Unsupervised Labeled Parsing with Deep Inside-Outside Recursive Autoencoders

Andrew Drozdov, Pat Verga, Yi-Pei Chen, Mohit Iyyer, and Andrew McCallum

College of Information and Computer Sciences
University of Massachusetts Amherst

{adrozdov, pat, yipeichen, miyyer, mccallum}@cs.umass.edu

Abstract

Understanding text often requires identifying meaningful constituent spans such as noun phrases and verb phrases. In this work, we show that we can effectively recover these types of labels using the learned phrase vectors from deep inside-outside recursive autoencoders (DIORA). Specifically, we cluster span representations to induce span labels. Additionally, we improve the model's labeling accuracy by integrating latent code learning into the training procedure. We evaluate this approach empirically through unsupervised labeled constituency parsing. Our method outperforms ELMo and BERT on two versions of the Wall Street Journal (WSJ) dataset and is competitive to prior work that requires additional human annotations, improving over a previous state-of-the-art system that depends on ground-truth part-of-speech tags by 5 absolute F1 points (19% relative error reduction).

1 Introduction

The deep inside-outside recursive autoencoder (Drozdov et al., 2019, DIORA) is part of a recent trend in fully unsupervised neural constituency parsers (Shen et al., 2018; Williams et al., 2018a; Htut et al., 2018; Shen et al., 2019; Kim et al., 2019). However, these works and nearly all previous research (Klein and Manning, 2002; Seginer, 2007; Ponvert et al., 2011; Spitkovsky et al., 2013) have focused on *unlabeled* constituency parsing.

In this paper, we instead focus on *labeled* constituency parsing for English. The small number of previous works that exist in this area suffer from substantial weaknesses: 1) the models depend on ground-truth part-of-speech tags, which are not always available and known to boost constituency parsing scores (Kitaev and Klein, 2018), 2) none can simultaneously identify and label constituents (instead they typically depend on an external latent parser), and 3) they ignore sentences longer than ten tokens because previous latent parsers do not scale to longer sentences (Haghighi and Klein, 2006; Borensztajn and Zuidema, 2007; Reichart and Rappoport, 2008).

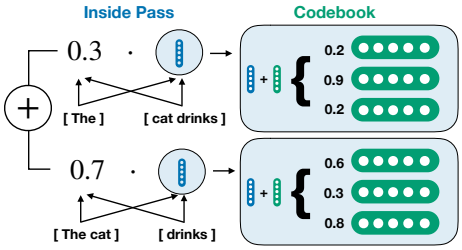

Figure 1: The left half of this figure depicts the inside-pass of the DIORA model as described in (Drozdov et al., 2019). We are interested in clustering the learned vectors $a(i, j)$ such that each span may be mapped to a phrase type. To enhance this clustering based approach, we augment the DIORA architecture with latent codes, shown in the right half of the figure.

Unlike previous work, we achieve strong results in unlabeled constituency parsing using a single model for both bracketing and labeling. Our approach relies on clustering span representations, which are fixed-length continuous vectors learned end-to-end using DIORA and do not require external resources such as part-of-speech tags. Furthermore, we enhance the DIORA architecture with latent codes: the model learns a distribution over these codes that loosely aligns with the ground-truth assignment of phrase types and, more importantly, improves the quality of the clusters.

Our code-enhanced DIORA architecture outperforms DIORA and achieves a new state of the art of 76.7 F1 on WSJ-10 when labeling a gold bracketing (19% relative error reduction over the previous best model, Haghighi and Klein 2006, which unlike our approach uses gold part-of-speech tags). Furthermore, we show DIORA is competitive when a ground truth bracketing is not provided, and instead must be induced. On

the full WSJ test set, DIORA outperforms two strong baselines, ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2019). We analyze the clustered constituents and observe they are separated syntactically (i.e. past tense vs. present participle verbs) and semantically (i.e. time-related phrases vs. references to people).

2 DIORA: Deep Inside-Outside Recursive Autoencoders

DIORA is a recursive autoencoder that learns to reconstruct an input sentence. A fundamental step in the reconstruction is to build a chart using the inside-outside algorithm (Baker, 1979), which represents a soft weighting over all possible binary trees of the input sentence. For all the model details, we refer the reader to Drozdov et al. (2019). For this work, it is key to understand two capabilities that DIORA provides: each span in a sentence is represented as a vector and DIORA induces a maximally likely binary tree for the sentence.

We can directly label the constituents of a sentence by clustering the learned span vectors from DIORA and assigning a label to each cluster. DIORA's autoencoder objective incentivizes the model to learn representations that compress the sentence well in order to reconstruct the input leading to the discovery of syntactic structure.

To encourage phrase representations to be easily clusterable into a small set of phrase types, we add an additional component to DIORA that forces phrase vectors to be representable by a small number of latent codes. Recent models have integrated ideas from vector quantization into variational autoencoders (Kingma and Welling, 2013) and key-value memory layers (Lample and Conneau, 2019), forcing the model to compress inputs into a single discrete latent embedding (van den Oord et al., 2017; Kaiser et al., 2018). Given a trained model, one could then assign labels to each of the latent variables and use this to label inputs directly.

We instead use a less restrictive modeling approach by assigning each input to a soft weighting over the K latent embeddings. This is similar to the soft EM training used by Roy et al. (2018) and can be thought of analogously to fuzzy/soft K-means clustering (Dunn, 1974; Bezdek, 1981) rather than hard K-means clustering.

Implementation and training details for our model are described in Appendix A.1.

2.1 DIORA with Codebook

DIORA is constrained to binary trees and its composition is represented as:

$$a(i,j) = \text{Compose}(\bar{a}(i), \bar{a}(j)), \quad (1)$$

where i and j are neighboring spans, $\bar{a}$ is summary vector for all possible parses over a span, and Compose is a function such as tree-LSTM or multi-layer perceptron.

To add the latent codebook into the model, we modify Eq. 1 to combine each constituent vector with a weighted summation over latent codes:

$$f_{cb}(x) = C^T \sigma(CWx),$$

where C is a codebook in $\mathbb{R}^{N \times M}$, x is a constituent vector in $\mathbb{R}^M$, and W is a bi-linear matrix used to compute the affinity between the constituent vector and the latent codes. One way to think of this equation is that each code (row in C) is a centroid, and the vector of affinity scores, $\sigma(CWx)$,[1] is a soft assignment of the constituent vector over the latent codes. The modified DIORA equation when incorporating the codebook is:

$$a'(i,j) = a(i,j) + f_{cb}(a(i,j)) \quad (2)$$

This codebook-enhanced architecture is visually depicted in Fig. 1. We use 70 codes when training this model (representing the 25 phrase types, 45 part-of-speech types, and ignoring the ROOT label), although we explore different configurations in §4.4.

3 Unsupervised Labeled Parsing

We perform unsupervised labeled constituency parsing with a multi-step approach.

Tree assignment. Assign a tree to each input sentence where the leaves of the tree are the words in the sentence. The tree is not labeled. This may be derived from the ground truth parse or induced using DIORA. When induced, we extract a binary tree by running the CKY algorithm[2] over DIORA's learned compatibility scores.

[1] More details about the equation $C^\top \sigma(CWx)$ are discussed in Appendix A.3. It's worth noting that σ can be an arbitrary function, in this work we use the identity function.

[2] The CKY algorithm is an efficient dynamic programming approach for recognizing constituency trees using exact inference (Kasami, 1966; Younger, 1967; Rush et al., 2010).

– WSJ (Test) –	Gold		Induced	
Model	**F1**$_\mu$	**F1**$_{max}$	**F1**$_\mu$	**F1**$_{max}$
Upper Bound	76.3	76.3	59.7	59.7
Majority (NP)	30.6	30.6	24.5	24.5
ELMo	58.5	59.4	43.5	48.2
ELMo$_{CI}$	53.4	56.3	38.5	40.2
BERT	41.8	42.2	38.1	38.3
DIORA	62.5 $\pm$0.5	63.4	50.2 $\pm$0.5	51.4
DIORA$_{CB}$	64.5 $\pm$0.6	65.5	49.8 $\pm$0.7	50.6
DIORA$^*_{CB}$	**66.4** $\pm$0.7	**67.8**	50.4 $\pm$0.7	51.5

Table 1: Results on the full Wall Street Journal test set.

– WSJ-10 –	Gold		Induced	
Model	**F1**$_\mu$	**F1**$_{max}$	**F1**$_\mu$	**F1**$_{max}$
Upper Bound	86.0	86.0	64.6	64.6
Majority (NP)	32.0	32.0	25.2	25.2
ELMo	67.8	68.9	50.1	53.0
ELMo$_{CI}$	65.9	67.3	46.0	47.6
BERT	54.6	57.8	44.5	45.2
DIORA	72.7 $\pm$1.5	76.2	55.2 $\pm$0.7	56.3
DIORA$_{CB}$	73.2 $\pm$1.7	75.7	54.5 $\pm$1.2	56.6
DIORA$^*_{CB}$	**74.9** $\pm$1.1	**76.7**	53.9 $\pm$0.8	55.1
PCFG†	-	51.6	-	35.3
BMM†	-	(76.8)	-	59.5
Proto†	-	71.1	-	65.2

Table 2: WSJ-10 unsupervised labeled constituency parsing with punctuation removed. † indicates that the model relies on gold part-of-speech tags, and results in parentheses are related but not comparable to others in the table. Proto (Haghighi and Klein, 2006) uses additional hand written rules. BMM (*Gold*) (Borensztajn and Zuidema, 2007) is evaluated using more than K clusters (where K is the size of the tag set) by mapping ground truth labels to induced labels, therefore it is not strictly comparable to the other results. Neither BMM (Reichart and Rappoport, 2008) nor Proto are effective at inducing unlabeled structured, so depend on external latent parsers for the *Induced* evaluation, either CCM (Klein and Manning, 2002) or CCL (Seginer, 2007). ELMo and BERT do not induce structure whatsoever and depend on DIORA for the *Induced* evaluation. ELMo$_{CI}$ uses only the context-insensitive character embeddings produced by ELMo.

Vector assignment. Assign the corresponding span vector to each constituent in these trees over the entire dataset. For DIORA without the codebook, this will be the concatenation of inside and outside vector. When using the codebook, this will be one of two options: the same as for DIORA, except using the output of Eq. 2, or it will be the soft score assignment of the codebook $\sigma(CWx)$. The first option is referred to as DIORA$_{CB}$ and the soft score assignment as DIORA$^*_{CB}$.

Cluster and label assignment. Cluster the collection of constituent vectors using K centroids learned with K-means. Finally, we use the ground truth phrase labels to assign each cluster to a phrase type — each constituent is mapped to the most common label within its cluster. We set K equal to the number of distinct phrase types in order to match previous work.

4 Experiments and Results

4.1 DIORA

We compare multiple configurations of DIORA. The first is the original model DIORA using the concatenation of the inside and outside vectors to represent a phrase. We also look at the codebook-enhanced architecture DIORA$_{CB}$, and when clustering the codebook scores we refer to the model as DIORA$^*_{CB}$.

4.2 Baselines

While ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2019) do not produce phrase vectors or induce recognizable constituency parse structure,[3] we show that they can still be used for unsupervised labeled parsing. When a reference parse is provided, it is only necessary to derive ad-hoc phrase vectors using the contextualized token vectors from these models. Peters et al. (2018b) describe an effective way to do so for ELMo, which involves concatenating the token vectors at the beginning and end of the phrase.[4] For BERT, it is critical to look at all layers as lower layers tend to be more syntactic in nature (Tenney et al., 2019). For both models, we report the max F1 and mean F1, and for the *Induced* evaluation we use the parses extracted from DIORA.

4.3 WSJ

Unsupervised constituency parsing has often been evaluated on different splits of the WSJ. For *labeled* constituency parsing, models that produce

[3] BERT does not strictly output word-level vectors. Rather, the output are subword vectors which we aggregate with mean-pooling to achieve a word-level representation.

[4] We tried many combinations (4 variants for ELMo and nearly 200 for BERT). They are described in Appendix A.2.

binary trees as output have a performance ceiling on this n-ary data — unary-chains limit recall[5] and more-than-binary nodes limit precision.

In some cases, an unlabeled tree structure over a sentence can be readily accessed. The algorithm described in §3 is robust to this case — simply replace the first step with the ground truth parse. We evaluate our model using the ground truth parse (Gold) and when inducing a parse (Induced). These results, comparison to baseline methods, and an upper-bound on binary tree performance are shown in Tables 1 and 2.

The Upper Bound in the *Induced* column of these tables represents a perfect labeling of the most accurate induced binary tree from DIORA, and the Majority (NP) row is the same tree labeled with the most common tag.

4.4 Model Ablations

As an alternative to clustering the constituent vectors with K-means, one can treat the codebook affinity scores, $\sigma(C^\top Wx)$, as a soft assignment over the clusters represented by each code. To examine this alternative, we replace K-means in the algorithm from §3 with the $\arg\max$ over the affinity scores. A model trained with 25 codes[6] achieves greater than 60% recall at labeling the ground truth trees for WSJ-10, indicating the codes represent some syntactic patterns although not as effectively as when using K-means.

Given these results, we are curious to see how model performance changes as the number of codes varies. We train codebook DIORA with $\{25, 70, 100, 200, 300, 400\}$ codes and evaluate each configuration using the procedure from §3 on the WSJ validation set. We compare the performance to non-codebook DIORA trained with $\{2, 3, 4, 5\}$ layers.[7] Results are shown in Fig. 2.

5 Qualitative Analysis of Clusters

We investigate phrase clusters from a single experiment (DIORA$_{CB}$ on WSJ-10), which are assigned to 9 NP, 5 VP, 5 S, 4 PP, 1 ADJP and 1 QP, according to the majority gold labels in that cluster. These 6 assigned phrase types correspond with the 6 most frequent labels.

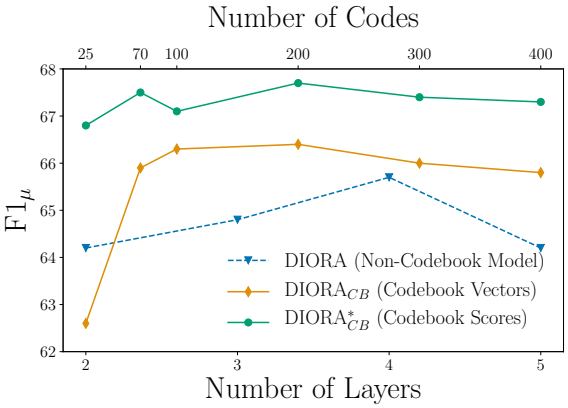

Figure 2: WSJ validation set results for different DIORA variants. The non-codebook DIORA sees improved performance as layers are added, but its average F1 never exceeds that of the codebook-enhanced architecture. Both clustering of codebook cells $a'(i, j)$ and codebook scores $\sigma(CWx)$ see performance improvements then diminishing returns as the number of codes increases. The dashed line uses the bottom x-axis and the solid lines use the top x-axis.

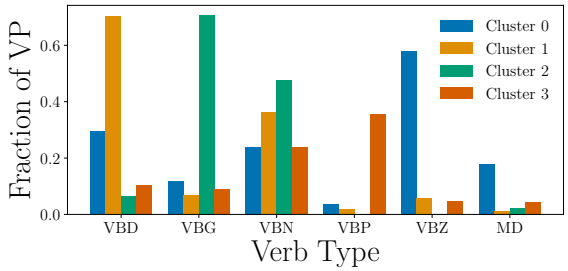

Figure 3: DIORA$_{CB}$ cluster assignment analyzed on WSJ-10 using part-of-speech tags. The four clusters shown were all assigned the VP label, yet seem to have finer-grained properties related to verb tense.

We find some semantic properties are evident in the clusters. For example, a 100% correct NP cluster (all phrases in this cluster have gold label NP) are all *possessive NPs*. One of the NP clusters consists of NPs that are mostly related to *time* (15 minutes, last year, this fall), even the incorrectly labeled phrases are time-related such as the ADVP 'no longer' and 'so far'. Another NP cluster identifies *people*, which includes "ms. parks 's mother" but excludes "mr. noriega 's proposal" even though both phrases have the same part-of-speech tag sequence [NNP NNP POS NN].

One of the five VP clusters completely takes the form *to + verb* — 7 out of 11 mislabeled cases contain "to" in the phrase, for example, "not

[5] Labeled parsing is usually evaluated on whether a span has the correct label. An NP prediction for a span would be correct if there is an NP-QP or QP-NP unary-chain over this span. A binary tree could only ever get one of QP or NP correct in this case, hence limiting recall.

[6] We use 25 codes here instead of 70 so that the model may be fairly compared with previous systems.

[7] Elsewhere in this paper, DIORA uses two layers.

to mention" (CONJP). The four other VP clusters present some degree of tense and singular/plural properties. A bar-chart showing the finer-grained properties of the VP clusters is shown in Fig. 3. Cluster 0 includes the majority of VBZ and MD (will, won't, can, could), Cluster 1 is mainly composed of past tense VPs (VBD), Cluster 2 has many VBG, and Cluster 3 consists of 86% VBP.

One of the S clusters captures instances of S that do not cover the whole sentence. Another starts with coordinating conjunctions such as "and" or "but", yet another captures phrases beginning with personal pronouns or determiners.

6 Conclusions

In this paper, we show that DIORA can be used for unsupervised *labeled* constituency parsing. We also introduce a new codebook-enhanced variant of DIORA that improves labeling performance. Our model outperforms the previous state of the art in unsupervised labeled constituency parsing for the WSJ-10 dataset, even though the previous best uses ground truth part-of-speech tags and ours does not, and introduces the first results on the full WSJ test set. The results indicate that grammar induction with types is viable using recent neural-network-based models, and our analysis warrants further exploration in this area.

Acknowledgements

We are grateful to our colleagues at UMass for help and advice, and to the UMass NLP reading group and the anonymous reviewers for feedback on drafts of this work. This work was supported in part by the Center for Intelligent Information Retrieval, in part by the National Science Foundation (NSF) grant numbers DMR-1534431, IIS-1514053 and CNS-0958392. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

James C Bezdek. 1981. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media.

Gideon Borensztajn and Willem Zuidema. 2007. *Bayesian model merging for unsupervised constituent labeling and grammar induction*. Institute for Logic, Language and Computation (ILLC): Technical Report.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*.

Andrew Drozdov, Pat Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive autoencoders. In *North American Association for Computational Linguistics (NAACL)*.

Joseph C Dunn. 1974. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. In *Journal of Cybernetics*.

Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *Association for Computational Linguistics (ACL)*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Phu Mon Htut, Kyunghyun Cho, and Samuel R Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Empirical Methods in Natural Language Processing (EMNLP): Short Paper*.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning (ICML)*.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Yoon Kim, Alexander M. Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019. Unsupervised recurrent neural network grammars. In *North American Association for Computational Linguistics (NAACL)*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Association for Computational Linguistics (ACL)*.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Association for Computational Linguistics (ACL)*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Pytorch Core Team. 2019. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. http://pytorch.org/. Accessed: 2019-05-18.

Roi Reichart and Ari Rappoport. 2008. Unsupervised induction of labeled parse trees by clustering with syntactic features. In *Conference on Computational Linguistics (COLING)*.

Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar. 2018. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Empirical Methods in Natural Language Processing (EMNLP): Short Paper*.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Association for Computational Linguistics (ACL)*.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations (ICLR)*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Empirical Methods in Natural Language Processing (EMNLP): Short Paper*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Association for Computational Linguistics (ACL)*.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2018a. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association of Computational Linguistics (TACL)*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018b. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Association for Computational Linguistics (NAACL)*.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.

A Supplemental Material

A.1 Implementation Details

We use the same settings as the best performing model for unsupervised unlabeled constituency parsing from the DIORA paper (Drozdov et al., 2019). We vary only in the batch size, using 32 instead of 128. A summary of the settings is below:

- Batch Size: 32
- Cell Size: 400
- Learning Rate: 0.002
- Optimization Algorithm: Adam
- Gradient Clipping: 5 (max-L2-norm)
- Training Data: SNLI (Bowman et al., 2015) + MultiNLI (Williams et al., 2018b)
- Negative Samples: 100 per batch
- Maximum Sentence Length: 20
- Composition Function: 2-layer MLP

All experiments were written using Pytorch (Pytorch Core Team, 2019) and using the publicly available DIORA codebase.[8]

A.2 ELMo/BERT Variants

For ELMo, we use the heuristic from Peters et al. (2018b) for each of the 3 layers individually and all combined, resulting in 4 variants.

With ELMo$_{CI}$, there is no notion layers and each token representation is completely context independent. For this reason, we use the same heuristic as for ELMo, but we also try averaging all tokens in a phrase, using the max, and concatenating the average with max. The latter of the 4 approaches worked best.

With BERT, each token is contextualized but retrieving a phrase representation is less clear cut. In addition, BERT operates over subword-tokens, so in all our methods, we tried aggregating subword-tokens using their average (same as in Hewitt and Manning (2019)) or using them as provided. To extract phrases, we tried 6 different approaches: using the max, using the average, concatenating the max and the average, using the heuristic, concatenating the max with the heuristic, and concatenating the max with the CLS and EOS tokens. We tried every variant with every layer of BERT (we used the base model that has 12 layers), and tried concatenating representations from each consecutive third of layers. This resulted in $2 \times 6 \times 15 = 180$ variants of the BERT baseline.

A.3 Codebook Notes

For the codebook, we use equation:

$$x' = x + C^\top \sigma(CWx)$$

In practice, we found that for σ it is effective to use the identity function. The matrix W is a learned bi-linear mapping, but if it is an identity matrix, than the equation above reduces to a much simpler form:

$$x' = (C^\top C + I)\, x$$

When interpreted this way, there could be some interesting properties to explore or enforce in the codebook equation, although we leave this for future work.

[8] https://github.com/iesl/diora