Syntax-Based Translation: The Good, The Bad, and How to Win Big

Adam Lopez

with thanks to Ondřej Bojar
(and apologies to Richard P. Gabriel)

- Why do we care about syntax-based MT?
- How does it work?
- What are the open problems?

Disclaimer
Fast-moving field, we only scratch the surface
Phrase-based models are good, but not perfect

- computing all possible reorderings is NP-complete
- can’t generalize
- can’t model long-distance dependencies
- can’t model grammaticality
The Good

Syntax-based models aim to solve these problems

- polynomial complexity
- can generalize
- can model long-distance dependencies
- can model grammaticality
the green witch
la bruja verde

NP $\rightarrow$ DT$_1$JJ$_2$NN$_3$/DT$_1$NN$_3$JJ$_1$

the wicked green witch
la bruja malvada verde

NP $\rightarrow$ DT$_1$JJ$_2$JJ$_3$NN$_4$/DT$_1$NN$_4$JJ$_2$JJ$_3$
Problem Stack decoding doesn’t apply
Idea Decoding is parsing
**Problem**  Phrase-based decoding with full reordering has exponential complexity.

**Idea**  Use binary-bracketing SCFG for polynomial complexity.
**Problem** Phrase-based cannot model grammaticality.

**Idea** Constrain SCFG to target-side syntax.
The Bad
It doesn’t really work.
  - Bracketing grammar doesn’t capture all alignments.
  - Tree isomorphism at production level is too strict.
Where do we go next?
  - More theory?
  - More articulated models?

Modeling translational equivalence using weighted finite state transducers is like approximating a high-order polynomial with line segments... the relatively low expressive power of weighted finite state transducers limits the quality of SMT systems.

–Burbank et al. 2005

But language is hierarchical.

–anonymous MT researcher

I think phrases are a passing fad.

–anonymous MT researcher
This type of difficulty has happened in other research areas.


Lisp = syntax-based models
Unix and C++ = phrase-based models

**Simplicity**  the design must be simple, both in implementation and interface. It is more important for the interface to be simple than the implementation.

**Correctness**  the design must be correct in all observable aspects. Incorrectness is simply not allowed.

**Consistency**  the design must not be inconsistent. A design is allowed to be slightly less simple and less complete to avoid inconsistency. Consistency is as important as correctness.

**Completeness**  the design must cover as many important situations as is practical. All reasonably expected cases must be covered. Simplicity is not allowed to overly reduce completeness.
Simplicity the design must be simple. Simplicity is the most important consideration in a design.

Correctness the design must be correct in all observable aspects. It is slightly better to be simple than correct.

Consistency the design must not be overly inconsistent. It is better to drop those parts of the design that deal 

Worse is Better

Completeness the design must cover as many important situations as is practical. Completeness can be sacrificed in favor of any other quality. In fact, completeness must sacrificed whenever implementation simplicity is jeopardized.

The good news is that in 1995 we will have a good operating system and programming language. The bad news is that they will be Unix and C++.

–Richard Gabriel

In 2018, will we have a good translation system based on phrases?
How to Win Big

Observation  Phrase-based models good at local reordering.
Idea  Use phrases to reorder phrases.
However, the sky remained clear under the strong north wind.

Although north wind howls, but sky still extremely limpid.

Observation  Phrase-based models good, but not grammatical.

Idea  Add syntax, but keep the phrases.
Current status
▶ Syntax-based models competitive with phrase-based
  ▶ Slightly better for Chinese-English
  ▶ Slightly worse for Arabic-English
  ▶ Open question for European languages
  ▶ Language models make a bigger difference
▶ Not as fast as advertised
  ▶ With 5-gram language model – $O(n^{11})$
  ▶ Easy tricks in phrase-based models not applicable
  ▶ Work on clever search algorithms
▶ Parsing progress – 1997: 88.1%, 2007: 92.4%

Many, many more angles
▶ Different formal models with different properties
  ▶ Dependency grammar
  ▶ Synchronous tree substitution grammar
  ▶ Synchronous tree adjoining grammar
▶ Parsing: source, target, or both?

See handout for some further reading