Objectives & Introduction

Question: How can we computationally model reduplication in a way that

- Covers its typology and productivity as a transduction?
- Is practically useful, convenient, compact, and efficient?
- Matches or provides theoretical insights?

Answer & contribution: Use 2-way finite-state transducers to get all 3 results.

Reduplication

Languages can use either affixes or copying to mark meanings

- English plural: book → books
- Indonesian plural: buku → buku-buku

Significant variation in reduplication (Rubino, 2013)

1. Total reduplication = unbounded copy (83%)
   - ‘woman’ → ‘women’ ( Indonesian)

2. Partial reduplication = bounded copy (77%)
   - CV / CV analogues
     - ‘to go’ → ‘to go repeated’ (Sundanese)
   - CVC / CVC analogues
     - ‘leg’ → ‘legs’ (Agta)
   - CVCV / CVCV analogues
     - ‘return’ (Dyirbal)

3. And more (Moravcsik, 1978; Rubino, 2005; Inkelas and Zoll, 2005)

Computing reduplication

Reduplication is a stable challenge in computational linguistics (Sprat, 1992; Roark and Sproat, 2007)

- Total reduplication: not finite-state
- Partial reduplication: state transducer
- Approximate (Hulden, 2009) or go above finite-state tools (Albro, 2005; Crysmann, 2017)?

2-way Finite-State Transducers

2-way FSTs can reconstitute parts of the input string unlike 1-way FSTs (Engelfriet and Hoogeboom, 2001; Shallit, 2008; Piloy and Recuyer, 2016)

$$\left(\Sigma, \Sigma, +1 \right) \rightarrow \left(\Sigma, \lambda, -1 \right) \rightarrow \left(\Sigma, \lambda, +1 \right)$$

- 1-way FSTs ≤ 2-way FSTs ≤ Turing Machines
- 2-way FSTs ≠ MOO string transducers ≠ Streaming String Transducers
- Closed under composition & non-deterministic version is invertible
- Existing applications are efficient (Alur and Černý, 2011; Alur et al., 2014).

2-way FSTs do a lot for computing reduplication

- Empirical coverage: cover virtually all of reduplicative typology
- Practical utility: are easy to design, build, debug, and run
- Linguistic motivation: capture linguistic generalization on reduplication

Empirical coverage = Typology

2-way FSTs can model virtually the entire typology of reduplication (Moravcsik, 1978; Hurcol, 2005; Inkelas and Zoll, 2005; Rubino, 2005; Samuels, 2010) productively

- local or non-local
- total, partial, or sub-constituent
- opaque or transparent phonology

Limited cases of over- and under-generation (Dolatian and Heinz, in press)

Practical utility = RedTyp

Created RedTyp database using 2-way FSTs

- SQL database of reduplicative morphemes and 2-way FSTs for them
- 138 reduplicative processes from 91 languages with 57 distinctive 2-way FSTs

- 2-way FSTs were easy to design, implement, and run with Python
- Compact with average 8.8 states and no state explosion
- URL: github.com/jheins/RedTyp

Linguistic motivation = Origin semantics

2-way FSTs match intentional description of reduplication

- 2-way FSTs actively copy segments from input to output
- 1-way FSTs remember string of segments

Capture copying vs. remembering with origin semantics (Bojarczyk, 2014).

Illustrating origin semantics

The origin information for each position, $\sigma_i$, in output string is the position, $\tau_i$, of the real head on the input string during the transduction

Equivalent general semantics but different origin semantics

- 2-way FST associates only identical ‘copied’ segments from input-to-output while 1-way FST remembers segments in bulk
- 2-way FST matches linguistic theories (Matras, 1982; Inkelas and Zoll, 2005)

Next Steps

Research questions

- Morphological analysis: non-deterministic 2-way FSTs
- Integrate into NLP?
- Copying outside of reduplication? Potentially the same
- Subclasses & learning? We did it

Results

Subclasses: Bulk of typology in RedTyp doesn’t need full power of 2-way FSTs

- Majority (87%) is in C-OSL subclass
- Concatenated OSL functions (Chandlee et al., 2015)
- Interfaces with morpho-phonology at best go up to function composition

Learnability: Well-defined subclasses pave way to learnability (Dolatian and Heinz, 2018)

Conclusions

- Reduplication can be modeled with 2-way FSTs
- 2-way FSTs are compact, convenient, and capture linguistic generalizations and typology
- 2-way FSTs opens many doors for research and applications for computational morphology

References