Know Yourself and Know Your Neighbour: A Syntactically Informed Self-Supervised Compositional Sentence Representation Learning Framework using a Recursive Hypernetwork

Anonymous authors
Paper under double-blind review

Abstract

~~In this work, we propose a syntactically informed self-supervised learning framework for generating sentence representations. In our framework, we train a recursive hypernetwork to compose sentence representation from any word-level representation by using a set of newly proposed self-supervised tasks. We verify that the newly proposed framework can generate sentence representations that encode more linguistic information than state-of-the-art sentence representations and verify the stability and adaptability of our model.~~ Sentence representation learning is still an open challenge in Natural Language Processing. In this work, we propose a new self-supervised framework for learning sentence representations, using a special type of neural network called a recursive hypernetwork. Our proposed model composes the representation of a sentence from representations of words by applying a recursive composition through the parse tree. We maintain a separate syntactic and semantic representation, and the semantic composition is guided by the information from the syntactic representation. To train this model, we introduce a novel set of six self-supervised tasks. We validate that the generated sentence representation encodes richer linguistic information than both averaging baselines and state-of-the-art alternatives. Furthermore, we assess the impact of the proposed tasks through ablation studies. We also demonstrate that the representations generated by our model are stable for sentences of varying length and that the model is adaptable to different syntactic categories.

1 Introduction

Most neural language models generate vector representations (embeddings) of words or tokens. For example, earlier models such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) generate representations of words, and more recent models such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) generate contextual representations of each token in a sentence. Most recent large language models, such as GPT (Brown et al., 2020) and Llama (Touvron et al., 2023), also internally generate representations of the tokens in the prompt for text generation. However, many natural language processing tasks require representations of sentences.

The standard way to create a sentence representation is to average the embeddings of the tokens in the sentence. However, this averaging ignores syntactic information and the position of tokens in a sentence. Although Transformer-based language models internally encode some syntactic information (Raganato & Tiedemann, 2018; Hewitt & Manning, 2019; Clark et al., 2019; Reif et al., 2019; Jawahar et al., 2019; Lin et al., 2019; Manning et al., 2020; Arps et al., 2022; Pimentel et al., 2022), such internal syntactic information is limited (Sinha et al., 2021b; Pham et al., 2021; Sinha et al., 2021a; Nedumpozhimana & Kelleher, 2024), and there is a growing body of work across a range of NLP tasks that points to the benefits of explicitly injecting syntactic information into transformer models (Pang et al., 2019; Moradshahi et al., 2019; Min et al., 2020; Wang et al., 2020; Pham et al., 2021; Sachan et al., 2021; Xu et al., 2021b; Tian et al., 2022; Hou et al., 2024; Kai et al., 2024).

Muennighoff et al. (2023) introduced the Massive Text Embedding (MTEB) benchmark and evaluated a range of embedding models across a variety of tasks. Their results indicate that no single embedding method obtains the best performance across all tasks, and when performance is averaged across tasks, supervised methods, in which at least one supervised task (such as NLI) with labelled dataset is used to train the sentence representation model, outperform pure ~~unsupervised~~ self-supervised methods. However, such supervised methods also have larger variance in performance across tasks. It has long been known that in deep networks the features learned in later layers are fitted to the training task and less transferable across tasks, see e.g. (Yosinski et al., 2014). Consequently, the greater variance in performance across tasks for supervised methods may be caused by their representations being overfitted to the learning task e.g., NLI task. Indeed, Carlsson et al. (2021) argue that self-supervised methods for sentence representations are preferable to supervised because they avoid the pre-training objective bias, and do not require labelled datasets.

Murty et al. (2023) observe that Transformer-based models trained in a supervised manner on compositional generalisation benchmarks become more tree-like (syntactic) in their processing over the course of training, and the more tree-like the processing of a model, the better its performance on tests of compositional generalizability. Moreover, they find that models trained in a self-supervised manner do not learn tree-structured processing. This suggests that one way to improve the performance of self-supervised methods for generating sentence representations is to improve the encoding of syntax within the generated embeddings.

We propose a novel self-supervised framework for generating syntactically informed sentence representations using a recursive hypernetwork trained. This framework has two major components: (1) a recursive hypernetwork architecture that implements a pair of trainable compositional operators, one semantic and one syntactic, and (2) a set of 6 self-supervised learning tasks used to train the compositional operators. The framework explicitly infuses syntactic information in two ways. First, the order in which embeddings are composed is defined by the constituency parse tree of the sentence being processed. Second, the self-supervised tasks are designed to guide the compositional operators to encode syntactic information as well as distributional semantic information in the embeddings they generate.

The rest of the paper is structured as follows. Section 2 presents a review of previous work, Section 3 describes the framework, and Section 4 describes the training of the models. Section 5 presents results from a set of probing tasks designed to assess whether the framework improves the encoding of linguistic information when compared with state-of-the-art baselines, and Section 6 reports an analysis of semantic compositional operator in terms of: (a) robustness to parse tree depth, and (b) sensitivity to syntactic information. Finally, Section 7 presents our conclusions and plans for future work.

2 Sentence Representation Learning

There is a growing body of work on generating neural sentence representations; see Kashyap et al. (2023) for a recent survey. The technical novelty of our work arises from combining a recursive hypernetwork with a self-supervised learning objective. Recursive ~~hyper-networks~~ hypernetwork have previously been used to train sentence representations; however, all of this work has been done using supervised learning. In the following two sections, we first review previous research on recursive ~~hyper-networks~~ hypernetwork for sentence representations and then review supervised and self-supervised approaches to sentence representation learning. Although to date supervised approaches tend to outperform unsupervised methods, this may be because unsupervised methods do not explicitly integrate syntactic information into their objectives (see discussion regarding Murty et al. (2023) in Section 1). Consequently, we hypothesise that a self-supervised framework (recursive hypernetwork model+objective) that encourages the encoding of more linguistic information into a representation may result in improved performance. The benefit of improving the performance of self-supervised methods is that it avoids the need for expensive labelled datasets, and may also result in more transferable (as in downstream task-agnostic) representations.

2.1 Recursive Neural Networks

Recursive Neural Networks are a deep learning architecture used to process hierarchical input, particularly the tree-structured input. These networks apply the same set of weights recursively over the structured input in topological order, from leaf to root in the case of a tree. In our work, we choose to use a recursive neural network sentence representation learning because of its ability to explicitly use the syntactic information from a parse tree by composing the representation of sentences (or phrases) recursively.[1] Socher et al. (2011b;a) proposed a recursive neural network with a simple feedforward network as the composition operator, which was later generalised and extended by Socher et al. (2012; 2013a;b) for sentence representation learning. However, these recursive neural networks based on feedforward networks suffer from the vanishing gradient problem when processing sentences with deep tree structures. To address this problem, Tai et al. (2015) adapted the LSTM architecture to recursive networks and proposed a compositional operator called Tree-LSTM. Contemporaneously, and in parallel, Le & Zuidema (2015) and Zhu et al. (2015) proposed similar extensions that use an LSTM over a tree structure.

Kim et al. (2018) modified the Tree-LSTM by augmenting the syntactic tag information as a supplementary input to the gate functions of the Tree-LSTM to compose the representation of each node in the constituency parse tree. The traditional Tree-LSTM is designed to be applied on a binarised constituency parse tree, which can be one of its limitations. To overcome this, Xu et al. (2021a) extended the Tree-LSTM to an ARTree-LSTM, which can be applied to constituency parse trees with any number of child nodes. Liu et al. (2017) extended the Tree-LSTM by exploiting the advantages of hypernetworks or meta-networks. They used a small meta-network and a dynamic parameter prediction method to enable different types of compositions. Shen et al. (2020) extended this work further by explicitly using syntactic (tag) information for the dynamic parameter prediction and improved the performance. Xu et al. (2020) also used a similar combination of Tree-LSTM and hypernetwork for the compositional operator as that of Shen et al. (2020).

In the literature, the recursive neural networks are used to generate task-specific sentence representations. In our work, we use the combination of the hypernetwork with the Tree-LSTM (please see section 3 for more details) to compose sentence representations that avoid the need for separate training for each downstream task. To our knowledge, no one has explored the possibility of using the state-of-the-art recursive neural network (hypernetwork+Tree-LSTM) for composing such a general sentence representation.

2.2 Supervised versus Unsupervised

Some of the earlier models, such as by Hill et al. (2016); Mitchell & Lapata (2010); Mikolov et al. (2013); Arora et al. (2017), use simple non-parametric compositions for generating sentence representation. Apart from such simple baseline models, almost all approaches use some task(s) to learn or fine-tune the model to generate sentence representation, and Hill et al. (2016) observed that such tasks will significantly impact the quality of the model.

A considerable amount of literature has used supervised learning approaches to generate sentence representations specific to target downstream tasks. Such models have to train separately for each of the tasks, which is a limitation of this approach. Therefore, in this work, our focus is on generating representations of sentences that are not fitted to specific downstream tasks. There are some notable models in the literature that utilise the supervised learning approach for generating such sentence representations. For example, Conneau et al. (2017) proposed a supervised model called Infersent for learning universal sentence representation by using labelled Stanford Natural Language Inference (SNLI) datasets, and Wieting et al. (2016) proposed a model which captures paraphrastic similarity by using a paraphrase dataset for learning sentence representation. The Universal Sentence Encoder by Cer et al. (2018) utilised unlabeled data along with the labelled SNLI data to train a Transformer encoder model for generating sentence representation with a multitask learning framework. Reimers & Gurevych (2019) proposed another supervised model called SBERT which is finetuned by using the SNLI labelled dataset after a self-supervised pre-training. Even though these supervised learning tasks empirically show their transfer effect to other tasks, there is no convincing reason

[1] Note that Recursive Neural Networks are not the only model that explicitly utilises the syntactic information from the parse tree. For example, the hypertree neural network proposed by Zhou et al. (2022) learns representations of each node in the constituency parse tree through iterative updates rather than a recursive composition.

observed for their generalizability (Carlsson et al., 2021). Also, such a supervised approach requires highly valuable labelled data, which restricts the scalability of such models. Therefore, generally, self-supervised or unsupervised tasks are preferred over supervised tasks for learning scalable models.

Self-supervised learning is widely used for generating sentence representations. One such model is the sentence representation model proposed by Le & Mikolov (2014), which is trained to predict words in the sentence (or document). The CPHRASE model by Pham et al. (2015) learns the representation by using the task of context predictions for phrases at all levels of the constituency parse tree generated by a supervised parser. The Skipthought model, Kiros et al. (2015) extended the skip-gram model for words (Mikolov et al., 2013) to the sentence level by proposing a task of decoding the next and previous sentences from the encoded sentence representation by using the encoder-decoder architecture. Ba et al. (2016) used the same task and architecture of Skipthought with an additional layer-norm regularisation and improved the performance. Hill et al. (2016) used an additional autoencoder task along with the task used by Kiros et al. (2015). In the same work, Hill et al. (2016) further proposed a Sequential Denoising Autoencoder model for sentence representation learning that has the objective of reconstructing the original sentence from a corrupted (noise-added) input representation.

Logeswaran & Lee (2018) simplified the tasks of Kiros et al. (2015) and used the task of identifying a context sentence from other contrastive sentences using the sentence representation, rather than regenerating the sentence itself. Pagliardini et al. (2018) used a CBOW-like objective (Mikolov et al., 2013) to predict a word within the sentence for composing the sentence representation. In another model called DeCLUTR, based on the concept of contrastive loss, Giorgi et al. (2021) extended the task of Logeswaran & Lee (2018) by allowing overlapping and subsuming context segments along with adjacent segments as positive samples. Carlsson et al. (2021) also proposed a self-supervised training objective called contrastive tension to learn sentence representation from pre-trained language models by removing the bias posed by the pre-training objective and achieved a new state-of-the-art in semantic text similarity tasks. Hu et al. (2022) used another self-supervised task of token prediction by using its left context and right context.

Inspired by various approaches in the literature, in this work, we propose a set of novel self-supervised tasks for learning representations of phrases and sentences, which we will describe in more detail in section 3.1. Some of our proposed tasks have some similarities with the tasks proposed or used by Pham et al. (2015) and Kiros et al. (2015). However, our tasks are novel and have many notable differences from previous tasks, and we will discuss more about this in section 3.1.

3 Proposed Model

The proposed model (shown in Figure 1) consists of two recursive neural networks. One network composes the syntactic embeddings of the tokens in the tree, the other composes the semantic embeddings. Both networks implement a pairwise composition operator. For both networks, the order in which embeddings are combined is defined by the structure of the constituency parse tree of the input sentence. Constituency parse tree represents the grammatical structure of a sentence by breaking it down into hierarchical constituents. For example, the constituent parse tree of the sample sentence '*a sentence is parsed*' first breaks down the sentence into two phrases: '*a sentence*' and '*is parsed*'. Then it breaks down the phrase '*a sentence*' into '*a*' and '*sentence*', and the phrase '*is parsed*' into '*is*' and '*parsed*'. To use the parse tree to sequence the pair-wise compositions, we convert the parse tree into a binary tree at the start of the composition process by introducing auxiliary nodes. Then we compose the representation of each node in the binarised parse tree, i.e., words, phrases, and the sentence, by recursively applying the compositional operator from leaves to the root. In the previous sample sentence, first we compose the representations of phrases '*a sentence*' and '*is parsed*' from representations of leaf nodes (i.e., words): '*a*', '*sentence*', '*is*', and '*parsed*'. Then we compose the representation of the sentence from representations of these phrases. Importantly, each composition performed by the semantic composition network is preceded by the corresponding syntactic composition. This is important because the result of the syntactic composition is fed into the semantic network. Consequently, while the syntactic compositional operator ($\oplus$) is a regular operator with learnable parameters, some of the parameters of the semantic compositional operator ($\otimes$) are generated from the syntactic embeddings created from the syntactic compositions implemented by the recursive syntactic network. ~~Therefore, the~~

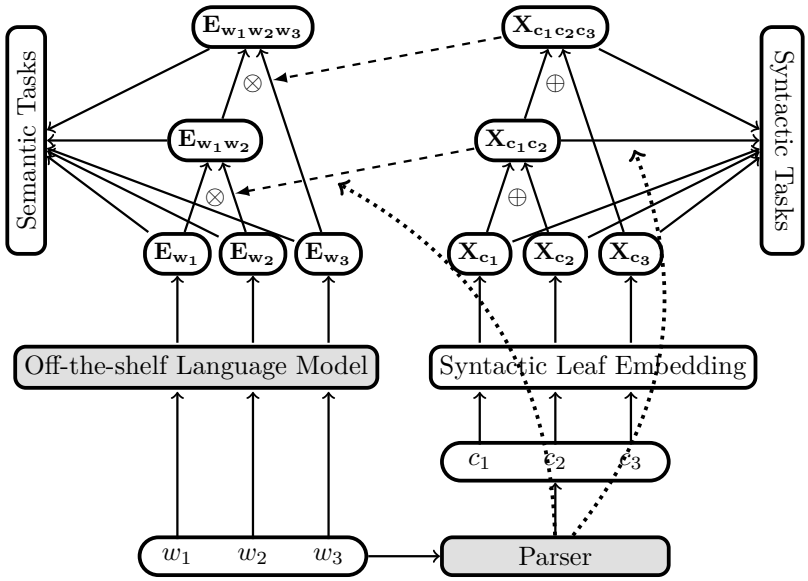

Figure 1: Model Architecture

~~syntactic recursive network acts as the hypernetwork that generates parameters of the semantic recursive network.~~ Generally, hypernetworks are deep learning architectures used to generate parameters of a primary network. In this case, the semantic recursive network acts as the primary network, and the syntactic recursive network acts as the corresponding hypernetwork that generates parameters of the primary network(semantic recursive network).

In a regular recursive network, there will be a unique composition operator to generate the representation of any node in the constituency parse tree. However, one can argue that the composition operator to generate the semantic representation of a noun phrase can be different from the compositional operator to generate the semantic representation of a sentence or a verb. The hypernetwork architecture enables the model to learn different semantic compositional operators for different syntactic categories.

We formally define the syntactic compositional operator $\oplus$ as a binary operator on a dx-dimensional syntactic embedding space and the semantic compositional operator $\otimes$ as a binary operator on a dm-dimensional semantic embedding space.

$$\oplus : \mathbb{R}^{dx} \times \mathbb{R}^{dx} \longrightarrow \mathbb{R}^{dx} \qquad \otimes : \mathbb{R}^{dm} \times \mathbb{R}^{dm} \longrightarrow \mathbb{R}^{dm}$$

Given these operators, the syntactic and semantic representations of a parent node p in a constituency parse tree are generated by the composition of its left child l and its right child r,

$$X_p = \oplus(X_l, X_r) \qquad E_p = \otimes(E_l, E_r; X_p)$$

where X_p, X_l, and X_r are the dx dimensional syntactic embeddings and E_p, E_l, and E_r are the dm dimensional semantic embeddings of p, l, and r.

As an initial basis for our work, we selected a Tag-Guided Hyper-Tree-LSTM compositional operator proposed by Shen et al. (2020), in which the syntactic compositional operator is a regular Tree-LSTM and the semantic compositional operator is a modified Tree-LSTM in which the parameters are generated by the syntactic embedding. The state of each node in the Tree-LSTM is a tuple (h, c) where h is the hidden state and c is the cell state of the node [2]. The regular Tree-LSTM architecture of the syntactic composition operator ($\oplus$) takes the state of both the left child ($X_l = (\bar{h}_l, \bar{c}_l)$) and the right child ($X_r = (\bar{h}_r, \bar{c}_r)$) from the

[2]We can concatenate both h and c to treat it as a single input vector.

syntactic recursive network as its input if the node is not a leaf node. If it is a leaf node, then it will take the syntactic leaf embedding (x) as its input. These two cases can be combined and considered X_l, X_r, and x as the input to the syntactic compositional operator, in which x is set to 0 if the node is not a leaf node, and both X_l and X_r are set to 0 if the node is a leaf node. The syntactic compositional operator outputs the state of the parent ($X_p = (\bar{h}_p, \bar{c}_p)$). In the semantic compositional operator ($\otimes$), along with the state of the left child ($E_l = (h_l, c_l)$), the right child ($E_r = (h_r, c_r)$), and the leaf embedding (e), it takes the state of the parent from the syntactic recursive network ($X_p = (\bar{h}_p, \bar{c}_p)$) and outputs the semantic representation of the parent ($E_p = (h_p, c_p)$). Both the syntactic compositional operator and the semantic compositional operator are mathematically described as:

$$
\begin{aligned}
&\oplus : ((\bar{h}_l, \bar{c}_l), (\bar{h}_r, \bar{c}_r), x) \longrightarrow (\bar{h}_p, \bar{c}_p),\\
&\bar{h}_p = \sigma(\bar{o}_p) \odot tanh(\bar{c}_p),\\
&\bar{c}_p = \sigma(\bar{i}_p) \odot tanh(\bar{u}_p) + \sigma(\bar{f}_l) \odot \bar{c}_l + \sigma(\bar{f}_r) \odot \bar{c}_r,\\
&\bar{i}_p = \bar{V}_{ip} x + \bar{W}^l_{ip} \bar{h}_l + \bar{W}^r_{ip} \bar{h}_r + \bar{b}_{ip},\\
&\bar{f}_l = \bar{V}_{fl} x + \bar{W}^l_{fl} \bar{h}_l + \bar{W}^r_{fl} \bar{h}_r + \bar{b}_{fl},\\
&\bar{f}_r = \bar{V}_{fr} x + \bar{W}^l_{fr} \bar{h}_l + \bar{W}^r_{fr} \bar{h}_r + \bar{b}_{fr},\\
&\bar{o}_p = \bar{V}_{op} x + \bar{W}^l_{op} \bar{h}_l + \bar{W}^r_{op} \bar{h}_r + \bar{b}_{op},\\
&\bar{u}_p = \bar{V}_{up} x + \bar{W}^l_{up} \bar{h}_l + \bar{W}^r_{up} \bar{h}_r + \bar{b}_{up},
\end{aligned}
$$

$$
\begin{aligned}
&\otimes : ((h_l, c_l), (h_r, c_r), e; (\bar{h}_p : \bar{c}_p)) \longrightarrow (h_p, c_p)\\
&h_p = \sigma(o_p) \odot tanh(c_p),\\
&c_p = \sigma(i_p) \odot tanh(u_p) + \sigma(f_l) \odot c_l + \sigma(f_r) \odot c_r,\\
&i_p = z^v_{ip} \odot V_{ip} e + z^w_{ip} \odot (W^l_{ip} h_l + W^r_{ip} h_r) + z^b_{ip},\\
&f_l = z^v_{fl} \odot V_{fl} e + z^w_{fl} \odot (W^l_{fl} h_l + W^r_{fl} h_r) + z^b_{fl},\\
&f_r = z^v_{fr} \odot V_{fr} e + z^w_{fr} \odot (W^l_{fr} h_l + W^r_{fr} h_r) + z^b_{fr},\\
&o_p = z^v_{op} \odot V_{op} e + z^w_{op} \odot (W^l_{op} h_l + W^r_{op} h_r) + z^b_{op},\\
&u_p = z^v_{up} \odot V_{up} e + z^w_{up} \odot (W^l_{up} h_l + W^r_{up} h_r) + z^b_{up},\\
&z^j_i = U^j_i \bar{h}_p + a^j_i
\end{aligned}
$$

Note that, in the semantic compositional operator, we are using a similar set of parameters as in the syntactic compositional operator and then scaling each output with z, and this z is generated from the syntactic embedding of the parent. Through this scaling, we are allowing the model to modify the parameters of the semantic compositional operator depending on the syntactic embedding of the parent.

The model recursively applies the compositional operators on a constituency parse tree of input text, and for that, the embeddings of all leaf nodes are required. These leaf node embeddings can be learned, or we can provide them from off-the-shelf language models. We introduced an embedding layer to learn the embeddings of the syntactic categories of leaf nodes. In the semantic recursive network, we use the embeddings of word tokens generated by off-the-shelf language models such as GloVe or RoBERTa. This allows the semantic recursive network to use the linguistic information from existing language models. Then, to generate the representation of any node p, we concatenate its syntactic embedding (X_p) and semantic embedding (E_p), which results in a $dx + dm$ dimensional vector (note that this concatenation of the syntactic and semantics embeddings is additional to the hypernetwork process whereby the syntactic embedding is used as input to the semantic compositional operators, and so the syntactic embedding is used at two levels in our framework: within the hypernetwork process and in this concatentation operation).

3.1 Proposed Tasks

We train our model on a set of 6 self-supervised tasks that take the embedding (syntactic or semantic) of any node of the parse tree and output the target label, which is generated from the input itself. Tasks that take syntactic embedding as input and output syntax-related targets (for example, the syntactic category of the node) are called syntactic tasks and tasks which take semantic embedding as input and output semantic targets (for example, generating the string that corresponds to the node) are called semantic tasks. All the proposed 6 tasks are listed below.

1. ***self category prediction (SelfCat)***: Predict the syntactic category (e.g., NP, VP, etc.) of a node from its syntactic embedding. This is to encourage the model to encode the category information in its syntactic embedding.

2. ***neighbour category prediction (SibCat)***: Predict the syntactic category of the sibling of a node by using its syntactic embedding. This is to encourage the model to encode the category information of its sibling in its syntactic embedding.

3. ***syntactic position prediction (SynPos)***: Predict whether the node is a left child or right child from its syntactic embedding. This is because the *neighbour category prediction* task does not use the information about the position of the node, and such position information can be useful. This task will force the model to encode the positional information in its syntactic embedding.

4. ***self text generation (SelfStr)***: Generate the text that corresponds to a node from its semantic embedding. This is similar to an auto-encoder (tree auto-encoding), which forces the semantic embedding of each node to encode the information in the corresponding text. Note that this task is different from the Recursive Auto Encoder used by Socher et al. (2011b), in which the representation of the parent is predicted from the representations of the children. In our case, instead of generating representations, we directly generate or decode the string of the node itself. In this text generation task, phrases in the higher levels of the parse tree will have to predict longer sequences. This can be seen as analogous to the decision to consider a longer context window for higher-level phrases in the CPHRASE model by Pham et al. (2015). To justify this decision, they argue that: *"for shorter phrases, narrower contexts are likely to be most informative (e.g., a modifying adjective for a noun), whereas for longer phrases and sentences it might be better to focus on broader "topical" information spread across larger windows"*. This argument justifies our task, which predicts longer sequences from phrases and sentences from higher levels of the parse tree.

5. ***neighbour text generation (SibStr)***: Generate the text corresponding to the sibling node by using the semantic embeddings of a node. This task is to force the model to encode the information about the left or right context of each node in its semantic embedding.

6. ***semantic position prediction (SemPos)***: Predict whether the node is a left child or a right child from its semantic embedding. This is to encode the positional information in the semantic embedding.

Here, our objective is to learn a general (task-agnostic) representation of sentences, and therefore, we excluded supervised downstream tasks for the training. However, if someone wants to use this framework for a specific downstream task, it would be beneficial to fine-tune the model for that particular downstream task.

The task proposed in the Skipthought model (Kiros et al., 2015) and used by many other succeeding models predicts the next and previous sentences. This resembles our task of predicting the neighbouring context. However, in our case, instead of predicting the entire sentences, we predict neighbours of each word, phrase, and sentence in the constituency-parse tree using a recursive neural network and thereby define our context based on the syntactic information from the parse tree. Our tasks are more syntactic and therefore guide the model to learn more syntactic information. The task proposed by the CPHRASE model (Pham et al., 2015) is also similar to our proposed model tasks because both tasks are to predict (or decode) the context from all levels of the parse tree. However, rather than selecting a fixed context window for each node in different levels of the parse tree, our model uses the neighbouring phrase in the parse tree as the context. This enables the model to make sure that the context has a linguistic boundary and that it is syntactically related to the target phrase. Also, instead of predicting words in a context, we generate the context phrase using a decoder as in the Skipthought model (Kiros et al., 2015). Along with the context phrase of each phrase, we generate the phrase itself, and in this way, we integrate the information within a phrase and information from the context of the phrase when training the compositional operators to generate the representation of the phrase.

We train our model using multiple tasks simultaneously, and therefore, it is a multi-task learning problem. For this multi-task learning, we generate and minimise a single aggregate loss from syntactic and semantic tasks. To generate the aggregate loss, we take a weighted sum of each loss from tasks by using learnable weight parameters. To avoid trivial solutions, we followed Liebel & Körner (2018) and added a regularisation

term. The final aggregate loss (L_{agg}) function is shown in the equation 1.

$$L_{agg} = \sum_t \frac{1}{2c_t^2 + \epsilon} L_t + ln(1 + c_t^2) \tag{1}$$

where L_t is the loss and c_t is the learnable weight parameter corresponding to a task t. ϵ is a small positive real constant to make sure that the denominator will not be 0.

4 Training

We experimented with two different off-the-shelf language models–GloVe (Pennington et al., 2014) and RoBERTa (Liu et al., 2019)–as a means of initialising the semantic leaf embedding. In both cases, we used a learnable embedding layer for initialising the syntactic leaf embedding. While initialising the semantic leaf embedding by using these two off-the-shelf language models, we handled some edge cases that are worth noting. In the case of GloVe, some words may not have a pre-generated GloVe embedding. To handle this, we assign a random 300-dimensional embedding for each of such out-of-vocabulary terms. In the case of RoBERTa, instead of word embedding, the model generates representations corresponding to each token. Some words may have multiple tokens (note that a token will not span into multiple words), and in such a situation, we took the average of the token embeddings corresponding to each word to generate the leaf embedding.

We fixed the dimension of the syntactic embedding as 64[3], which resulted in 20K parametered syntactic compositional operators for both the GloVe-based and the RoBERTa-based models. The dimension of the semantic embedding is fixed as 1024[4], which resulted in semantic compositional operators with 3.7M parameters on the GloVe-based model and 4.9M parameters on the RoBERTa-based model. For the classification tasks (4 out of 6), we selected a simple multi-layer perceptron model with one 100-dimensional hidden layer and ReLU hidden layer activation. For the two text generation tasks, we used a simple LSTM-based auto-regressive decoder by setting its hidden dimension and token embedding dimension to the same as the model dimension. This model iteratively generates tokens by using the semantic embedding of the node and the embedding of the previous token as input to the LSTM cell.

For training the model, we used 100,000 sentences with 2,280,900 tokens from the Wikipedia dataset[5]. We parsed each sentence using the *benepar* parser (Kitaev & Klein, 2018), one of the best performing state-of-the-art parsers, to generate the constituency parse tree and syntactic categories. In some cases, the parser may generate multiple parse trees from an input text (e.g., if the input text contains multiple sentences). In such cases, we introduced a new category named SS, and joined all the parse trees with this newly introduced category node to make a single tree for each input. We trained the model for 5 epochs with Adam optimiser (Kingma & Ba, 2014) with learning rate scheduling (initial learning rate 0.001) and a batch size of 8. The model training, excluding parsing and preprocessing, finished within 20 hours for the GloVe-based model and within 26 hours for the RoBERTa-based model on a single A100 GPU.

5 Analysis of Sentence Representation

We first analysed whether the sentence representation generated by our model encodes more linguistic information, as compared with a number of standard sentence encoder models. Probing is one of the standard approaches for investigating whether a representation encodes linguistic information in it. Conneau et al. (2018) proposed a set of 10 standard probing tasks to test the presence of various linguistic information, and we selected these tasks for our analysis.

The first two tasks in the standard probing tasks are the *Sentence Length* (*SentLen*), which is to predict the number of words in the sentence, and the *Word Content* (*WC*), which is to predict the presence of 1000

[3]Shen et al. (2020) used 50 as the dimension of syntactic embedding in their hypernetwork architecture, so we ensured that the dimension we use is greater than 50.

[4]We ensured that the semantic embedding has more dimensions than the initialised leaf embeddings. Specifically, in our experiments, we used 300-dimensional and 768-dimensional leaf embeddings.

[5]enwiki-20240620-pages-articles-multistream.xml.bz2

preselected words in the sentence. These two tasks are to measure the presence of surface-level information about the sentence. The next set of tasks is the ~~*Bigram Shift (Bshift),*~~ *Tree Depth* (*TreeDepth*), and *Top Constituent* (*TC*), which are to measure the presence of syntactic information in the sentence representation. ~~The *Bshift* task is to detect the inverted sentences in which two randomly selected adjacent words are swapped.~~ The *TreeDepth* is the task to predict the depth of the parse tree of the sentence, and the *TC* is the task to predict the categories of top constituents (i.e., sequence of categories of constituents immediately below the root node of the parse tree, e.g., "*ADVP NP VP*"). The final set of tasks–*Past Present* (*PP*), *Subject Number* (*Subj*), *Object Number* (*Obj*), ~~*Semantic Odd Man Out (SOMO),*~~ and *Coordination Inversion* (*CI*)–is to measure the presence of semantic information in the sentence representation. The *PP* is to predict the tense (past or present) of the main clause of the sentence. *Subj* and *Obj* are to predict the number (*NN* or *NNS*) of the subject and the object of the main clause. ~~The *SOMO* is to detect modified sentences in which a random noun or verb in the sentence is replaced with another noun or verb.~~ The final task, *CI*, is to detect sentences in which the order of clauses is inverted. We excluded two probing tasks: *Bigram Shift* and *Semantic Odd Man Out* because datasets of both of these tasks contain 50% non-fluent samples, which can affect the parsing of the sentence. Our sentence representation generation relies on a valid parse tree, and therefore, on both of these datasets, we can not generate a good quality sentence representation by using our model.

We selected Skip-Thought (Kiros et al., 2015), Infersent (Conneau et al., 2017), and SBERT-WK (Wang & Kuo, 2020), as three well-known baseline sentence representation models from the literature that have shown the best performance. Note that within these baseline models, Skip-Thought and Infersent are RNN-based (GRU and BiLSTM) models, and SBERT-WK is a Transformer-based model. In terms of tasks, Skip-Thought is based on a self-supervised task (next and previous sentence prediction), Infersent is based on a supervised task (SNLI), and SBERT-WK is not based on any extra learning task (it generates sentence representation from BERT representation by doing subspace analysis). To include more recent baseline models, we selected two more baselines: the RoBERTa-based supervised SimCSE model (*sup-simcse-roberta-base*) (Gao et al., 2021) and the most recent and recommended version of SBERT model (*all-mpnet-base-v2*) (Reimers & Gurevych, 2019).

We first parsed all sample sentences from all the datasets by using *benepar* parser and then generated the sentence representation by using the trained model (representations of the root node, i.e., the concatenation of semantic and syntactic embeddings of the root node). ~~Note that both the *Bshift* and the *SOMO* datasets contain non-fluent samples, which can affect the parsing of the sentence. Our sentence representation generation relies on a valid parse tree, and therefore, on both of these datasets, we can not expect a good quality sentence representation by using our model.~~ We also generated baseline sentence representations by taking the average of all semantic leaf embeddings. Then, for each task, we trained a probing model, a simple MLP model with one hidden layer with 100 nodes and ReLU hidden activation function, with corresponding training samples and reported the accuracy on the corresponding test samples.

Table 1: Accuracy on ~~10~~ 8 standard probing tasks.

	SentLen	TreeDepth	TC	PP	Subj	Obj	CI	WC	Avg	Avg (wo WC)
Skip-thought	0.8603	0.4122	0.8277	**0.9005**	0.8606	0.8355	0.7189	0.7964	0.7765	0.7737
Infersent	0.8425	0.4513	0.7814	0.8802	0.8613	0.8231	0.7034	**0.8974**	0.7801	0.7633
SBERT-WK	0.9240	0.4540	0.7920	0.8888	0.8645	0.8453	0.7187	0.7750	**0.7828**	**0.7839**
SBERT	0.6043	0.2994	0.6302	0.8730	0.7856	0.7673	0.6141	0.6617	0.6545	0.6534
SimCSE	0.5865	0.2889	0.5668	0.8680	0.8157	0.7869	0.5995	0.6077	0.6400	0.6446
Avg GloVe	0.5936	0.3477	0.6553	0.8427	0.7822	0.7467	0.5468	**0.8669**	0.6727	0.6450
Ours+GloVe	**0.9553**	**0.4784**	**0.8756**	**0.8809**	**0.9036**	**0.8794**	**0.6895**	0.4436	**0.7633**	**0.8090**
Avg RoBERTa	0.8213	0.4290	0.7496	0.8759	0.8645	0.8417	0.7322	**0.6260**	0.7425	0.7592
Ours+RoBERTa	**0.9480**	**0.4879**	**0.8712**	**0.8837**	**0.9164**	**0.8867**	**0.7387**	0.4090	**0.7677**	**0.8189**

Our results shown in Table 1 are really promising. Adding our model to GloVe or RoBERTa results in a significant performance improvement over the corresponding baseline GloVe or RoBERTa representations on ~~7 of the 10 tasks, the three exceptions being *Bshift*, *SOMO*, and *WC*. Interestingly, these 3 tasks are distinct in that *Bshift* and *SOMO* have shuffled sentences (shuffled input sentences may affect the parsing, one of the core parts of our model) and the~~ all tasks except *WC*, where *WC* task is a somewhat different from all other tasks, and we will come back to this point.~~, which has 1000 target labels (such a large number of target labels can increase the chance of overfitting)~~. Moreover, our model, in combination with GloVe or RoBERTa, outperforms other sentence representation models in the literature on ~~6 of the 10 tasks (*SentLen*, *TreeDepth*, *TC*, *Subj*, *Obj*, and *CI*).~~ all tasks except *PP* and *WC*. Most interestingly, we note that even in combination with a non-neural language model, GloVe, our model achieves better performance than the RoBERTa averaged baseline and other neural state-of-the-art sentence representation models such as SBERT-WK (Wang & Kuo, 2020) on 5 of the ~~10~~ 8 tasks (*SentLen*, *TreeDepth*, *TC*, *Subj*, *Obj*). This indicates that the sentence representation generated by our model encodes more linguistic information than baseline representations and other state-of-the-art representations. ~~From our analysis, we also observed that the performance of our model is limited if the input sentence is ill-formed or non-fluent and which highlights the dependency of our model on good quality input sentences.~~

Out of 8 probing tasks considered for this analysis, we think WC is somewhat an outlier task. One reason is that the performances of most of the recent language model baselines on WC (62% with RoBERTa average, 61% with BERT average, and 50% with BERT CLS token) are much worse than other older models and very simple bag-of-words baselines (80% with GloVe average and 98% with tf-idf features). Another reason is that WC has a much larger number of target labels (1000) in comparison with other tasks. Such a huge number of target labels will significantly increase the number of parameters in the MLP classifier for this task, and this increases the chance of overfitting[6]. Probably due to this property, we observed a heavy variance in the performance of WC task with changes in the architecture of the MLP classifier. For reporting the accuracy in Table 1, we fixed the architecture, which is the same as that used in the evaluation of other baseline models, and which is not the best accuracy we obtained. This suggests that the reported scores in Table 1 may not be a direct reflection of the linguistic information encoded in the representation. Because of these reasons, we treat WC as an outlier task and therefore we excluded it from our analysis.

5.1 Ablation Study: Impact of Self-supervised Tasks

To study the impact of proposed self-supervised task(s), we conducted an ablation study by systematically excluding some of these task(s) from the training of the model and analysed how much performance drop it resulted in. To measure the performance drop, we evaluated on 7 of the probing tasks, excluding the WC, the outlier task on which our model is not performing well and calculated accuracies. Then we reported the average accuracy drop as the measure of the impact of the excluded self-supervised task(s).

In the first set of ablation studies, we dropped each task in turn and measured the impact of each of the individual self-supervised tasks. The performance drop corresponds to each of the 6 self-supervised tasks are shown in the first 6 rows of Table 2. Our self-supervised tasks can be grouped into two categories: 1. syntactic tasks, i.e., tasks that are based on the syntactic representation (*SynPos*, *SelfCat*, and *SibCat*) and 2. semantic tasks, i.e., tasks that are based on semantic representations (*SemPos*, *SelfStr*, and *SibStr*). The impact of syntactic tasks on learning sentence representation is an interesting question, and to address this question, we did an ablation study by dropping 3 syntactic tasks (*SynPos*, *SelfCat*, and *SibCat*)[7] and reported the corresponding performance drop in the 7th row of Table 2.

From our ablation study, we found a significant performance drop (8.8%) on both GloVe-based and RoBERTa-based models after excluding the self text generation task (*SelfStr*) from the set of 6 tasks. This

[6]Note that, for this evaluation, we trained an MLP classifier for each of the probing tasks in a supervised fashion by using the standard probing task dataset. Such supervised learning with a large number of target labels increases the chance of overfitting.

[7]One can think about dropping the semantic task to measure the impact of semantic tasks on our model. However, if we drop semantic tasks, then the semantic compositional operator, and therefore the entire semantic network, will not be able to learn. This will result in a situation where the information in the initialised leaf embedding from the existing language model will not be used in the final representation. In that situation, the final representation will encode only the category information of the root node, and most likely, this will result in an obvious and drastic performance drop. Such an ablation study will be less interesting, and therefore, we didn't include it.

Table 2: Impact of Self-supervised tasks. Average drop in performance by excluding WC

Task Ablation	GloVe	RoBERTa
$-SynPos$	0.000	0.002
$-SelfCat$	−0.002	0.004
$-SibCat$	−0.001	0.005
$-SemPos$	0.000	0.001
$-SelfStr$	**0.088**	**0.088**
$-SibStr$	−0.001	0.010
$-SynPos, -SelfCat, -SibCat$	0.005	0.008

shows the crucial role of self text generation task for training our model. We found that other tasks alone are not as crucial as self text generation task.

On RoBERTa-based model, we found performance drops by ablating each of the 6 individual self-supervised tasks, and that suggests that all tasks are useful for the training of the RoBERTa-based model. On the contrary, dropping three tasks: *SelfCat*, *SibCat*, and *SibStr* from the training of the GloVe-based model resulted in a very small performance improvement (less than 0.2%). One possible reason is that dropping less useful tasks may enable the model to focus more on other useful tasks, and that can result in a performance improvement. However, given the positive impact of these tasks on the RoBERTa-based model, it is not advisable to drop any of these three tasks for the training of our model.

Our ablation study on syntactic tasks shows a performance drop of 0.5% and 0.8% on GloVe-based and RoBERTa-based models, respectively. Although these results show that the impact of syntactic tasks is not very significant, they validate their usefulness on both GloVe-based and RoBERTa-based models. Note that, although the individual syntactic tasks do not contribute to the training of the GloVe-based model, together these tasks contribute positively.

6 Analysis of Compositional Operator

In our model, we are applying the compositional operator recursively to generate the representations of phrases and sentences. The cumulative effect of this recursive composition may lead to the case where some of the properties of the sentence representation can vanish or explode if the constituency parse tree is deep. To evaluate how robust our model is to such pitfalls, we measured some properties of representations of nodes in the constituency parse tree. Then we analysed how these properties vary with varying heights of the node. Note that the height of a node is the same as the number of times the compositional operator is applied to generate the node's representation, and therefore, if some properties are decreasing or increasing rapidly with the increasing height, this indicates that the property is likely to vanish or explode for deeper constituency trees.

One of the interesting properties of vector representations is their isotropy. Isotropy is the measure of how uniformly the representation vectors are distributed in the embedding space, and it indicates how well the model utilises the embedding space. If we apply composition recursively, the generated representations can become more and more skewed and shrink to smaller and smaller representational subspaces. If the set of all representations lies in a smaller subspace of embedding space or if it is distributed skewedly towards some direction, then it limits its capacity to encode various information. Different approaches to measure the isotropy of representations are used in the literature (Mu et al., 2018; Cai et al., 2021; Rudman et al., 2022). Of these approaches, we use the measure called IsoScore proposed by Rudman et al. (2022) because it has become relatively standard in the literature and has the properties of being mean agnostic and rotation invariant. Another interesting property of vector representations is the norm of the generated representation. If we apply composition recursively, one potential risk is that the composed representations shrink to the origin or explode.

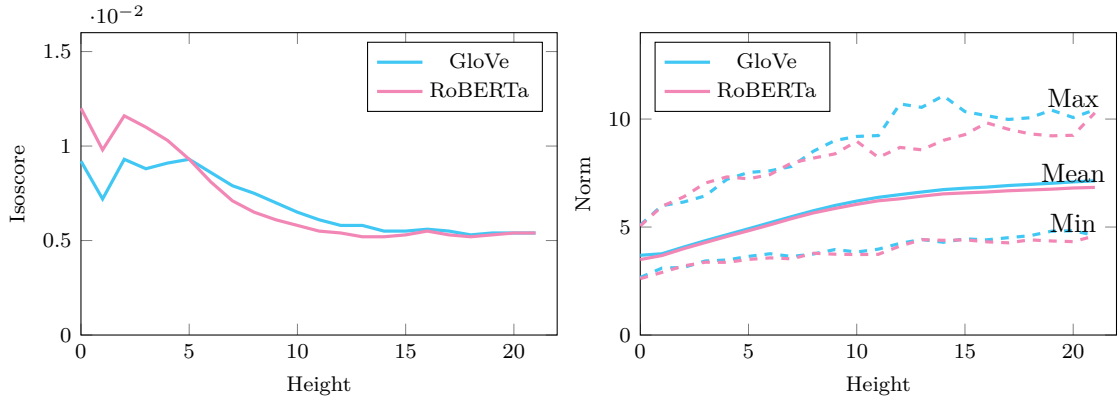

Figure 2: IsoScores and Norms across varying height of nodes

We analysed both the isotropy and the norm of representations of nodes (semantic embedding concatenated with syntactic embedding) with varying height. For the analysis, we selected 10,000 sample sentences from the Wikipedia dataset. To avoid any possible bias of our model towards the dataset used for training, we ensured that none of the samples used for this analysis were from the data used to train our models. The norm (minimum, maximum and mean values) and the isotropy of nodes with varying height are presented in Figure 2. Although there is a slight trend of increasing norm length and decreasing isotropy during the initial compositions, both properties stabilise for higher values of height. This rules out the possible issue of vanishing or exploding values of norm or isotropy on longer sentences (or deeper parse trees).

6.1 Sensitivity of Semantic Composition on Syntactic Embedding

An interesting question is to what extent the semantic composition operation is affected by the syntactic information fed into it via the hypernetwork architecture. To analyse this, we measure the sensitivity of the semantic compositional operator to changes in the syntactic embeddings. If the operator is not sensitive to changes in its syntactic embedding, this would indicate that the hypernetwork architecture does not affect the semantic composition and vice versa.

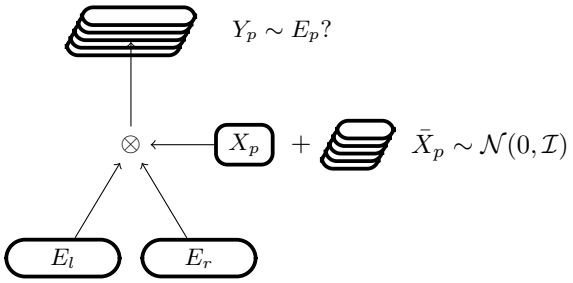

Figure 3: Sensitivity of Semantic Compositional Operator on Syntactic Embedding

To measure the sensitivity of the learned semantic compositional operator to the syntactic embedding it receives as input, we systematically altered the syntactic embedding (X_p) and measured how much change is present in the output of the semantic operator in comparison with its normal output (E_p) when the syntactic embedding is not modified. Our framework for this analysis is shown in Figure 3. The change in the output can be measured either in terms of isotropy (whether these output variants are distributed uniformly across all dimensions?) or in terms of its spread (how close are these output variants to the normal output E_p?). To measure the isotropy, we used the method of IsoScore proposed by Rudman et al. (2022), and to measure the spread, we calculated the average Euclidean distance from the normal output.

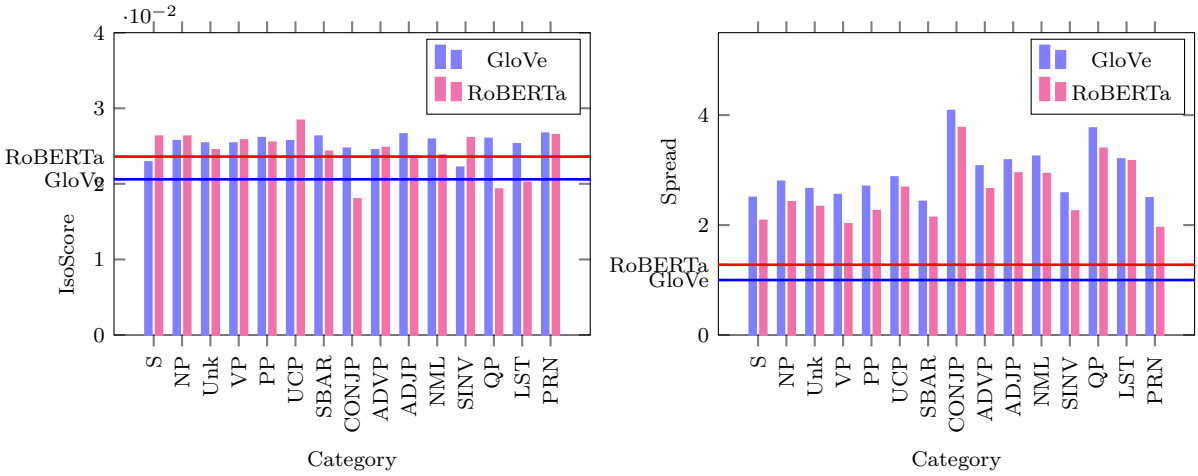

Figure 4: Category-wise IsoScore and Spread of Semantic Compositional Operator to Changes in Syntactic Embedding

For this analysis, we selected all non-leaf nodes from 100 sample sentences from the Wikipedia dataset (ensuring that the selected samples were not used for the training). Then, for each node, we generated 10,000 altered variants of its syntactic embedding (X_p) by adding a random embedding ($\bar{X}_p$) sampled from a multivariate normal distribution with zero mean and identity standard deviation. Then we generated 10,000 semantic embeddings of the node (Y_p) from the semantic embeddings of its left child (E_l) and right child (E_r) by using the semantic compositional operator ($\otimes$) with each of the 10,000 altered variants of the syntactic embedding ($X_p + \bar{X}_p$). Then the IsoScore and spread are calculated from these 10,000 altered variants of the semantic embeddings of the node (Y_p). We averaged the IsoScore and spread of nodes according to the syntactic category of the node.

To get an idea about the goodness of these IsoScores and spreads, and in turn the sensitivity of semantic compositional operator to syntactic embedding, we calculated the baseline IsoScore and the baseline spread values from semantic embeddings (non-perturbed) of all the nodes used for this analysis (total 2162 non-leaf nodes from the selected 100 Wikipedia sample sentences). The spread of semantic embeddings is calculated by taking the average Euclidean distance from its centroid. If the average IsoScore or spread of a category is greater than (or at least comparable to) these baseline scores, then that indicates that, for that category, the semantic compositional operator is sensitive to the changes in the syntactic embedding.

The average IsoScores and spreads of the semantic nodes broken down by their corresponding syntactic categories, along with the baseline values, are plotted in Figure 4. We found that, on both GloVe-based and RoBERTa-based compositional operators, the average spread values of all syntactic categories are notably larger than the baseline spread, and the average IsoScores of almost all categories are larger than the baseline IsoScore. The only exceptions to this are the average IsoScores of CONJP (Conjunction Phrase), QP (Quantifier Phrase; used within NP), and LST (List marker) categories on the RoBERTa-based compositional operator, and even for these three categories, the differences in IsoScores are not very large (Compared to 0.0236, the CONJP has 0.0179, the QP has 0.0192, and the LST has 0.0201 IsoScores). This indicates that, regardless of the syntactic category, the semantic compositional operator is sensitive to changes in syntactic embeddings.

A possible interpretation of these results is that, for less frequent categories (less exposed to the model), the semantic compositional operator can have a lower sensitivity to the syntactic embedding due to insufficient training. To investigate this possibility, we calculated the frequencies of syntactic categories in the training data and calculated the correlations with the IsoScores and spreads. We found that the Pearson correlation coefficient of the IsoScore with the frequency is 0.05931 on the GloVe-based compositional operator and 0.26544 on the RoBERTa-based compositional operator. These correlation scores suggest that, in general, less

frequent categories are less sensitive to the syntactic embedding in terms of IsoScore. While this impact is very small on the GloVe-based operator, the RoBERTa-based operator has shown relatively more impact. The Pearson correlation coefficient of the spread with the frequency of the GloVe-based compositional operator is -0.31309, and the RoBERTa-based compositional operator is -0.32843. This negative correlation implies the less frequent categories have a relatively high spread, even though they have a relatively low IsoScore. In summary, even if the altered variants of semantic embedding of less frequent categories lie in a smaller subspace, they disperse more across this space. This implies that even for an extremely infrequent category, the semantic compositional operator does not become insensitive to the syntactic embedding.

7 Conclusion and Future Work

We proposed a self-supervised recursive hypernetwork architecture to generate sentence representations that encode more linguistic information within the generated embeddings. We validated that the generated sentence representations encode more linguistic information than the standard average-based baselines and state-of-the-art sentence representation models. Our ablation study validated the impact of each of the proposed self-supervised tasks on our model training and ~~O~~ our analysis of the composition operator verified the stability of our framework with increasing length of sentence (more precisely, the depth of the parse tree). We also observed that the hypernetwork architecture has an impact on our framework, and our semantic compositional operator has adaptability depending on the syntactic category of the node.

Our approach for generating sentence representations has a broader impact on the performance of sentence-level tasks, because generating a representation with more linguistic information will likely be useful for downstream tasks. Also, in that case, fine-tuning our framework for specific downstream tasks will likely result in further performance improvement. More importantly, our framework can be used with any language model, and such scalability opens up the potential benefit of our framework with more recent language models for generating better sentence representations that can lead to better performance.

Compared to pretrained language models, our framework is generally lightweight with a few million parameters, and can be trained with a relatively small dataset (we trained the model with 100,000 sentences). Yet, we observed that our model, even with a relatively simple GloVe representation, encodes more linguistic information than RoBERTa's average baseline. This indicates that our model has implications from an efficiency perspective.

One of the limitations of the recursive neural network is its dependence on the constituency parse tree of each input. Sachan et al. (2021) observe that there is a significant performance improvement by using gold standard parse trees instead of trees generated by the off-the-shelf stanza parser. In our model, we used another off-the-shelf parser, called *benepar*, to generate parse trees. To verify the dependence of our model towards the quality of the generated parse tree by using *benepar* parser, we evaluated our model on the *Bigram Shift* and *Semantic Odd Man Out* task from the standard probing task. 50% of the samples in the datasets of these tasks contain non-fluent sentences, and on such samples, the parse tree generated by our parser is expected to be worse. We found that on both of these tasks, our model's performance is not good, and that shows the dependence of our model towards the good-quality parse trees. We haven't explored other parsers for this task, and we are expecting a better performance with a better parser. We propose such exploration for future work.

Another limitation of our work is that we experimented with our model only on the English language. Indeed, our model can be applied to any language; however, the requirement of a good-quality parser can be a major limitation. Maillard et al. (2017) and Hu et al. (2022) learned the parse tree along with the sentence representation by considering all possible branching of the parse tree and selecting the best branching to apply composition. Expanding our experiment to other languages by adopting such techniques to overcome the requirement of the parser is also proposed for future work.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*, 2017. URL `https://api.semanticscholar.`

org/CorpusID:64908139.

David Arps, Younes Samih, Laura Kallmeyer, and Hassan Sajjad. Probing for constituency structure in neural language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6738–6757, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.502. URL https://aclanthology.org/2022.findings-emnlp.502.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xYGNO86OWDH.

Fredrik Carlsson, Amaru Cuba Gyllensten, Evangelia Gogoulou, Erik Ylipää Hellqvist, and Magnus Sahlgren. Semantic re-tuning with contrastive tension. In *International Conference on Learning Representations*, 2021. URL https://api.semanticscholar.org/CorpusID:235613354.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In Eduardo Blanco and Wei Lu (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 169–174, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2029. URL https://aclanthology.org/D18-2029.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1070. URL https://aclanthology.org/D17-1070.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2126–2136, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1198. URL https://www.aclweb.org/anthology/P18-1198.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

John Giorgi, Osvald Nitski, Bo Wang, and Gary Bader. DeCLUTR: Deep contrastive learning for unsupervised textual representations. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 879–895, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.72. URL https://aclanthology.org/2021.acl-long.72.

John Hewitt and Christopher D. Manning. 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)*, pp. 4129–4138, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1419. URL https://www.aclweb.org/anthology/N19-1419.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In Kevin Knight, Ani Nenkova, and Owen Rambow (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1367–1377, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1162. URL https://aclanthology.org/N16-1162.

Zheheng Hou, Sheng Bi, Guilin Qi, Yuanchun Zheng, Zuomin Ren, and Yun Li. Syntax-guided question generation using prompt learning. *Neural Computing and Applications*, 36(12):6271–6282, 2024.

Xiang Hu, Haitao Mi, Liang Li, and Gerard de Melo. Fast-R2D2: A pretrained recursive neural network based on pruned CKY for grammar induction and text representation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2809–2821, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.181. URL https://aclanthology.org/2022.emnlp-main.181.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1356. URL https://aclanthology.org/P19-1356/.

Jushi Kai, Shengyuan Hou, Yusheng Huang, and Zhouhan Lin. Leveraging grammar induction for language understanding and generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4501–4513, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.259. URL https://aclanthology.org/2024.findings-emnlp.259/.

Abhinav Ramesh Kashyap, Thang-Tung Nguyen, Viktor Schlegel, Stefan Winkler, See-Kiong Ng, and Soujanya Poria. A comprehensive survey of sentence representations: From the bert epoch to the chatgpt era and beyond. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2023. URL https://api.semanticscholar.org/CorpusID:258832362.

Taeuk Kim, Jihun Choi, Daniel Edmiston, Sanghwan Bae, and Sang goo Lee. Dynamic compositionality in recursive neural networks with structure-aware tag representations. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://api.semanticscholar.org/CorpusID:52175957.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In C. Cortes, N. Lawrence, D. Lee,

M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/f442d33fa06832082290ad8544a8da27-Paper.pdf.

Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2676–2686, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1249. URL https://www.aclweb.org/anthology/P18-1249.

Phong Le and Willem Zuidema. Compositional distributional semantics with long short term memory. In Martha Palmer, Gemma Boleda, and Paolo Rosso (eds.), *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pp. 10–19, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-1002. URL https://aclanthology.org/S15-1002/.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1188–1196, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/le14.html.

Lukas Liebel and Marco Körner. Auxiliary tasks in multi-task learning, 2018.

Yongjie Lin, Yi Chern Tan, and Robert Frank. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 241–253, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4825. URL https://aclanthology.org/W19-4825.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Dynamic compositional neural networks over tree structure. In *International Joint Conference on Artificial Intelligence*, 2017. URL https://api.semanticscholar.org/CorpusID:24551661.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJvJXZb0W.

Jean Maillard, Stephen Clark, and Dani Yogatama. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *CoRR*, abs/1705.09189, 2017. URL http://arxiv.org/abs/1705.09189.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054, 2020.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Junghyun Min, R Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. Syntactic data augmentation increases robustness to inference heuristics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2339–2352, 2020.

Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8): 1388–1429, 2010. doi: https://doi.org/10.1111/j.1551-6709.2010.01106.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1551-6709.2010.01106.x.

Mehrad Moradshahi, Hamid Palangi, Monica S Lam, Paul Smolensky, and Jianfeng Gao. Hubert untangles bert to improve transfer across nlp tasks. *arXiv preprint arXiv:1910.12647*, 2019.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations, 2018. URL https://arxiv.org/abs/1702.01417.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 2014–2037, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.148. URL https://aclanthology.org/2023.eacl-main.148/.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D Manning. Characterizing intrinsic compositionality in transformers with tree projections. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sAOOeI878Ns.

Vasudevan Nedumpozhimana and John D. Kelleher. Topic aware probing: From sentence length prediction to idiom identification how reliant are neural language models on topic? *Natural Language Processing*, pp. 1–29, 2024. doi: 10.1017/nlp.2024.43.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 528–540, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1049. URL https://aclanthology.org/N18-1049/.

Deric Pang, Lucy H Lin, and Noah A Smith. Improving natural language inference with a pretrained parser. *arXiv preprint arXiv:1909.08217*, 2019.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. Jointly optimizing word representations for lexical and sentential tasks with the C-PHRASE model. In Chengqing Zong and Michael Strube (eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 971–981, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1094. URL https://aclanthology.org/P15-1094.

Thang Pham, Trung Bui, Long Mai, and Anh Nguyen. Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 1145–1160, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.98. URL https://aclanthology.org/2021.findings-acl.98.

Tiago Pimentel, Josef Valvoda, Niklas Stoehr, and Ryan Cotterell. The architectural bottleneck principle, 2022.

Alessandro Raganato and Jörg Tiedemann. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 287–297, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5431. URL https://aclanthology.org/W18-5431.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. Visualizing and measuring the geometry of bert. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*,

volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/159c1ffe5b61b41b3c4d8f4c2150f6c4-Paper.pdf.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

William Rudman, Nate Gillman, Taylor Rayne, and Carsten Eickhoff. IsoScore: Measuring the uniformity of embedding space utilization. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 3325–3339, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.262. URL https://aclanthology.org/2022.findings-acl.262/.

Devendra Sachan, Yuhao Zhang, Peng Qi, and William L. Hamilton. Do syntax trees help pre-trained transformers extract information? In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 2647–2661, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.228. URL https://aclanthology.org/2021.eacl-main.228/.

Gehui Shen, Zhi-Hong Deng, Ting Huang, and Xi Chen. Learning to compose over tree structures via pos tags for sentence representation. *Expert Systems with Applications*, 141:112917, 2020. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2019.112917. URL https://www.sciencedirect.com/science/article/pii/S0957417419306359.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2888–2913, Online and Punta Cana, Dominican Republic, November 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.230. URL https://aclanthology.org/2021.emnlp-main.230.

Koustuv Sinha, Prasanna Parthasarathi, Joelle Pineau, and Adina Williams. UnNatural Language Inference. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7329–7346, Online, August 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.569. URL https://aclanthology.org/2021.acl-long.569.

Richard Socher, Cliff Chiung-Yu Lin, A. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *International Conference on Machine Learning*, 2011a. URL https://api.semanticscholar.org/CorpusID:18690358.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Regina Barzilay and Mark Johnson (eds.), *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 151–161, Edinburgh, Scotland, UK., July 2011b. Association for Computational Linguistics. URL https://aclanthology.org/D11-1014/.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In Jun'ichi Tsujii, James Henderson, and Marius Paşca (eds.), *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1201–1211, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://aclanthology.org/D12-1110/.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In Hinrich Schuetze, Pascale Fung, and Massimo Poesio (eds.), *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 455–465, Sofia, Bulgaria, August 2013a. Association for Computational Linguistics. URL https://aclanthology.org/P13-1045/.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013b. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170/.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In Chengqing Zong and Michael Strube (eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1150. URL https://aclanthology.org/P15-1150/.

Yuanhe Tian, Yan Song, and Fei Xia. Improving relation extraction through syntax-induced pre-training with dependency masking. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 1875–1886, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.147. URL https://aclanthology.org/2022.findings-acl.147/.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Bin Wang and C.-C. Jay Kuo. Sbert-wk: A sentence embedding method by dissecting bert-based word models. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 28:2146–2157, July 2020. ISSN 2329-9290. doi: 10.1109/TASLP.2020.3008390. URL https://doi.org/10.1109/TASLP.2020.3008390.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BJgQ4lSFPH.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.08198.

Chunlin Xu, Hui Wang, Shengli Wu, and Zhiwei Lin. Treelstm with tag-aware hypernetwork for sentence representation. *Neurocomputing*, 434:11–20, 2020. URL https://api.semanticscholar.org/CorpusID:230622737.

Chunlin Xu, Hui Wang, Shengli Wu, and Zhiwei Lin. Tag-enhanced dynamic compositional neural network over arbitrary tree structure for sentence representation. *Expert Systems with Applications*, 181:115182, 2021a. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.115182. URL https://www.sciencedirect.com/science/article/pii/S0957417421006199.

Zenan Xu, Daya Guo, Duyu Tang, Qinliang Su, Linjun Shou, Ming Gong, Wanjun Zhong, Xiaojun Quan, Daxin Jiang, and Nan Duan. Syntax-enhanced pre-trained model. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5412–5422, Online, August 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.420. URL https://aclanthology.org/2021.acl-long.420/.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

Hao Zhou, Gongshen Liu, and Kewei Tu. Improving constituent representation with hypertree neural networks. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1682–1692, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.121. URL https://aclanthology.org/2022.naacl-main.121/.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over tree structures. *ArXiv*, abs/1503.04881, 2015. URL https://api.semanticscholar.org/CorpusID:17629631.