To tree or not to tree?

The Quest for Sentence Structure in Natural Language Processing

Zdeněk Žabokrtský

Institute of Formal and Applied Linguistics
Charles University in Prague

Prague Gathering of Logicians, February 12-13, 2016

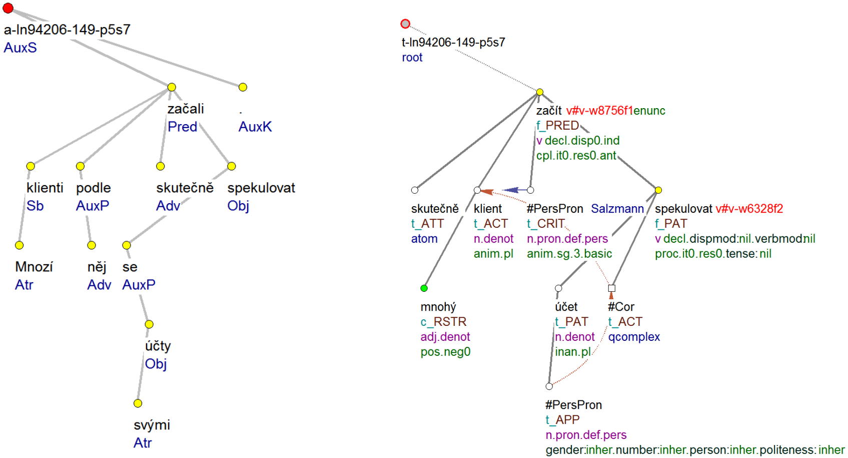
I'll be shamelessly borrowing all kinds of materials from my colleagus throughout the talk.

Dependency trees – a first glimpse

- tree-shaped sentence analysis
  - familiar to everyone who went through the Czech education system:

Credit: http://konecekh.blog.cz

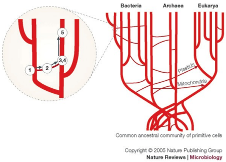
Dependency trees – a more modern look



Credit: Prague Dependency Treebank 2.0, sample selection by Jan Hajič

To tree or not to tree, that is the question.

- A tree is an **irresistibly attractive** data structure, but . . .
- Formal linguists are not the only ones to face this question.
  - geneticists hesitate because of horizontal gene transfer



Credit: Nature Publishing Group

  - interfaith families hesitate before Christmas

Credit: http://www.frumsatire.net

Outline of the talk

Actually there are more questions to discuss today:

- **WHAT?** What kind of creatures are those dependency trees?
- **HOW?** How can we build such trees automatically?
- **WHY?** Are the trees really useful in NLP applications?

Part 1:

WHAT?

What kind of trees do we search for?

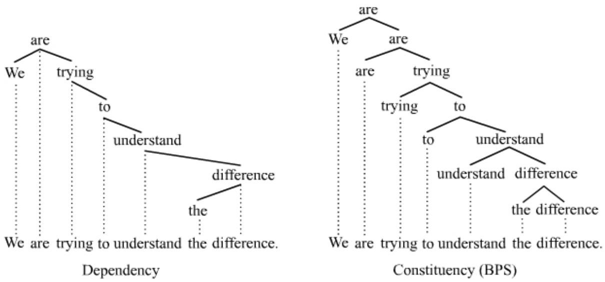
Initial thoughts

1. We believe sentences can be reasonably represented by discrete units and relations among them.
2. Some relations among sentence components (such as some word groupings) make more sense than others.
3. In other words, we believe there is an latent but identifiable discrete structure hidden in each sentence.
4. The structure must allow for various kinds of nestedness (*. . . a já mu řek, že nejsem Řek, abych mu řek, kolik je v Řecku řeckých řek . . .*).
5. This resembles recursivity. Recursivity reminds us of trees.
6. Let's try to find such trees that make sense linguistically and can be supported by empirical evidence.
7. Let's hope they'll be useful in developing NLP applications such as Machine Translation.

So what kind of trees?

There are two types of trees broadly used:

- constituency (phrase-structure) trees
- dependency trees



Credit: Wikipedia

Constituency trees simply don't fit to languages with freer word order, such as Czech. Let's use dependency trees.

How do we know there is a dependency between two words?

- There are various clues manifested, such as
  - word order (juxtapositon): "... *přijdu zítra* ..."
  - agreement: "... *novými*$_{.pl.instr}$ *knihami*$_{.pl.instr}$ ..."
  - government: "... *slíbil Petrovi*$_{.dative}$ ..."

- Different languages use different mixtures of morphological strategies to express relations among sentence units.

Basic assumptions about building units

If a sentence is to be represented by a dependency tree, then we need to be able to:

- identify **sentence boundaries**.
- identify **word boundaries** within a sentence.

Basic assumptions about dependencies

If a sentence is to be represented by a dependency tree, then:

- there must be a **unique parent word** for each word in each sentence, except for the root word
- there are **no loops** allowed.

Even the most basic assumptions are violated

- Sometimes **sentence boundaries are unclear** – generally in speech, but e.g. in written Arabic too, and in some situations even in written Czech (e.g. direct speech)
- Sometimes **word boundaries are unclear**, (Chinese, "ins" in German, "abych" in Czech).
- Sometimes its **unclear which words should become parents** (A preposition or a noun? An auxiliary verb or a meaningful verb? ...).
- Sometimes there are too many relations ("Zahlédla ho bosého."), which implies **loops**.

Life's hard. Let's ignore it and insist on trees.

Counter-examples revisited

If we cannot find lingustically justified decisions, then make them at least consistent.

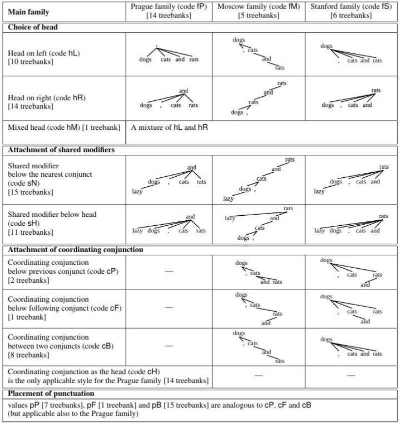
- Sometimes sentence boundaries are unclear (generally in speech, but e.g. in written Arabic too. . .)
  - **OK, so let's introduce annotation rules for sentence segmentation.**
- Sometimes word boundaries are unclear, (Chinese, "ins" in German, "abych" in Czech).
  - **OK, so let's introduce annotation rules for tokenization.**
- Sometimes it's not clear which word should become parent (e.g. a preposition or a noun?).
  - **OK, so let's introduce annotation rules for choosing parent.**
- Sometimes there are too many relations ("Zahlédla ho bosého."), which implies loops.
  - **OK, so let's introduce annotation rules for choosing tree-shaped skeleton.**

Treebanking

- Is our dependency approach viable? Can we check it?
- Let's start by building the trees manually.
- a treebank - a collection of sentences and associated (typically manually annotated) dependency trees
- for English: Penn Treebank [Marcus et al., 1993]
- for Czech: Prague Dependency Treebank [Hajič et al., 2001]
  - layered annotation scheme: morhology, surface syntax, deep syntax
  - dependency trees for about 100,000 sentences
- high degree of design freedom and local linguistic tradition bias
- different treebanks $\Longrightarrow$ different annotation styles

Case study on treebank variability: Coordination

- coordination structures such as "*lazy dogs, cats and rats*" consists of
  - conjuncts
  - conjunctions
  - shared modifiers
  - punctuations
- 16 different annotation styles identified in 26 treebanks (and many more possible)
- different expressivity, limited convertibility, limited comparability of experiments...
- **harmonization of annotation styles badly needed!**



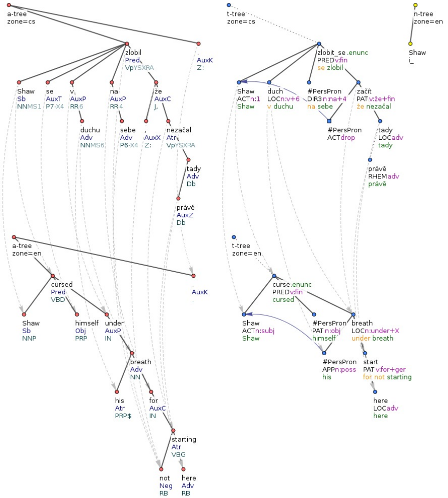
Main family	Prague family (code fP) [14 treebanks]	Moscow family (code fM) [5 treebanks]	Stanford family (code fS) [6 treebanks]
Choice of head			
Head on left (code hL) [10 treebanks]	, dogs cats and rats	dogs , cats and rats	dogs , cats and rats
Head on right (code hR) [14 treebanks]	and dogs , cats rats	rats and cats dogs ,	rats dogs , cats and
Mixed head (code hM) [1 treebank]	A mixture of hL and hR		
Attachment of shared modifiers			
Shared modifier below the nearest conjunct (code sN) [15 treebanks]	and dogs , cats rats lazy	rats and cats dogs , lazy	rats dogs , cats and lazy
Shared modifier below head (code sH) [11 treebanks]	and lazy dogs , cats rats	rats lazy and cats dogs ,	rats lazy dogs , cats and
Attachment of coordinating conjunction			
Coordinating conjunction below previous conjunct (code cP) [2 treebanks]	—	dogs , cats and rats	dogs , cats rats and
Coordinating conjunction below following conjunct (code cF) [1 treebank]	—	dogs , cats rats and	dogs , cats rats and
Coordinating conjunction between two conjuncts (code cB) [8 treebanks]	—	dogs , cats and rats	dogs , cats and rats
Coordinating conjunction as the head (code cH) is the only applicable style for the Prague family [14 treebanks]		—	—
Placement of punctuation			
values pP [7 treebanks], pF [1 treebank] and pB [15 treebanks] are analogous to cP, cF and cB (but applicable also to the Prague family)			

How many treebanks are there out there?

- growing interest in dependency treebanks in the last decade or two
- existing treebanks for about 50 languages now (but roughly 7,000 languages in the world)
- UFAL participated in several treebank unification efforts:
  - 13 languages in CoNLL in 2006
  - 29 languages in HamleDT in 2011
  - 37 languages in Universal Dependencies in 2015:

We don't do only monolingual data

- parallel Czech-English treebank CzEng
- 15 million sentence pairs in version 1.0 [Bojar,2012]
- annotated fully automatically



Conclusion from Part 1

- No assumptions can be taken for granted.
- But we can hopefully live with that, as
  - dependencies are often manifested in a relatively tangible way,
  - simplifications can be introduced,
  - artificial annotation rules for deciding unclear cases can be added,
  - annotation schemes can be verified by manual annotations,
  - massively crosslingual view helps us not to be trapped in a local linguistic tradition.
- Nowadays, dependency trees seem to be the most viable syntactic model applicable accross languages.

Part 2:

HOW?

How can we build dependency trees automatically?

Dependency parsing

Task specification:

- **Input**: a sequence of words (typically also their lemmas and morphological tags)
- **Output**: for each word (except the root word) find its parent word

Evaluation criterion:

- **Unlabelled attachment score**: percentage of words for which correct parents were found
- **Labelled attachment score**: percentage of words for which correct parents were found and whose dependency label were correct too
- Obvious drawback: all types of errors considered equally important

Typology of parsers in NLP

- rule-based
- data-driven
  - supervised – big amount of manually annotated trees available
  - unsupervised – no manually annotated trees available
  - semi-supervised – something in between

Rule-based parsers

- more or less **obsolete**
- although hand-coded grammars are immensely successful in computer science...
- ...it is surprisingly difficult (if not impossible) to design a reliable hand-written a grammar for a natural language
- the law of diminishing returns applies very quickly
  - a few simplest grammar patterns (such as determiner-adjective-noun) are easy to exploit
  - but errors start interfering with more complex rules very soon and the system becomes unmaintainable

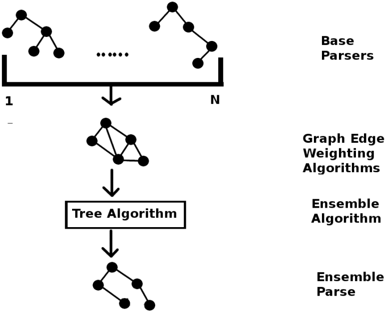
Supervised parsing

Main approaches:

- **graph-based**: we learn a model for scoring graph edges, and search for the highest scoring tree (global optimization e.g. by Maximum Spanning Tree algorithm)
- **transition-based**: a shift-reduce parser gradually processing words stored in a queue,
- **CFG-based**: a constituency parser applied first, then resulting constituency trees converted to dependencies

Supervised parsing: ensamble parsing

- Task:
  - Input: dependency trees resulting from several parsers
  - Output: a single dependency tree
- Intuition: different parsers are correct in different places.
- Greedy *argmax* parent selection insufficient
- Treeness constraint kept e.g. by applying Maximum Spanning Tree again [Green-Žabokrtský, 2012])



Unsupervised parsing

- Treebanks for about 50 languages exist . . .
- . . . but what about the remaining 6950 languages?
- How can we build parsers from nothing, without having a single hand-annotated tree?
- Extremely challenging task!

Unsupervised parsing by Gibbs sampling

- we can employ the rich-gets-richer principle to amplify detected regularities
- for instance by Gibbs sampling [Mareček, 2011]
  1. build a probabilistic model (assign probability to each tree) using e.g.:
     - ★ prior knowledge: edge length, node fertility,
     - ★ sentence fragment reducibility
     - ★ word frequency (tendency: frequent $\Longrightarrow$ auxiliary $\Longrightarrow$ leaf)
     - ★ above all: prefer repeated patterns
  2. initialize trees randomly
  3. iterate:
     - ★ generate a random small change of some of the trees (sampled proportionally to its probability)
     - ★ update the model

Semi-supervised parsing

- typically an under-resourced scenario:
- some hand-annotated trees are available ...
- ... but they are not sufficient for supervised approach, because
  - the data is too small (sometimes only a few trees)
  - or no data available for a particular languages, only for some other languages

Semi-supervised parsing example: weighted multisource delexicalized parser transfer

- **parser transfer** = we neet to parse language A, but have only training data for language B
- **delexicalized** = we ignore words, we use only part-of-speech tags (*Noun Verb Noun* instead of *John loves Mary*)
- **multisource** = treebanks for more languages (B,C,D...) are used
- **weighted** = we give different weight to information gained from different languages, according to similarity A-B, A-C, A-D, ...
- a possible similarity measure: Kullback-Leibler divergence on distribution of part-of-speech trigrams [Rosa-Žabokrtský, 2015]

Part 3:

WHY?

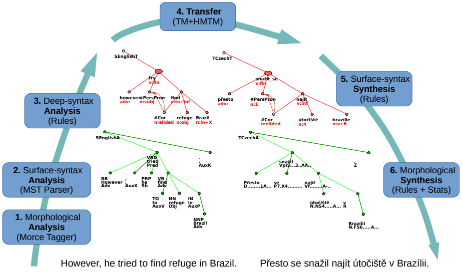
Are the trees useful?

Golden Rule of Natural Language Processing

- Whatever task you try to solve in NLP, you can convincily argue that it will be useful for Machine Translation ...
- ... but it hardly ever really is.
- (but this time there will be a happy ending eventually)

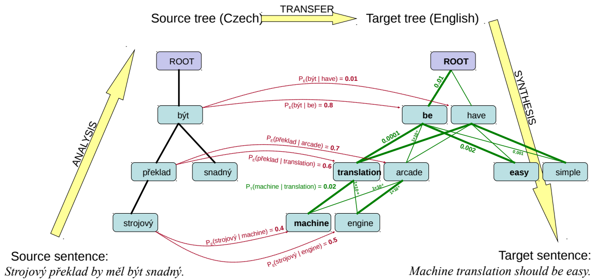
TectoMT:
a dependency-based machine translation system

- developed in UFAL
- three phases:
  1. analysis up to deep-syntactic trees,
  2. transfer on the deep-syntactic level
  3. synthesis down to sentence string level
- most components trainable, for instance Maximum-Entropy based translation dictionary [Mareček et al., 2010]



Hidden Tree Markov Model for MT

- inspirid by noisy-channel model
- combination of translation model and target side language model
- but this time on dependency trees
- global optimum searched by tree-modified Viterbi algorithm [Žabokrtský-Popel, 2009]

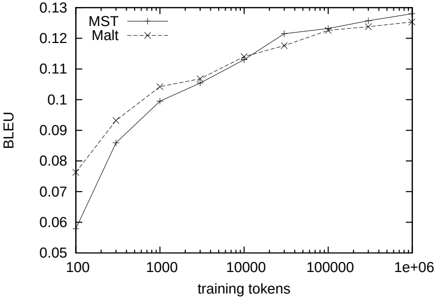


P_E(source | target) … emission probabilities … **translation model**

P_T(dependent | governing) … transition probabilities … **target-language tree model**

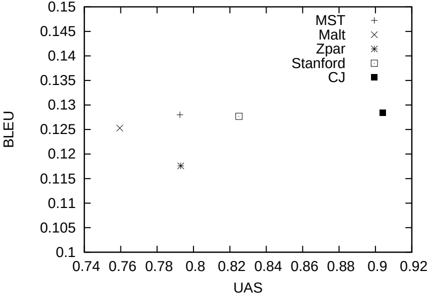
TectoMT: what about more training trees for parsing?

- in fact we have no more extra annotated data
- but we can downscale the data and try to extrapolate
- BLEU (horizontal axis) – an automatized estimate of parsing quality
- close-to-log growth $\Longrightarrow$ exponentially growing annotation costs



TectoMT: what about different parsers?

- five different parsers plugged into the translation system [Popel et al., 2011]
- higher parsing quality does not imply higher translation quality



DeepFix: dependency-based post-editting of an MT system's output

- Example: *EU criticizes not only the Greek government.*
- Google translation: *EU kritizuje nejen <u>řecká vláda</u>.*
- intuition: it should be possible to fix such errors if we model target language grammar
- in this case we model valency frames:
  - P(nominative | kritizovat, object) = 0.03
  - P(accusative | kritizovat, object) = 0.80
- DeepFix post-editted sentence: *EU kritizuje nejen <u>řeckou vládu</u>.*
- dependency trees needed, e.g., for imposing attribute agreement
- improvement of state-of-the-art systems' translation quality

Thank you!