Proceedings of the Society for Computation in Linguistics

Volume 2 Article 56

2019

Re(current) reduplication: Interpretable neural network models of morphological copying

Colin Wilson
Johns Hopkins University, colin.wilson@jhu.edu

Re(current) reduplication: Interpretable neural network models of morphological copying

Colin Wilson (colin@cogsci.jhu.edu) / Johns Hopkins University

Introduction. Patterns of reduplication, in which the realization of a morpheme involves complete or partial copying of a stem, have a rich descriptive typology. The study of reduplication has also been central to many areas of linguistic theory, including the morphology/phonology interface, prosodic morphology, and constraint-based approaches to grammar (see Inkelas & Downing 2015, Downing & Inkelas 2015 for a comprehensive review). In contrast, computational research on morphology has tended to avoid reduplication because of its apparent unwieldiness: for full reduplication, a straightforward finite-state implementation requires a state set that grows exponentially with the maximum length of stems that can be copied, and even for bounded partial reduplication this approach requires exhaustive enumeration of copyable sequences. Most computational analyses of reduplication patterns that do exist involve specialized mechanisms and, most importantly, are hand-written rather than learned from data (e.g., Walther 2000, Beesley & Karttunen 2003, Hulden & Bischoff 2009); one exception, proposed by Frank & Tenenbaum (2011), is limited to learning syllable reduplication in ABA and ABB patterns.

This talk introduces a recurrent neural network model of reduplication that provides analyses for a range of attested patterns, including full reduplication (e.g., *talk* → *talk-talk*), reduplication with overwriting (e.g., *talk* → *talk-shmalk*), and various forms of partial copying. While similar at a high level to other encoder-decoder networks that have recently been applied to morphological patterns (e.g., Malouf 2017), the model proposed here is distinguished by being highly interpretable: it contains functionally-specialized layers whose representations and processes have clear counterparts in linguistic analyses of reduplication. Furthermore, because all of the operations in the network are differentiable, it offers a novel approach to learning reduplication patterns from examples. Several representative simulations probe the potential of this approach.

Network representations. The network embeds stems, reduplicants and other affixes, and output forms as (exact or approximate) tensor product representations (TPRs; e.g., Smolensky 1990). Each symbol x is embedded as an m-dimensional vector $\mathbf{x}$; these vectors are arbitrary here except that they have privative components indicating the presence of a symbol (as opposed to ε), the identities of the word boundaries (start ⋊ and end ⋉), and the consonant/vowel status of each non-boundary symbol. Separately, each ordinal position $i \in 0, ..., n-1$ is embedded as a one-hot vector $\mathbf{r}_i$ (where n is the maximum sequence length that a given network can represent). An exact TPR is created by summing bindings ($\mathbf{x} \otimes \mathbf{r} = \mathbf{x}\,\mathbf{r}^T$) of symbol and position vectors; this is how input stems are provided to the model (e.g., $\rtimes\otimes\mathbf{r}_0 + \mathbf{t}\otimes\mathbf{r}_1 + \mathbf{a}\otimes\mathbf{r}_2 + \mathbf{k}\otimes\mathbf{r}_3 + \ltimes\otimes\mathbf{r}_4$ for *talk*). Representations learned by the network (e.g., for fixed affixes) and outputs created during its processing are typically approximate TPRs in which 'blends' of symbol vectors are combined with position vectors. Multiplying a TPR matrix $\mathbf{X}$ with a designated unbinding vector $\mathbf{u}_i$ returns a symbol or blend; computing the negative Euclidean distance of the result to each symbol embedding and applying softmax yields a probability distribution over symbols in the ith position.

Network processing. The *reduplicant* layer receives the TPR $\mathbf{S}$ of the stem as input and performs two operations: it extracts a contiguous portion of the stem (i.e., annotates the remaining stem symbols for deletion) and adds a possibly empty affix to the stem. Extraction is performed with a bidirectional scan of the stem by a recurrent unit that has minimal hidden state (two scalars);

Proceedings of the Society for Computation in Linguistics (SCiL) 2019, pages 379-380.
New York City, New York, January 3-6, 2019

affixation works as described below. The output of this layer is an approximate TPR **R**. The *pivot* layer identifies the point in the stem at which the reduplicant is added (e.g., prefixed or suffixed) with a bidirectional scan of **S** by a recurrent unit defined as in the reduplicant layer. The *combination* layer than transforms [**S**, **R**, *pivot*] into an output TPR **O** by (i) unbinding symbol vectors from **S** and binding them to successive position vectors in the output until the *pivot* point is reached, then (ii) unbinding symbol vectors from **R** and binding them into the output until the reduplicant is exhausted, and finally (iii) unbinding any remaining symbol vectors of **S** and binding them successively into the output. The result **O** can be decoded as described above and thereby used to assess the log-probability of each target output symbol for the given stem.

Simulations. The results here are typical of several learning simulations with the same hyperparameters and other details (Adagrad optimizer, epochs = 1000, minibatch size = 40, learning rate = 0.1, affix regularizer coefficient = 0.001). In each case ∼3/4 of the available stem-output pairs were used for training with the rest held out for testing. No morpheme boundaries or other annotations were supplied during training. English full reduplication was learned from a large set of monomorphemic nouns (e.g., *talk* → *talk-talk*): train accuracy 1.0 (2739/2739), test accuracy 1.0 (914/914). The simulation for English shm-reduplication involved a smaller corpus of examples gathered from the internet (e.g., *talk* → *talk-shmalk*): train 1.0 (42/42), test 0.80 (12/15); test errors were minimal edits of target outputs (e.g., *easy* → *easy-shmaasy*). Bengali echo reduplication, which involves full copy with *t* overwriting an initial consonant (e.g., *boj* 'books' → *boj toj* 'books or anything'), was learned from the examples in Khan (2007): train 1.0 (93/93), test 1.0 (31/31). Amele iterative reduplication, in which copying the initial consonant-vowel of a verb indicates simultaneous action (e.g., *bagawen* 'he came out' → *ba-bagawen* 'as he came out'), was learned from examples in Roberts (1987): train 1.0 (75/75), test 1.0 (26/26). Finally, Ilokano plural reduplication is a textbook case of partial copying in which an initial heavy syllable is extracted and prefixed: train 1.0 (27/27), test 0.9 (8/9).

Summary. These results support the development of neural networks that successfully learn and generalize morphological operations using representations and processes that are linguistically interpretable, and motivate extensions to an even wider typology (e.g., foot copying).

References. Beesley, K. R., & Karttunen, L. (2003). *Finite-state morphology: Xerox tools and techniques*. Stanford, CA: CSLI Publications. • Downing, L. J., & Inkelas, S. (2015). What is reduplication? Typology and analysis part 2/2: The analysis of reduplication. *Language and Linguistics Compass*, 9(12), 516–528. • Frank, M. C., & Tenenbaum, J. B. (2011). Three ideal observer models for rule learning in simple languages. *Cognition*, 120(3), 360–371. • Hulden, M., & Bischoff, S. T. (2009). A simple formalism for capturing reduplication in finite-state morphology. In *Proceedings of FSMNLP* (pp. 207–214). Association for Computational Linguistics. • Inkelas, S., & Downing, L. J. (2015). What is reduplication? Typology and analysis part 1/2: The typology of reduplication. *Language and Linguistics Compass*, 9(12), 502–515. • Khan, S. (2007). Similarity avoidance in East Bengali fixed-segment reduplication. MA thesis, University of California, Los Angeles. • Malouf, R. (2017). Abstractive morphological learning with a recurrent neural network. *Morphology*, 27(4), 431–458. • Roberts, J. R. (1987). *Amele*. London: Croom Helm. • Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2), 159–216.