Dependency parsing with structure preserving embeddings

Ákos Kádár[*] [†] **Lan Xiao**[*] **Mete Kemertas**[‡] **Federico Fancellu**
Allan Jepson **Afsaneh Fazly**
Samsung AI Centre Toronto (SAIC Toronto)
{lan.xiao, federico.f, allan.jepson, a.fazly}@samsung.com

Abstract

Modern neural approaches to dependency parsing are trained to predict a tree structure by jointly learning a contextual representation for tokens in a sentence, as well as a head–dependent scoring function. Whereas this strategy results in high performance, it is difficult to interpret these representations in relation to the geometry of the underlying tree structure. Our work seeks instead to learn interpretable representations by training a parser to explicitly preserve structural properties of a tree. We do so by casting dependency parsing as a tree embedding problem where we incorporate geometric properties of dependency trees in the form of training losses within a graph-based parser. We provide a thorough evaluation of these geometric losses, showing that the majority of them yield strong tree distance preservation as well as parsing performance on par with a competitive graph-based parser (Qi et al., 2018). Finally, we show where parsing errors lie in terms of tree relationship in order to guide future work.

1 Introduction

Dependency grammars are syntactic formalisms that represent the syntactic structure of a sentence as asymmetric binary grammatical relations among words (Tesnière, 1959; Hudson, 1984; Melcuk, 2003). An example dependency structure is given in Figure 1. Formally, a dependency structure is defined as a directed graph where words are vertices and relations are labelled directed edges (the *arcs*) between a child (the *dependent*) and its parent (the *head*). In practice, dependency structures considered for syntactic analysis are trees. Dependency trees have long been used to improve the performance of many NLP applications, including machine translation (Ding and Palmer, 2004; Menezes et al., 2010; Bastings et al., 2017), relation extraction (Kambhatla, 2004; Bunescu and Mooney, 2005; Miwa and Bansal, 2016; Zhang et al., 2018), and semantic role labeling (Hacioglu, 2004; Marcheggiani and Titov, 2017; He et al., 2018).

In order to assign the correct dependency tree to a sentence, dependency parsers are trained to correctly identify head–dependent relations between pairs of words. Modern neural approaches do so by jointly learning contextual feature representations for the tokens in a sentence, as well as a parsing decision function. This is the case for recent graph-based parsers (Zhang et al., 2017; Dozat et al., 2017; Mohammadshahi and Henderson, 2020, *inter alia*) where an encoder feature extractor is complemented by a score function predicting the likelihood of a word to be the head of another. However, whereas this joint learning strategy results in state-of-the-art performance, the representations learned by these parsers are opaque.

As a first step towards learning interpretable parser representations, here we take a different approach: in addition to learning-to-parse, we seek to learn representations from which tree distances between words in dependency trees can be recovered. This stems from one simple observation: previous approaches do not take into account the *geometry* of the tree they try to model. That is, parsers are unaware of the structural properties of the tree (e.g., distance between nodes, depth from root), and as such are not trained to *explicitly* preserve these properties. In this respect, our approach is aligned with recent work looking at these geometric properties in the context of *probing* the BERT (Devlin et al., 2019) embedding space for syntactic structure (Hewitt and Manning, 2019; Reif et al., 2019). In particular, Hewitt and Manning (2019)

[*] Equal contribution.

[†] Now at Borealis AI (akos.kadar@borealisai.com).

[‡] Now at University of Toronto (kemertas@cs.toronto.edu).

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 1684–1697
April 19 - 23, 2021.

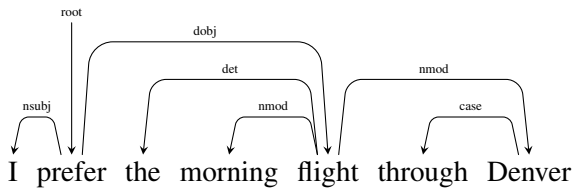


Figure 1: Example dependency tree.

have shown that it is possible to recover approximate syntactic trees from BERT embeddings by a linear transformation trained to minimize the difference between predicted and ground-truth tree distances. Given these results we then ask: is it possible to extend this idea to directly train tree-aware dependency parsers? We argue that using the geometric tree structure to embed the dependency trees enhances the interpretability of the learned representations

In this paper, we show that this is indeed possible by casting dependency parsing as a tree embedding problem. Specifically, we view a dependency tree as a finite metric space, and compute head–dependent scores for all word pairs within a sentence as follows: Given a sentence $s = (w_0, \ldots, w_{n_s})$, where w_0 is a special ROOT token, we compute a geometric tree embedding $\phi : \{w_0, \ldots, w_{n_s}\} \rightarrow \mathbb{R}^m$ that maps tokens w_i to m-dimensional vectors. Geometric properties of an ideal isometric tree embedding are used to define the functional form of head–dependent (edge) scores $\psi(w_i, w_j)$. Concretely, our approach predict dependency trees only from pairwise embedded node distances, which completely specify the score function. We consider this as a step in the direction of interpretable end-to-end dependency parsing. We start with a straight-forward application of Hewitt and Manning (2019) and consider a mean absolute error (MAE) loss that encourages the embedding ϕ to approximate an isometric embedding of the ground-truth tree T_s into $(\mathbb{R}^m, d_R)$. In this paper, we use the squared Euclidean distance semi-metric (d_2^2) as in Hewitt and Manning (2019) and also consider the distance obtained from the ℓ_1-norm (d_1)[1]. We show formally that any isometric d_2^2 embedding can be simply rotated to form an isometric d_1 embedding.

[1]The squared Euclidean distance is a semi-metric, not a metric, as it does not respect the triangle inequality. Here $d_1(\mathbf{u}, \mathbf{v}) := ||\mathbf{u} - \mathbf{v}||_1$. Other distance functions could also be considered for d_R but that is beyond the scope of current work.

We learn the tree embedding ϕ and the edge score function ψ through end-to-end training, by incorporating geometric properties of dependency trees (in terms of distance and depth) in the form of geometric losses within a graph-based parser. As our base parser, we use a simplified version of the biaffine parser of Qi et al. (2018). This setup allows us to directly compare the performance of our losses and score functions to the biaffine score function used in several state-of-the-art graph parsers. We propose three additional losses for training a dependency parser expressed explicitly through tree distances: a maximum likelihood estimation function, a margin-based loss function, and one based on cross-entropy.

Finally, we explore whether adding a soft global constraint on the isometry of the learned trees helps with parsing performance; to this end, we combine our novel loss functions with the MAE loss.

We evaluate our approach on 16 languages from different families. We complement unlabeled accuracy of head–dependent attachement scores (UAS) with a Spearman's ρ correlation between predicted and true distances (DSpr) to directly assess geometric properties of the output trees. We also provide labeled attachment scores (LAS) for completeness. Through extensive experimentation, we make the following observations:

- All of our novel tree distance based losses outperform the MAE loss of Hewitt and Manning (2019)
- All losses using the d_1 metric provide better distance preservation properties and dependency parsing performance than using the d_2^2 semi-metric.
- Five of the six loss combinations (using d_1) show both strong distance preservation properties and parsing performance, indicating that distance preservation can be obtained without trading off parsing performance. Only the maximum likelihood estimation loss (on its own) has poor distance preservation; however, we find that the combination of this loss with the MAE loss greatly improves distance preservation, while achieving similar or better parsing performance.
- We show that the majority of parsing errors are local in tree distance, with by far the most frequent incorrect head assignments being either true sisters or grandparents.

Our results in the direction of accurate dependency parser that closely preserve tree distances are encouraging.

2 Background: Metric Tree Embeddings

A tree $T = (V, E)$ along with the distance $d_T(u, v)$ is a finite metric space,[2] where $d_T(u, v)$ is the length of the shortest path between any $u, v \in V$. In this paper we consider tree embeddings, $\phi : V \to \mathbb{R}^m$, which map nodes $v \in V$ to points $\phi(v) \in \mathbb{R}^m$ such that the mapping ϕ approximately preserves tree distance. That is, for all pairs $u, v \in V$, the tree distance $d_T(u, v)$ is roughly the corresponding distance $d_R(\phi(u), \phi(v))$ in the embedding space $\mathbb{R}^m$, where we select the metric or semi-metric $d_R(x, y)$ in $\mathbb{R}^m$.

In this section, we discuss the choice of d_R and illustrate distortion free (i.e., isometric) embeddings $\phi : V \to \mathbb{R}^m$. These distortion free embeddings motivate the formulation of losses that we use for training suitable embeddings, as discussed in § 3.

To choose the distance measure d_R in the embedding space, we note that for a sufficiently large dimension m: i) any tree can be embedded isometrically into ℓ_1; ii) any metric space (including trees) can be embedded into ℓ_∞; and iii) for ℓ_p spaces, with $1 < p < \infty$, trees can only be embedded with distortion (Linial et al., 1995). The power transform of the Euclidean distance $d_2(\mathbf{x}, \mathbf{y})^c$, with $c \geq 2$, allows for isometric embedding of trees (Reif et al., 2019). However, the squared Euclidean distance $d_2(\mathbf{x}, \mathbf{y})^2$ does not satisfy the triangle inequality[3] and therefore is only a semi-metric. Nevertheless, both $(\mathbb{R}^m, d_1)$ and $(\mathbb{R}^m, d_2^2)$ are natural choices for embedding spaces for trees and, in this paper, we restrict our attention to these.

We follow Reif et al. (2019) to explicitly construct squared Euclidean embeddings. Specifically, all distortion free embeddings into $(\mathbb{R}^m, d_2^2)$ can be simply expressed in terms of the edge displacement vectors $\{\mathbf{z}_i\}_{i=1}^{|E|}$, where $\mathbf{z}_i \in \mathbb{R}^m$ is the displacement between the embedded endpoints of edge $e_i \in E$ (i.e., $\mathbf{z} := \phi(c) - \phi(p)$, where $c, p \in V$ are a pair of child and parent nodes). For an isometric embedding, it turns out we require these $\mathbf{z}_i$'s to be orthonormal, that is

$$\mathbf{Z}^T \mathbf{Z} = \mathbb{1}, \tag{1}$$

where $\mathbf{Z} \in \mathbb{R}^{m \times |E|}$ is the matrix having $\mathbf{z}_i$ as the i^{th} column and $\mathbb{1}$ denotes the $|E| \times |E|$ identity matrix. In addition, it is useful to define $\rho(u, v)$ to denote the shortest path between two vertices $u, v \in V$. And, finally, define the indicator vector $\mathbf{b}(u, v) \in \mathbb{Z}^{|E|}$ such that $b_i(u, v) = 1$ when edge e_i is on the shortest path between u and v, and $b_i(u, v) = 0$ otherwise. With this notation we have the following theorem:

Theorem 2.1. Pythagorean Embeddings. Given a rooted tree (V, E, r), where $r \in V$ denotes the root, then for any $m \geq |E|$ there exists an embedding $\phi : V \to \mathbb{R}^m$ such that,

$$\forall_{u,v \in V}\ d_T(u, v) = ||\phi(u) - \phi(v)||_2^2. \tag{2}$$

Moreover, ϕ satisfies (2) if and only if

$$\phi(v) = \phi(r) + \mathbf{Z}\,\mathbf{b}(r, v) \tag{3}$$

for some matrix $\mathbf{Z} \in \mathbb{R}^{m \times |E|}$ with orthonormal columns (i.e., Eqn. (1) is satisfied).

See Appendix B-D for an example that demonstrates this construction along with proofs.

The edge-on-path indicator vectors $\mathbf{b}(r, v)$ provide some additional intuition about these squared Euclidean embeddings. Specifically, for $v, w \in V$ we have

$$\begin{aligned} g(v) \coloneqq \mathbf{Z}^T \phi(v) &= \mathbf{Z}^T \phi(r) + \mathbf{b}(r, v), && (4) \\ g(v) - g(w) &= \mathbf{Z}^T \mathbf{Z}(\mathbf{b}(r, v) - \mathbf{b}(r, w)), \\ &= \mathbf{b}(r, v) - \mathbf{b}(r, w) && (5) \\ ||g(v) - g(w)||_1 &= ||\mathbf{b}(r, v) - \mathbf{b}(r, w)||_1 \\ &= d_T(v, w) && (6) \end{aligned}$$

Here (5) follows from (1), (3) and (4). Therefore g is an isometric ℓ_1 embedding which expresses tree distance in terms of the ℓ_1 norm of the difference between two path vectors $\mathbf{b}$. One interesting consequence of Theorem 2.1, along with Eqn.'s (4) and (6), is:

Corollary 2.1. ℓ_1 Embeddings. Given any isometric embedding $\phi : V \to \mathbb{R}^m$ using the squared Euclidean distance, the embedding $g(v) = \mathbf{Z}^T \phi(v)$ is an isometric embedding of the same tree (V, E, r) into $(\mathbb{R}^{|E|}, d_1)$. Here $\mathbf{Z}$ is as described in (3).

[2] Metric spaces are 2-tuples (X, d_X) consisting of a set of elements X and a metric $d_X : X \times X \mapsto [0, \infty)$ quantifying notion of distance between any pair of elements of X.

[3] Consider three points on a line with successive pairs separated by a d_2^2 distance of 1. Then the outer two are separated by an d_2^2 of $2^2 = 4$, which is larger than the sum of the distances between the successive pairs.

That is, any distance preserving tree embedding using the squared Euclidean norm can simply be rotated to an isometric ℓ_1 tree embedding.[4] Note the converse of Cor. 2.1 is not true. For example, three equally spaced points on a line form an ℓ_1 embedding that cannot be linearly transformed to a d_2^2 embedding. Indeed, it is shown in (Aksoy et al., 2020) that a finite metric tree can be embedded into $(\mathbb{R}^m, d_1)$ if and only if it has at most $2m$ leaves. Thus, for a fixed dimensional embedding space $\mathbb{R}^m$, the metric d_1 allows for more trees to be isometrically embedded than d_2^2.

In this paper we use an embedding dimension m that is larger than the number of edges and thus isometric tree embeddings are feasible using both d_2^2 and d_1. The learning problem considered in this paper is, given a sentence s, we seek to embed each of the sentence's tokens into $(\mathbb{R}^m, d_R)$ such that the embedded tree is nearly isometric to the dependency parse tree for s. Here we evaluate using both d_1 and d_2^2 for d_R.

3 Geometric Losses for Tree Embeddings

Given a sentence $s = (w_0, \ldots, w_{n_s})$, where w_0 is a special ROOT token inserted at the beginning of every sentence, dependency parsing seeks to recover the correct dependency tree $T_s = (V_s, E_s)$. For simplicity, we label the tree nodes in V_s with the tokens themselves, so $V_s = \{w_0, \ldots, w_{n_s}\}$. A geometric tree embedding maps tokens w_i within a sentence s to embedded points $\mathbf{v}_i = \phi(i, s) \in \mathbb{R}^m$ (for brevity, we drop the dependence on the whole sentence s and simply write $\phi(w_i)$). In § 2, we describe the exact geometry of the isometric embeddings using the d_2^2 semi-metric. For d_1 isometric embeddings we show one sub-class are simply rotations of isometric d_2^2 embeddings, but there are other forms. This section examines the use of auxiliary losses on the embedding ϕ that encourage approximately isometric embeddings. We expect an approximation of this geometry holds in d_2^2 when the losses are sufficiently small. Since we do not have a similar proof of necessity as in Appendix D for d_1 embeddings, the local geometry is more open.

Given such an embedding, we then follow first-order graph-based dependency parsers (McDonald et al., 2005) which compute a pairwise score $\psi(\mathbf{v}_i, \mathbf{v}_j)$ that indicates how likely it is for w_j to be the head of w_i. These scores provide edge weights on a fully connected embedded graph on $\phi(V_s) \times \phi(V_s)$. Having trained a network to compute suitable embeddings $\mathbf{v}_i = \phi(w_i)$ and edge weights $\psi(\mathbf{v}_i, \mathbf{v}_j)$, parsing then amounts to finding the maximum spanning tree in this weighted graph that is rooted at $\mathbf{v}_0$; a detailed description of the parser architecture is provided in § 4.

Given this general approach, a natural choice for an auxiliary loss on ϕ is to consider the mean absolute error (MAE) in the distances between the embeddings of any two nodes:

$$\mathcal{L}_{MAE}(\phi, T) = \frac{1}{|V|(|V|-1)} \sum_{\substack{w_i, w_j \in V, \\ i \neq j}} |\, d_R(\mathbf{v}_i, \mathbf{v}_j) - d_{T_s}(w_i, w_j)|, \quad (7)$$

where $\mathbf{v}_i = \phi(w_i)$ (we drop the subscripts s for brevity). This MAE loss treats the distance errors for all pair of nodes as equally important, and we therefore refer to it as a *global* loss. Note the loss in (7) is zero only when the embedding ϕ is an isometric tree embedding with respect to d_R.

We next consider the edge scoring function ψ, whose role is to assign costs to proposed head–dependent pairs (w_i, w_j). We denote this ground-truth head–dependent relation by $\varrho(w_i, w_j)$ (which is true when w_j is the head of w_i in T) and note it can be defined **only** in terms of the distance $d_T(w_i, w_j)$ and depth difference $\Delta_s(w_i, w_j) = d_T(w_0, w_i) - d_T(w_0, w_j)$ as follows:

$$\forall_{w_i, w_j \in V} \varrho(w_i, w_j) \iff d_T(w_i, w_j) = 1 \wedge \Delta(w_i, w_j) = 1. \quad (8)$$

Each node w_i then has a unique head w_j defined by (8), except for the ROOT w_0 which has none. Given the embedding ϕ is providing a near-isometric tree, we define an edge scoring function in the embedding space, namely $\psi(\mathbf{v}_i, \mathbf{v}_j)$, by rewriting (8) in the following probabilistic form:

$$\mathbf{v}_i \coloneqq \phi(w_i), \quad (9)$$

$$\Delta_{ij}^{\phi} \coloneqq d_R(\mathbf{v}_i, \mathbf{v}_0) - d_R(\mathbf{v}_j, \mathbf{v}_0), \quad (10)$$

$$\ell_{ij}^{\phi} \coloneqq |d_R(\mathbf{v}_i, \mathbf{v}_j) - 1| + |\Delta_{ij}^{\phi} - 1|, \quad (11)$$

$$\psi(\mathbf{v}_i, \mathbf{v}_j) \coloneqq -\ell_{ij}^{\phi}, \quad (12)$$

$$p_{\phi}(w_j | w_i) \coloneqq \frac{e^{\psi(\mathbf{v}_i, \mathbf{v}_j)/\tau}}{\sum_{\substack{0 \leq k \leq n \\ k \neq i}} e^{\psi(\mathbf{v}_i, \mathbf{v}_k)/\tau}}. \quad (13)$$

where τ is a temperature parameter. Here we refer to ℓ_{ij}^{ϕ} as the "head–dependent cost" for the pair w_i and w_j.

[4] This rotation aligns the edge directions $\mathbf{z}_i$ with the standard bases vectors $\pm e_i$ of $\mathbb{R}^m$.

A natural loss on both the embedding $\phi : V \rightarrow \mathbb{R}^m$ and the edge-cost $\psi(\mathbf{v}_i, \mathbf{v}_j)$ is the maximum likelihood loss

$$\mathcal{L}_{MLE}(\phi, T) = \sum_{\substack{w_i, w_j \in V, \\ i \neq j}} \varrho(w_i, w_j) \left[-\log(p_\phi(w_j \mid w_i)) \right] \tag{14}$$

where $p_\phi(w_j|w_i)$ is defined in (13).

As an alternative to the MLE loss we consider a margin based approach where the task is to minimize ℓ^ϕ_{ij} for the true head j, subject to $\ell^\phi_{ik} \geq \ell^\phi_{ij} + \alpha$ for all $k \in \{0, \ldots, n\} \backslash \{i, j\}$, where $\alpha > 0$ is a margin.[5] We explore such a margin-based approach using the soft triplet loss (Sohn, 2016)

$$\mathcal{L}_\alpha(\phi, T) = \tag{15}$$
$$\sum_{\substack{w_i, w_j \in V, \\ i \neq j}} \varrho(w_i, w_j) \log \left[1 + \sum_{w_k \in V \backslash \{w_j, w_i\}} e^{\alpha - \ell^\phi_{ik} + \ell^\phi_{ij}} \right].$$

As a third alternative for the head–dependent loss we consider the cross-entropy between the probability distribution $p_\phi(w_j|w_i)$ and the corresponding distribution $p_T(w_i|w_j)$ formed by using an isometric embedding ϕ_T in Eqns. (9 - 13). Note that for an isometric embedding ϕ_T we have $d_R(\mathbf{v}_i, \mathbf{v}_j) = d_T(w_i, w_j)$, and this can be used to simplify the resulting expression. Specifically, we find $p_T(w_j|w_i)$ depends only on the true tree distance d_T and not on the details of ϕ_T. The cross-entropy loss is then

$$\mathcal{L}_{CE}(\phi, T) = \sum_{\substack{w_i \in V \backslash \{w_0\}, \\ w_j \in V \backslash \{w_i\}}} p_T(w_j|w_i) \left[-\log(p_\phi(w_j|w_i)) \right]. \tag{16}$$

In summary, we investigate the choice between several different head–dependent losses (i.e., Eqn.'s (14), (15), or (16), and optionally combine each of these with the explicit global MAE loss (7).

4 Evaluation

4.1 Model

We use a simplified version of the Biaffine dependency parser of Qi et al. (2018).[6] First, we give an overview of the Biaffine parser, and then describe our modifications. Biaffine is composed of a highway-BiLSTM encoder (Srivastava et al., 2015) that takes as input a sequence of n_s+1 embeddings $\mathbf{x}_0, \ldots, \mathbf{x}_{n_s}$, where each $\mathbf{x}_i$ is a concatenation of word-level, character-level, part-of-speech and morphological feature embeddings. We use pre-trained word2vec (Mikolov et al., 2013) when available, and fastText embedding (Bojanowski et al., 2017) otherwise. We train the rest of the embeddings from scratch.

Given such an input sequence, the Biaffine parser predicts the most likely head for each word (referred to as unlabelled attachment prediction), along with the grammatical relation between each pair of head and dependent words (labelled attachment prediction). Biaffine first calculates contextual embeddings $\mathbf{h}_i$ (through the encoder), and then projects these into separate head and dependent representations for each word (through two separate MLP networks):

$$\mathbf{h}_i = \text{BiLSTM}_i(\mathbf{x}_0, \ldots, \mathbf{x}_{n_s}), \tag{17}$$
$$\mathbf{h}^{\text{head}}_i, \mathbf{h}^{\text{dep}}_j = \text{MLP}^{\text{head}}(\mathbf{h}_i), \text{MLP}^{\text{dep}}(\mathbf{h}_j), \tag{18}$$

where $\mathbf{h}^{\text{head}}_i$ and $\mathbf{h}^{\text{dep}}_i$ are the *head* and *dependent* representations. Next, for each pair of words w_i and w_j, a head–dependent score s_{ij} and a corresponding probability $p(w_j|w_i)$ are calculated with a learnable biaffine weight $\mathbf{U}$:

$$\begin{aligned} s_{ij} &= \mathbf{h}^{\text{head}}_i \mathbf{U} \mathbf{h}^{\text{dep}}_j, \\ &\coloneqq \text{Biaffine}(\mathbf{h}_i, \mathbf{h}_j), \end{aligned} \tag{19}$$

$$p(w_j|w_i) = \frac{e^{s_{ij}}}{\sum_{w_k \in V_s \backslash \{w_i\}} e^{s_{ik}}}. \tag{20}$$

Our geometric tree embedding ϕ computes a single representation for each node, as such we replace the separate head and dependent MLP networks with a single MLP network[7]:

$$\mathbf{v}_i \coloneqq \phi(w_i), \tag{21}$$
$$= \text{MLP}(\mathbf{h}_i). \tag{22}$$

Given $\mathbf{v}_0, \ldots, \mathbf{v}_n$, head–dependent scores s_{ij} are defined as in:

$$s_{ij} = \psi(\mathbf{v}_i, \mathbf{v}_j), \tag{23}$$

where ψ is calculated as in Eqn. (12). We obtain asymmetry in our score function $\psi(\mathbf{v}_i, \mathbf{v}_j)$ from the depth difference term $|\Delta^\phi_{ij} - 1|$ in Eqn. (11).

[5] Note that an isometric embedding ϕ provides one solution for which $\ell^\phi_{ij} = 0$ for all head–dependent pairs, and $\ell^\phi_{ij} \geq 2$ otherwise.

[6] We use the codebase provided at `https://github.com/stanfordnlp/stanfordnlp`.

[7] To focus on learning unlabeled tree structures, we also remove the auxiliary losses that penalize rightward attachments or long dependencies, since the model behaves differently from the unlabeled prediction.

During inference, we use the Chu-Liu-Edmonds algorithm (Chi, 1999; Edmonds, 1967) to find the highest-scoring dependency tree. While our main focus is embedding unlabeled trees for dependency relation prediction, for completeness we also report results on labeled dependency tree prediction. We use the same classifier as Biaffine — with the same setting as described in Qi et al. (2018) — to estimate the probabilities of dependency labels $\mathbf{l}_{ij}$ for a given head–dependent pair w_i and w_j:

$$p(l_{k,ij}|w_j, w_i) = \frac{e^{s^{\text{rel}}_{k,ij}}}{\sum_n e^{s^{\text{rel}}_{n,ij}}} \quad (24)$$

$$\mathbf{s}^{\text{rel}}_{ij} = \text{Biaffine}^{\text{rel}}(\mathbf{h}_i, \mathbf{h}_j) \quad (25)$$

where $\mathbf{s}^{\text{rel}}_{ij} \in \mathbb{R}^K$ is a K-dimensional vector containing the dependency relation scores for each of the K dependency labels.

4.2 Experimental Setup

Data. We perform experiments on Universal Dependencies (UD; Nivre et al. 2020). Relying on criteria proposed by Kulmizev et al. (2019), we select 16 languages from different families, with different scrips and training sizes, all with good annotation quality. We use the following treebanks from UD v2.2: Arabic-PADT, Basque-BDT, Bulgarian-BTB, Chinese-GSD, Czech-PDT, Danish-DDT, English-EWT, Finnish-TDT, Hebrew-HTB, Hindi-HDTB, Italian-ISDT, Japanese-GSD, Korean-GSD, Russian-SynTagRus, Swedish-Talbanken, Turkish-IMST.

Performance Measures. We report the overall accuracy of head and relation predictions for all tokens in the test portion of the data sets. Given our model prediction and a reference parse for a given input, accuracy is calculated using two standard measures: Unlabelled Attachment Score (UAS), that is the percentage of tokens that are assigned the correct head; and Labelled Attachment Score (LAS) that is the percentage of tokens that are assigned the correct head and the correct grammatical relation. We use UAS for model selection.

To assess how well the learned tree embeddings preserve distances, we follow Hewitt and Manning (2019) and Hall Maudslay et al. (2020), and measure the correlation between the learned and ground-truth tree distances. Specifically, for all words in all sentences, we compute Spearman's ρ between predicted and true distances. We first average the correlation coefficients for sentences of the same length. We report the macro-average over these averages for sentences of length 5–50, referred to as DSpr.

	d_2^2	d_1	d_1 [+ $\mathcal{L}_{MAE}$]
$\mathcal{L}_{MAE}$	90.28 (0.89)	90.53 (0.89)	N/A
$\mathcal{L}_{MLE}$	91.47 (0.60)	91.51 (0.71)	91.77 (0.85)
$\mathcal{L}_{\alpha}$	91.43 (0.83)	91.57 (0.88)	91.41 (0.89)
$\mathcal{L}_{CE}$	86.82 (0.38)	91.79 (0.81)	91.85 (0.88)

Table 1: UAS (DSpr) of d_2^2 and d_1 embeddings, plus effects of adding the auxiliary MAE loss; reported on the development portion of English-EWT.

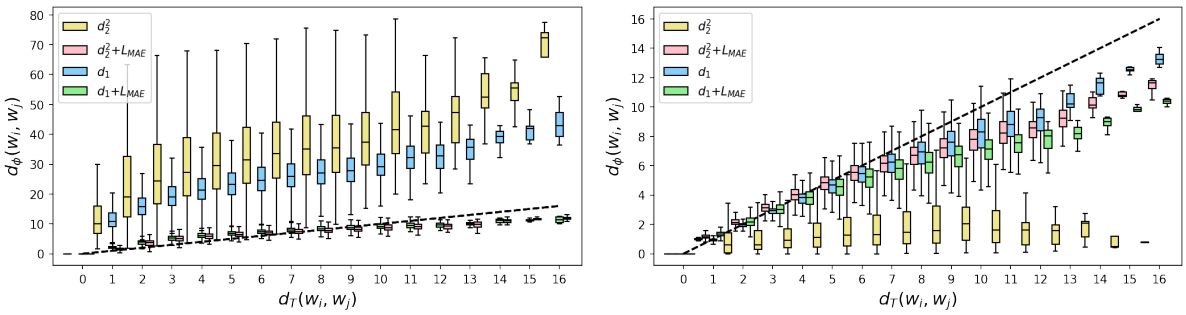
Hyperparameters. We adopt the same hyperparameter configuration as in the original Biaffine model (Qi et al., 2018) up to the BiLSTM layer for the head–dependent classifier, and the same configuration for the entire dependency label classifier. We perform grid search on the remaining hyperparameters and select best hyperparameter configurations based on UAS on the development portion of the English-EWT data. Based on the results, we set the margin $\alpha = 3$ for $\mathcal{L}_{\alpha}$, and the temperature $\tau = 1$ for $\mathcal{L}_{MLE}$ and $\tau = 0.2$ for $\mathcal{L}_{CE}$. In our evaluation, we run experiments that involve the combination of any of $\mathcal{L}_{\alpha}$, $\mathcal{L}_{CE}$, and $\mathcal{L}_{MLE}$ with $\mathcal{L}_{MAE}$ as an auxiliary loss, with a coefficient λ_1. We find the best values for λ_1 to be 0.2 for $\mathcal{L}_{\alpha}$, and 0.1 for both $\mathcal{L}_{CE}$ and $\mathcal{L}_{MLE}$. We refer the reader to Appendix A for the full list of hyperparameters and training details.

5 Results

5.1 Metric Spaces and Geometric Losses

We first verify whether the choice of metric space impacts performance. Table 1 reports UAS (and DSpr) on the development portion of the English-EWT corpus, for tree embeddings in both $(\mathbb{R}^m, d_1)$ and $(\mathbb{R}^m, d_2^2)$ metric spaces (referred to as d_1 and d_2^2 for brevity, respectively).

The first row shows results when trained with $\mathcal{L}_{MAE}$ only. It learns an embedding that provides good approximation of global tree distances using d_2^2 (DSpr: 0.89), which is similar to the findings reported by Hewitt and Manning (2019), but is suboptimal in terms of parsing (UAS: 90.28). On the other hand we can see that all head–dependent losses achieve stronger parsing performances in both spaces, with the proper metric d_1 leading to higher UAS and DSpr scores across the board (including $\mathcal{L}_{MAE}$) when compared to d_2^2. We will then report results in the rest of this section only



(a) (b)

Figure 2: Distance distributions for (a) $\mathcal{L}_{MLE}$ and (b) $\mathcal{L}_{CE}$, for different metric spaces (d_2^2/d_1) and trained with or without the auxiliary loss $\mathcal{L}_{MAE}$. The four box plots at each x-tick summarize the distribution of predicted distances for a particular ground-truth distance. Each box covers data distribution between 25 to 75 percentile, with median showing by the black line. The dashed line represents perfect correlation.

with the d_1 metric, unless otherwise stated.

Table 1 also shows a comparison between the different head–dependent losses, with and without the auxiliary loss $\mathcal{L}_{MAE}$ that further constrains ϕ to be globally isometric. In isolation, $\mathcal{L}_{CE}$ yields the best UAS while $\mathcal{L}_\alpha$ the best DSpr; interpolating the losses with $\mathcal{L}_{MAE}$ improves results for $\mathcal{L}_{MLE}$ and $\mathcal{L}_{CE}$, especially in terms of DSpr. This is in line with our expectations that the auxiliary loss encourages the parser to learn an embedding that more faithfully preserves global tree distances.

$\mathcal{L}_{MLE}$ seeks to correctly identify all head–dependent relations by maximizing the probability $p_\phi(w_j|w_i)$ of true head–dependent pairs with no explicit constraints on global isometry; see Eqn. (14). We thus hypothesize that $\mathcal{L}_{MLE}$ can learn an embedding that produces good UAS, but is far from being isometric to the ground-truth tree. We verify this empirically: Figure 2(a) shows that the addition of $\mathcal{L}_{MAE}$ greatly regulates the embedding distances of the trees produced by $\mathcal{L}_{MLE}$, hence improving the DSpr score for this loss.[8] $\mathcal{L}_{CE}$ on the other hand seeks to match $p_\phi(w_j|w_i)$ with $p_T(w_j|w_i)$, which encourages all embedding distances to be correlated with tree distances; see Eqn. (16). However, the $p_T(w_j|w_i)$ term in this loss gives higher weights to word pairs that are closer in terms of ground-truth tree distance and therefore the model is trained to focus more on preserving short distances.[9] Figure 2(b) confirms that the embedding obtained using $\mathcal{L}_{CE}$ underestimates the ground-truth tree distances, and adding the auxiliary MAE loss helps regulate this distortion of tree distances.

§ 2 describes the exact geometry of the embeddings that out model learns using the d_2^2 semi-metric in the special case that $\mathcal{L}_{MAE}$ is reduced to zero. In Figure 2(b), we observe small losses up to tree distance five when d_2^2 is used with $\mathcal{L}_{CE}$ and $\mathcal{L}_{MAE}$. Therefore we expect the local geometry of d_2^2 trees to approximately follow Eqn. (3).

Overall, for both losses we observe that, on average, d_2^2 embeddings lead to predicted tree distances that have large variances, as well as medians further away from the ground-truth, whereas the d_1 embeddings are more stable and accurate. This agrees with the observations in Table 1 that embeddings in d_1 have better DSpr than embeddings in d_2^2 for all losses we considered. The same comparison between d_ϕ and d_T for $\mathcal{L}_\alpha$ is provided in Appendix E.

5.2 Comparison with Qi et al. (2018)

We compare the parsing results for the six geometric loss combinations against the biaffine parser of Qi et al. (2018) for all treebanks in Table 2. For a fair comparison, we re-run all experiments and report our results for the biaffine parser. We report UAS for models trained without the dependency label prediction loss in order to focus on unlabeled tree structure and LAS for completeness. In general, the parsing performance is stable across

[8] We observe that the optimized $\mathcal{L}_{MLE}$ (without MAE) is lower than that produced by an isometric embedding on the development portion of English-EWT, indicating that the observed distortion of tree distances is an overfitting issue.

[9] The local emphasis is stronger with lower temperature. We chose temperature $\tau = 0.2$ based on UAS on the development portion of English-EWT. However, we find a temperature of 1 to provide better approximation of global distances, especially for tree distances less than five (see Appendix E and Figure 6).

	Biaffine		$\mathcal{L}_{MLE}$		$\mathcal{L}_{MLE+MAE}$		$\mathcal{L}_{\alpha}$		$\mathcal{L}_{\alpha+MAE}$		$\mathcal{L}_{CE}$		$\mathcal{L}_{CE+MAE}$	
Arabic	**87.70**	**84.83**	86.75	83.44	86.60	82.96	86.16	82.93	85.81	82.65	86.01	83.14	86.82	82.77
Basque	**89.03**	**87.45**	88.01	86.08	87.75	86.17	87.49	85.97	87.57	86.05	87.14	85.92	87.61	86.03
Bulgarian	**94.97**	92.44	94.06	92.06	94.58	92.06	93.97	91.97	94.18	91.54	94.22	92.12	94.51	**92.45**
Chinese	**86.73**	**86.16**	85.51	84.54	86.29	84.75	85.92	84.82	85.79	85.11	85.95	85.23	85.79	85.22
Czech	92.43	90.92	93.18	92.04	93.10	91.72	93.40	91.54	93.24	91.97	93.29	91.64	**93.42**	**92.12**
Danish	**88.23**	**87.70**	87.08	86.31	87.51	86.40	87.11	86.45	87.26	86.50	87.08	86.55	87.91	87.30
English	**91.21**	**90.15**	90.71	89.37	90.81	89.57	90.82	89.66	90.81	89.63	90.78	89.69	90.87	89.86
Finnish	**91.55**	**91.29**	90.54	90.06	90.51	90.24	90.14	89.60	90.32	89.73	90.55	90.02	90.85	90.75
Hebrew	**91.00**	**90.28**	90.07	89.14	90.11	88.90	89.14	88.46	89.16	88.94	89.50	88.95	90.29	89.51
Hindi	96.78	94.80	96.55	94.62	96.55	94.35	96.65	94.44	96.40	94.58	96.40	94.58	**96.81**	**94.69**
Italian	**94.14**	**92.98**	93.56	92.62	93.35	92.27	93.21	92.49	93.27	92.34	93.64	92.26	93.30	92.51
Japanese	**96.03**	**95.69**	95.43	94.94	95.41	94.83	95.14	95.16	95.14	94.87	94.77	94.97	95.53	95.50
Korean	**88.84**	**86.75**	87.80	86.24	87.98	85.78	87.59	85.66	87.73	85.97	87.03	85.72	88.28	86.40
Russian	**93.86**	**93.22**	93.16	92.61	93.37	92.17	93.06	91.80	92.97	91.96	93.50	91.99	92.93	92.61
Swedish	**91.52**	**90.16**	90.06	89.07	90.52	89.13	90.85	89.74	90.61	89.33	90.74	89.67	91.12	89.58
Turkish	73.14	**70.25**	72.69	69.16	72.85	67.80	72.18	68.48	72.61	69.14	70.34	68.15	**73.45**	67.34
Average	**90.45**	**89.07**	89.70	88.29	89.83	88.07	89.55	88.07	89.56	88.04	89.43	88.16	90.18	88.42

Table 2: UAS (left) and LAS (right) on the test set of 16 treebanks in the UD dataset, plus the macro averages over all treebanks. We mark the highest performing system for both UAS and LAS in **bold** and second highest in blue.

	d_1	d_1 [+ $\mathcal{L}_{MAE}$]
$\mathcal{L}_{MLE}$	0.72 ± 0.03	0.85 ± 0.02
$\mathcal{L}_{CE}$	0.89 ± 0.03	0.86 ± 0.05
$\mathcal{L}_{\alpha}$	0.87 ± 0.03	0.89 ± 0.03

Table 3: Average DSpr over all (test) treebanks.

different languages for all geometric losses. Overall, $\mathcal{L}_{CE+MAE}$ is our best performing model and the average UAS/LAS across languages are on par with the Biaffine parser in spite of only having a single representation for each token. Moreover, it achieves top performance on Czech, Hindi and Turkish. All other geometric losses also achieve competitive results that are within 1% of the Biaffine parser for both UAS and LAS.

We also report the mean DSpr along with standard deviation across the 16 treebanks in Table 3. Unlike the UAS and LAS we find a substantial difference in the DSpr score for the loss $\mathcal{L}_{MLE}$. When combined with the auxiliary loss $\mathcal{L}_{MAE}$ we find a pronounced increase in DSpr, this agrees with the findings in § 5.1.

5.3 Parsing Errors w.r.t. Tree Relationships

Inspired by the geometric structure of tree embeddings, we investigate the sources of errors in terms of ground-truth dependency trees. Given a sentence s and a dependency tree T_s, we define the type of relation between a pair of token (w_i, w_j) by a 2-tuple consisting the distance $d_{T_s}(w_i, w_j)$ and depth difference $\Delta_s(w_i, w_j) = d_{T_s}(w_0, w_i) - d_{T_s}(w_0, w_j)$. This definition follows naturally from the geometric interpretation of trees: for a node w_i, (1, -1) defines its children, (2, 0) defines its sisters, and (2, 2) defines its grandparents.

To visualize the distribution of errors, for each trained model, we plot the percentage of wrong edges for each relation type on the development set of English-EWT. We show an example plot in Figure 3 for our best $\mathcal{L}_{CE+MAE}$, with results for other losses provided in Appendix F.

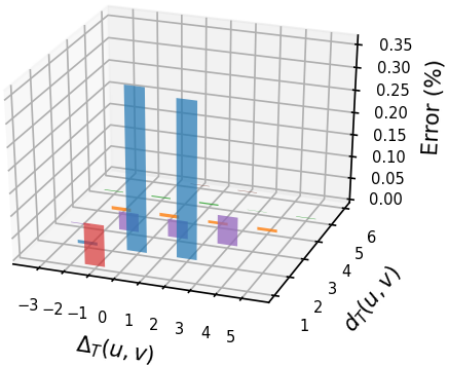


Figure 3: Distribution of edge errors on the development portion of English-EWT, optimized with $\mathcal{L}_{CE+MAE}$; the two axes are $(d_T(i, v), \Delta_T(u, v))$.

Surprisingly, we do not identify long distance ambiguities as a major source of errors (i.e., 99.8% of UAS errors have incorrectly assigned a head node that is within a tree distance of 5 from the dependent). Moreover, we find that sisters and grandparents account for 36.2% and 34.8% of all the UAS errors, respectively. To put these results into context, we construct a synthetic random error distribution: for each tree in the English-EWT development set we generate all trees with a single attachment error. We observe 81.6% of the errors

to be local (up to distance 5) and sisters account for 22.5% and grandparents only account for 5.4% of the errors. We further compare with the biaffine parser and find 99.5% of UAS errors are local with 40.2% for sister and 32.7% for grandparent errors. Therefore parsing errors for a trained parser are dominated by errors that are more local in terms of tree distance than expected from a uniform error distribution. One immediate question that may arise is how can we reduce these specific high-frequency errors. One intuitive extension of the current work is to modify the formulation of edge scores in Eqn. (12) to push the decision boundary away from sisters or grandparents during inference. By training to explicitly model the geometry of the tree, our approach is one step closer towards addressing specific high-frequency errors.

6 Conclusions

In this work, we propose to use the geometry of (dependency) tree structures to construct a neural dependency parser that improves the interpretability of the learned representations without compromising parsing performance. We propose several geometric loss functions, and show that for a majority of them, our simple network learns distance-preserving embeddings through end-to-end training. In doing so, we also compare squared Euclidean distance (d_2^2) with the distance obtained from ℓ_1-norm (d_1), as the (semi-)metric in the embedding space $\mathbb{R}_m$, and provide empirical evidence for using the proper d_1 metric. We compare our results with a competitive and widely-used graph-parser proposed by Qi et al. (2018) on 16 languages from different families, and show overall parser performances that are on par with it. Our experiments also suggest a new way of looking at the sources of parsing errors in terms of tree distances, and show that the majority of errors are local (e.g., sisters or grandparents).

For future work, we suggest looking at potential ways to correct such high-frequency head prediction errors, defined by their relationship within a tree. Another interesting direction that's worth exploring is to use the continuous tree distances predicted by our methods as features for downstream tasks instead of the discrete tree structures produced by conventional parsers. As recent work has been exploring, this differentiable representation of tree structure is potentially useful within the iterative-refinement framework (Mohammadshahi and Henderson, 2020), or as additional tree-specific positional features in a transformer (Omote et al., 2019).

Acknowledgments

We thank three anonymous reviewers for their useful comments. Research was conducted at Samsung AI Centre Toronto and funded by Samsung Research, Samsung Electronics Co.,Ltd

References

Asuman Aksoy, Mehmet Kili, and Sahin Koak. 2020. Isometric embeddings of finite metric trees into $(\mathbf{R}^n, d_1)$ and $(\mathbf{R}^n, d_\infty)$. *arXiv preprint arXiv:2002.00062*.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Z. Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25:131–160.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yuan Ding and Martha Palmer. 2004. Synchronous dependency insertion grammars: A grammar formalism for syntax based statistical MT. In *Proceedings of the Workshop on Recent Advances in Dependency Grammar*, pages 90–97, Geneva, Switzerland. COLING.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency

parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Kadri Hacioglu. 2004. Semantic role labeling using dependency trees. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1273–1276, Geneva, Switzerland. COLING.

Rowan Hall Maudslay, Josef Valvoda, Tiago Pimentel, Adina Williams, and Ryan Cotterell. 2020. A tale of a probe and a parser. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7389–7395, Online. Association for Computational Linguistics.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2061–2071, Melbourne, Australia. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Richard A Hudson. 1984. *Word grammar*. Blackwell Oxford.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 178–181, Barcelona, Spain. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2755–2768, Hong Kong, China. Association for Computational Linguistics.

Nathan Linial, Eran London, and Yuri Rabinovich. 1995. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245.

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, Copenhagen, Denmark. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Igor Melcuk. 2003. Levels of dependency in linguistic description: Concepts and problems. *Dependency and Valency. An International Handbook of Contemporary Research*, 1:188–229.

Arul A Menezes, Christopher B Quirk, and Colin A Cherry. 2010. Machine translation system incorporating syntactic dependency treelets into a statistical framework. US Patent 7,698,124.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119. Curran Associates, Inc.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Alireza Mohammadshahi and James Henderson. 2020. Recursive non-autoregressive graph-to-graph transformer for dependency parsing with iterative refinement. *Transactions of the Association for Computational Linguistics(under submission)*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Yutaro Omote, Akihiro Tamura, and Takashi Ninomiya. 2019. Dependency-based relative positional encod-

ing for transformer NMT. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 854–861, Varna, Bulgaria. INCOMA Ltd.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D Manning. 2018. Universal dependency parsing from scratch. In *CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *International Conference on Learning Representations*.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. In *Advances in Neural Information Processing Systems*, volume 32, pages 8594–8603. Curran Associates, Inc.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, volume 29, pages 1857–1865. Curran Associates, Inc.

Rupesh Kumar Srivastava, Klaus Greff, , and Jürgen Schmidhuber. 2015. Highway networks. In *Proceedings of the Deep Learning Workshop at the International Conference on Machine Learning*.

Lucien Tesnière. 1959. *Eléments de Syntaxe Structurale*. Klincksieck, Paris.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. 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, Brussels, Belgium. Association for Computational Linguistics.

A Hyperparameters

We use 2 layers of MLP with leaky-relu as activation to map the biLSTM outputs into a 800-dimensional embedding space[10]. The layer weights are initialized with the values from uniform distribution $\mathcal{U}(-0.05, 0.05)$, and biases are initialized to zero.

We train the models with Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 1e-8$ for up to $50,000$ iterations, where each iteration is a batch of up to 5000 tokens or the maximum number of tokens we can fit in the GPU memory. We evaluate the models every 100 steps and save them only if we see improvement in UAS on development data. We switch to AMSGrad (Reddi et al., 2018) after 3000 iterations with no observed improvement on development set UAS, at which point we terminate training when another 3000 iterations pass without improving development set UAS.

B Example d_2^2 and d_1 Tree Embeddings

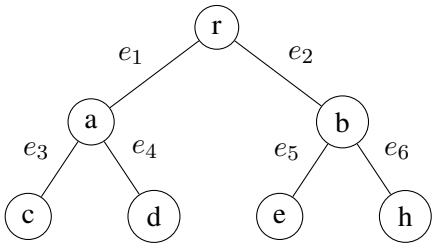


Figure 4: Simple binary tree example.

Let us take as a working example the binary tree on Figure 4. The embedding of node v is the embedding of the root r plus the sum of all direction vectors on the path $\rho(r, v)$, from r to v. If can take $f(r)$ to be $\mathbf{0}$ or any random vector then embedding of all nodes in the tree are by definition:

$$
\begin{aligned}
f(a) &= f(r) + \mathbf{u}_1 \\
f(c) &= f(r) + \mathbf{u}_1 + \mathbf{u}_3 \\
f(d) &= f(r) + \mathbf{u}_1 + \mathbf{u}_4 \\
f(b) &= f(r) + \mathbf{u}_2 \\
f(e) &= f(r) + \mathbf{u}_2 + \mathbf{u}_5 \\
f(h) &= f(r) + \mathbf{u}_2 + \mathbf{u}_6
\end{aligned}
$$

[10]The maximum length of any sentence in the dataset is smaller than 800 and thus an isometric tree embeddings are feasible using both d_2^2 and d_1

We took an arbitrary root embedding $f(r)$ and $n-1$ orthogonal unit-length vectors $\mathbf{u}_i$. If the $\mathbf{u}_i$-s are the standard unit basis vectors, for example, then the tree is embedded on the edges of the unit cube and it is an isometric embedding for both d_2^2 and d_1.

C Proof of Thm. 2.1, Sufficiency

Proof. **Eqn. (3) sufficient.** We first show that an embedding of the form given in Eqn. (3) necessarily satisfies (2). This is Thm. 1 in Reif et al. (2019) and here we find it useful to expand on their proof to introduce notation and assist the reader.

Let $T = (V, E, r)$, $\mathbf{Z}$ and $b(r, u)$ be as described in § 2. Then Eqn. (3) is simply

$$
f(v) = f(r) + \sum_{\{i \in \rho(r,v)\}} \mathbf{z}_i, \tag{26}
$$

where the notation $\{i \in \rho(r, v)\}$ is short for $\{i \,|\, e_i \in \rho(r, v)\}$, which is the set of i's for which $b_i(r, v) = 1$. Consider any two vertices v and w. Notice the two paths $\rho(r, v)$ and $\rho(r, w)$ must share a common prefix, namely $\rho(r, a)$, the sub-path from r to the lowest common ancestor a of v and w, with the remaining paths $\rho(a, v)$ and $\rho(a, w)$ being edge disjoint. Therefore

$$
\begin{aligned}
&||f(v) - f(w)||_2^2 \\
&= || \sum_{\{i \in \rho(r,v)\}} \mathbf{z}_i - \sum_{\{j \in \rho(r,v))\}} \mathbf{z}_j ||_2^2 && (27) \\
&= || \sum_{\{i \in \rho(a,v)\}} \mathbf{z}_i - \sum_{\{j \in \rho(a,w)\}} \mathbf{z}_j ||_2^2 && (28) \\
&= |\rho(a, v)| + |\rho(a, w)| && (29) \\
&= |\rho(v, w)| \equiv d_T(v, w). && (30)
\end{aligned}
$$

Here we canceled the common term $f(r)$ and common prefix edges to derive Eqn. (27) and (28). Eqn. (29) follows from the fact that paths $\rho(a, v)$ and $\rho(a, w)$ are edge disjoint and the orthonormality of the $\mathbf{z}_i$'s. Here $|\rho(a, v)|$ denotes the number of edges on the a to v path. Finally, (30) follows since a is the least common ancestor of v and w.

□

D Proof of Thm. 2.1, Necessity

We provide a proofs that any isometric d_2^2 embedding must have the form defined in (3) that relies only on linear algebra and may provide the reader with additional intuition about the construction of d_2^2 embeddings.

Proof. First, for the case $|V| \leq m+1$, we use induction to show that any isometric d_2^2 embedding f must have the form described in Eqn. (3). Then we show that for $|V| > m+1$, such an f cannot exist.

We use the notation $V = \{v_j \mid 0 \leq j < k\}$, where $v_0 = r$, $|V| = k$ and $|E| = k-1$. For $2 \leq k \leq m+1$ define the the $|E| \times |E|$ path matrix B to be

$$\mathbf{B} = (\mathbf{b}(r, v_1), \ldots, \mathbf{b}(r, v_{k-1})), \tag{31}$$

where $\mathbf{b}(r, v)$ are defined to be the binary path vectors used in (3).

We use induction to prove that if $f : V \to \mathbb{R}^m$ is an isometric embedding for any $k \leq m+1$ then (3) must hold and $\mathbf{B}$ must be full rank. Note this statement is trivially true for $k = 1$ and 2.

Let $k \geq 2$ and $k \leq m$. Let $T = (V, E, r)$ be a tree of size $|V| = k$. Consider the induction hypothesis that any isometric embedding (using d_2^2) of T must have the form in (3) and, moreover, the matrix $\mathbf{B}$ in (31) has full rank. We next show that it follows the same is true for any tree of size $k+1$.

Suppose $T' = (V', E', r)$ is a rooted tree with $|V'| = k+1 \leq m+1$. Suppose $f : V' \to \mathbb{R}^m$ is an isometric embedding using d_2^2. Let $c \neq r$ be a leaf and p be its parent in T'[11], and consider the subtree $T = (V, E, r)$ formed by removing the vertex c and the edge (c, p) from the tree. So $|V| = k$, and the restriction of f to V must provide an isometric embedding of this subtree T. By the induction hypothesis this embedding f, when restricted to V, must satisfy (3) and have the form in (3). Moreover the path matrix $\mathbf{B}$ in (31) for T must be full rank.

Since the full embedding f of T' (including the new vertex c) is isometric it follows that

$$\begin{aligned} ||f(c) - f(v)||_2^2 &= d_T(c, v), \\ &= d_T(p, v) + 1, \ \forall v \in V. \end{aligned} \tag{32}$$

Specifically, when $v = p$ this constrains the embedded displacement for the (p, c) edge, namely $\mathbf{z} := f(c) - f(p)$, to be length 1. Moreover, for any $v \in V' \setminus \{p, c\}$, consider the triangle with vertices $f(c)$, $f(p)$, and $f(v)$. Then (32) states that the squared Euclidean distances of the triangle edges $(f(c), f(p))$, $(f(p), f(v))$ and $(f(c), f(v))$, must be 1, $d_T(p, v)$, and $d_T(p, v) + 1$, respectively. By the Pythagorean Theorem, this must be a right triangle with

$$\mathbf{z}^T(f(p) - f(v)) = 0, \forall v \in V' \setminus \{p, c\}.$$

By using Eqn. (3) on the subtree T this can be rewritten as

$$\mathbf{z}^T \mathbf{Z}(\mathbf{b}(r, p) - \mathbf{b}(r, v)) = 0, \forall v \in V' \setminus \{p, c\}. \tag{33}$$

By setting $v = r$ in (33) and using $\mathbf{b}(r, r) = \mathbf{0}$ (i.e., null vector in $\mathbb{Z}^{|E|}$) we find $\mathbf{z}^T \mathbf{Z} \mathbf{b}(r, p) = 0$. So (33) is equivalent to

$$\mathbf{z}^T \mathbf{Z} \mathbf{b}(r, v) = 0, \forall v \in V' \setminus \{c\}. \tag{34}$$

That is $z^T \mathbf{Z} B = 0$. However, by the induction hypothesis, the $(k-1) \times (k-1)$ matrix $\mathbf{B}$ (for T) is full rank, and so we must have $\mathbf{z}^T \mathbf{Z} = \mathbf{0}$. That is, the direction of the edge (p, c) in the embedding space, $\mathbf{z} = f(c) - f(p)$, must be perpendicular to all the other edges in the tree T' and have unit length, so

$$\begin{aligned} \mathbf{z}^T \mathbf{Z} &= \mathbf{0}, \\ ||\mathbf{z}||_2 &= 1. \end{aligned} \tag{35}$$

Since $\mathbf{Z}$ is a $m \times (k-1)$ matrix with $k \leq m$ it follows that such $\mathbf{z}$ must exist.

Note the path vector $\mathbf{b}(r, c)$, in the tree T', is just the path vector for the parent, $\mathbf{b}(r, p)$, modified to have a 1 in the k^{th} row, while all other path vectors $\mathbf{b}(r, v_j)$ must have zero in this k^{th} row (since c is a leaf). The path matrix $\mathbf{B}'$ for T' therefore has the form

$$\mathbf{B}' = \begin{pmatrix} \mathbf{B} & \mathbf{b}(r, p) \\ \mathbf{0} & \mathbf{1} \end{pmatrix}. \tag{36}$$

Since the induction hypothesis ensures that $\mathbf{B}$ has full rank, it follows that $\mathbf{B}'$ has full rank. Finally, defining $\mathbf{Z}' = (\mathbf{Z}\mathbf{z}) \in \mathbb{R}^{m \times k}$, it follows from (35) that $(\mathbf{Z}')^T \mathbf{Z}' = \mathbb{1}$ and therefore we have shown that f must have the form in (3) with this $\mathbf{Z}'$ and these path vectors $\mathbf{b}(r, v)$. This completes the proof of the induction step, the desired result follows by induction.

To show that there is no embedding for $|V'| \geq m+2$ it is sufficient to show that there is no embedding for $|V'| = m+2$. We use contradiction. Suppose $f : V' \to \mathbb{R}^{\mathfrak{m}}$ is such an embedding. We can proceed as above, with c a leaf node of $T' = (V', E', r)$, and p its parent. Then, by using the previous result, any isometric embedding of the subtree $T = (V, E, r)$ that is formed by removing

[11] Note $c \neq r$ is possible since $k \geq 2$

this child (so $|V| = m + 1$) must have the form described in (3) with a full rank $m \times m$ path matrix $\mathbf{B}$ and orthonormal $m \times m$ matrix $\mathbf{Z}$. The same line of reasoning then shows that $z := f(c) - f(p)$ must satisfy (35). But here $\mathbf{Z}$ is full rank and so $z = 0$, which contradicts the constraint that the (c, p) edge in the embedding must have length 1. □

E Distance plots for learned embedding

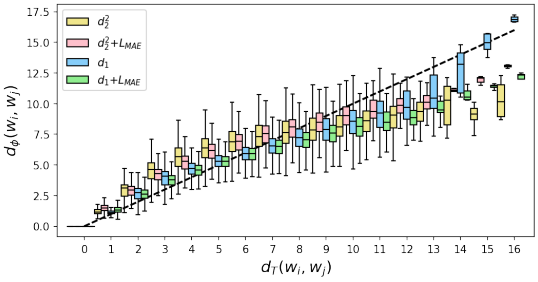


Figure 5: Distance distribution for triplet loss in d_2^2, d_1 and d_1 with MAE on dev set of English-EWT.

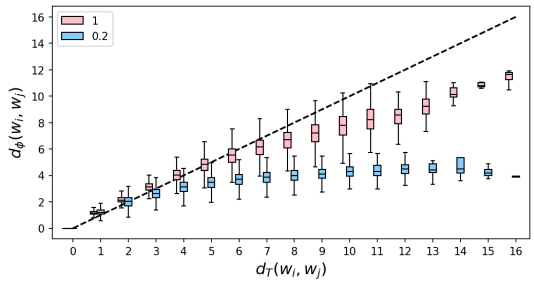
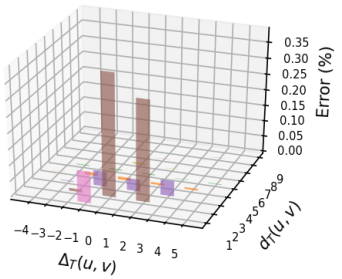
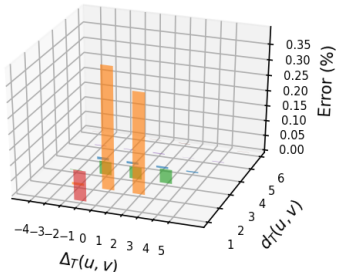


Figure 6: Distance distribution for CE loss using d_1 for different softmax temperature on dev set of English-EWT.

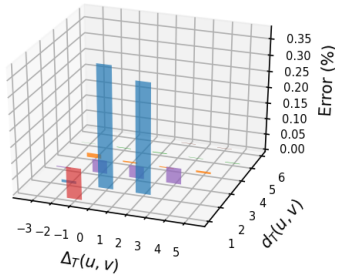
F Error distribution trained with different losses



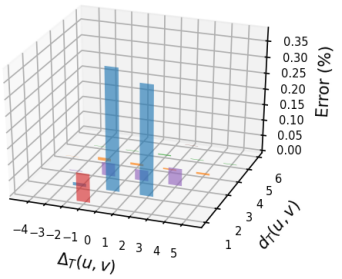
(a)



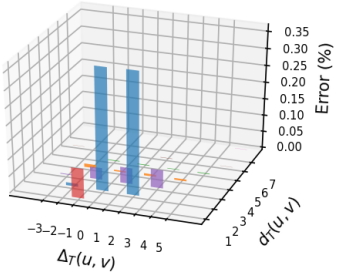
(b)



(c)



(d)



(e)

Figure 7: Distribution of edge errors for (a) $\mathcal{L}_{MLE}$ (b) $\mathcal{L}_{MLE+MAE}$ (c) $\mathcal{L}_{\alpha}$ (d) $\mathcal{L}_{\alpha+MAE}$ and (e) $\mathcal{L}_{CE}$ on dev set of English-EWT.