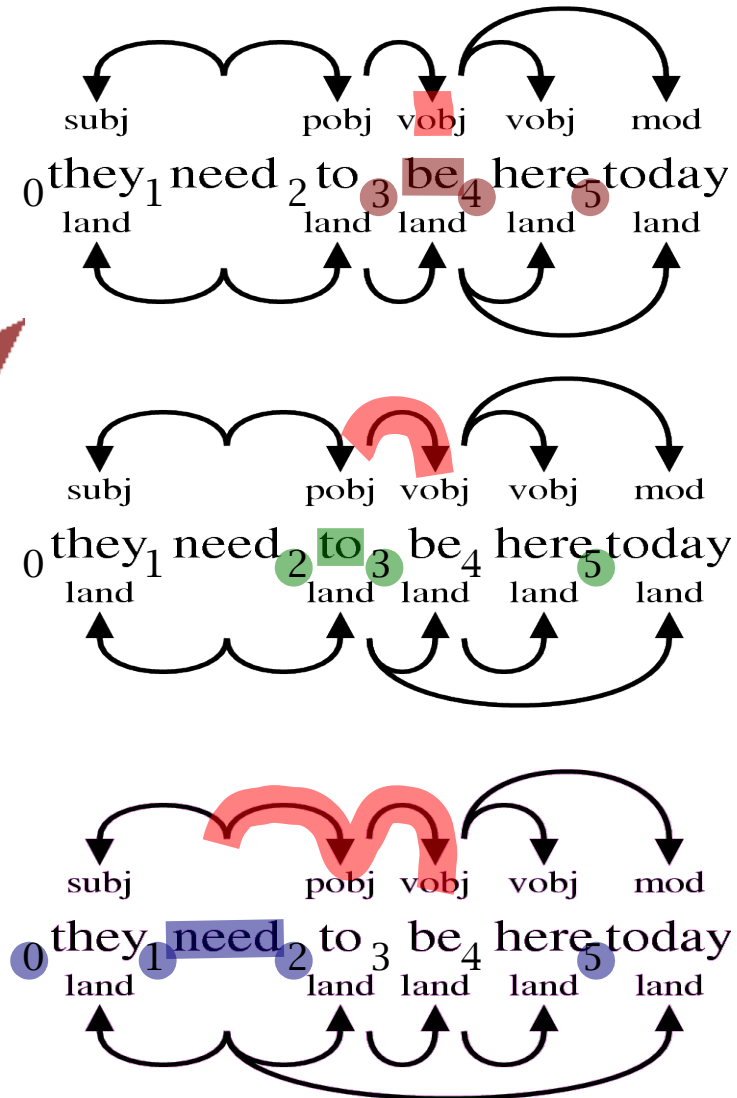
- 3.1. Generative procedure
- 3.2. T1 Select landing sites and word order
- 3.3. T2 Translate arguments of translation unit
- 3.4. T3 Delete singular source adjuncts
- 3.5. T4 Translate parallel adjuncts
- 3.6. T5 Add singular target adjuncts

S1/T1. Select landing site and word order.

Eg, we must choose a **landing site** for “today” among its transitive governors (“be”, “to”, “need”) and a **valid position** (0-5) relative to the other landed nodes at the landing site.



The extraction path needs to be checked for **blocking edges** that prevent the extraction.





1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 3.1. Generative procedure
- 3.2. T1 Select landing sites and word order
- 3.3. T2 Translate arguments of translation unit
- 3.4. T3 Delete singular source adjuncts
- 3.5. T4 Translate parallel adjuncts
- 3.6. T5 Add singular target adjuncts

Relative **probability of extraction path**:

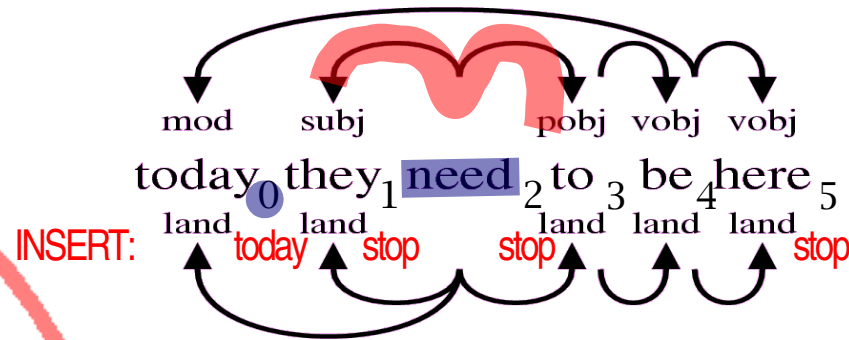
$$\min \left(1, \frac{P_{\text{exptpath}}(\text{to} \xleftarrow{\text{vobj}} \text{be} | \text{today}^*)}{P_{\text{deptree}}(\text{to} \xleftarrow{\text{vobj}} \text{be})} \right)$$

$$\cdot \min \left(1, \frac{P_{\text{exptpath}}(\text{need} \xleftarrow{\text{pobj}} \text{to} | \text{today}^*)}{P_{\text{deptree}}(\text{need} \xleftarrow{\text{pobj}} \text{to})} \right)$$

Rel. prob. of chosen local **word order**:

$$P_{\text{worder}}(\text{today} | c_{\text{need},0}) \prod_{\omega=1,2,5} P_{\text{worder}}(\text{STOP} | c_{\text{need},\omega})$$

Finally **normalize** so that probabilities of all choices of landing site and local word order sum to 1.



Probability of finding edge in extraction path given extr. word.

Probability of finding edge in dependency tree.

Ratio > 1: above chance-level
(with prob. 1 don't block)
Ratio < 1: below (blocking edge)

$c_{\text{need},0}$ is the local word order context in position 0 at "need".



1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 3.1. Generative procedure
- 3.2. T1 Select landing sites and word order
- 3.3. T2 Translate arguments of translation unit
- 3.4. T3 Delete singular source adjuncts
- 3.5. T4 Translate parallel adjuncts
- 3.6. T5 Add singular target adjuncts

T2 Translate arguments of a tunit:

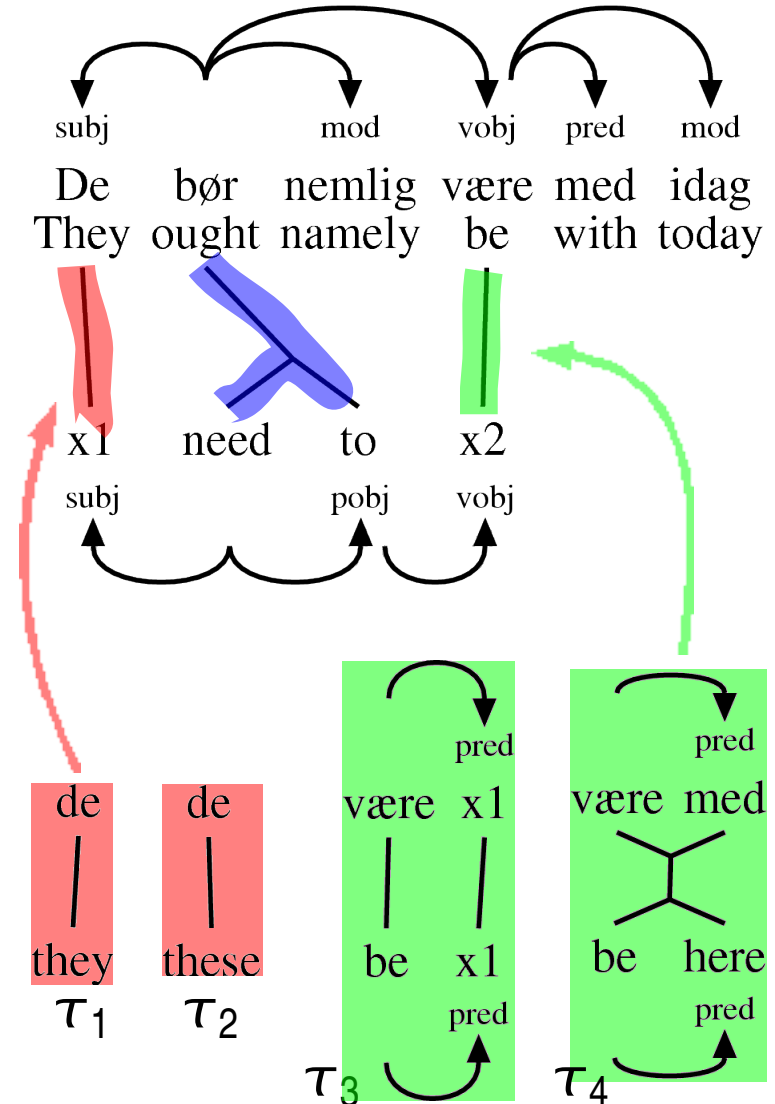
select tunits for the arguments, striking a compromise btw. adequacy and fluency.

Adequacy: How probable is the target argument dependency structure as a translation of the source arg. dep. structure?

$$A(\text{være}, \tau_4) = P_{\text{tunit}}(\tau_4 | \text{være}^*)$$

Fluency: How well does the target arg. structure fit into existing target structure?

$$F(\text{være}, \tau_4) = P_{\text{comp}'}(\text{be} | \text{vobj}, x_2^*) \cdot P_{S'_{234}}(\text{be}^{(\text{pred} \text{ here})})$$





1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 3.1. Generative procedure
- 3.2. T1 Select landing sites and word order
- 3.3. T2 Translate arguments of translation unit
- 3.4. T3 Delete singular source adjuncts
- 3.5. T4 Translate parallel adjuncts
- 3.6. T5 Add singular target adjuncts

Compromise: Weigh adequacy and fluency with some parameter λ (say 1/2), and normalize:

$$P_{\text{transfer}}(\text{være}, \tau_4) = \frac{A^{1/2} F^{1/2}}{\text{normalization}}$$

Iterate through all arguments:

$$P_{\text{transfer}}(\text{de}, \tau_1) \cdot P_{\text{transfer}}(\text{være}, \tau_4)$$



1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 3.1. Generative procedure
- 3.2. T1 Select landing sites and word order
- 3.3. T2 Translate arguments of translation unit
- 3.4. T3 Delete singular source adjuncts
- 3.5. T4 Translate parallel adjuncts
- 3.6. T5 Add singular target adjuncts

T3. Delete singular source adjuncts

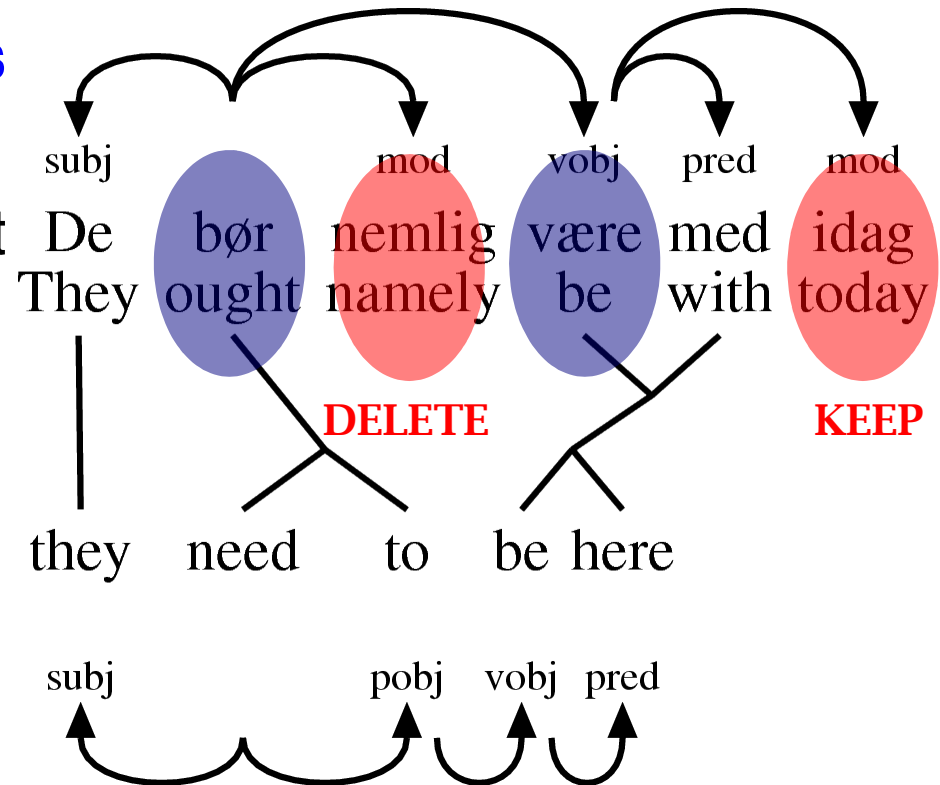
To **model deletions** in the translation, we must decide for each adjunct phrase whether we will delete it.

Eg, when translating “være” (“be”), we choose to **keep** “idag” (“today”).

$$P_{\text{del}}(\text{KEEP}|\text{idag}^*)$$

Later on at “bør” (“ought”), we decide to **delete** “nemlig” (“namely”).

$$P_{\text{del}}(\text{DELETE}|\text{nemlig}^*)$$





1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 3.1. Generative procedure
- 3.2. T1 Select landing sites and word order
- 3.3. T2 Translate arguments of translation unit
- 3.4. T3 Delete singular source adjuncts
- 3.5. T4 Translate parallel adjuncts
- 3.6. T5 Add singular target adjuncts

T4. Translate parallel adjuncts

To translate the non-deleted adjunct “idag” (“today”), we must:

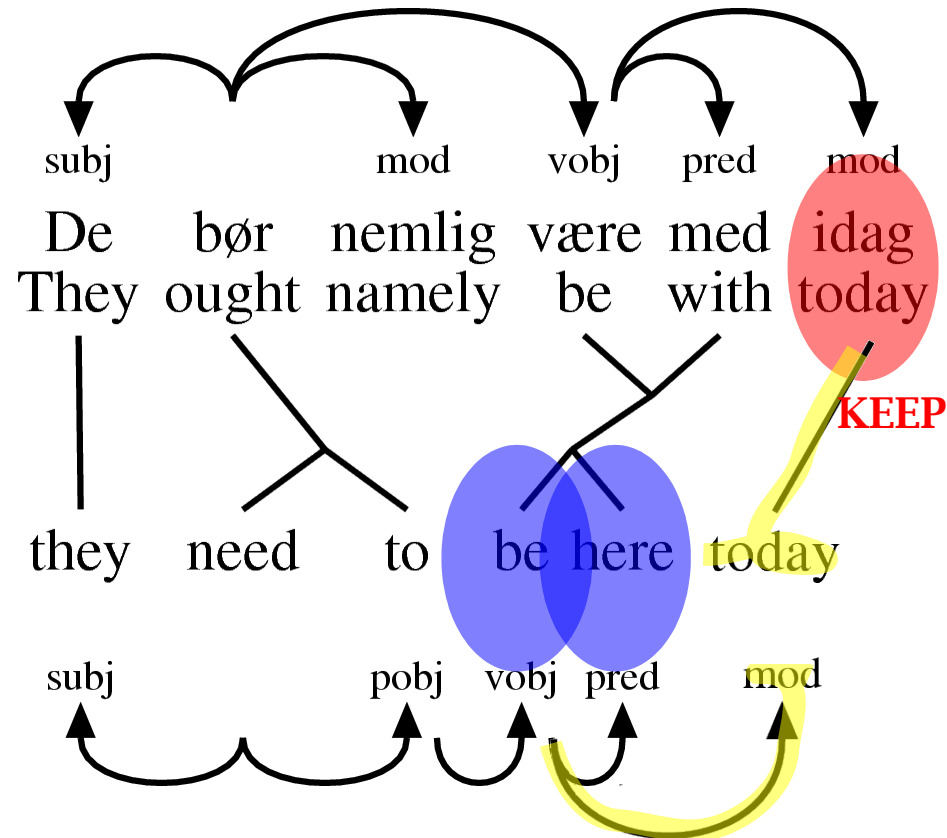
(a) **select governor and adjunct role** within the target unit “be(predhere)”.

Eg, select “be” + adjunct role “mod”:

$$\frac{P_{\text{adjtrans}}(\text{be}, \text{mod} | \text{idag}^*, \text{mod})}{P_{\text{adjtrans}}(\text{be}, \text{mod})}$$

normalization

(b) **translate** “idag” (“today”) by selecting a translation unit for it, emitting probabilities as in step T2.





1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 3.1. Generative procedure
- 3.2. T1 Select landing sites and word order
- 3.3. T2 Translate arguments of translation unit
- 3.4. T3 Delete singular source adjuncts
- 3.5. T4 Translate parallel adjuncts
- 3.6. T5 Add singular target adjuncts

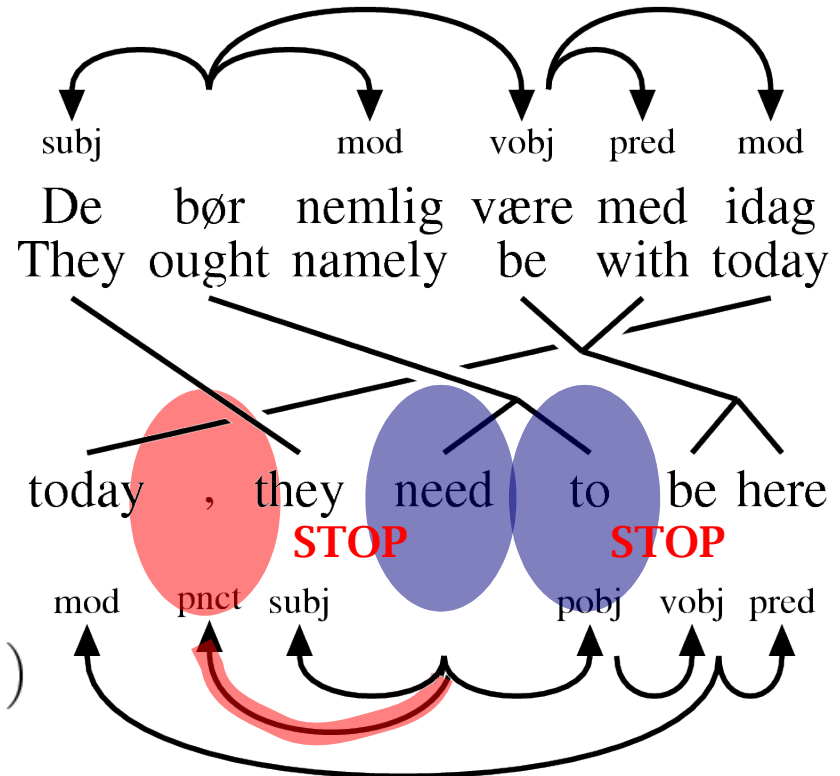
T5. Add singular target adjuncts

Generate **singular target adjuncts** (adjunct role first, then adjunct tree) and then STOP for each node in the target unit in target top-down order.

Eg, for “være” (“be”), generate “,” and STOP for “need”, and STOP for “to”.

$$P_{\text{add-arole}}(\text{pnct}|\text{need}^*)P_{\text{add-adj}}(,|\text{need}^*)$$

- $P_{\text{add-arole}}(\text{STOP}|\text{to}^*)$
- $P_{\text{add-arole}}(\text{STOP}|\text{need}^*)$





Conclusion: I have:

- argued that it is important to **develop syntax-based SMT** models;
- argued that context-free SMT models **cannot model non-projectivity**;
- proposed a **dependency-based model** that can handle non-projectivity, island constraints, complement-adjunct distinction, deletions and additions (and head-switching, see extra stuff) – at least in theory.

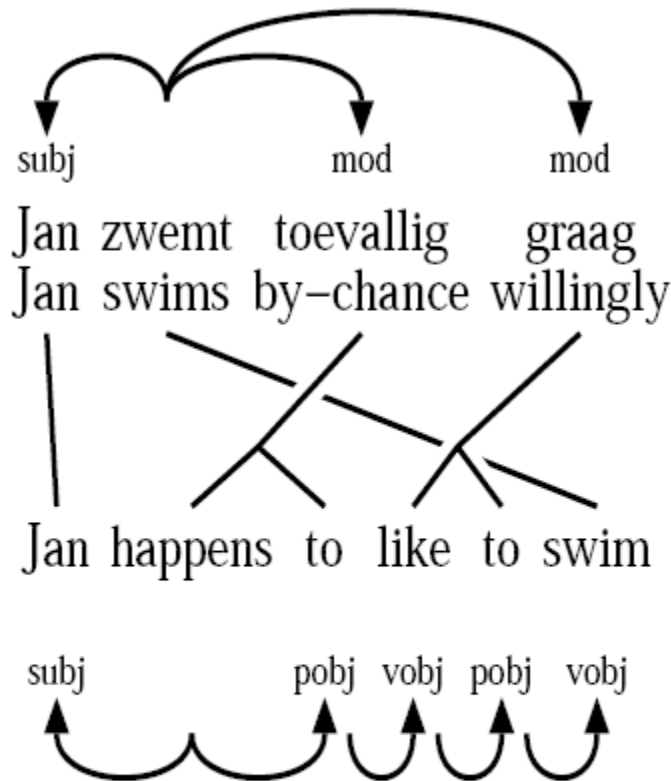
Future work: Much still remains to be done:

- build a **functioning SMT system** and evaluate it
- **extend the model** wrt morphology, syntax and discourse structure

THANKS FOR LISTENING



Lexical tunit alignments



Syntactic tunit alignments

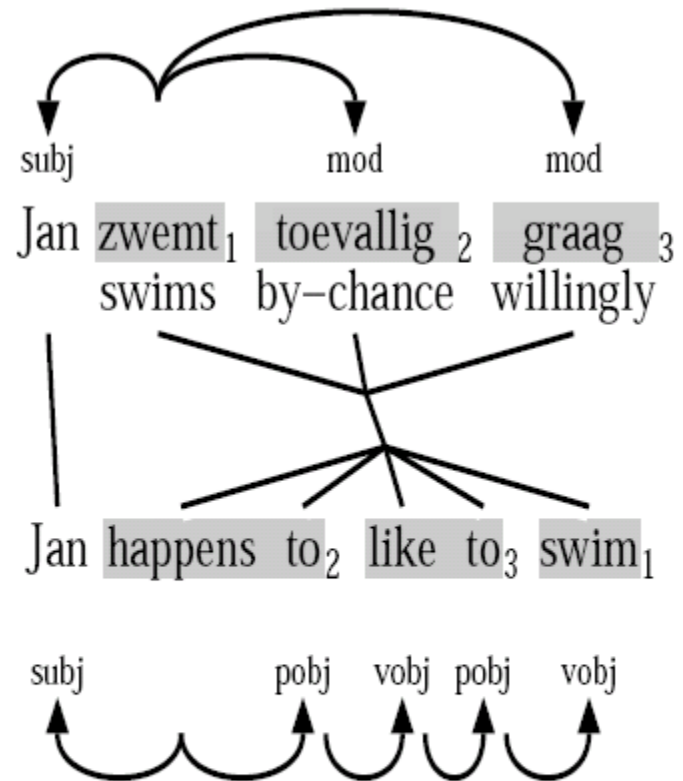
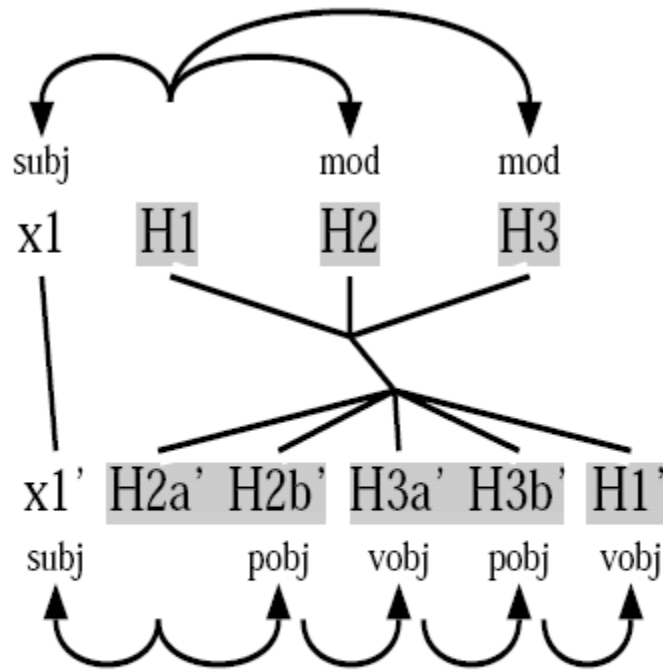


Figure 4: A head-switching example (left) and the associated minimal reduction (right).



Source match

$H_1 = \text{zwemt}$

$H_2 = \text{toevallig}$

$H_3 = \text{graag}$

Target match

$H'_1 = \text{swims}$

$H'_{2a,b} = \text{happens}(\text{pobj to})$

$H'_{3a,b} = \text{like}(\text{pobj to})$

Figure 5: Syntactic translation template induced from Figure 4, with source and target match.



- **Time complexity (exact):** Exact translation and parallel parsing with our language model is NP-hard (Buch-Kromann, 2006).
- **Time complexity (heuristic):** For monolingual parsing, it is possible to find heuristic algorithms based on local search with time complexity $O(n \log^k n)$ (cf. Buch-Kromann 2006, 2007b).
- **Algorithm:** Start with the empty analysis, construct analysis incrementally by applying local probability-increasing parsing operations that introduce new words, or create or modify dependency edges.
Alternative for translation: parse, then use generative procedure greedily.
- **Errors:** Probabilistic dependency model can be used to pinpoint errors in the analysis (error = unusually low probability in generative step).
- **Error-driven operations:** A chain of elementary operations each of which corrects an error in the graph introduced by the previous operation.



1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 5.1. Head-switching
- 5.2. Translation and parallel parsing
- 5.3. Statistical estimation
- 5.4. Parallel dependency treebanks

- **Statistical estimation:** Use your **favorite statistical methods**, eg, Generalized Linear Models and Generalized Additive Models (log-linear methods are a special case).
- **Correction estimators:** In steps S1, T1, and T4 of the model, we need to estimate a **posterior divided by a prior** ($P(x|y) / P(x)$). Thus a correction estimator, where we first estimate $P(x)$ and then $C(x|y) = P(x|y) / P(x)$, may be particularly useful for this purpose – eg, the XHPM method proposed by (Buch-Kromann, 2006), which is a generalization of (Li and Abe, 1997).



1. Dependency-based SMT
2. Parallel dependency analyses
3. Generative dependency model
4. Conclusion
5. Extra stuff

- 5.1. Head-switching
- 5.2. Translation and parallel parsing
- 5.3. Statistical estimation
- 5.4. Parallel dependency treebanks

Examples of parallel dependency treebanks include:

**The Prague Czech-English
Dependency Treebank.**

**The Copenhagen Danish-English
Dependency Treebank.**

**Constituency-based parallel
treebanks converted to
dependency representation.**

**Any pair of monolingual treebanks
with the same underlying corpus.**

