Syntax in SMT

Ondřej Bojar

April 11, 2019




EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research,
Development and Education

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Outline

- Motivation for grammar in MT.
- Constituency vs. dependency trees.

Constituency Syntax:

- Context Free Grammars.
- MT as parsing.
- Hierarchical phrase-based model (Hiero, Joshua).
    - Synchronous CFG, LM integration.
- Using real source/target parse trees.
    - Tricks to avoid data loss.

Dependency Syntax:

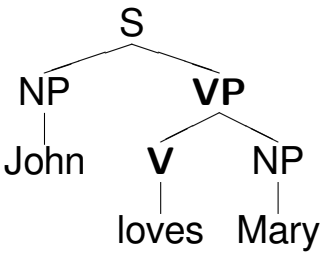
- Surface syntax (STSG), problems.
- Deep syntax (t-layer); factorization is a must.

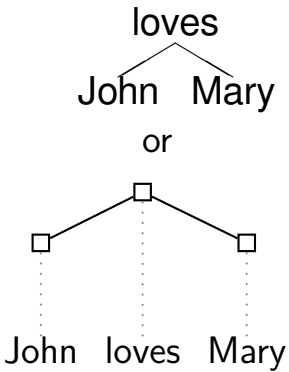
Motivation

Simple phrase-based models:

- Don't check grammatical coherence.
  $\Rightarrow$ 3-grams fluent, overall word salad.
- Don't allow gaps in phrases:
  I do not know... $\leftrightarrow$ Je ne sais pas...
  "do not" $\leftrightarrow$ "ne ...pas"
- Reordering models have little explicit knowledge:
  - No movement of longer blocks.
  - No grammar constraints possible.

Constituency vs. Dependency Trees





Constituency trees = syntactic structure of a sentence as "bracketting":

$$(_S\ (_NP\ \text{John})\ (_VP\ (_V\ \text{loves})\ (_NP\ \text{Mary})\ )\ )$$

Contituency Syntax

Context Free Grammar

Context-free grammar (CFG) describes infinite set of valid trees using a finite set of rules, e.g.:

$$\text{S} \rightarrow \text{NP VP}$$

Probabilistic CFG assign weights to rules, e.g.:

$$\text{VP} \rightarrow \begin{cases} \text{V} & 0.1 \\ \text{V NP} & 0.5 \\ \text{V NP NP} & 0.4 \end{cases} \tag{1}$$

Generative model for probabilitic CFG:

$$p(\text{tree } T|\text{sentence } s) = \prod_{X \rightarrow \alpha \in T} p(\alpha|X) \tag{2}$$

Parsing

PARSING is finding the most probable constituency tree:

$$\hat{T} = \underset{T \in \{\text{trees of sentence } s\}}{\text{argmax}} \; p(T|s) \tag{3}$$

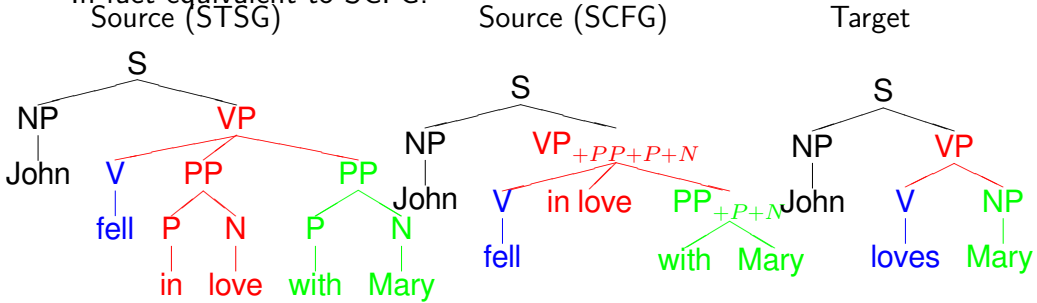
CKY (CYK) algorithm for $O(n^3)$ parsing. ("dynamic programming"):

length:							
7	S						
6		VP					
5							
4	S						
3		VP			PP		
2	S		NP			NP	
1	NP	V, VP	Det	N	P	Det	N
	$_0$she$_1$	$_1$eats$_2$	$_2$a$_3$	$_3$fish$_4$	$_4$with$_5$	$_6$a$_6$	$_6$fork$_7$

MT as Parsing (SCFG, STSG)

While parsing input sequence, construct output.

- Synchronous CFG capture the "double generation".
  - Nonterminals must map 1-1.
- Synchronous Tree Substitution Grammars (STSG)
  - Whole subtrees are attached ⇒ structural changes ok.
  - In fact equivalent to SCFG.



Hierarchical P-B Model

Some of the syntax hype in MT started by Chiang (2005).
Chiang (2005) addresses the gaps in phrases:

- Gaps don't have labels $\Rightarrow$ any phrase fits.
  "do not X" $\leftrightarrow$ "ne X pas"
  "do not cat" $\leftrightarrow$ "ne chat pas"
  $\Rightarrow$ Not really a grammar, but not an issue for correct input.
- Phrase extraction similar to phrase-based:
  - All phrases consistent with alignment.
  - If a subphrase exists, make a synchronous gap.

Joshua: Hierarchical Decoder

- Open-source replacement for Hiero (Chiang, 2005).
- Support SCFGs now, possibly still undocumented.

See slides by Li et al. (2009):

- Synchronous CFG.
- Heuristic learning of hierarchical rules.
- Decoding via chart parsing.
- **State splitting due to LM.**

Other implementations:

- Moses
- cdec (`http://cdec-decoder.org/`)

Making Syntax Work

- Hierarchical P-B is not a syntactic model.
  - With a special treatment of unaligned words, a regular P-B system can reach most of hierarchical hypotheses (Auli et al., 2009).

Syntax-Augmented MT (Zollmann and Venugopal, 2006)[1]:

- The first C++ version of Joshua.
- Uses proper nonterminals coming from source-tree parse.

See slides by Chiang (2010):

- Why is tree-to-tree translation hard.
- Learning how much syntax to use.

[1] `http://www.cs.cmu.edu/~zollmann/samt/`

Summary of Constituency Syntax in MT

- CFG capture syntax of some natural languages.
- Translating while chart parsing.
  - SCFG/STSG parse input
    and construct syntactically-parallel output.
- Hierarchical model: a simplification.
  - Only a single nonterminal used.
  - LM integrated by state splitting.
- When real source and/or target parse trees are used:
  - Tricks/binarization necessary to extract non-isomorphic trees.
  - Fuzzy matching must be allowed during decoding.

- Joshua, Moses, SAMT, cdec as available tools.

Dependency Syntax in MT

Outline

- Motivation for dependency trees. Non-projectivity.
- Synchronous Tree Substitution Grammars
  - STSG Formally, STSG vs. SCFG, TAG.
  - "Adjunction" in dependency and constituency STSG.
- Heuristic treelet dictionary extraction
- Decoding STSG (Top-down beam-search).

Deep-syntactic approach:

- Analytical and tectogrammatical trees, t-layer hope.
- TectoMT
  - HMTM for selecting the best combination of node labels.

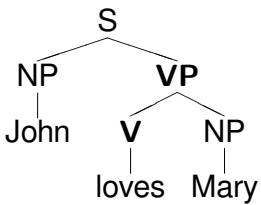
Constituency vs. Dependency Again

Constituency trees (CFG) represent only bracketing:
= which adjacent constituents are glued tighter to each other.
Dependency trees represent which words depend on which.
+ usually, some agreement/conditioning happens along the edge.

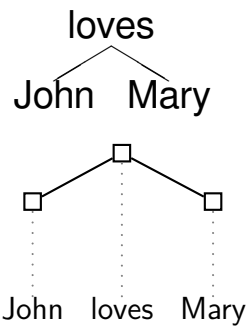
Constituency

John (loves Mary)

John $_{\text{VP}}$(loves Mary)



Dependency



What Dependency Trees Tell Us

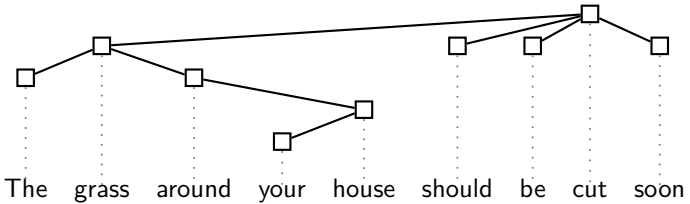
Input: The **grass** around your house should be **cut** soon.
Google: **Trávu** kolem vašeho domu by se měl **snížit** brzy.

- Bad lexical choice for *cut* = *sekat/snížit/krájet/řezat/...*
  - Due to long-distance dependency with *grass*.
  - One can "pump" many words in between.
  - Could be handled by full source-context (e.g. maxent) model.
- Bad case of *tráva*.
  - Depends on the chosen active/passive form:

active⇒accusative	passive⇒nominative
trávu ... by**ste** se měl posekat	tráva ... by **se** měl**a** posekat
	tráva ... by měl**a** **být** posek**ána**

Examples by Zdeněk Žabokrtský, Karel Oliva and others.

Tree vs. Linear Context

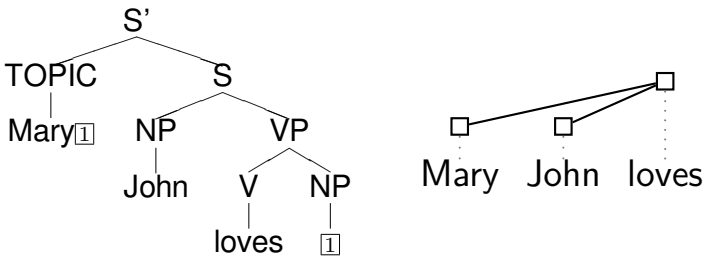


- Tree context (neighbours in the dependency tree):
  - is better at predicting lexical choice than n-grams.
  - often equals linear context:
    Czech manual trees: 50% of edges link neighbours,
    80% of edges fit in a 4-gram.
- Phrase-based MT is a very good approximation.
- Hierarchical MT can even capture the dependency in one phrase:

$$X \rightarrow <\text{ the grass } X \text{ should be cut, trávu } X \text{ byste měl posekat } >$$

“Crossing Brackets”

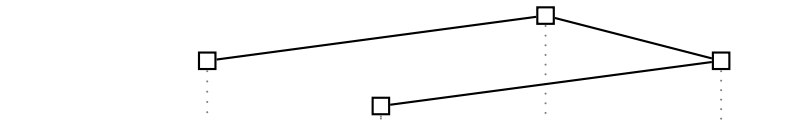
- Constituent outside its father’s span causes “crossing brackets.”
  - Linguists use “traces” (1) to represent this.
- Sometimes, this is not visible in the dependency tree:
  - There is no “history of bracketing”.
  - See Holan et al. (1998) for dependency trees including derivation history.



Despite this shortcoming, CFGs are popular and “the” formal grammar for many. Possibly due to the charm of the father of linguistics, or due to the abundance of dependency formalisms with no clear winner (Nivre, 2005).

Non-Projectivity

= a gap in a subtree span, filled by a node higher in the tree.
Ex. Dutch "cross-serial" dependencies, a non-projective tree with one gap caused by *saw* within the span of *swim*.



- 0 gaps ⇒ projective tree ⇒ can be represented in a CFG.
- ≤ 1 gap & "well-nested" ⇒ mildly context sentitive (TAG).

See Kuhlmann and Möhl (2007) and Holan et al. (1998).

Why Non-Projectivity Matters?

- CFGs cannot handle non-projective constructions:

  Imagine John **grass** saw **being-cut**!

- No way to glue these crossing dependencies together:
  - Lexical choice:

    $$X \rightarrow <\text{grass } X \text{ being-cut}, \text{trávu } X \text{ sekat}>$$

  - Agreement in gender:

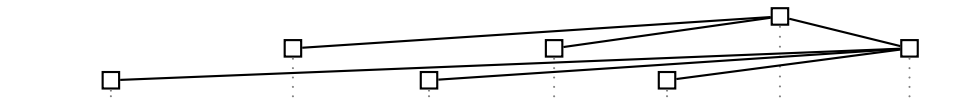
    $$X \rightarrow <\text{John } X \text{ saw}, \text{Jan } X \text{ viděl}>$$
    $$X \rightarrow <\text{Mary } X \text{ saw}, \text{Marie } X \text{ viděl}\mathbf{a}>$$

- Phrasal chunks can memorize fixed sequences containing:
  - the non-projective construction
  - and all the words in between! ($\Rightarrow$ extreme sparseness)

Is Non-Projectivity Severe?

Depends on the language.
In principle:

- Czech allows long gaps as well as many gaps in a subtree.



Peter decided to object against the dismissal at work tomorrow.

In treebank data:

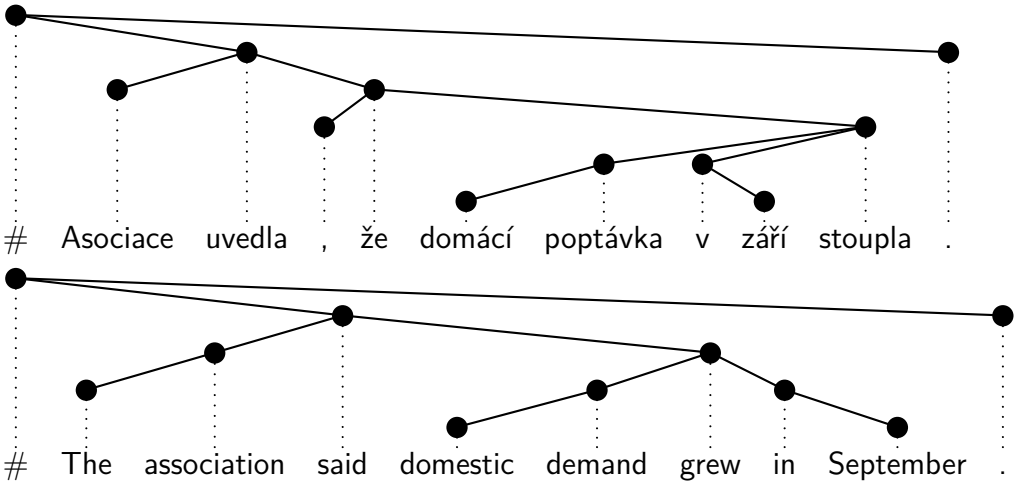
- ⊖ 23% of Czech sentences contain a non-projectivity.
- ⊕ 99.5% of Czech sentences are well nested with ≤ 1 gap.

Summary of Dependency Trees

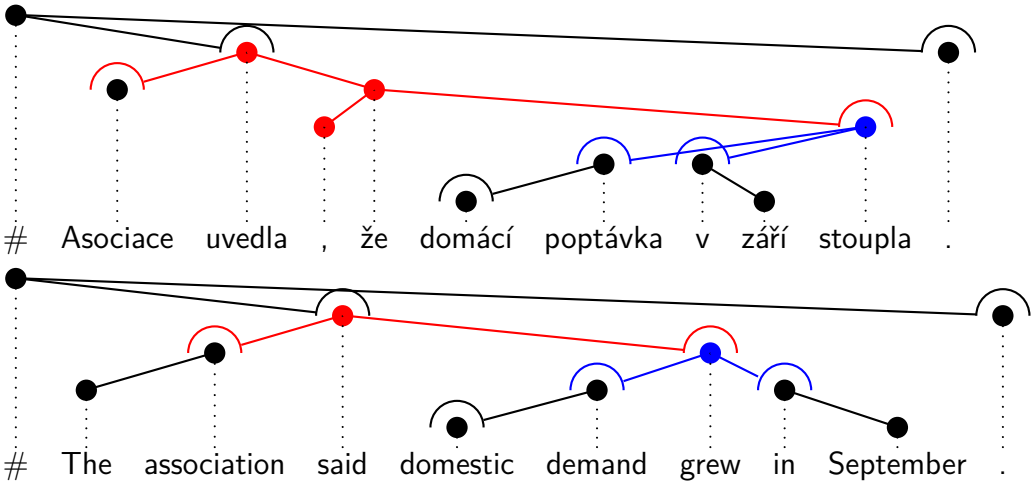
- More appropriate for Czech (frequent non-projectivity).
- Exhibit less divergence across languages (Fox, 2002).
- Dependency context more relevant than adjacency context.

So let's look if we can apply them in MT.

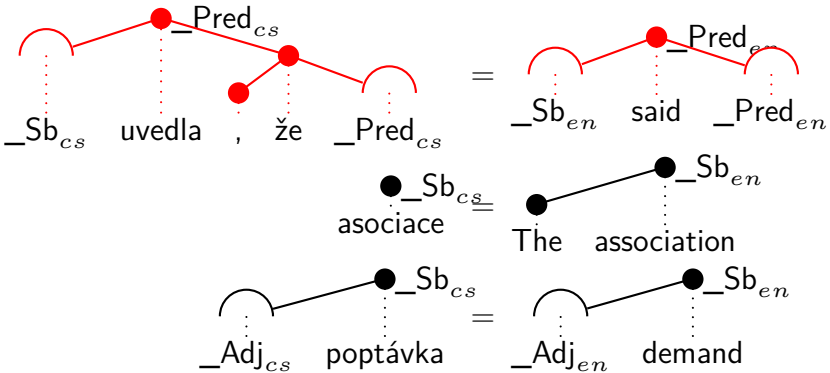
Idea: Observe a Pair of Trees...



...Decompose into Treelets...



...Collect Dict. of Treelet Pairs

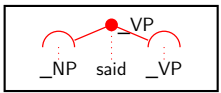


...**Synchronous Tree Substitution Grammar**,

e.g. Čmejrek (2006).

Little Trees Formally (TSG)

Given a set of states Q and a set of word labels L, we define:
A LITTLE TREE or TREELET t is a tuple (V, V^i, E, q, l, s) where:



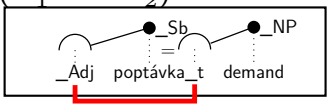
- V is a set of NODES,
- $V^i \subseteq V$ is a nonempty set of INTERNAL NODES. The complement $V^f = V \setminus V^i$ is called the set of FRONTIER NODES,
- $E \subseteq V^i \times V$ is a set of directed edges starting from internal nodes only and forming a directed acyclic graph,
- $q \in Q$ is the ROOT STATE,
- $l : V^i \to L$ is a function assigning labels to internal nodes,
- $s : V^f \to Q$ is a function assigning states to frontier nodes.

Optionally, we can keep track of local or global ordering of nodes in treelets.

Treelet Pair, Synch. Derivation

A TREELET PAIR $t_{1:2}$ is a tuple (t_1, t_2, m) where:

- t_1 and t_2 are little trees for source and target languages (L_1 and L_2) and states (Q_1 and Q_2),
- m is a 1-1 MAPPING of frontier nodes in t_1 and t_2.
  Unlike Čmejrek (2006), I require all frontier nodes mapped, i.e. equal number of left and right frontier nodes.



From a starting SYNCHRONOUS STATE $Start_{1:2} \in Q_1 \times Q_2$,
a SYNCHRONOUS DERIVATION δ constructs a pair of dependency trees by:

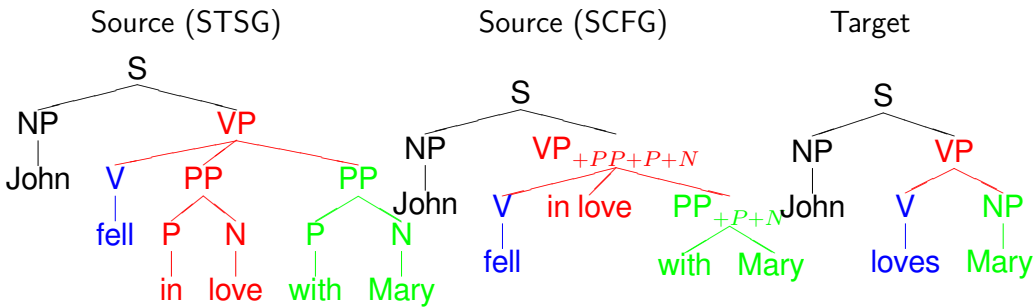
- attaching treelet pairs $t^0_{1:2}, \ldots, t^k_{1:2}$ at corresponding frontier nodes, and
- ensuring that the root states $q^0_{1:2}, \ldots, q^k_{1:2}$ of the attached treelets pairs $t^0_{1:2}, \ldots, t^k_{1:2}$ match the frontier states of the corresponding frontier nodes.

Can define probability of a derivation: $p(\delta) = p(t^0_{1:2}|Start_{1:2}) * \prod_{i=1}^{k} p(t^k_{1:2}|q^k_{1:2})$

Related: SCFG

Synchronous Context Free Grammars:

- Don't encode internal structure of rules.
  - SCFG "treelets" are one level only.
- STSG can be encoded as SCFG:



Related: TAG

Tree-adjoining grammars (TAG, Joshi et al. (1975), see also the review by Joshi et al. (1990)) use:

- tree substitution (at frontier F) and
- tree adjunction (at node A):

There is no adjunction in (S)TSG (Eisner, 2003).
A few remarks on expressive power in Bojar and Lopez (2008).

Treelet Alignments: Heuristics

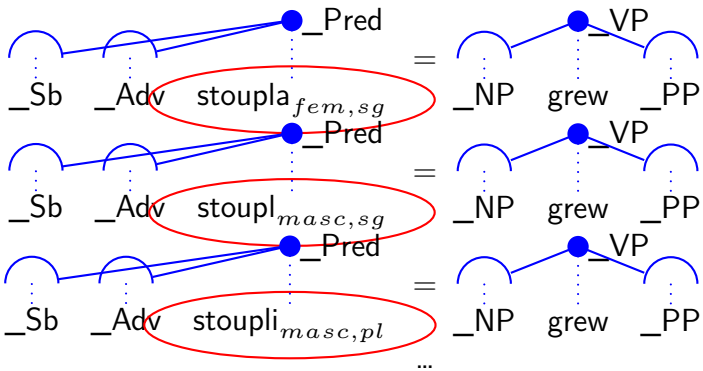
- Similar to common phrase-extraction techniques given word alignments.
- Basic units are little trees instead of word spans.

1. Parse both sides of the parallel corpus.
2. Obtain **node-to-node alignments** (GIZA++ on linearized trees).
3. Extract all treelet pairs satisfying these conditions:
   - no more than i internal nodes and f frontier nodes,
   - **compatible with node alignment**: e.g. no node-alignment link leads outside the treelet pair and frontiers are linked.
   - satisfying **STSG property**:
     All children of an internal node have to be included in the treelet (as frontiers or internals), ie. assume no adjunction operation was necessary to construct the full tree.

Sources of Data Sparseness (1)

Morphological richness:

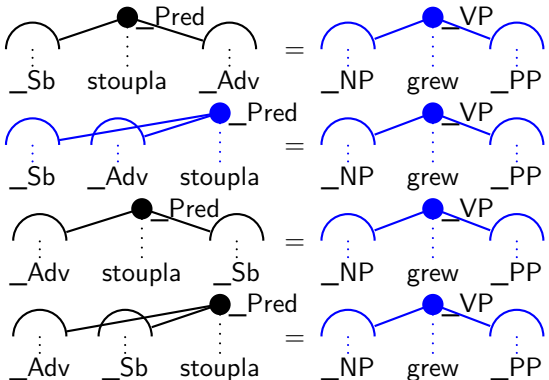
- not an issue at a higher layer, where nodes hold lemmas.



Sources of Data Sparseness (2)

Ordering of nodes:

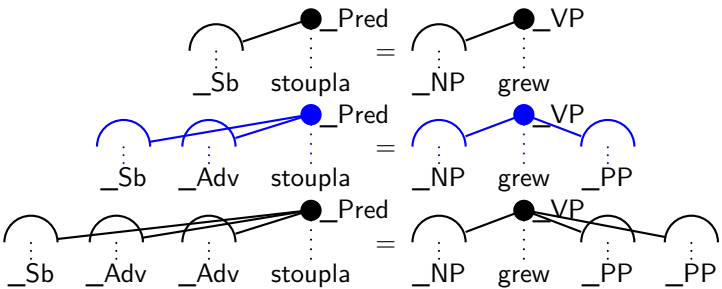
- In Czech many permutations are possible.
- Not an issue if we decide to leave the tricky part for someone else, e.g. a tecto→analytical generator.



Sources of Data Sparseness (3)

Frontiers for additional adjuncts, state labels for root and frontiers:

- There is no adjunction operation in STSG, treelets explicitly encode the number of frontiers.

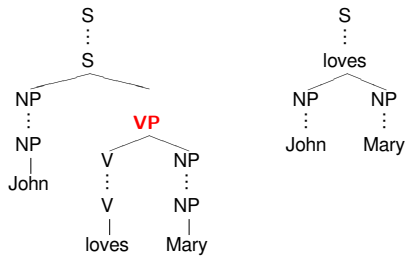


A Remark on "Adjunction"

If TSG use constituency trees, substitution can introduce adjuncts.
Consider CFG rule for "adjunct":

$$\text{VP} \rightarrow \text{fiercely VP}$$

- Can be applied in constituency tree.
- Dependency tree needs a new rule: "loves→NP ADV NP".



Decoding STSG

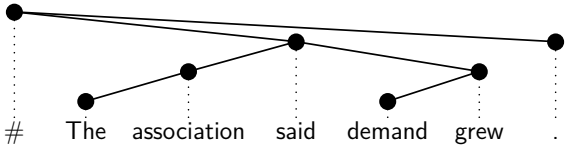
Given an input dependency tree:

- decompose it into known treelets,
- replace treelets by their treelet translations,
- join output treelets and produce output final tree; linearize or generate plaintext.

Implemented as two-step top-down beam-search similar to Moses:

1. Prepare **translation options table**:
   - For every source node consider every subtree rooted at that node.
   - If the subtree matches the source treelet in a treelet pair, we've got a translation option.
   - Keep only best τ translation options at a node.
2. Gradually **expand partial hypotheses**:
   - Starting at root use translation options to cover source tree.
   - Keep only best σ partial hypotheses of a given size (input nodes covered).

Translation Options Example

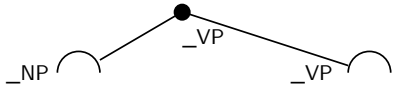


Sample translation options at root:

⇒ # _Pred _AuxK

⇒ # _Pred .

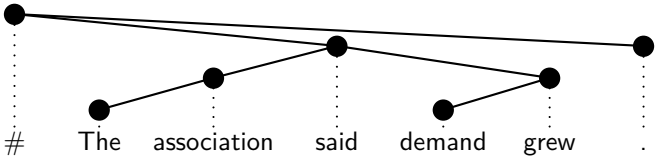
Sample translation options at 'said':



⇒ _Sb uvedla , že _Pred

Sample translation options at '.':

Hypothesis Expansion Example



Sample Derivation:

Linearized output:

h_0 _#

⇒ _#

h_1 _# _VP

⇒ # _Pred .

_# _NP _VP

h_2

⇒ # _Sb uvedla , že _Pred .

Back-off Schemes

Preserve all. Full-featured treelets are collected in training phase.
Required treelets often never seen in training data ⇒ back-off needed.

Drop frontiers. Observed treelets reduced to internal nodes only.
Given a source treelet, internals translated by the dictionary, frontiers generated on the fly, labelled and positioned probabilistically.

Keep a word non-translated to handle unknown words.
Allowed only for single-internal treelets, frontiers mapped probabilistically.

Transfer numeric expression, showing possibility to include hand-coded rules.

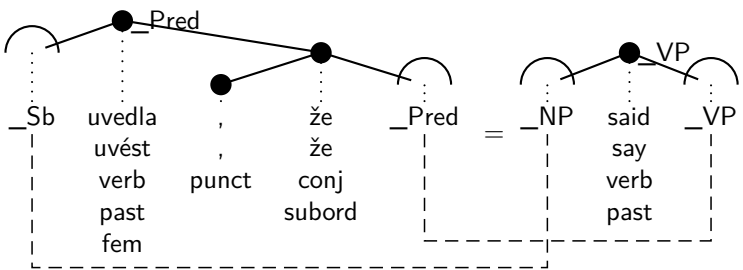
Adjoin on the fly like Quirk et al. (2005); not implemented.

Modular approach to back-off schemes, config says:

- which methods to use
- in which order, or whether more should be attempted at simultaneously.

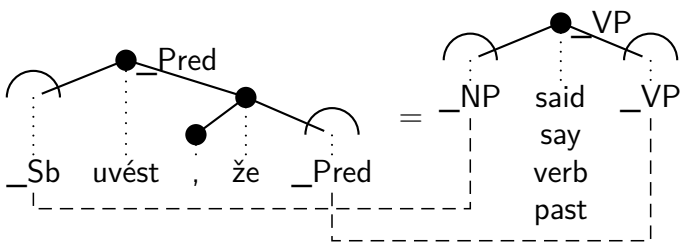
Treelet Construction 1: Preserve All

- Most basic method (no back-off).
- Preserves:
  - shapes of treelets, ordering of nodes,
  - all factors (attributes) of internal nodes,
  - position and states of frontier nodes.



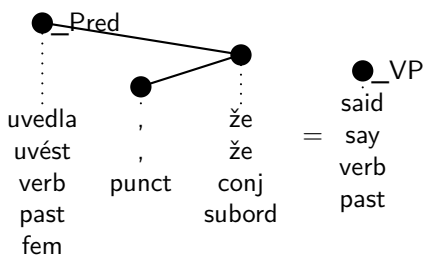
Construction 2: Drop Input Factors

- Back-off by ignoring some of input factors ⇒ reduced source data sparseness.
- Output factors fully specified (i.e. their values guessed).



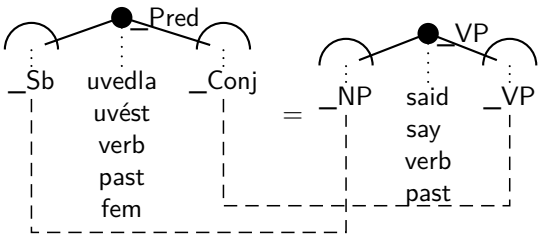
3: Drop Frontiers

- We preserve: treelet shape, all factors of all internal nodes.
- In training, frontier nodes are ignored (dropped).
- In translation, frontier nodes are translated and positioned one-by-one.
  - Available only when producing linearized output (no need to reconstruct the structure).
  - Frontiers are placed before or after internal nodes, not in between.



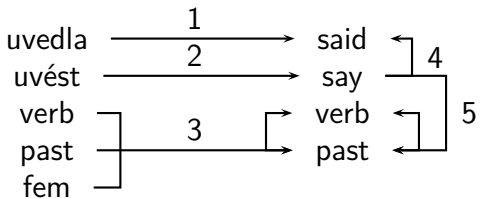
4: Translate Node-by-Node

- Like (3), but treelets limited to one internal only.
  $\Rightarrow$ trivial to reconstruct treelet structure.
- Frontiers ignored in training, and translated one-by-one.
  - Ordering preserved for the sake of simplicity.
- If used alone, the source and target trees will have equal number of nodes:
  $\Rightarrow$ Not suitable for transfer at a-layer.



5: Factored (Node-by-Node)

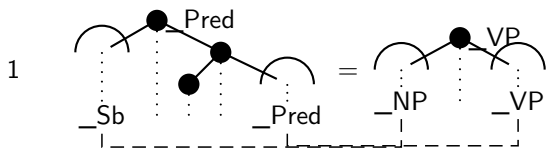
- Like factored phrase-based translation. (Koehn and Hoang, 2007)
- The configuration specifies a sequence of steps:
  - **Mapping steps** convert input factors to output factors.
  - **Generation steps** bind values of output factors.
  - The order of steps is important due to the limited stack of partial hypotheses.



6: Factored (with Structure)

- Like (5) but a zeroth mapping step used to predict:
  - Target treelet structure.
  - Position and labels of frontier nodes.
- Subsequent mapping and generation steps applied **synchronously** at all nodes.

  (Structure factorization goes directly against attribute factorization.)



Combining Models

- The original STSG (Eisner, 2003) extended to a log-linear:

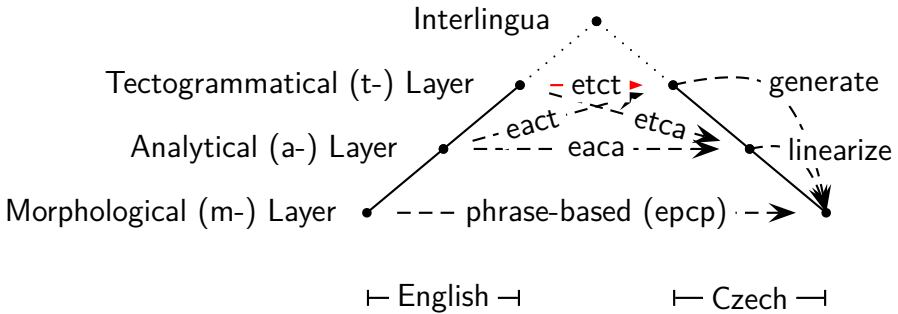
$$\text{search for best derivation } \hat{\delta} = \underset{\delta \in \Delta(T_1)}{\text{argmax}} \exp\Big(\sum_{m=1}^{M} \lambda_m h_m(\delta) \Big) \tag{4}$$

$$\text{instead of } \hat{\delta} = \underset{\delta \in \Delta(T_1)}{\text{argmax}}\, p(t^0_{1:2} | Start_{1:2}) * \prod_{i=1}^{k} p(t^k_{1:2} | q^k_{1:2}) \tag{5}$$

- The configuration specifies treelet-construction methods to use:
  - E.g. prefer "Preserve everything" but back-off to factored node-by-node.
- Weights λ_m of simultaneously used models chosen to achieve high BLEU.
  - Implemented binding to two MERT methods: (Och, 2003) a (Smith and Eisner, 2006)
  - Fails to converge (too many weights) $\Rightarrow$ manually pick some values.

Transfer at Various Layers

- Main goal: Transfer at t-layer.
- Applicable anywhere with dependency trees.



Deep Syntax

Going Deeper

- Motivation for tectogrammatical layer.
  - Including some slides by Zdeněk Žabokrtský.
- TectoMT Transfer.
  - TectoMT vs. tectogrammatical theory.
  - Hidden Markov Tree Model.
- Treex Platform.
- Remarks on manual annotation.
- T-Layer in STSG:
  - Complexity of t-layer attributes.
  - Factorization inevitable, but how to factorize?
  - Empirical evaluation.

Tectogrammatics: Deep Syntax Culminating

Background: Prague Linguistic Circle (since 1926).

Theory: Sgall (1967), Panevová (1980), Sgall et al. (1986).

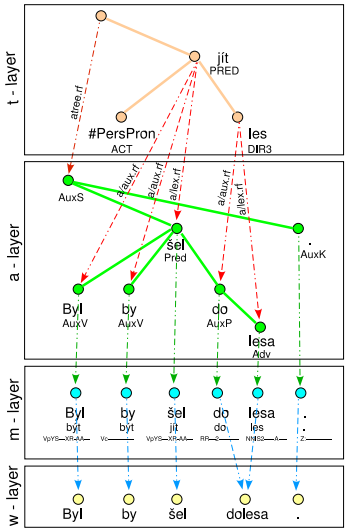
Materialized theory — Treebanks:

- Czech: PDT 1.0 (2001), PDT 2.0 (2006)
- Czech-English: PCEDT 1.0 (2004), PCEDT 2.0 (2012)
- Arabic: PADT (2004)

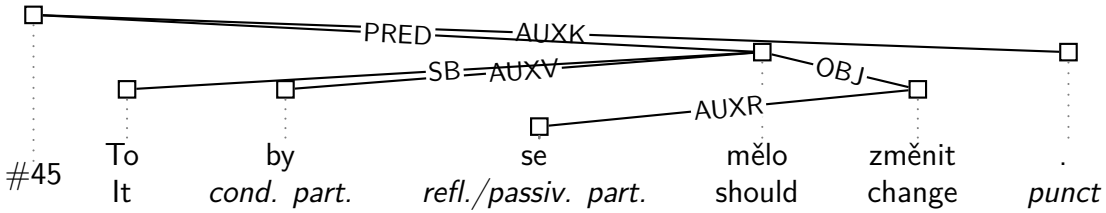
Practice — Tools:

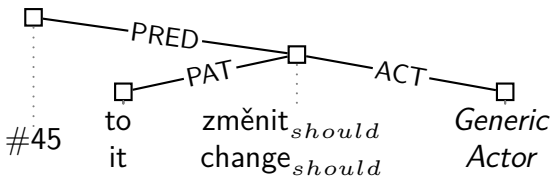
- parsing Czech to surface: McDonald et al. (2005)
- parsing Czech to deep: Klimeš (2006)
- parsing English to surface: well studied (+rules convert to dependency trees)
- parsing English to deep: heuristic rules (manual annotation in progress)
- generating Czech surface from t-layer: Ptáček and Žabokrtský (2006)

Layers in PDT



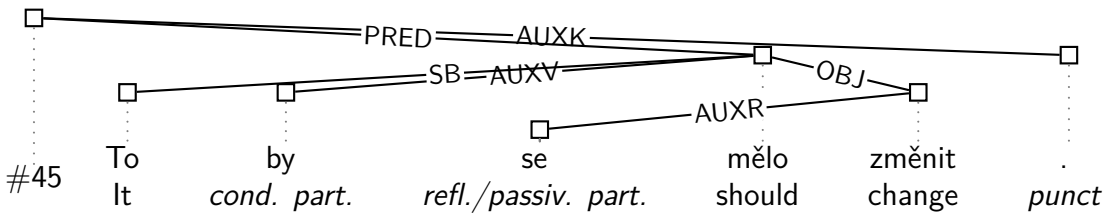
Analytical vs. Tectogrammatical

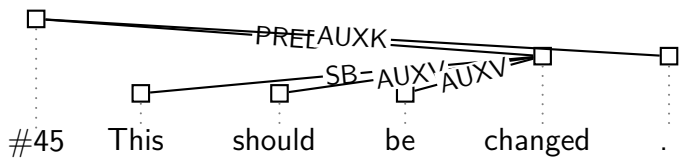




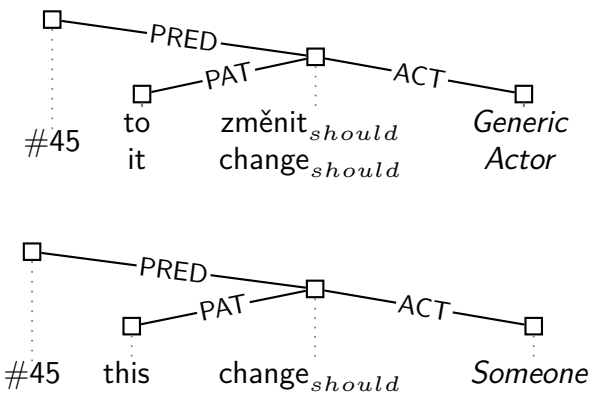
- hide auxiliary words, add nodes for "deleted" participants
- resolve e.g. active/passive voice, analytical verbs etc.
- "full" t-layer resolves much more, e.g. topic-focus articulation or anaphora

Czech and English A-Layer





Czech and English T-Layer



Predicate-argument structure: change$_{should}$(ACT: someone, PAT: it)

The Tectogrammatical Hope

Transfer at t-layer should be easier than direct translation:

- Reduced structure size (auxiliary words disappear).
- Long-distance dependencies (non-projectivites) solved at t-layer.
- Word order ignored / interpreted as information structure (given/new).
- Reduced vocabulary size (Czech morphological complexity).
- Czech and English t-trees structurally more similar
  $\Rightarrow$less parallel data might be sufficient (but more monolingual).
- Ready for fancy t-layer features: co-reference.

The complications:

- 47 pages documenting data format (PML, XML-based, sort of typed)
- 1200 pages documenting Czech t-structures
  "Not necessary" once you have a t-tree but useful understand or to blame the right people.

How could tecto help? (cont.)

- n-gram view:
  - manifestations of lexemes are mixed with manifestations of language means expressing the relations between the lexemes and of other grammar rules
    - inflectional endings, agglutinative affixes, functional words, word order, punctuation orthographic rules ...
    - *It will be delivered to Mr. Green's assistants at the nearest meeting.*
  - → **training data sparsity**

- tectogrammar view:
  - clear separation of meaningful "signs" from "signs" which are only imposed by grammar (e.g. imposed by agreement)
  - clear separation of lexical, syntactical and morphological meaning components
  - → modularization of the translation task → **potential for a better structuring of statistical models** → more effective exploatation of the (limited) training data

Tecto transfer factorization

- Three transfer "channels" can be separated:
  - translation of lexicalization
    - E.g. 'koupit' goes to 'buy'
  - translation of syntactization
    - e.g. relative clause goes to attributive adjective
  - Translation of morphological meanings
    - e.g. singular goes to singular
- The channels are relatively loosely coupled (esp. the third one) which could be used for smoothing.

Tecto transfer factorization (cont.)

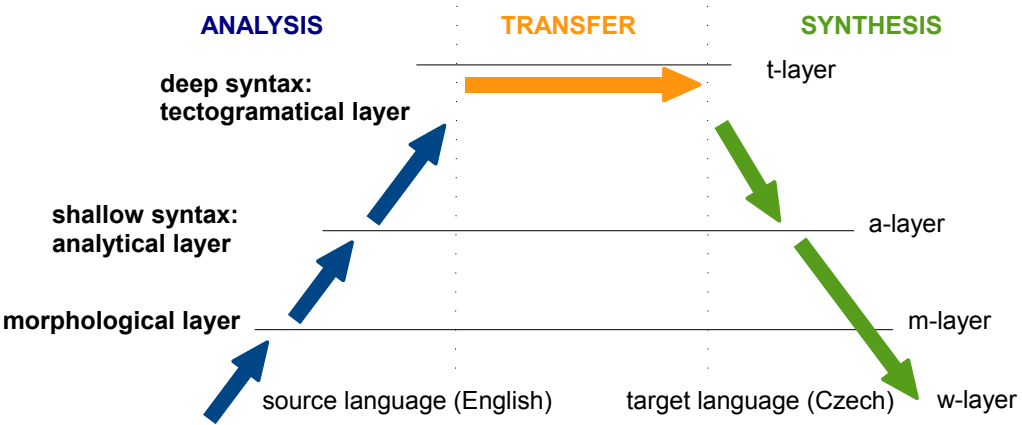
- Example: three ways to express future tense in Czech
  - (1) aux.verb: ***budu*** *… chodit – I will walk …*
  - (2) prefix: ***po****letím – I will fly …*
  - (3) ending: *uvař****ím*** *– I will boil …*
- nontrivial tense translation from the n-gram view
- but once we work with tecto analyses, we can translate the future tense just to future tense, separately from translating the lemma
  - similarly, plural goes mostly to plural, comparative to comparative, etc.

Tecto transfer factorization (cont.)

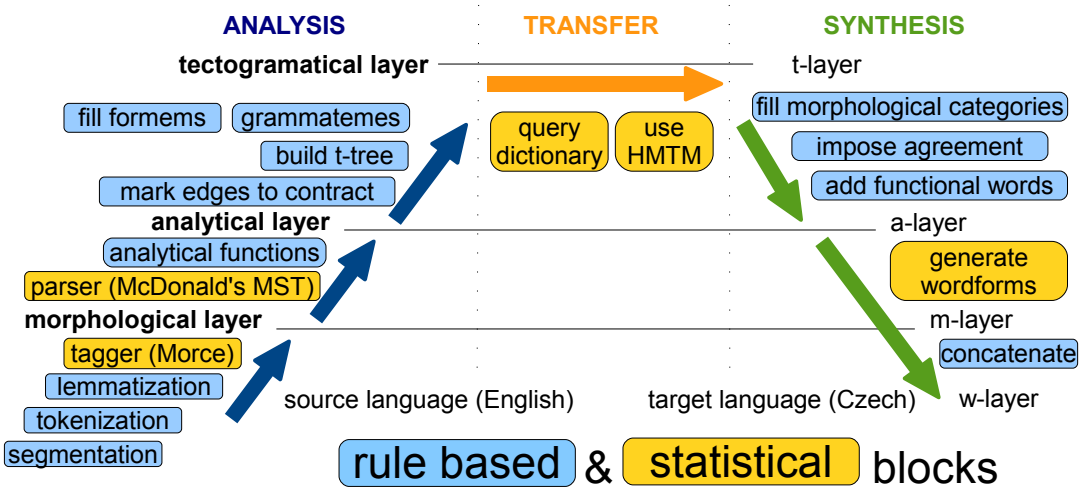
- we introduce the notion of **formemes** - morphosyntactic language means expressing the dependency relation
- example values:
  - **n:v+6** (in Czech) = semantic noun which is on the surface expressed in the form of prepositional group in locative with preposition "v"
  - **v:that+fin/a** (in English) = semantic verb expressed in active voice as a head of subordinating clause introduced with the sub.conjunction "that"
  - **v:rc** (in Czech and English) = head of relative clause
  - **n:sb** (in English) = noun in subject position
  - **n:1** (in Czech) = noun in nominative case
  - **adj:attr** (in Czech and English) = adjective in attributive position
- formemes allow us to introduce a **separate syntactization factor** and to train it using a parsed parallel corpus
- trained estimates of $P(F_{cz}|P_{en})$:

v:to+inf	v:inf	0.4817
v:to+inf	v:aby+fin	0.0950
v:to+inf	n:k+3	0.0702
v:to+inf	v:že+fin	0.0621
n:for+X	n:pro+4	0.2234
n:for+X	n:2	0.1669
n:for+X	n:4	0.0788
n:for+X	n:za+4	0.0775

“TectoMT Transfer” (1/3)



“TectoMT Transfer” (2/3)



"TectoMT Transfer" (3/3)

Warning: TectoMT's t-layer against the spirit of FGD:

- FGD: A unit (e.g. a sentence) at a higher (deeper) layer represents the meaning of several units at a lower level.
  - A t-tree can be expressed by many a-trees (choose passive/active, ...).
- TectoMT: Generation from t-layer should be deterministic.
  - Good for (e.g. MERT) optimization: no need to run the generation.
  - Some features (e.g. n-gram LM) not applicable at t-layer, replacements have to be found.

More: Slides 6–28 by Martin Popel (2009):

- Illustration of TectoMT transfer.
- Analysis of translation errors.
- Hidden Markov Tree Model (HMTM).

Treex Platform

Originally called also TectoMT (Žabokrtský and Bojar, 2008)

- Modular Perl environment.
- Common data format (XML), API. Tree editor TrEd used.
- Many NLP tools wrapped (4+ parsers, 4+ taggers, NER, ...).
- Support for Unix pipes and parallel processing (SGE).

Used in many NLP tasks:

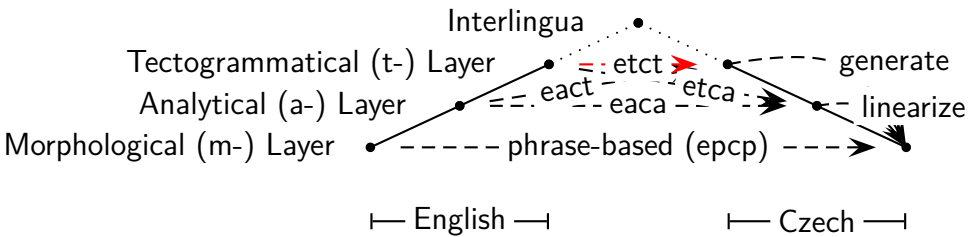
- All our automatic annotations. Incl. 15M parallel sentences.
- "TectoMT transfer", an MT system implemented in TectoMT.
- Dialogue systems, automatic MT evaluation, ...

See also: `http://ufal.mff.cuni.cz/treex/`

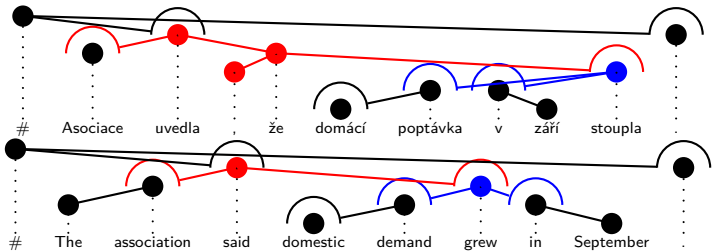
Public packages on CPAN, Docker.

STSG for Tree Transfer

- Synchronous Tree Substitution Grammars described last time.
- Transfer source dependency tree into target dependency tree.
- Applicable at or across layers.
- Main goal: Transfer at t-layer.



Reminder: STSG



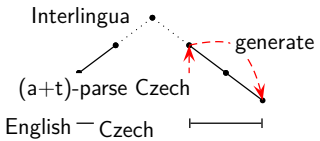
1. Decompose input tree into treelets.
2. Replace treelets with their translations.
3. Join output treelets.

In Reality, t-nodes are not Atomic!

t-nodes have about 25 attributes: t-lemma, functor, gender, person, tense, iterativeness, dispositional modality, ...

Upper Bound on MT Quality via t-layer:

- Analyse Czech sentences to t-layer.
- Optionally ignore some node attributes.
- Generate Czech surface.
- Evaluate BLEU against input Czech sentences.

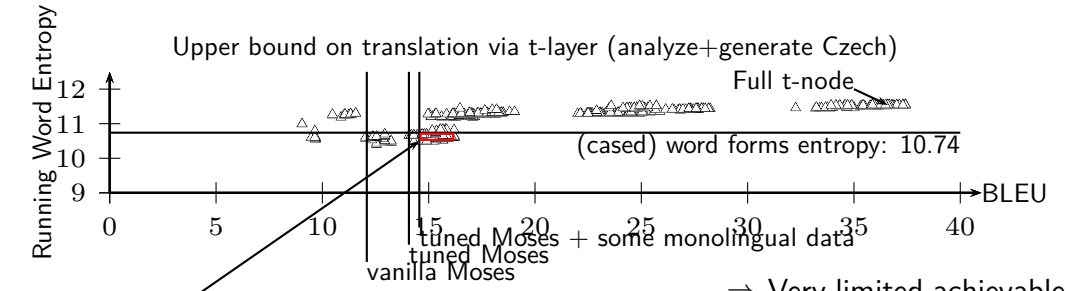


	BLEU
Full automatic t-layer, no attributes ignored	36.6±1.2
Ignore sentence mood (assume indicative)	36.6±1.2
Ignore verbal fine-grained info (resultativeness, ...)	36.6±1.2
Ignore verbal tense, aspect, ...	24.9±1.1
Ignore all grammatemes	5.3±0.5

⇒ Node attributes obviously very important.

Fairy Tales on Vocabulary Reduction

Is there a balance of small vocab. and high achievable BLEU?



Space for improvement assuming:

- t-nodes atomic (with a restricted set of attributes)
- we wish to stay below the entropy of plain text

⇒ Very limited achievable BLEU even if transfer were absolutely perfect.

Consequence: Must Factorize

- Zdeněk's slide 11 (here slide 43) mentions three "channels" than can be separated.
- t-layer by itself increases complexity of node label choice.
  $\Rightarrow$ The factorization is not an option, it is a must.

Structure vs. Attributes

Factorization = introduction of independence assumptions.

- STSG factorizes along structure (input into treelets).
- T-layer requires factorization along attributes.

Which should go first?

- Treelets of attributes?
  - Similar to phrases of factors, synchronous approach.
  - Can easily fill up stacks with treelets differing too little.
- Layers of trees?
  - Would be hard to ensure matching tree structure.
  - Rather use a few attributes to construct structure and postpone the choice of others until the tree is finished.

Empirical Evaluation

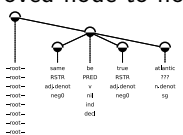
Back-off Sequence	**43k**	**643k**
strall+wfwindep	5.51±0.49	5.26±0.51
strall+strindep+strdelay+wfwall+wfwindep	5.29±0.49	5.09±0.48
strdelay+wfwindep	4.99±0.45	5.30±0.54
strindep+wfwindep	4.86±0.49	5.39±0.59
wfwindep	4.58±0.51	5.09±0.50
strdelay	4.22±0.43	4.39±0.50
strindep	4.09±0.44	4.50±0.55
wfwall	3.81±0.52	4.11±0.50
strall	3.62±0.42	3.81±0.42

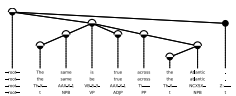
- Preserve treelet **str**ucture vs. translate "**w**ord-**f**or-**w**ord" (i.e. node-to-node).
- Translate **all** attributes at once, in **indep**endent mapping steps when constructing treelets or **delay** their value generation until tree is constructed.

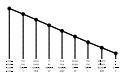
Transfer at Various Layers

Layers \ Language Models	no LM	n-gram/*binode*
epcp, no factors	8.65±0.55	10.90±0.63
eaca, no factors	6.59±0.52	8.75±0.61
etct 2009; 43k	-	7.39±0.52
etca, no factors	-	6.30±0.57
etct factored, preserving structure	5.31±0.53	*5.61±0.50*
eact, source factored, output atomic	-	*3.03±0.32*
etct, no factors, all attributes	1.61±0.33	*2.56±0.35*
etct, no factors, just t-lemmas	0.67±0.19	-

etct 2009: strall + wfwindep. LM rescoring. Formemes (not functors) as frontier labels.
Improved node-to-node alignment (Mareček et al., 2008). New generation pipeline.

t: 

a: 

p: 

(WMT07 DevTest)

Reasons of STSG Bad Performance

- **Cumulation of Errors** in annotation pipeline.
- **Data Loss** due to incompatible structures:
  - Error in parses or word-alignment prevents treelet pair extraction.
- **Combinatorial Explosion** of factored output (**indep**):
  - Translation options are first fully built, before combination is attempted.
  - Abundance of t-node attribute combinations

    $\Rightarrow$ e.g. lexically different translation options pushed off the stack

    $\Rightarrow$ n-bestlist varies in unimportant attributes.
  - Not an issue with **delay**ed factors.
- **Too Strong Independence Assumtions**:
  - Should never analyze and factorize phrases seen often enough.
- **Complex models hard to tune:**
  - Independence assumption need to be validated: which are ok?
  - More models $\Rightarrow$ Minimum error rate training has harder time.

Summary

- Dependency trees linguistically more promising.
  - Tree context vs. linear context. Non-projectivity. T-layer.
- STSG to transfer dependency trees:
  - Severe issues of sparseness, i.a. due to missing adjunction.
- TectoMT system with HMTM transfer.
  - T-tree isomorphism is still viable assumption.

Rich annotation **hurts** if not **backed-off**.

- Due to sparser data (incompatible trees).
- Due to cummulation of errors.
- Due to too strong independence assumptions.
- Due to harder optimization problem for MERT.

References

Michael Auli, Adam Lopez, Hieu Hoang, and Philipp Koehn. 2009. A systematic analysis of translation model search spaces. In Proceedings of the Fourth Workshop on Statistical Machine Translation, pages 224–232, Athens, Greece, March. Association for Computational Linguistics.

Ondřej Bojar and Adam Lopez. 2008. Tree-based Translation. Handout for MT Marathon Tutorial, May.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), pages 263–270, Ann Arbor, Michigan, June. Association for Computational Linguistics.

David Chiang. 2010. Learning to translate with source and target syntax. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1443–1452, Uppsala, Sweden, July. Association for Computational Linguistics.

Martin Čmejrek. 2006. Using Dependency Tree Structure for Czech-English Machine Translation. Ph.D. thesis, ÚFAL, MFF UK, Prague, Czech Republic.

Jason Eisner. 2003. Learning Non-Isomorphic Tree Mappings for Machine Translation. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), Companion Volume, pages 205–208, Sapporo, July.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing, pages 304–3111. Association for Computational Linguistics.

Tomáš Holan, Vladislav Kuboň, Karel Oliva, and Martin Plátek. 1998. Two Useful Measures of Word Order Complexity. In A. Polguere and S. Kahane, editors, Proceedings of the Coling '98 Workshop: Processing of Dependency-Based Grammars, Montreal. University of Montreal.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. J. Comput. Syst. Sci., 10(1):136–163.