Long-Distance Dependencies don't have to be Long: Simplifying through Provably (Approximately) Optimal Permutations

Rishi Bommasani
Department of Computer Science
Cornell University
rb724@cornell.edu

Abstract

Neural models at the sentence level often operate on the constituent words/tokens in a way that encodes the inductive bias of processing the input in a similar fashion to how humans do. However, there is no guarantee that the standard ordering of words is computationally efficient or optimal. To help mitigate this, we consider a dependency parse as a proxy for the inter-word dependencies in a sentence and simplify the sentence with respect to combinatorial objectives imposed on the sentence-parse pair. The associated optimization results in permuted sentences that are provably (approximately) optimal with respect to minimizing dependency parse lengths and that are demonstrably simpler. We evaluate our general-purpose permutations within a fine-tuning schema for the downstream task of subjectivity analysis. Our fine-tuned baselines reflect a new state of the art for the **SUBJ** dataset and the permutations we introduce lead to further improvements with a **2.0%** increase in classification accuracy (absolute) and a **45%** reduction in classification error (relative) over the previous state of the art.

1 Introduction

Natural language processing systems that operate at the sentence level often need to model the interaction between different words in a sentence. This kind of modelling has been shown to be necessary not only in explicit settings where we consider the relationships between words (GuoDong et al., 2005; Fundel et al., 2006) but also in opinion mining (Joshi and Penstein-Rosé, 2009), question answering (Cui et al., 2005), and semantic role labelling (Hacioglu, 2004). A standard roadblock in this process has been trying to model long-distance dependencies between words. Neural models for sentence-level tasks, for example, have leveraged recurrent neural networks (Sutskever et al., 2014) and attention mechanisms (Bahdanau et al., 2015; Luong et al., 2015) as improvements in addressing this challenge. LSTMs (Hochreiter and Schmidhuber, 1997) in particular have been touted as being well-suited for capturing these kinds of dependencies but recent work suggests that the modelling may be insufficient to various extents (Linzen et al., 2016; Liu et al., 2018; Dangovski et al., 2019). Fundamentally, these neural components do not restructure the challenge of learning long-distance dependencies but instead introduce computational expressiveness as a means to represent and retain inter-word relationships efficiently (Kuncoro et al., 2018).

Models that operate at the sentence level in natural language processing generally process sentences word-by-word in a left-to-right fashion. Some models, especially recurrent models, consider the right-to-left traversal (Sutskever et al., 2014) or a bidirectional traversal that combines both the left-to-right and right-to-left traversals (Schuster and Paliwal, 1997). Other models weaken the requirement of sequential processing by incorporating position embeddings to retain the sequential nature of the data and then use self-attentive mechanisms that don't explicitly model the sequential nature of the input (Vaswani et al., 2017). All such approaches encode the prior that computational processing of sentences should appeal to a cognitively plausible ordering of words.

Nevertheless in machine translation, reorderings of both the input and output sequences have been considered for the purpose of improving *alignment* between the source and target languages. Specifically, *preorders*, or permuting the input sequence, and *postorders*, or permuting the output sequence, have been well-studied in statistical machine translation (Xia and McCord, 2004; Goto et al., 2012) and have been recently integrated towards fully neural machine translation

(De Gispert et al., 2015; Kawara et al., 2018). In general, these re-ordering methods assume some degree of supervision (Neubig et al., 2012) and have tried to implicitly maintain the original structure of the considered sequence despite modifying it to improve alignment. Similar approaches have also been considered for cross-lingual transfer in dependency parsing (Wang and Eisner, 2018) based on the same underlying idea of improving alignment.

In this work, we propose a general approach for permuting the words in an input sentence based on the notion of *simplification*, i.e. reducing the length of inter-word dependencies in the input. In particular, we appeal to graph-based combinatorial optimization as an unsupervised approach for producing permutations that are provably optimal, or provably approximately optimal, in minimizing inter-word dependency parse lengths.

Ultimately, we hypothesize that our simplification-based permutations over input sentences can be incorporated as a lightweight, drop-in preprocessing step for neural models to improve performance for a number of standard sentence-level NLP problems. As an initial case study, we consider the task of sentence-level subjectivity classification and using the **SUBJ** dataset (Pang and Lee, 2004), we first introduce baselines that achieve a state of the art **95.8%** accuracy and further improve on these baselines with our permutations to a new state of the art of **97.5%** accuracy.

2 Limitations

This work considers simplifying inter-word dependencies for neural models. However, we measure inter-word dependencies using dependency parses and therefore rely on an incomplete description of inter-word dependencies in general. Further, we assume the existence of a strong dependency parser, which is reasonably well-founded for English which is the language studied in this work. This assumption is required for providing theoretical guarantees regarding the optimality of sentence permutations with respect to the gold-standard dependency parse.[1] In spite of these assumptions, it is still possible for the subsequent neural models to recover from errors in the initial sentence permutations.

Beyond this, we consider dependency parses which graph theoretically are edge-labelled directed trees. However, in constructing optimal sentence permutations, we simplify the graph structure by neglecting edge labels and edge directions. Both of these are crucial aspects of a dependency parse tree and in §6 we discuss possible future directions to help address these challenges.

Most concerningly, this approach alters the order of words in a sentence for the purpose of simplifying one class of dependencies — binary inter-word dependencies marked by dependency parses. However, in doing so, it is likely that other crucial aspects of the syntax and semantics of a sentence that are a function of word order are obscured. We believe the mechanism proposed in §3.3 helps to alleviate this by making use of powerful initial word representations that are made available through recent advances in pretrained contextual representations and transfer learning (Peters et al., 2018; Devlin et al., 2018; Liu et al., 2019).

3 Model

Our goal is to take a dependency parse of a sentence and use it is as scaffold for permuting the words in a sentence. We begin by describing two combinatorial measures on graphs that we can use to rank permutations of the words in a sentence by, and therefore optimize with respect to, in order to find the optimal permutation for each measure. Given the permutation, we then train a model end-to-end on a downstream task and exploit pretrained contextual word embeddings to initialize the word representations for our model.

3.1 Input Structure

Given a sentence as an input for some downstream task, we begin by computing a dependency parse for the sentence using an off-the-shelf dependency parser. This endows the sentence with a graph structure corresponding to an edge-labelled directed tree $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ where the vertices correspond to tokens in the sentence ($\mathcal{V}^* = \{w_1, w_2, \dots, w_n\}$) and the edges correspond to dependency arcs. We then consider the undirected tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \mathcal{V}^*$ and $\mathcal{E} = \mathcal{E}^*$ without the edge labels and edge directions.

[1]The generated permutations are always (approximately) optimal with respect to the system-generated dependency parse.

3.2 Combinatorial Objectives

We begin by defining a *linear layout* on $\mathcal{G}$ to be a bijective, i.e. one-to-one, ordering on the vertices $\pi : \mathcal{V} \to \{1, 2, \ldots, n\}$. For a graph associated with a sentence, we consider the *identity linear layout* π_I to be given by $\pi_I(w_i) = i$: the linear layout of vertices is based on the word order in the input sentence. For any given linear layout π we can further associate each edge $(u, v) \in \mathcal{E}$ with an *edge distance* $d_{u,v} = |\pi(u) - \pi(v)|$.[2]

By considering the modified dependency parse $\mathcal{G}$ alongside the sentence, we recognize that a computational model of the sentence may need to model any given dependency arc $(w_i, w_j) \in \mathcal{E}$. As a result, for a model that processes sentences word-by-word, information regarding this arc must be stored for a number of time-steps given by $d_{w_i,w_j} = |\pi_I(w_i) - \pi_I(w_j)| = |j - i|$. This implies that a model may need to store a dependency for a large number of time-steps (a long-distance dependency) and we instead consider finding an optimal linear layout π^* (that is likely not to be the identity) to minimize these edge distances with respect to two well-studied objectives on linear layouts.

Bandwidth Problem The *bandwidth* problem on graphs corresponds to finding an optimal linear layout π^* under the objective:

$$\operatorname*{argmin}_{\pi \in \Pi} \max_{(u,v) \in \mathcal{E}} d_{u,v} \tag{1}$$

The bandwidth problem is a well known problem dealing with linear layouts with applications in sparse matrix computation (Gibbs et al., 1976) and information retrieval (Botafogo, 1993) and has been posed in equivalent ways for graphs and matrices (Chinn et al., 1982). For dependency parses, it corresponds to finding a linear layout that minimizes the length of the longest dependency. Papadimitriou (1976) proved the problem was NP-hard and the problem was further shown to remain NP-hard for trees and even restricted classes of trees (Unger, 1998; Garey et al., 1978). In this work, we use the better linear layout of those produced by the guaranteed $\mathcal{O}(\log n)$ approximation of Haralambides and Makedon (1997) and the heuristic of Cuthill and McKee (1969) and refer to the resulting linear layout as π_b^*.

Minimum Linear Arrangement Problem Similar to the bandwidth problem, the *minimum linear arrangement* (minLA) problem considers finding a linear layout given by:

$$\operatorname*{argmin}_{\pi \in \Pi} \sum_{(u,v) \in \mathcal{E}} d_{u,v} \tag{2}$$

While less studied than the bandwidth problem, the minimum linear arrangement problem considers minimizing the sum of the edge lengths of the dependency arcs which may more closely resemble how models need to not only handle the longest dependency well, as in the bandwidth problem, but also need to handle the other dependencies. Although the problem is NP-hard for general graphs (Garey et al., 1974), it admits polynomial time exact solutions for trees (Shiloach, 1979). We use the algorithm of Chung (1984), which runs in $\mathcal{O}(n^{1.585})$, to find the optimal layout π_m^*.

3.3 Downstream Integration

Given a linear layout π, we can define the associated permuted sentence s' of the original sentence $s = w_1\ w_2\ \ldots\ w_n$ where the position of w_i in s' is given by $\pi(w_i)$. We can then train models end-to-end taking the permuted sentences as direct replacements for the original input sentences. However, this approach suffers from the facts that (a) the resulting sentences may have lost syntactic/semantic qualities of the original sentences due to the permutations and (b) existing pretrained embedding methods would need to be re-trained with these new word orders, which is computationally expensive, and pretraining objectives like language modelling may be less sensible given the problems noted in (a). To reconcile this, we leverage a recent three-step pattern for many NLP tasks (Peters et al., 2019):

1. **Pretrained Word Representations**: Represent each word in the sentence using a pretrained contextualizer (Peters et al., 2018; Devlin et al., 2018).

2. **Fine-tuned Sentence Representation**: Learn a task-specific encoding of the sentence using a task-specific encoder as a fine-tuning step on top of the pretrained word representations.

3. **Task Predictions**: Generate a prediction for the task using the fine-tuned representation.

[2]Refer to Díaz et al. (2002) for a survey of linear layouts, related problems, and their applications.

As a result, we can introduce the permutation between steps 1 and 2. What this means is the initial pretrained representations model the sentence using the standard ordering of words and therefore have access to the unchanged syntactic/semantic properties. These properties are diffused into the word-level representations and therefore the fine-tuning encoder may retrieve them even if they are not observable after the permutation. This allows the focus of the task-specific encoder to shift towards modelling useful dependencies specific to the task.

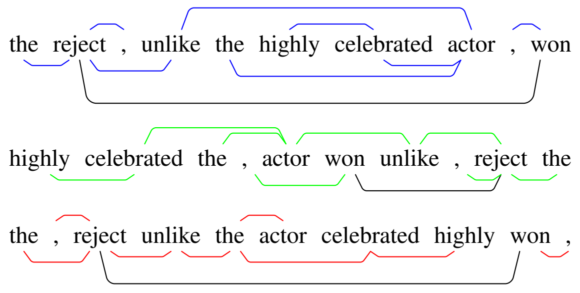
4 Experiments

Using our approach, we begin by studying how optimizing for these combinatorial objectives affects the complexity of the input sentence as measured using these objective functions. We then evaluate performance on the downstream task of subjectivity analysis and find our baseline model achieves a new state of the art for the dataset which is improved further by the permutations we introduce.

For all experiments, we use the spaCy dependency parser (Honnibal and Montani, 2017) to find the dependency parse. In studying properties of the bandwidth optimal permutation π_b^* and the minLA optimal permutation π_m^*, we compare to baselines where the sentence is not permuted/the identity permutation π_I as well as where the words in the sentence are ordered using a random permutation π_R. A complete description of experimental and implementation details is provided in Appendix A.

Our permutations do not introduce or change the size or runtime of existing models while providing models with dependency parse information implicitly. The entire preprocessing process, including the computation of permutations for both objectives, takes **21 minutes** in aggregate for the 10000 examples in the **SUBJ** dataset. A complete description of changes in model size, runtime, and convergence speed is provided in Appendix B.

Data and Evaluation To evaluate the direct effects of our permutations on input sentence simplification, we use 100000 sentences from Wikipedia; to evaluate the downstream impacts we consider the **SUBJ** dataset (Pang and Lee, 2004) for subjectivity analysis. The subjectivity analysis task requires deciding whether a given sentence is subjective or objective and the dataset is



the reject , unlike the highly celebrated actor , won
highly celebrated the , actor won unlike , reject the
the , reject unlike the actor celebrated highly won ,

Figure 1: Example of the sentence permutation along with overlayed dependency parses. Blue indicates the standard ordering, green indicates the bandwidth optimal ordering, and red indicates the minLA optimal ordering. Black indicates the longest dependency arc in the original ordering.

balanced with 5000 subjective and 5000 objective examples. We consider this task as a starting point as it is well-studied and dependency features have been shown to be useful for similar opinion mining problems (Wu et al., 2009).

Examples In Figure 1, we present an example sentence and its permutations under π_I, π_b^* and π_m^*. Under the standard ordering, the sentence has bandwidth 8 and minLA score 22 and this is reduced by both the bandwidth optimal permutation to 3 and 17 respectively and similarly the minLA permutation also improves on both objectives with scores of 6 and 16 respectively. A model processing the sequence word-by-word may have struggled to retain the long dependency arc linking 'reject' and 'won' and therefore incorrectly deemed that 'actor' was the subject of the verb 'won' as it is the only other viable candidate and is closer to the verb. If this had occured, it would lead an incorrect interpretation (here the opposite meaning). While both of the introduced permutations still have 'actor' closer to the verb, the distance between 'reject' and 'won' shrinks (denoted by the black arcs) and similarly the distance between 'unlike' and 'actor' shrinks. These combined effects map help to mitigate this issue and allow for improved modelling. Across the Wikipedia data, we see a similar pattern for the minLA optimal permutations in that they yield improvements on both objectives but we find the bandwidth optimal permutations on average increase the minLA score as is shown in Table 1. We believe this is natural given the relationship of the objectives; the longest arc is accounted for in the minLA objective whereas the other arcs don't contribute to the

	Bandwidth	minLA
π_I (Standard)	17.64	82.39
π_R (Random)	20.94	294.43
π_b^* (Bandwidth)	6.75	101.23
π_m^* (minLA)	9.43	54.57

Table 1: Bandwidth and minimum linear arrangement scores for the specified permutation type averaged across 100000 Wikipedia sentences.

	Accuracy
π_I (Standard)	95.8
π_R (Random)	94.8
π_b^* (Bandwidth)	96.2
π_m^* (minLA)	**97.5**
AdaSent (Zhao et al., 2015)[†]	95.5
CNN+MCFA (Amplayo et al., 2018)[†]	94.8

Table 2: Accuracy on the **SUBJ** dataset using the specified ordering of pretrained representations for the fine-tuning LSTM. [†] indicates prior models that were evaluated using 10-fold cross validation instead of a held-out test set.

bandwidth cost. We also find the comparison of the standard and random orderings to be evidence that human orderings of words to form sentences (at least in English) are correlated with these objectives, as they are significantly better with respect to these objectives as compared to random orderings. Refer to Figure 3 for a larger example.

Downstream Performance In Table 2, we present the results on the downstream task. Despite the fact that the random permutation LSTM encoder cannot learn from the word order and implicitly is restrained to permutation-invariant features, the associated model performs comparably with previous state of the art systems, indicating the potency of current pretrained embeddings and specifically ELMo. When there is a deterministic ordering, we find that the standard ordering is the least helpful of the three orderings considered. We see a particularly significant spike in performance when using permutations that are minLA optimal and we conjecture that this may be because minLA permutations improve on both objectives on average and empirically we observe they better maintain the order of the original sentence (as can be seen in Figure 1).

5 Related Work

This work draws upon inspiration from the literature on psycholinguistics and cognitive science. Specifically, dependency lengths and the existing minimization in natural language has been studied under the dependency length minimization (DLM) hypothesis (Liu, 2008) which posits a bias in human languages towards constructions with shorter dependency lengths.[3]

In particular, the relationship described between random and natural language orderings of words to form sentences as in Table 1 has been studied more broadly across 37 natural languages in Futrell et al. (2015). This work, alongside Gildea and Temperley (2010); Liu et al. (2017); Futrell et al. (2017) helps to validate the extent and pervasiveness of DLM in natural languages. More generally, this literature body has proposed a number of reasons for this behavior, many of which center around the related notions of efficiency (Gibson et al., 2019) and memory constraints (Gulordava et al., 2015) for humans. Recent research at the intersection of psycholinguistics and NLP that has tried to probe for dependency-oriented understanding in neural networks (primarily RNNs) does indicate relationships with specific dependency-types and RNN understanding. This includes research considering specific dependency types (Wilcox et al., 2018, 2019a), word-order effects (Futrell and Levy, 2019), and structural supervision (Wilcox et al., 2019b).

Prompted by this, the permutations considered in this work can alternatively be seen as linearizations (Langkilde and Knight, 1998; Filippova and Strube, 2009; Futrell and Gibson, 2015; Puzikov and Gurevych, 2018) of a dependency parse in a minimal fashion which is closely related to Gildea and Temperley (2007); Temperley and Gildea (2018). While such linearizations have not been well-studied for downstream impacts, the usage of dependency lengths as a constraint has been studied for dependency parsing itself. Towards this end, Eisner and Smith (2010) showed that using dependency length can be a powerful heuristic tool in dependency parsing (by either enforcing a strict preference or favoring a soft preference for shorter dependencies).

[3]In this work, we partially deviate from this linguistic terminology, which primarily deals with the measure defined in Equation 2, and prefer algorithmic terminology to accommodate the measure defined in Equation 1 and disambiguate these related measures more clearly.

6 Future Directions

Graph Structure Motivated by recent work on graph convolutional networks that began with undirected unlabelled graphs (Kipf and Welling, 2016; Zhang et al., 2018) that was extended to include edge direction and edge labels (Marcheggiani and Titov, 2017), we consider whether these features of a dependency parse can also leveraged in computing an optimal permutation. We argue that bidirectionally traversing the permuted sequence may be sufficient to address edge direction. A natural approach to encode edge labels would be to define a mapping (either learned on an auxiliary objective or tuned as a hyperparameter) from categorical edge labels to numericals edge weights and then consider the weighted analogues of the objectives in Equation 1 and Equation 2.

Improved Objective The objectives introduced in Equation 1 and Equation 2 can be unified by considering the family of cost functions:

$$f_p(\pi) = \sum_{(u,v)\in\mathcal{E}} |\pi(u) - \pi(v)|^p \qquad (3)$$

Here, minLA correspond to $p = 1$ and the bandwidth problem corresponds to $p = \infty$. We can then propose a generalized objective that is the convex combination of the individual objectives, i.e. finding a permutation that minimizes:

$$f^{\alpha}_{1,\infty}(\pi) = \alpha f_1(\pi) + (1 - \alpha) f_\infty(\pi) \qquad (4)$$

Setting α to 0 or 1 reduces to the original objectives. This form of the new objective is reminiscent of Elastic Net regularization in statistical optimization (Zou and Hastie, 2005). Inspired by this parallel, a Lagrangian relaxation of one of the objectives as a constraint may be an approach towards (approximate) optimization.

Task-specific Permutations The permutations produced by these models are invariant with respect to the downstream task. However, different tasks may benefit from different sentence orders that go beyond task-agnostic simplification. A natural way to model this in neural models is to learn the permutation in a differentiable fashion and train the permutation model end-to-end within the overarching model for the task. Refer to Appendix C for further discussion.

7 Conclusion

In this work, we propose approaches that permute the words in a sentence to provably minimize combinatorial objectives related to the length of dependency arcs. We find that this is a lightweight procedure that helps to simplify input sentences for downstream models and that it leads to improved performance and state of the art results (**97.5%** classification accuracy) for subjectivity analysis using the **SUBJ** dataset.

Acknowledgements

I thank Bobby Kleinberg for his tremendous insight into the present and future algorithmic challenges of layout-based optimization problems, Tianze Shi for his pointers towards the dependency length minimization literature in psycholinguistics and cognitive science, and Arzoo Katiyar for her advice on comparing against random sequence orderings and considering future work towards soft/differentiable analogues. I also thank the anonymous reviewers for their perceptive and constructive feedback. Finally, I thank Claire Cardie for her articulate concerns and actionable suggestions with regards to this work and, more broadly, for her overarching guidance and warm mentoring as an adviser.

References

Reinald Kim Amplayo, Kyungjae Lee, Jinyoung Yeo, and Seung won Hwang. 2018. Translations as additional contexts for sentence classification. In *IJCAI*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Rodrigo A Botafogo. 1993. Cluster analysis for hypertext systems. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 116–125. ACM.

P. Z. Chinn, J. Chvtalov, A. K. Dewdney, and N. E. Gibbs. 1982. The bandwidth problem for graphs and matricesa survey. *Journal of Graph Theory*, 6(3):223–254.

FRK Chung. 1984. On optimal linear arrangements of trees. *Computers & mathematics with applications*, 10(1):43–60.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 400–407. ACM.

E. Cuthill and J. McKee. 1969. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, pages 157–172, New York, NY, USA. ACM.

Rumen Dangovski, Li Jing, Preslav Nakov, Mico Tatalovic, and Marin Soljacic. 2019. Rotational unit of memory: A novel representation unit for rnns with scalable applications. *Transactions of the Association for Computational Linguistics*, 7:121–138.

Adrià De Gispert, Gonzalo Iglesias, and Bill Byrne. 2015. Fast and accurate preordering for smt using neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1012–1017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Josep Díaz, Jordi Petit, and Maria Serna. 2002. A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, 34(3):313–356.

Jason Eisner and Noah A Smith. 2010. Favor short dependencies: Parsing with soft and hard constraints on dependency length. In *Trends in Parsing Technology*, pages 121–150. Springer.

Katja Filippova and Michael Strube. 2009. Tree linearization in english: Improving language model based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 225–228, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2006. Relexrelation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Richard Futrell and Edward Gibson. 2015. Experiments with generative models for dependency tree linearization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1978–1983.

Richard Futrell, Roger Levy, and Edward Gibson. 2017. Generalizing dependency distance: Comment on dependency distance: A new perspective on syntactic patterns in natural languages by haitao liu et al. *Physics of life reviews*, 21:197–199.

Richard Futrell and Roger P. Levy. 2019. Do RNNs learn human-like abstract word order preferences? In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 50–59.

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):10336–10341.

Michael R Garey, Ronald L Graham, David S Johnson, and Donald Ervin Knuth. 1978. Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics*, 34(3):477–495.

Michael R Garey, David S Johnson, and Larry Stockmeyer. 1974. Some simplified np-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63. ACM.

Norman E. Gibbs, William G. Poole, and Paul K. Stockmeyer. 1976. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, 13(2):236–250.

Edward Gibson, Richard Futrell, Steven T Piandadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. 2019. How efficiency shapes human language. *Trends in cognitive sciences*.

Daniel Gildea and David Temperley. 2007. Optimizing grammars for minimum dependency length. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 184–191.

Daniel Gildea and David Temperley. 2010. Do grammars minimize dependency length? *Cognitive Science*, 34(2):286–310.

Isao Goto, Masao Utiyama, and Eiichiro Sumita. 2012. Post-ordering by parsing for japanese-english statistical machine translation. In *50th Annual Meeting of the Association for Computational Linguistics*, page 311.

Kristina Gulordava, Paola Merlo, and Benoit Crabbé. 2015. Dependency length minimisation effects in short spans: a large-scale analysis of adjective placement in complex noun phrases. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 477–482.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 427–434, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kadri Hacioglu. 2004. Semantic role labeling using dependency trees. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1273. Association for Computational Linguistics.

J. Haralambides and F. Makedon. 1997. Approximation algorithms for the bandwidth minimization problem for a large class of trees. *Theory of Computing Systems*, 30(1):67–90.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Mahesh Joshi and Carolyn Penstein-Rosé. 2009. Generalizing dependency features for opinion mining. In *Proceedings of the ACL-IJCNLP 2009 conference short papers*, pages 313–316.

Yuki Kawara, Chenhui Chu, and Yuki Arase. 2018. Recursive neural network based preordering for english-to-japanese machine translation. In *Proceedings of ACL 2018, Student Research Workshop*, pages 21–27, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 704–710. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Haitao Liu. 2008. Dependency distance as a metric of language comprehension difficulty. *Journal of Cognitive Science*, 9(2):159–191.

Haitao Liu, Chunshan Xu, and Junying Liang. 2017. Dependency distance: a new perspective on syntactic patterns in natural languages. *Physics of life reviews*, 21:171–193.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. *CoRR*, abs/1903.08855.

Nelson F Liu, Omer Levy, Roy Schwartz, Chenhao Tan, and Noah A Smith. 2018. Lstms exploit linguistic attributes of data. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 180–186.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 843–853, Jeju Island, Korea. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Christos H. Papadimitriou. 1976. The np-completeness of the bandwidth minimization problem. *Computing*, 16:263–270.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Matthew Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks.

Yevgeniy Puzikov and Iryna Gurevych. 2018. BinLin: A simple method of dependency tree linearization. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 13–28, Melbourne, Australia. Association for Computational Linguistics.

Rodrigo Santa Cruz, Basura Fernando, Anoop Cherian, and Stephen Gould. 2017. Deeppermnet: Visual permutation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3949–3957.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Yossi Shiloach. 1979. A minimum linear arrangement algorithm for undirected trees. *SIAM Journal on Computing*, 8(1):15–32.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

David Temperley and Daniel Gildea. 2018. Minimizing syntactic dependency lengths: Typological/cognitive universal? *Annual Reviews of Linguistics*.

Walter Unger. 1998. The complexity of the approximation of the bandwidth problem. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 82–91. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Dingquan Wang and Jason Eisner. 2018. Synthetic data made to order: The case of parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1325–1337, Brussels, Belgium. Association for Computational Linguistics.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do rnn language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221.

Ethan Wilcox, Roger P. Levy, and Richard Futrell. 2019a. What syntactic structures block dependencies in rnn language models? In *Proceedings of the 41st Annual Meeting of the Cognitive Science Society*.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019b. Structural supervision improves learning of non-local grammatical dependencies. In *Proceedings of the 18th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1533–1541. Association for Computational Linguistics.

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of the 20th international conference on Computational Linguistics*, page 508. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.

A Implementation Details

We implement our models in PyTorch (Paszke et al., 2017) using the Adam optimizer (Kingma and Ba, 2014) with its default parameters in PyTorch. We split the dataset using a 80/10/10 split and the results in Table 2 are on the test set whereas those in Figure 2 are on the development set. We use ELMo embeddings (Peters et al., 2018)[4], for the initial pretrained word representations by concatenating the two 1024 dimensional pretrained vectors, yielding a 2048 dimensional initial pretrained representation for each token. These representations are frozen based on the results of Peters et al. (2019) and passed through a single-layer bidirectional LSTM with output dimensionality 256. The outputs of the forward and backward LSTMs at position i are concatenated and a sentence representation is produced by max-pooling as was found to be effective in Howard and Ruder (2018) and Peters et al. (2019). The sentence representation is passed through a linear classifier $M \in \mathbb{R}^{512 \times 2}$ and the entire model is trained to minimize cross entropy loss. All models are trained for 13 epochs with a batch size of

[4]Specifically, we use embeddings available at: https://s3-us-west-2.amazonaws.com/allennlp/models/elmo/2x4096_512_2048cnn_2xhighway/elmo_2x4096_512_2048cnn_2xhighway_options.json

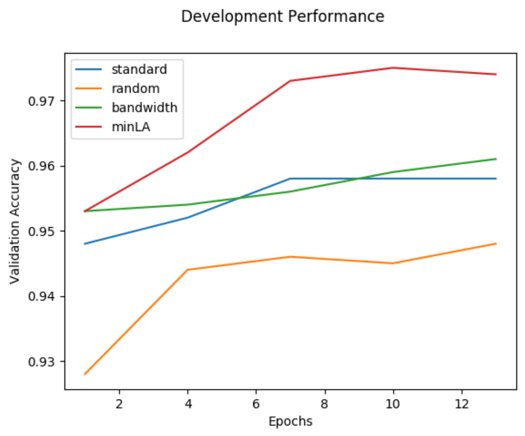


Figure 2: Development set performance for each ordering. Values are reported beginning at epoch 1 in intervals of 3 epochs.

16 with the test set results reported being from the model checkpoint after epoch 13. We also experimented with changing the LSTM task-specific encoder to be unidirectional but found the results were strictly worse.

B Efficiency Analysis

Model Size The changes we introduce only impact the initial preprocessing and ordering of the pretrained representations for the model. As a result, we make no changes to the number of model parameters and the only contribution to the model footprint is we need to store the permutation on a per example basis. This can actually be avoided in the case where we have frozen pretrained embeddings as the permutation can be computed in advance. Therefore, for the results in this paper, the model size is entirely unchanged.

Runtime The wall-clock training time, i.e. the wall-clock time for a fixed number of epochs, and inference time are unchanged as we do not change the underlying model in any way and the permutations can be precomputed. As noted in the paper, on a single CPU it takes **21** minutes to complete the entire preprocessing process and 25% of this time is a result of computing bandwidth optimal permutations and 70% of this time is a result of computing minLA optimal permutations. The preprocessing time scales linearly in the number of examples and we verify this as it takes **10** minutes to process only the subjective examples (and the dataset is balanced). Figure 2 shows the development set performance for each of the permutation types over the course of the fine-tuning process.

C End-to-End Permutations

In order to approach differentiable optimization for permutations, we must specify a representation. A standard choice that is well-suited for linear algebraic manipulation is a permutation matrix, i.e $P_\pi \in \mathbb{R}^{n \times n}$, where $P_\pi[i, j] = 1$ if $\pi(i) = j$ and 0 otherwise. As a result, permutation matrices are discrete, and therefore sparse, in the space of real matrices. As such they are poorly suited for the gradient-based optimization that supports most neural models. A recent approach from vision has considered a generalization of permutation matrices to the associated class of *doubly stochastic matrices* and then considered optimization with respect to the manifold they define (the *Sinkhorn Manifold*) to find a discrete permutation (Santa Cruz et al., 2017). This approach cannot be immediately applied for neural models for sentences since the algorithms exploits that images, and therefore permutations of the pixels in an image, are of fixed size between examples. That being said we ultimately see this as being an important direction of study given the shift from discrete optimization to soft/differentiable alternatives for similar problems in areas such as structured prediction.

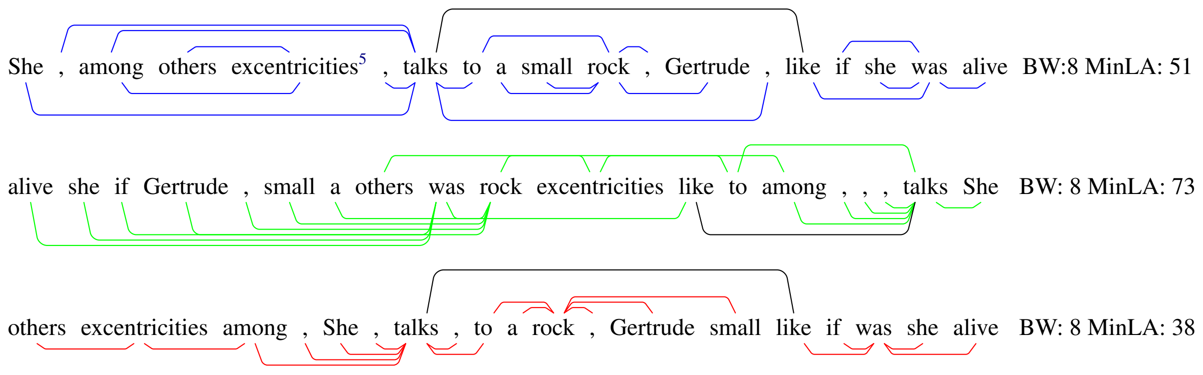

She , among others excentricities[5] , talks to a small rock , Gertrude , like if she was alive BW:8 MinLA: 51
alive she if Gertrude , small a others was rock excentricities like to among , , , talks She BW: 8 MinLA: 73
others excentricities among , She , talks , to a rock , Gertrude small like if was she alive BW: 8 MinLA: 38

Figure 3: Addition example sentence with sentence permutations and overlayed dependency parses. Blue indicates the standard ordering, green indicates the bandwidth optimal ordering, and red indicates the minLA optimal ordering. Black indicates the longest dependency arc in the original ordering.