

Weighted posets: Learning surface order from dependency trees

William Dyer

Oracle Corp

william.dyer@oracle.com

Abstract

This paper presents a novel algorithm for generating a surface word order for a sentence given its dependency tree using a two-stage process. Using dependency-based word embeddings and a Graph Neural Network, the algorithm first learns how to rewrite a dependency tree as a partially ordered set (poset) with edge-weights representing dependency distance. The subsequent topological sort of this poset reflects a surface word order. The algorithm is evaluated against a naive baseline of average dependency distances across 14 languages, performing well in terms of rank correlation and resulting rate of projectivity based on Universal Dependencies corpora.

1 Introduction

In a tradition dating at least back to Tesnière (1959), the words in a phrase or sentence can be thought of as a set of heads and dependents. Each word save the root is a dependent of another word, its head, and heads and dependents exist in a one-to-many relationship (Polguère and Mel’čuk, 2009). This arrangement of heads and their dependents forms a tree, or more formally an unordered directed acyclic graph (DAG), in which words are nodes and edges are the dependency relations. A sentence is one possible linearization or surface order of the DAG.

This paper describes a method for learning how to generate a valid¹ surface order from a dependency tree. Determining the underlying dependency tree from a surface order is the rather extensively studied task of parsing; this paper concerns the opposite task.

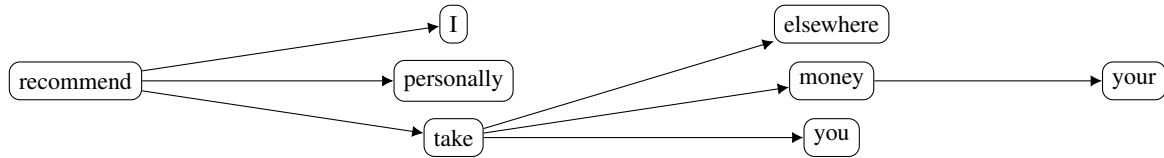
The key insight of the paper is that rather than learning to directly convert a dependency tree to surface order, the target is instead an edge-weighted partially ordered set (poset). The poset’s edge direction represents linear precedence in the surface order, while edge weight represents dependency distance, the number of words intervening between dependent and head in the surface order. The topological sort or linear extension of this poset—performed such that nodes connected by edges with smaller weights are placed closest to each other—reflects the surface order of the dependency tree.

For example, Figure 1 shows (a) the dependency tree, (b) edge-weighted poset, and (c) surface order of the sentence *Personally I recommend you take your money elsewhere*. Rather than attempting to learn how to convert (a) directly into (c), the approach outlined here rewrites (a) to (b) by learning edge directions and weights, then rewrites (b) to (c) via topological sort. Given examples of dependency trees and their corpus-attested surface orders, a neural network can learn to convert previously unseen dependency trees into surface orders by way of a weighted poset.

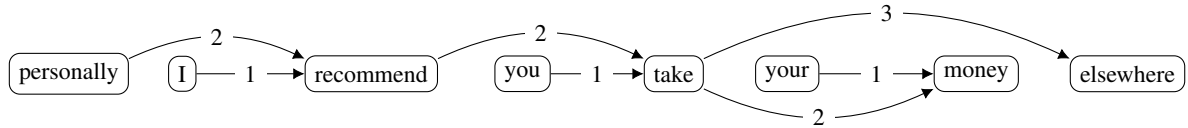
Implemented as a Graph Neural Network, the machine-learning algorithm treats inputs, targets, and outputs as directed graphs. Further, by representing words with their dependency-based embeddings—that is, embeddings trained on syntactic rather than linear contexts—the model generates a linearized surface order as the final step only, performing all other analysis within a graph framework. In this way surface order is treated as an emergent consequence of topologically sorting an edge-weighted poset, the weights of which represent learned dependency distances.

¹Validity here should be taken neither grammatically nor prescriptively, but rather as a stand-in for attested in a corpus. That is, the model developed herein learns to order the words in a dependency tree based on the structural regularities of a corpus, not intuitive or prescribed grammaticality judgments.

(a) unordered dependency DAG



(b) edge-weighted poset



(c) surface order

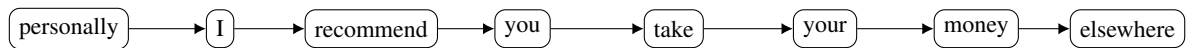


Figure 1: Three graph-theoretic representations of a sentence. (a) A dependency tree as an unordered directed acyclic graph (DAG). (b) A poset in which edge weight indicates dependency distance in the surface order. (c) A surface order generated by a topological sort of the poset in (b).

2 Background literature

2.1 Related linguistic work

Word order is one of the oldest and most prominent areas in the field of linguistics, and as such a wide variety of models have been advanced seeking to describe and understand word-order variation (Song, 2012). It has been approached from generalist perspectives, as in Behaghel’s “what belongs together semantically is also placed close together” (1932, p. 4) or Uniform Information Density (Jaeger and R. Levy, 2006), as well as from specific constituent types, such as the ordering of adpositions and adverbials by manner, place and time (Boisson, 1981); demonstratives, numerals, and descriptive adjectives (Greenberg, 1963; Dryer, 2009); or adjectives by size, shape, and so on (Scott, 2002).

Building on principles such as Head Proximity (Rijkhoff, 1986), Early Immediate Constituents (Hawkins, 1994), Dependency Locality Theory (Gibson, 2000), and Minimize Domains (Hawkins, 2004), a recent approach to word order holds that the dependency distance²—the number of words intervening between a dependent and its head—should be minimized and long-distance dependencies should be avoided (Hudson, 1995; H. Liu et al., 2017). Dependency Distance Minimization (DDM) proposes that a surface order with a smaller cumulative or mean dependency distance is generally favored over alternatives, a tendency that may be universal (Futrell et al., 2015).

However, DDM alone cannot fully explain word order: it does not distinguish between total mirror orders—the dependency distances of *the cat purrs* are presumably the same as *purrs cat the*—or, more plausibly, partial mirror orders such as the swapped adjectives in *big red barn* or *red big barn*. Methods for extending DDM include employing phonemes or syllables as the unit of distance (Ferrer-i-Cancho, 2017), or exploring the relationship between dependents and heads in information-theoretic terms (Dyer, 2018; Hahn et al., 2018). Another avenue is some sort of linear principle that could operate to differentiate mirror surface orders, such as “old concepts come before new ones” (Behaghel, 1932, p. 4), or the possibly contradictory “provide the most important information first” (Gundel, 1988, p. 229).

One way to conceive of surface order is as the result of rewriting a dependency graph by modifying its edge directions to reflect linear order. This process represents an intermediate stage between syntactic structure and surface order in which the linear order of certain word pairs is expressed as a series of precedence relations (Gerdes and Kahane, 2001; Kahane and Lareau, 2016). These precedence relations form a partially ordered set (poset) which can be topologically sorted into a non-unique linearization.

²Dependency distance is also referred to as dependency length in the literature.

2.2 Related NLG work

The field of natural language generation (NLG) seeks to model word order in the service of generating accurate natural language. Contra Harris (1954)³, language is often seen within NLG as a bag of words in which the task of realizing surface order is based on an n -gram language model (Filippova and Strube, 2009). A common implementation follows the bottom-up insights from dependency parsing (Y. Liu et al., 2015), and features such as syntactic category or dependency relations can improve algorithms for linearizing a bag of words (Zhang and Clark, 2015).

The First Multilingual Surface Realisation Shared Task (SR ‘18) brought together nine submissions in a shallow track requiring teams to determine word order and inflections of shuffled and lemmatized Universal Dependencies (UD) data, evaluated by both statistical and human assessment (Mille et al., 2018). For the linearization subtask, of the four submissions with the highest BLEU⁴ scores in at least one of the 10 supported languages: Puzikov and Gurevych (2018) use a bigram language model with binary neural-net classification; Elder and Hokamp (2018) treat the task as a machine-translation problem, using sequence-to-sequence models augmented with synthetic and outside data; Castro Ferreira et al. (2018) sort dependents into preceding and following groups which are then sorted by syntactic category or with a maximum entropy classifier; and King and White (2018) use features such as syntactic category, projectivity, and dependency distance to build a language model to incrementally linearize words.

It has long been noted that a reliance on statistical n -gram metrics like BLEU for measuring generated language is problematic given their inability to generalize seemingly unimportant word order variation or synonymy (Pastra and Saggion, 2003; Turian et al., 2006), as well as their lack of correlation with human assessment (Novikova et al., 2017). BLEU specifically has been criticized given its understudied technological biases, a sufficient reason to avoid using it alone to report scientific evidence (Reiter, 2018, p. 399). Further, while the target or reference of generated language is not necessarily a single sentence—there may be more than one semantically and syntactically valid surface realization of a given set of words, with context determining appropriateness—limited resources often result in a single human-produced reference being used, usually in the guise of an attested sentence in a corpus.

2.3 Projectivity

Projectivity refers to the constraint that a head and its dependents must occur in a contiguous sequence in the surface order (Marcus, 1965). Violations of projectivity—often referred to as discontinuities in the linguistics literature—are instances when a word occurring between a head h and dependent d is not dominated by h in the dependency tree. In the oft-cited non-projective sentence *The hearing is scheduled on the issue today*, both *is* and *scheduled* occur between *hearing* \rightarrow *issue*⁵, but are not dominated by *hearing*. A projective order would be *The hearing on the issue is scheduled today*.

It seems that all natural languages contain some amount of non-projective dependency relations, though calculating exact rates of non-projectivity is difficult given design decisions in the original parsing to create corpora. That is, some annotation schemes presuppose projectivity, and as a result corpora produced following those schemes will not exhibit discontinuities (Ferrer-i-Cancho and Gómez-Rodríguez, 2016). Observed percentages of non-projectivity range from single digits to the mid-teens depending on language, though sources disagree, likely due to differences in corpora, genre, and annotation scheme.

Non-projectivity must be accounted for in any model of word order. Parsers have been developed which allow pseudo-projective (Nivre and Nilsson, 2005), non-projective (Nivre, 2009), and mildly non-projective dependencies (cf. Gómez-Rodríguez, 2016). Similarly, the submissions to SR ‘18 vary with regard to projectivity: of the eight, three explicitly exclude non-projective arcs due to algorithmic design (Basile and Mazzei, 2018; Puzikov and Gurevych, 2018; Sobrevilla Cabezudo and Pardo, 2018), while one follows the tendency toward limited non-projectivity by “encourag[ing] the model to learn that most choices should yield continuous phrases” (King and White, 2018, p. 42).

³“[L]anguage is not merely a bag of words but a tool with particular properties which have been fashioned in the course of use” (p. 156).

⁴BLEU, for bilingual evaluation understudy (Papineni et al., 2002), is “the geometric mean of the n -gram precisions between generated text and reference texts and adds a brevity penalty for shorter sentences” (Mille et al., 2018, p. 4).

⁵Following the UD convention of adpositions depending on nouns, we have *hearing* \rightarrow *issue* and *issue* \rightarrow *on*.

The causal relationship between Dependency Distance Minimization and projectivity is unsettled. Ninio (2017) concludes that “projectivity appears to be not so much a side-effect of DDM as a mathematical requisite for a method to encode a two-dimensional tree in a one-dimensional sentence-string in a way that makes reconstruction possible” (p. 216), appealing to other linguistic structures such as catenae (Osborne et al., 2012) to explain discontinuities. This traditional view—that projectivity exists as a principle independent of DDM—is largely disproven by an analysis which positively correlates dependency distance and the number of crossing dependencies across a variety of multilingual corpora (Ferrer-i-Cancho and Gómez-Rodríguez, 2016). Park and R. Levy (2009) note that an avoidance of long-distance dependencies can result in non-projective surface orders.

2.4 Syntactic word embeddings

The relationship between words has long been thought of distributionally; as Firth (1957) memorably puts it: “you shall know a word by the company it keeps” (p. 11). The company or context of a word is often conceived in terms of the linear neighbors that commonly occur around that word, a context that can be quantified with a dense vector or series of numbers called an embedding. Algorithms have been developed to learn a word’s embedding in a corpus, such as skip-grams (Mikolov et al., 2013). O. Levy and Goldberg (2014) extend the notion of context beyond linear neighbors in their `word2vec-f` to use dependency relations in learning syntactic embeddings: a word’s context is based on the heads and dependents it takes in a corpus.

The number of dimensions necessary for a given task is an understudied problem. It is widely accepted that larger dimensions are better, up to a point of diminishing returns; for example, O. Levy and Goldberg (2014) use 300 in their evaluation, mentioning that 600 produces similar results. However, Spirling and Rodriguez (2019) note that very large dimensions relative to corpus size result in greater instability of embeddings, where instability refers to the rate at which the cosine-similar nearest neighbors differ between models (Wendlandt et al., 2018). Patel and Bhattacharyya (2017) explore the lower bound of embedding dimensions, below which performance suffers, providing a rather complicated method for calculating the minimum based on the maximum clique of a cosine-similarity matrix of word co-occurrence. An industry rule-of-thumb⁶ is to use the fourth root of vocabulary size.

2.5 Graph neural networks

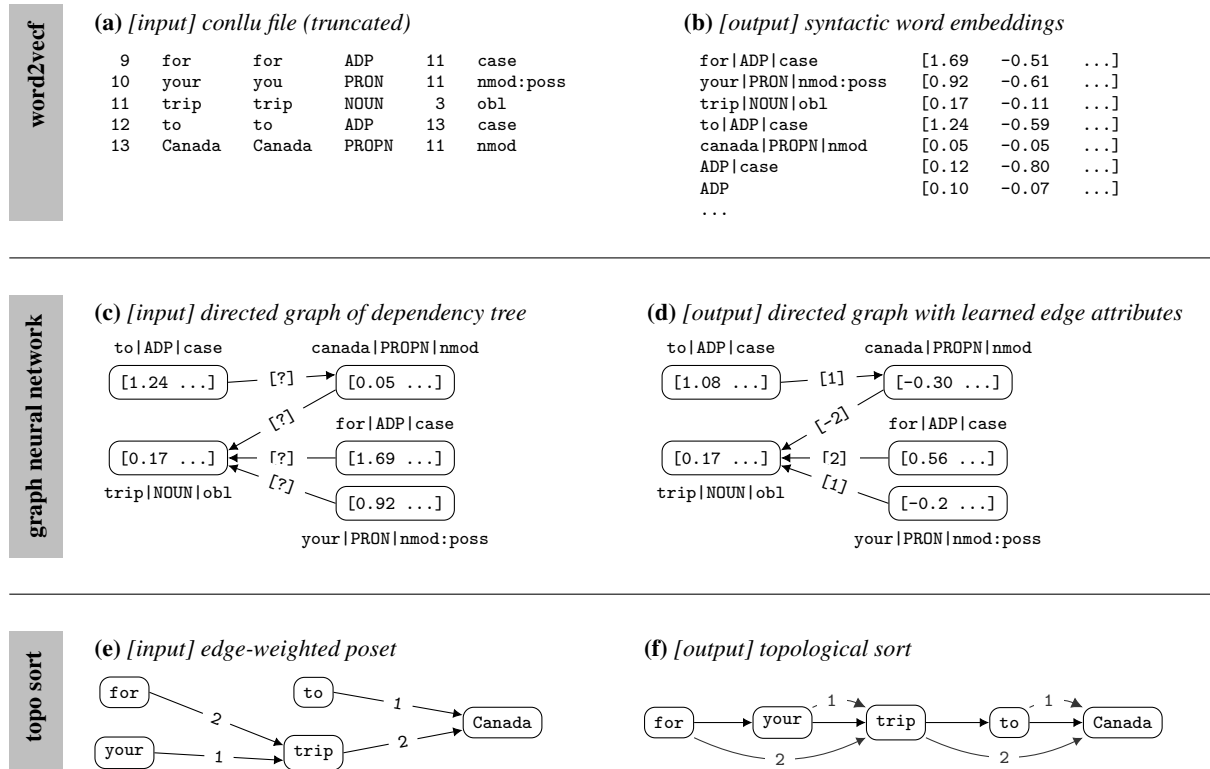
While machine-learning algorithms, deep or otherwise, have traditionally operated on data represented in Euclidean space—for example, image data can be represented as a regular grid of pixel values—graph neural networks (GNN) allow the complexity of graph structures to be analyzed (Wu et al., 2019). The Graph Nets (GN) framework relies on a graph-to-graph model called a GN block “which takes a graph as input, performs computations over the structure, and returns a graph as output” (Battaglia et al., 2018, p. 11). In this framework, a graph is composed of nodes and their attributes, edges and their attributes, and a set of global attributes. Input and target graphs may contain different node and edge configurations; only the attributes for nodes and the attributes for edges must be of a consistent form. It is these sets of attributes which form the learned parameters of the neural network.

GN blocks also support message-passing neural networks (MPNN) (Gilmer et al., 2017), a method by which a graph’s node and edge attributes undergo spatial-based graph convolutions and pooling (Wu et al., 2019, p. 8). In this manner a graph’s connected nodes influence each other’s node and edge attributes, passing information along directed edges.

3 Methodology

The approach described in the current study rests on the notion that adding dependency distances as positive or negative edge weights to a dependency tree allows the DAG to be rewritten as a poset whose topological sort reflects a surface order. Edge weights are therefore the number of words intervening between a dependent and its head, where negative weights indicate a dependent that precedes its head and positive a dependent following its head. Learning these edge weights is the core goal of the model.

⁶<https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html>



3.2 Graph neural network implementation

The machine-learning algorithm is implemented using Graph Nets and Sonnet, two DeepMind libraries⁹ for building graph neural networks using Google’s Tensorflow (Abadi et al., 2015). The network’s layers contain 18 neurons each and follow an ‘encode-process-decode’ model common to many Graph Nets implementations. Because learned edge weights in the GNN can be positive or negative, loss is calculated as the absolute difference between target and output. An Adam optimizer with a learning rate of 1^{-3} is used, there are 6 message-passing steps, and the network is run through 10,000 iterations.

The input is a series of networkx¹⁰ directed graphs, one for each sentence in the training and testing sets. In order to effectively utilize message passing, edges are constructed as *dependent* \rightarrow *head*, opposite the usual syntactic dependency-parsing edge direction. Each node has an attribute which is the vector produced by `word2vecf`’s syntactic embedding. In the GNN, edge weights are used to track dependency distance, both negative and positive. A negative edge weight indicates that a head precedes its dependent, and a positive weight that a dependent precedes its head. Target edge weights are calculated as the difference between the dependent and head location in the original surface order, normalized to $[-1, 1]$ by dividing each distance by the maximum dependency distance of a given sentence.

For example, Figure 2 (c) and (d) show the input and output for the phrase *for your trip to Canada*. The input to the GNN is the dependency tree, where each node’s attribute is the word’s syntactic embedding. The output is the same dependency tree with learned edge attributes reflecting the distance between dependent and head.

3.3 Weighted topological sort

Performing a topological sort of an edge-weighted poset such that connected nodes are placed in ascending order of edge weight is conceptually quite simple, but implementation is more complicated than it may appear. A straightforward approach of simply merging nodes with the smallest weights before those with larger weights does not properly order the nodes, since the weight of arcs crossing the merged nodes are not necessarily updated to reflect the merge. Instead, as outlined in Algorithm 1, each edge (u, v) from the poset can be added to a new directed graph *order* such that the edge’s weight is maintained, even though u and v may not be adjacent in *order*.

When inserting edge (u, v) with weight w_{uv} into *order*, if u is already in *order*, then traverse the successor nodes of u until the total distance from u —a value maintained by w_{sum} —exceeds w_{uv} . At that point, insert v and update the weights of v ’s neighbors. This process is shown in lines 5-16. Similarly, as shown in lines 17-28, if v is already in *order*, traverse the predecessor nodes of v until w_{sum} exceeds w_{uv} , insert u , and update u ’s neighbors’ weights. Finally, if neither u nor v are in *order*, add edge (u, v) with weight w_{uv} to *order*, as shown in line 30. When all edges from *poset* have been added to *order*, the topological sort of *order* is returned as the surface realization. Each edge in *poset* must be added to *order*, and in the worst-case scenario the weight of each existing edge in *order* must be examined. Therefore Algorithm 1 runs in $O(n \log n)$ time, where n is the number of edges in *poset*.

3.4 Baseline (AVG)

Rather than generating syntactic word embeddings and running the GNN, a naive approach to determining dependency distances is to average the distance between any two words in the training set for use on the testing set. Similar to the set of word embeddings (§3.1), in order to generalize to unseen words in the test set, average distances are created for each pair dependent pair of word | POS | relation, POS | relation, and POS. For example, if the |DET|det has an average dependency distance of 1.2 from horse |NOUN|nsubj, and brown |ADJ|amod has an average of 0.9 from horse |NOUN|subj, then using those two average distances as weights in a poset would result in a surface order of *the brown horse*. If red |ADJ|amod was unseen during training, then the average of all instances of ADJ | amod dependent on horse |NOUN|subj would be used—if that average distance were 1.3, then this naive approach would return *red the horse*.

⁹<https://github.com/deepmind/>

¹⁰<https://networkx.github.io/>

Algorithm 1: Given an edge-weighted *poset*, construct a total order such that nodes with smallest weights are adjacent.

```

1:  function WEIGHTED_TOPO_SORT(poset)
2:    order  $\leftarrow \emptyset$                                  $\triangleright$  empty directed graph to hold totally ordered set
3:    for  $(u, v, w_{uv}) \in \textit{poset}$  do
4:       $w_{sum} \leftarrow 0$                                  $\triangleright$  a sum of traversed weights
5:      if  $u \in \textit{order}$  then
6:        while  $w_{uv} > w_{sum}$  do                         $\triangleright$  traverse successors of  $u$ 
7:           $s \leftarrow \textit{order}.u.\textit{successor}$ 
8:           $w_{us} \leftarrow \textit{order}[u][s].\textit{weight}$ 
9:           $w_{sum} \leftarrow w_{sum} + w_{us}$ 
10:         if  $w_{uv} < w_{sum}$  then
11:            $u \leftarrow s$                                  $\triangleright u$  becomes its successor  $s$ 
12:         end if
13:       done
14:        $w_{vs} \leftarrow w_{sum} - w_{uv}$                      $\triangleright w_{vs}$  is how much  $w_{sum}$  overshot  $w_{uv}$ 
15:       order.UPDATE_EDGE( $u, s, \_$ )  $\leftarrow$                  $\triangleright$  change existing  $(u, s)$ ...
16:          $[(u, v, w_{us} - w_{vs}), (v, s, w_{vs})]$            $\triangleright$  ... to  $(u, v)$  and  $(v, s)$  and update weights
17:     else if  $v \in \textit{order}$  then
18:       while  $w_{uv} > w_{sum}$  do                         $\triangleright$  traverse predecessors of  $v$ 
19:          $p \leftarrow \textit{order}.v.\textit{predecessor}$ 
20:          $w_{pv} \leftarrow \textit{order}[p][v].\textit{weight}$ 
21:          $w_{sum} \leftarrow w_{sum} + w_{pv}$ 
22:         if  $w_{uv} < w_{sum}$  then
23:            $v \leftarrow p$                                  $\triangleright v$  becomes its predecessor  $p$ 
24:         end if
25:       done
26:        $w_{pu} \leftarrow w_{sum} - w_{uv}$                      $\triangleright w_{pu}$  is how much  $w_{sum}$  overshot  $w_{uv}$ 
27:       order.UPDATE_EDGE( $p, v, \_$ )  $\leftarrow$                  $\triangleright$  change existing  $(p, v)$ ...
28:          $[(p, u, w_{pu}), (u, v, w_{pv} - w_{pu})]$            $\triangleright$  ... to  $(p, u)$  and  $(u, v)$  and update weights
29:     else
30:       order.ADD_EDGE( $u, v, w_{uv}$ )
31:     end if
32:   done
33:   return TOPO_SORT(order)                             $\triangleright$  return topological sort of order graph
34: end function

```

3.5 Evaluation

To evaluate the performance of the GNN algorithm compared to the AVG baseline in an automated way across various languages, we must unfortunately use a single target reference to compare the generated sentences. Thus the reference for each sentence is the attested version in the source UD corpus; the generated sentences from both AVG and GNN will be measured for similarity to the attested version.

The algorithm is attempting to order a set of words as closely as possible to their original surface realization in the corpus. Because words may repeat in the sentence, each order is instead represented by a list of integers, and it is these lists of integers which are compared. For example, assuming a target reference order of $[1, 2, 3]$ for *the red horse*, the generated order of *red the horse* would be $[2, 1, 3]$. An obvious way to quantify how similar these integer lists are is with the widely used Spearman’s rank correlation coefficient (Spearman, 1904), also known as Spearman’s ρ (rho), which non-parametrically measures the similarity of two rankings. It ranges from -1, indicating that one order is the reverse of the other, to 1, for perfect correlation. The example of $[1, 2, 3]$ $[2, 1, 3]$ returns a ρ of 0.5, since in the second order 1 and 2 both precede 3, but 1 does not precede 2. This measure tells us which approach, AVG or GNN, generates orders closest to the attested UD order, as well as a loose gauge of overall effectiveness for both the general approach as well as each algorithm.

Further, to address the question of projectivity, the percentage of projective dependency arcs generated by the AVG baseline, the GNN algorithm, and the attested sentences is evaluated. In each case, projectivity is calculated as the number of instances in which a word appearing between a head h and dependent d is not dominated by h . This measure allows us to explore how dependency distance might result in known rates of projectivity in natural language.






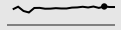


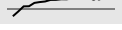

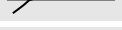

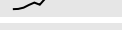





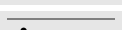







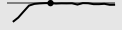


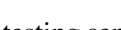
					SPEARMAN'S ρ [-1,1]			PROJECTIVITY [0,1]			
		N _{TR}	N _{TE}	D _V	AVG		GNN	AVG		GNN	UD
Afrikaans	AfriBooms	1315	425	18	0.707		0.773	0.530		0.650	0.939
Armenian	ArmTDP	560	470	18	0.628		0.672	0.413		0.585	0.987
Czech	CLTT	755	121	18	0.665		0.659	0.359		0.469	0.982
English	ParTUT	1781	153	20	0.634		0.775	0.496		0.680	0.995
French	ParTUT	803	110	18	0.677		0.729	0.531		0.669	0.998
Greek	GDT	1632	450	22	0.731		0.754	0.503		0.651	0.996
Hungarian	Szeged	910	449	20	0.635		0.609	0.440		0.598	0.969
Irish	IDT	566	452	18	0.674		0.753	0.461		0.603	0.978
Italian	ParTUT	1781	153	22	0.657		0.796	0.482		0.651	0.996
Latin	Perseus	1334	939	20	0.614		0.582	0.613		0.729	0.855
Maltese	MUDT	1119	516	18	0.729		0.750	0.498		0.682	0.995
Slovenian	SST	1669	890	18	0.549		0.567	0.663		0.798	0.967
Telugu	MTG	1051	146	14	0.916		0.931	0.925		0.971	0.997
Uyghur	UDT	1656	900	20	0.728		0.727	0.629		0.762	0.976

Table 1: Results. Each language is listed by its corpus; number of training and testing sentences; embedding dimension; Spearman’s ρ rank correlation coefficient for AVG and GNN; and rate of projectivity for AVG, GNN, and as attested in the UD corpus. Boldfaced numbers indicate cases in which GNN performed better than AVG. Sparklines show trends over 10K iterations with horizontal gray lines indicating AVG performance and black dots showing peak performance of GNN.

4 Results & Discussion

Table 1 shows the results of running both the AVG baseline and GNN algorithm on 14 v2.4 UD corpora representing a range of language families. These are relatively small corpora—between 500 and 2000 training sentences—and as a consequence their small vocabularies result in embedding vector dimensions between 14 and 22 due to the use of twice the fourth root of vocabulary size (§2.4, §3.1). While smaller than the more usual 50- or 300-element vectors, tying dimensionality to corpus vocabulary size seemed to avoid instability in the embedding space, though perhaps not in every case. Further, experiments with larger dimensions resulted in poor generalization to the testing set, possibly due to a lack of correlation between embeddings seen and unseen during training.

Results from Spearman’s ρ rank correlation show that both AVG and GNN were able to positively correlate surface order with the source UD corpora. Because Spearman’s ρ ranges from -1 to 1, positive values are better than chance; values above 0.5 seem rather promising. A large part of surface order can apparently be predicted based on dependency distance, averaged or learned. In all cases the GNN was able to approach AVG, exceeding it 10 out of 14 times. For many languages, the GNN achieved its peak value before training was complete, probably indicating overfitting. In the cases in which the GNN did not best AVG, the sparkline trends for Czech, Hungarian, and Latin suggest problems during training, perhaps due to overzealous learning rates or unstable embeddings, while Uyghur came very close.

In terms of projectivity, the GNN outperformed AVG in all cases, even when it did not best AVG in terms of Spearman’s ρ . While it is of course true that were the AVG or GNN method able to perfectly capture the word order of the UD corpora, the rate of projectivity and Spearman’s ρ would match exactly, but it is intriguing that short of perfection, Spearman’s ρ and projectivity are not necessarily correlated. Nor do many of the intralanguage trends match between the two measures—the highest GNN projectivity was generally achieved late in the training process, and the two sparklines of, for example, Armenian, are not very similar. While the GNN outperformed AVG in generating surface orders with higher rates of projectivity, even those rates lagged quite a bit behind the actual rates for almost all languages. This is likely due to even seemingly minor word transpositions leading to non-projective arcs (§4.1).

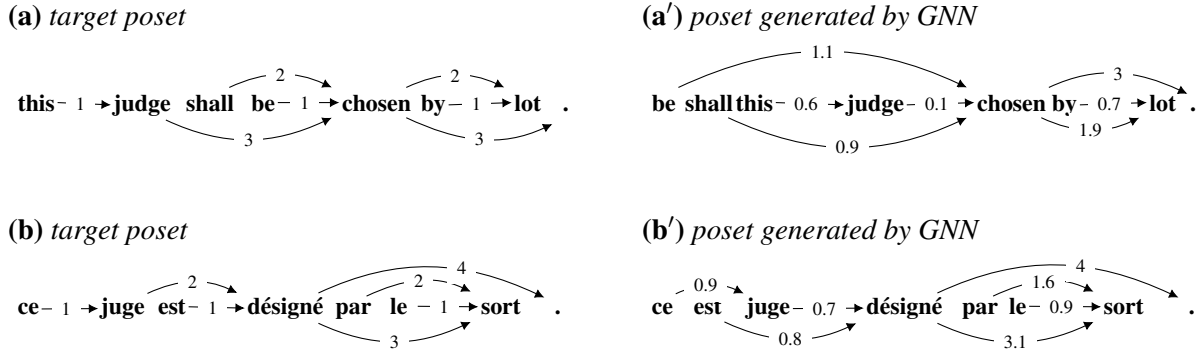


Figure 3: Target and generated posets from English- and French-ParTUT corpora.

Importantly, AVG is a naive approach, not a learning algorithm. As such there is very little room for improvement by adjusting how the averaged dependency distances are determined—employing morphological data or using lemmata instead of wordforms, for example. Conversely, changes to number of iterations, architecture, or hyperparameters of the GNN, especially tailored to each corpus, would almost certainly yield even better results, with a hypothetical upper bound limited only by the irreducible error present in a language’s word-order variation.

The results confirm that dependency distances can be learned from dependency trees by the GNN algorithm, usually better than a naive approach. Those distances can be used to generate surface realizations with word orders that positively correlate with attested UD sentences. Because these promising results can be generated from an essentially off-the-shelf GNN with relatively standardized parameters across a wide variety of languages, future endeavors improving the GNN architecture is certainly warranted.

4.1 Error analysis

Delving a bit into the sorts of errors in the surface orders generated by GNN, Figure 3 shows four versions of the same sentence: (a) the poset for a sentence from the UD English-ParTUT corpus; (a’) the poset generated by GNN with a Spearman’s ρ coefficient of 0.786—only slightly higher than the average ρ for that corpus, and therefore a typical generation; (b) the poset for the same sentence from French-ParTUT; and (b’) the poset generated by GNN with a non-projective¹¹ arc.

Figure 3 (a’) deviates from (a) in that the weight of $be \xrightarrow{1.1} chosen$ is larger than $shall \xrightarrow{0.9} chosen$, and both those edges have weights larger than the combination of $this \xrightarrow{0.6} judge$ and $judge \xrightarrow{0.1} chosen$. The result is a sentence in which the auxiliaries *be* and *shall* are transposed, and both appear in front of *this judge*. Similarly, (b’) deviates from (b) in that the weight of $est \xrightarrow{0.8} désigné$ is larger than the weight of $juge \xrightarrow{0.7} désigné$ —though unlike the English not larger than the combined weights of $ce \xrightarrow{0.9} juge$ and $juge \xrightarrow{0.7} désigné$. The result is a transposition of *juge* and *est*, causing a non-projective arc as *est* appears between *ce* and *juge* but is not dominated by either.

Aside from the transposition of the auxiliaries in (a’), both generated surface orders suffer from the weight of $judge/juge \rightarrow chosen/désigné$ being too small. While the offending edge in (a’) is quite small at 0.02, requiring an addition of over 2 to overcome the combined weights of the auxiliaries *be* and *shall*, an addition of just 0.11 to the weight of the edge would resolve (b’). In other words, if the weight of $est \xrightarrow{0.8} désigné$ were increased to 0.81, it would be larger than $est \xrightarrow{0.8} désigné$ and therefore *juge* would precede *est*, resolving (b’) to (b).

Neither training set for these corpora contains the word *judge/juge*, so the word’s embedding collapses to an average of all nouns acting as passive subjects, NOUN|nsubj:pass. This suggests that insufficient training size, lack of proper generalization from the available training data, and/or problematic embedding creation for unseen words is at fault here. These can all be addressed in future research.

¹¹The graphs in Figure 3 are posets, not dependency trees, and therefore the dependency concept of projectivity is not readily apparent. A poset analogue is planarity in the half-plane, or 1-planarity (cf. Pitler et al., 2013, p. 19). If all arcs in Figure 3 (b’) were drawn above the words, we would see that the $ce \rightarrow juge$ and $est \rightarrow désigné$ arcs would cross.

4.2 Dependency distance tolerance & projectivity

What is being learned by the GNN? That is, what do the edge weights, used to create a poset, actually represent? The question is perhaps conceptually a bit easier with AVG: the weights are the average distances between dependents and their heads in a corpus. AVG calculates how far a dependent tends to be from its head, or put another way, how many intervening words tend to be allowed between dependent and head in a collection of surface orders. It is a dependent word’s tolerance for how far it can be placed in front or behind its head in a surface realization. It seems that the GNN is learning this same information about dependency distance tolerance, but in a more subtle and context-sensitive way. Rather than simply an average distance, the GNN is learning how far a dependent can be placed from its head in concert with its syntactically related words¹² in a given dependency tree.

Dependency distance tolerance is effectively a maximum for how far apart a dependent and head can be in the surface realization of a given dependency tree. What factors determine this tolerance and how it might be encoded in a linguistic system is left for other research. However, dependency distance tolerance is a useful concept for exploring how projectivity might come about.

It was suggested in §2.3 that observed rates of projectivity might emerge from Dependency Distance Minimization (DDM). That is, the desire to minimize cumulative or mean dependency distances results in the high rates of projectivity seen across languages. A further goal within DDM is to avoid long-distance dependencies, though this avoidance may result in non-projective surface orders. The concept of dependency distance tolerance provides a more nuanced view of this second DDM motivation.

The topological sort of a poset whose edge weights correspond to contextual dependency tolerances, at least as implemented here, may place dependents closer to their heads than their tolerance, but not farther. As such, it defines an upper bound for each edge weight in a poset. A surface order can be seen as the result of assembling words such that dependents are placed no farther from their heads than their tolerance. In this way dependency distances in the surface order are not only minimized, but minimized in such a way that each word’s contextual dependency tolerance is taken into account.

Thus the topological sort of a weighted poset implements DDM’s goal of minimizing dependency distances generally, while the learned dependency tolerances provide a contextually sensitive definition of what ‘long distance’ means for each dependent pair in order to avoid generating surface orders with long-distance dependencies. Through this lens both the strong tendency towards projectivity across languages, as well as the occasional instances of non-projectivity, can be seen as an effect of avoiding dependency distances which exceed their contextual tolerances.

5 Summary

This paper describes a novel method for converting dependency trees to surface orders via syntactic word embeddings and edge-weighted posets. The embeddings are learned via `word2vecf`, and poset edge directions and weights are learned by a graph neural network (GNN), all trained on Universal Dependencies (UD) corpora. An algorithm is provided for topologically sorting a weighted poset. The output of the GNN is compared to a naive baseline in which average dependency distances are used as poset edge weight, both evaluated against attested word orders in UD corpora representing a variety of language families. The GNN outperforms the baseline on 10 of 14 corpora in terms of rank correlation and in all cases in terms of rate of projectivity.

The main contribution of the paper is the insight that a surface order can be represented by an edge-weighted poset, the weights of which can be learned by a graph neural network. Representing surface order as the result of topologically sorting this poset contributes to our understanding of how a tendency towards projectivity across natural languages might be explained.

Future research directions include improvement of the GNN architecture and hyperparameters; exploration of the interaction between word embedding dimension, performance, and generalizability; and the analysis of larger corpora.

¹²Due to the graph nature of the GNN, message passing, and the use of syntactic embeddings, a word’s context for determining dependency distance in this study is entirely dependency based, never linear.

References

- Martin Abadi et al. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. URL: <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf>.
- Valerio Basile and Alessandro Mazzei (2018). The DipInfo-UniTo system for SRST 2018. *Proceedings of the First Workshop on Multilingual Surface Realisation*. Melbourne, Australia: Association for Computational Linguistics, pp. 65–71.
- Peter W. Battaglia et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261 [cs, stat]*.
- Otto Behaghel (1932). *Deutsche Syntax eine geschichtliche Darstellung*. Heidelberg: Carl Winters Universitätsbuchhandlung.
- Claude Boisson (1981). Hiérarchie universelle des spécifications de temps, de lieu, et de manière. *Confluents* 7, pp. 69–124.
- Thiago Castro Ferreira, Sander Wubben, and Emiel Krahmer (2018). Surface Realization Shared Task 2018 (SR18): The Tilburg University Approach. *Proceedings of the First Workshop on Multilingual Surface Realisation*. Melbourne, Australia: Association for Computational Linguistics, pp. 35–8.
- Matthew S. Dryer (2009). On the order of demonstrative, numeral, adjective, and noun: an alternative to Cinque. *Conference on theoretical approaches to disharmonic word orders*.
- William Dyer (2018). Integration complexity and the order of cosisters. *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*. Brussels, Belgium: Association for Computational Linguistics, pp. 55–65.
- Henry Elder and Chris Hokamp (2018). Generating High-Quality Surface Realizations Using Data Augmentation and Factored Sequence Models. *Proceedings of the First Workshop on Multilingual Surface Realisation*. Melbourne, Australia: Association for Computational Linguistics, pp. 49–53.
- Ramon Ferrer-i-Cancho (2017). Towards a theory of word order. Comment on "Dependency distance: a new perspective on syntactic patterns in natural language" by Haitao Liu et al. *Physics of Life Reviews*.
- Ramon Ferrer-i-Cancho and Carlos Gómez-Rodríguez (2016). Crossings as a side effect of dependency lengths. *Complexity* 21 (S2), pp. 320–328.
- Katja Filippova and Michael Strube (2009). Tree Linearization in English: Improving Language Model Based Approaches. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*. Boulder, Colorado: Association for Computational Linguistics, pp. 225–8.
- John Rupert Firth (1957). A synopsis of linguistic theory 1930-1955. *Studies in Linguistic Analysis*. Oxford: Philological Society, pp. 1–32.
- Richard Futrell, Kyle Mahowald, and Edward Gibson (2015). Large-scale evidence of dependency length minimization in 37 languages. *Proceedings of the National Academy of Sciences* 112.33, pp. 10336–41.
- Kim Gerdes and Sylvain Kahane (2001). Word Order in German: A Formal Dependency Grammar Using a Topological Hierarchy. *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*. Toulouse, France: Association for Computational Linguistics, pp. 220–7.
- Edward Gibson (2000). The dependency locality theory: A distance-based theory of linguistic complexity. *Image, language, brain*, pp. 95–126.
- Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl (2017). Neural Message Passing for Quantum Chemistry. *arXiv:1704.01212 [cs]*.
- Carlos Gómez-Rodríguez (2016). Restricted Non-Projectivity: Coverage vs. Efficiency. *Computational Linguistics* 42.4, pp. 809–17.

- Joseph Greenberg (1963). Some universals of grammar with particular reference to the order of meaningful elements. *Universals of Grammar*. Ed. by Joseph Greenberg. Cambridge, Massachusetts: MIT Press, pp. 73–113.
- Jeanette Gundel (1988). Universals of topic-comment structure. *Studies in Syntactic Typology*. Ed. by Michael Hammond, Edith Moravcsik, and Jessica Wirth. Philadelphia: John Benjamins Publishing, pp. 209–39.
- Michael Hahn, Judith Degen, Noah Goodman, Dan Jurafsky, and Richard Futrell (2018). An Information-Theoretic Explanation of Adjective Ordering Preferences. *Proceedings of the 40th annual conference of the Cognitive Science Society*. London: Cognitive Science Society.
- Zellig S. Harris (1954). Distributional Structure. *WORD* 10.2, pp. 146–62.
- John A. Hawkins (1994). *A Performance Theory of Order and Constituency*. Cambridge: Cambridge University Press.
- John A. Hawkins (2004). *Efficiency and Complexity in Grammars*. Oxford: Oxford University Press.
- Richard Hudson (1995). Measuring syntactic difficulty. URL: <http://dickhudson.com/wp-content/uploads/2013/07/Difficulty.pdf>.
- T. Florian Jaeger and Roger Levy (2006). Speakers optimize information density through syntactic reduction. *Advances in neural information processing systems*, pp. 849–56.
- Sylvain Kahane and François Lareau (2016). Word Ordering as a Graph Rewriting Process. *Formal Grammar*. Ed. by Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald, and Sylvain Pogodalla. Springer Berlin Heidelberg, pp. 216–39.
- David King and Michael White (2018). The OSU Realizer for SRST '18: Neural Sequence-to-Sequence Inflection and Incremental Locality-Based Linearization. *Proceedings of the First Workshop on Multilingual Surface Realisation*. Melbourne, Australia: Association for Computational Linguistics, pp. 39–48.
- Omer Levy and Yoav Goldberg (2014). Dependency-Based Word Embeddings. *ACL (2)*. Citeseer, pp. 302–8.
- Haitao Liu, Chunshan Xu, and Junying Liang (2017). Dependency distance: A new perspective on syntactic patterns in natural languages. *Physics of Life Reviews* 21, pp. 171–93.
- Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin (2015). Transition-Based Syntactic Linearization. *HLT-NAACL*.
- Solomon Marcus (1965). Sur la notion de projectivité. *Mathematical Logic Quarterly* 11.2, pp. 181–92.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). Efficient estimation of word representations in vector space. *arXiv:1301.3781 [cs]*.
- Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner (2018). The First Multilingual Surface Realisation Shared Task (SR'18): Overview and Evaluation Results. *Multilingual Surface Realisation: Shared Task and Beyond: Proceedings of the Workshop*. Multilingual Surface Realisation: Shared Task and Beyond. Melbourne, Australia: Association for Computational Linguistics, pp. 1–12.
- Anat Ninio (2017). Projectivity is the mathematical code of syntax. *Physics of Life Reviews* 21, pp. 215–7.
- Joakim Nivre (2009). Non-projective dependency parsing in expected linear time. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, pp. 351–9.
- Joakim Nivre and Jens Nilsson (2005). Pseudo-projective dependency parsing. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 99–106.
- Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser (2017). Why We Need New Evaluation Metrics for NLG. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2231–40.

- Timothy Osborne, Michael Putnam, and Thomas Groß (2012). Catenae: Introducing a Novel Unit of Syntactic Analysis. *Syntax* 15.4, pp. 354–96.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–18.
- Y. Albert Park and Roger Levy (2009). Minimal-length linearizations for mildly context-sensitive dependency trees. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 335–43.
- Katerina Pastra and Horacio Saggion (2003). Colouring Summaries BLEU. *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: Are Evaluation Methods, Metrics and Resources Reusable? Evalinitatives '03*. event-place: Budapest, Hungary. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 35–42.
- Kevin Patel and Pushpak Bhattacharyya (2017). Towards Lower Bounds on Number of Dimensions for Word Embeddings. *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, pp. 31–6.
- Emily Pitler, Sampath Kannan, and Mitchell Marcus (2013). Finding Optimal 1-Endpoint-Crossing Trees. *Transactions of the Association for Computational Linguistics* 1, pp. 13–24.
- Alain Polguère and Igor Mel'čuk (2009). *Dependency in Linguistic Description*. John Benjamins Publishing.
- Yevgeniy Puzikov and Iryna Gurevych (2018). BinLin: A Simple Method of Dependency Tree Linearization. *Proceedings of the First Workshop on Multilingual Surface Realisation*. Melbourne, Australia: Association for Computational Linguistics, pp. 13–28.
- Ehud Reiter (2018). A Structured Review of the Validity of BLEU. *Computational Linguistics* 44.3, pp. 393–401.
- Jan Rijkhoff (1986). Word Order Universals Revisited: The Principle of Head Proximity. *Belgian Journal of Linguistics* 1, pp. 95–125.
- Gary-John Scott (2002). Stacked adjectival modification and the structure of nominal phrases. *Functional Structure in DP and IP: The Cartography of Syntactic Structures*. Vol. 1. New York: Oxford University Press, pp. 91–120.
- Marco Antonio Sobrevilla Cabezudo and Thiago Pardo (2018). NILC-SWORNEMO at the Surface Realization Shared Task: Exploring Syntax-Based Word Ordering using Neural Models. *Proceedings of the First Workshop on Multilingual Surface Realisation*. Melbourne, Australia: Association for Computational Linguistics, pp. 58–64.
- Jae Jung Song (2012). *Word Order*. New York: Cambridge University Press.
- Charles Spearman (1904). The Proof and Measurement of Association between Two Things. *The American Journal of Psychology* 15.1, pp. 72–101.
- Arthur Spirling and Pedro L Rodriguez (2019). What works, what doesn't, and how to tell the difference for applied research. URL: <https://www.nyu.edu/projects/spirling/documents/embed.pdf>.
- Lucien Tesnière (1959). *Éléments de syntaxe structural*. Paris: Klincksieck.
- Joseph P. Turian, Luke Shea, and I. D. Melamed (2006). *Evaluation of Machine Translation and its Evaluation*: Fort Belvoir, VA: Defense Technical Information Center.
- Laura Wendlandt, Jonathan K. Kummerfeld, and Rada Mihalcea (2018). Factors Influencing the Surprising Instability of Word Embeddings. *arXiv:1804.09692 [cs]*.
- Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu (2019). A Comprehensive Survey on Graph Neural Networks. *arXiv:1901.00596 [cs, stat]*.
- Yue Zhang and Stephen Clark (2015). Discriminative Syntax-Based Word Ordering for Text Generation. *Computational Linguistics* 41.3, pp. 503–38.