KERMIT: Complementing Transformer Architectures with Encoders of Explicit Syntactic Interpretations

Fabio Massimo Zanzotto(*)
Andrea Santilli(*)
Leonardo Ranaldi(‡,*)
(*) ART Group
Department of Enterprise Engineering
University of Rome Tor Vergata
Viale del Politecnico, 1, 00133 Rome, Italy
fabio.massimo.zanzotto@uniroma2.it

Dario Onorati(*)
Pierfrancesco Tommasino(*)
Francesca Fallucchi(‡)
(‡) Department of Innovation
and Information Engineering
Guglielmo Marconi University
Via Plinio 44, 00193 Rome, Italy
f.fallucchi@unimarconi.it

Abstract

Syntactic parsers have dominated natural language understanding for decades. Yet, their syntactic interpretations are losing centrality in downstream tasks due to the success of large-scale textual representation learners. In this paper, we propose KERMIT (Kernel-inspired Encoder with Recursive Mechanism for Interpretable Trees) to embed symbolic syntactic parse trees into artificial neural networks and to visualize how syntax is used in inference. We experimented with KERMIT paired with two state-of-the-art transformer-based *universal sentence encoders* (BERT and XLNet) and we showed that KERMIT can indeed boost their performance by effectively embedding human-coded universal syntactic representations in neural networks.

1 Introduction

Universal sentence embeddings (Conneau et al., 2018), which are task-independent, distributed sentence representations, are redesigning the way linguistic models in natural language processing are defined. These embeddings are usually created from scratch over large corpora without human supervision (Cho et al., 2014; Kiros et al., 2015; Conneau et al., 2017; Subramanian et al., 2018; Cer et al., 2018) or are crafted with compositional distributional semantics methods (Clark and Pulman, 2007; Mitchell and Lapata, 2008; Baroni and Zamparelli, 2010; Zanzotto et al., 2010).

Traditional task-independent, symbolic, human-defined syntactic interpretations for sentences, which may be referred to as *universal syntactic interpretations*, are losing their centrality in language understanding systems due to the success of transformer-based neural networks (Vaswani et al., 2017) that have boosted performances on a wide variety of linguistic tasks (Devlin et al., 2018; Liu et al., 2019; Yang et al., 2019).

There is evidence that universal sentence embeddings store bits of universal syntactic interpretations. Even if not explicitly designed for encoding syntax, these embeddings implicitly capture syntactic relations among words with different strategies. Transformers (Devlin et al., 2018; Liu et al., 2019; Yang et al., 2019; Dai et al., 2019) seem to capture syntactic relations among words by "focusing the attention". Yet, to be sure that syntax is encoded, many syntactic *probes* (Conneau et al., 2018) for neural networks have been designed to test for specific phenomena (Kovaleva et al., 2019; Jawahar et al., 2019; Hewitt and Manning, 2019; Ettinger, 2019; Goldberg, 2019) or for full syntactic trees (Hewitt and Manning, 2019; Mareček and Rosa, 2019). Indeed, some syntax is correctly encoded in these universal sentence embeddings.

However, universal sentence embeddings encode syntax in a way that is opaque and not so universal. Firstly, and perhaps surprisingly, task-adapted universal sentence embeddings encode syntax better than general universal sentence embeddings (Jawahar et al., 2019). Secondly, even if these embeddings contains syntactic information and may be "just another way in which traditional syntactic models are encoded" (Fodor and Pylyshyn, 1988), there is no clear view on how this information is encoded and, hence, on how syntactic information is *holistically* (Chalmers, 1992) used in inference. Then, it is difficult to envisage ways to symbolically control the behavior of neural networks.

In this paper, we investigate whether explicit *universal syntactic interpretations* can be used to improve state-of-the-art universal sentence embeddings and to create neural network architectures where syntax decisions are less obscure and, thus, syntactically explainable. For this purpose we propose KERMIT, a Kernel-inspired Encoder with a Recursive Mechanism for Interpretable Trees, and KERMIT*viz* . KERMIT is a lightweight en-

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 256–267,
November 16–20, 2020.

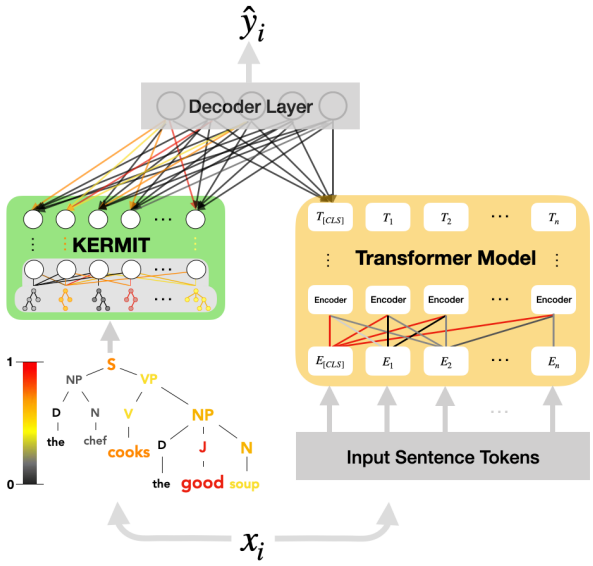


Figure 1: The KERMIT+Transformer architecture, forward and interpretation pass. During the forward pass parse trees are passed inactive (black and white trees). During the interpretation pass activations are back-propagated and heat parse trees are produced (colored trees).

coder for embedding syntax parse trees in universal-syntax-encoding vectors by explicitly embedding subtrees in the representation space. KERMIT*viz* is a visualizer to inspect how syntax is used in taking final decisions in specific tasks. We showed that KERMIT can effectively embed different syntactic information and KERMIT*viz* can explain KERMIT's decisions. Furthermore, paired with universal sentence embeddings, KERMIT outperforms state-of-the-art models - BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019) - in three different downstream tasks, albeit findings in Kuncoro et al. (2020), showing that traditional syntactic information is not represented in universal sentence embeddings.

2 Background and Related Work

Embedding symbolic syntactic or structured information within neural networks is a very active research field given the impression that using pre-existing syntactic knowledge in neural networks can be beneficial for many tasks. Initial attempts have tried to recursively encode structures in distributed representations to use them inside neural networks (Pollack, 1990; Goller and Kuechler, 1996). More recently, Socher et al. (2011) have defined the notion of Recursive Neural Networks (RecNN) that are Recurrent Neural Networks applied to binary trees. Initially, these RecNNs have been used to parse sentences and not to include pre-existing syntax in a final task (Socher et al., 2011). Then, these RecNNs have been used to encode pre-existing syntax in the specific task of Sentiment Analysis (Socher et al., 2012, 2013). With the rise of Long Short-Term Memories (LSTMs), Tai et al. (2015); Zhu et al. (2015) and Zhang et al. (2016) independently proposed TreeLSTM as an adapted version of LTSM that may use syntactic information. In TreeLSTM, the LSTM is applied following the structure of a binary tree instead of following an input sequence. In semantic relatedness and in sentiment classification, TreeLSTM has outperformed RecNN (Tai et al., 2015) by using pre-existing syntactic information. TreeLSTM has also been used to induce task-specific trees while learning a novel task (Choi et al., 2018). Moreover, Munkhdalai and Yu (2017) have specialized LSTM for binary and n-ry trees with their Neural Tree Indexers and Strubell et al. (2018) have encoded syntactic information by using multi-head attention within a transformer architecture.

However, there is a major problem with the methods for embedding syntactic structures in neural networks, it is unclear which parts of the parse trees are represented, and how. Hence, the behavior of neural networks that use these embeddings is obscure. It is then difficult to understand what kind of syntactic knowledge is encoded in the different layers and how this syntactic knowledge is used.

Some initial attempts to clarify which syntactic parts are encoded in embedding vectors exist. Zhang et al. (2018) have encoded parse trees by means of paths connecting the root of parse trees with words. Yet, these attempts are still far from completely representing parse trees.

For a long time, structural kernel functions have been the way to exploit syntactic information in learning but these functions cannot be used within neural networks. Kernel machines (Cristianini and Shawe-Taylor, 2000) exploit these, generally recursive, structural kernel functions that define a similarity measure between two trees counting common substructures. Hence, these structural *kernel functions* are built over a clear, although hidden, space of substructures. Structural kernels have been defined for both constituency-based (Collins and Duffy, 2002; Moschitti, 2006) and dependency-based parse trees (Culotta and Sorensen, 2004). As underlying spaces are well defined, it is even possible to extract back substructures that are relevant in

each decision (Pighin and Moschitti, 2010). However, these structural kernel functions are *generally recursive algorithms* that hide the real underlying space of features. Thus, structures are never represented as vectors in the target representation spaces as these spaces are generally huge. It is generally impossible then to use these clear spaces in learning with neural networks.

In the field of structural kernels, distributed tree kernels (Zanzotto and Dell'Arciprete, 2012) have opened an interesting possibility. To reduce the computational cost of tree kernels, these distributed tree kernels embed the huge space of substructures in a smaller space. This embedding is obtained by using recursive functions, which are linear with respect to the tree size. Hence, structures are represented in a smaller vector in an embedded space that represents the original space of structures. Hence, DTKs open an interesting path to include clear syntactic information in neural network architectures (Zanzotto and Ferrone, 2017; Santilli and Zanzotto, 2018).

3 The model

This section introduces our Kernel-inspired Encoder with a Recursive Mechanism for Interpretable Trees (KERMIT) (Sec.3.2) along its visualizer KERMIT*viz* (Sec.3.3). KERMIT is a lightweight encoder for universal syntactic interpretations which can be used in combination with transformer-based networks such as BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019) (Fig. 1). Some preliminary notations are given in Section 3.1.

3.1 Preliminary notation

This section fixes the notation for parse trees, random vectors and operations on random vectors as these are core representations in our model to deal with universal syntactic interpretations.

Parse trees $\mathcal{T}$ and parse subtrees τ are recursively represented as trees $t = (r, [t_1, ..., t_k])$ where r is the label representing the root of the tree and $[t_1, ..., t_k]$ is the list of child trees t_i. Leaves t are represented as trees $t = (r, [])$ with an empty list of children or directly as $t = r$.

On parse trees $\mathcal{T}$, our model KERMIT requires the definition of three sets of subtrees: the set $N(\mathcal{T})$, the set $\overline{S}(\mathcal{T})$ and the set of $S(\mathcal{T})$. The last two sets are defined according to subtrees we want to model in the embeddings of the universal syntactic interpretations. We use subtrees defined in Collins and Duffy (2002). The set $N(\mathcal{T})$ contains all the complete subtrees of $\mathcal{T}$. Given a tree $\mathcal{T}$ and r one of its nodes, a complete subtree of $\mathcal{T}$ from r is the subtree rooted in r that reaches the leaves, for example (see the parse tree in Fig. 1):

$$N\left(\begin{array}{c}\text{NP}\\ \text{the chef}\end{array}\right) = \boxed{\begin{array}{ccc}\text{NP} & \text{A} & \text{N}\\ \text{the chef} & \text{the} & \text{chef}\end{array}}$$

The set $\overline{S}(\mathcal{T})$ contains all the valid subtrees of $\mathcal{T} = (r, [t_1, ..., t_k])$ as follows $(r, [])$ is in $\overline{S}(\mathcal{T})$ and each $(r, [\tau_1, ..., \tau_k])$ where $\tau_i \in \overline{S}(t_i)$ are in $\overline{S}(\mathcal{T})$, for example:

$$\overline{S}\left(\begin{array}{c}\text{NP}\\ \text{the tasty soup}\end{array}\right) = \boxed{\begin{array}{cccc}\text{NP} & \text{NP} & \text{NP} & \\ \text{the tasty soup} & \begin{array}{ccc}\text{A}&\text{J}&\text{N}\\ &\text{tasty}&\text{soup}\end{array} & \begin{array}{ccc}\text{A}&\text{J}&\text{N}\\ \text{the}&&\text{soup}\end{array} &]\\ \text{NP} & \text{NP} & \text{NP} & \\ \begin{array}{ccc}\text{A}&\text{J}&\text{N}\\ \text{the}&&\end{array} & \begin{array}{ccc}\text{A}&\text{J}&\text{N}\\ &\text{tasty}&\end{array} & \begin{array}{ccc}\text{A}&\text{J}&\text{N}\\ &&\text{soup}\end{array} & \end{array}}$$

Finally, the set $S(\mathcal{T})$ is the union of the sets $\overline{S}(t)$ for all the trees $t \in N(\mathcal{T})$, that is:

$$S(\mathcal{T}) = \bigcup_{t \in N(\mathcal{T})} \overline{S}(t)$$

and it contains the subtrees used during training and inference.

Finally, to build the untrained KERMIT encoder, we use the properties of random vectors drawn from a multivariate Gaussian distribution $\boldsymbol{v} \sim \mathcal{N}(0, \frac{1}{\sqrt{d}}\mathbb{I})$. These vectors guarantee that $\boldsymbol{u}^T\boldsymbol{v} \approx 0$ if $u \neq v$ and $\boldsymbol{u}^T\boldsymbol{u} \approx 1$. This property is extremely important for interpretability. To compose vectors, we use the shuffled circular convolution $\boldsymbol{u} \otimes \boldsymbol{v}$. If these vectors are drawn from a multivariate Gaussian distribution, the function guarantees that $(\boldsymbol{u} \otimes \boldsymbol{v})^T\boldsymbol{u} \approx 0$, $(\boldsymbol{u} \otimes \boldsymbol{v})^T\boldsymbol{v} \approx 0$ and $(\boldsymbol{u} \otimes \boldsymbol{v}) \neq (\boldsymbol{v} \otimes \boldsymbol{u})$. This operation is a circular convolution $\star$ (as for Holographic Reduced Representations (Plate, 1995)) with a permutation matrix Φ: $\boldsymbol{u} \otimes \boldsymbol{v} = \boldsymbol{u} * \Phi\boldsymbol{v}$. This operation is extremely important for soundly composing node vectors.

3.2 The encoder for parse trees and its sub-network

KERMIT is a lightweight neural layer that allows the encoding and use of universal syntactic interpretations in neural networks architectures. This layer has two main components. The first component is the KERMIT encoder that actually encodes parse trees $\mathcal{T}$ in embedding vectors:

$$\boldsymbol{y} = \mathcal{D}(\mathcal{T}) \quad (1)$$

which corresponds to the gray arrow and the gray box in the KERMIT side of Fig. 1. The second component is a multi-layer perceptron that exploits these embedding vectors:

$$\boldsymbol{z} = mlp(\boldsymbol{y}) \tag{2}$$

which corresponds to green area in the KERMIT side of Fig. 1.

The KERMIT encoder $\mathcal{D}$ in Eq. 1 stems from tree kernels (Collins and Duffy, 2002) and distributed tree kernels (Zanzotto and Dell'Arciprete, 2012). It makes it possible to represent parse trees in vector spaces $\mathbb{R}^d$ that embed huge spaces of subtrees $\mathbb{R}^n$ where n is the huge number of different subtrees. Each tree $\mathcal{T}$ is represented by using the set of its valid subtrees $S(\mathcal{T})$. The encoder is based on an embedding layer for tree node labels $\boldsymbol{x_r} = \boldsymbol{W_o r} \in \mathbb{R}^d$ and on a recursive encoding function based on the shuffled circular convolution $\otimes$ introduced by Zanzotto and Dell'Arciprete (2012). The embedding layer $\boldsymbol{x_r} = \boldsymbol{W_o r} \in \mathbb{R}^d$ is an untrained encoding function that maps one-hot vectors $\boldsymbol{r}$ of tree node labels r to random vectors drawn from the previously introduced multivariate Gaussian distribution $\mathcal{N}(0, \frac{1}{\sqrt{d}}\mathbb{I})$. Hence, $\boldsymbol{W_o} \in R^{m \times d}$ is a matrix of m columns where m is the cardinality of the set of node labels and each column $\boldsymbol{w^{(i)}}$ of the matrix $\boldsymbol{W_o}$ is $\boldsymbol{w^{(i)}} \sim \mathcal{N}(0, \frac{1}{\sqrt{d}}\mathbb{I})$. The function $\mathcal{D}(\mathcal{T})$ is defined as a the sum of recursive function $\Upsilon(t)$ on parse trees:

$$\boldsymbol{y} = \mathcal{D}(\mathcal{T}) = \sum_{t \in N(\mathcal{T})} \Upsilon(t)$$

where $N(\mathcal{T})$ is the previously defined set of complete subtrees of $\mathcal{T}$. Then, $\Upsilon(t)$ is defined as:

$$\Upsilon(t) = \begin{cases} \sqrt{\lambda}\boldsymbol{W_o r} & \text{if } t = (r, []) \\ \sqrt{\lambda}(\boldsymbol{W_o r} + \boldsymbol{W_o r} \otimes \Upsilon(t_1) \otimes ... \otimes \Upsilon(t_k)) & \\ \quad \text{if } t = (r, [t_1, ..., t_k]) & \end{cases}$$

where $0 < \lambda \leq 1$ is a decaying factor penalizing large subtrees (Collins and Duffy, 2002; Zanzotto and Dell'Arciprete, 2012). By implementing $\mathcal{D}(\mathcal{T})$ with a dynamic algorithm, its computational cost is linear with respect to the nodes of the tree $\mathcal{T}$ and the cost of the basic function $\otimes$ is $d \log d$ where d is the size of the representation space $\mathbb{R}^d$. In fact, the circular convolution can be computed with Fast Fourier Transformation.

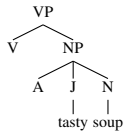
Given its nature, the tree neural encoder has a nice interpretation as a very simple embedding layer, that is, $\boldsymbol{W_\Upsilon} \in \mathbb{R}^{d \times n}$ that embeds the space of subtrees in a smaller space $\mathbb{R}^d$. This is in line with the Johnson-Lindenstrauss Transformation (Johnson and Lindenstrauss, 1984). Hence, $\mathcal{D}(\mathcal{T})$ can be seen as the following:

$$\boldsymbol{y} = \mathcal{D}(\mathcal{T}) = \boldsymbol{W_\Upsilon x} \tag{3}$$

where $\boldsymbol{x}$ is the vector representing the set of subtrees $S(\mathcal{T})$, that is, the sum of $\sqrt{\lambda^k}\boldsymbol{x}_t$ where $\boldsymbol{x}_t$ is one-hot vector representing $t \in S(\mathcal{T})$, λ is the decaying factor for penalizing large trees and k is the number of nodes of the tree t. It is possible and easy to show that columns $\boldsymbol{w_i}$ of $\boldsymbol{W_\Upsilon}$ encode subtrees t as follows:

$$w^{(i)} = \Gamma(t^{(i)}) = \begin{cases} \boldsymbol{W_o r} & \text{if } t^{(i)} = (r, []) \\ \boldsymbol{W_o r} \otimes \Gamma(t_1^{(i)}) \otimes ... \otimes \Gamma(t_k^{(i)}) & \\ \quad \text{if } t = (r, [t_1^{(i)}, ..., t_k^{(i)}]) & \end{cases}$$

for example:

$\Gamma($  $) =$

$$\begin{aligned} &\boldsymbol{W_o e_{VP}} \otimes (\boldsymbol{W_o e_N} \otimes \boldsymbol{W_o e_{NP}} \\ &\otimes (\boldsymbol{W_o e_A} \otimes (\boldsymbol{W_o e_J} \otimes \boldsymbol{W_o e_{tasty}}) \\ &\otimes (\boldsymbol{W_o e_N} \otimes \boldsymbol{W_o e_{soup}}))) \end{aligned}$$

where $\sqrt{\lambda^8}$ is the decay factor applied to the sample subtree with 8 nodes.

Given the properties of the vectors $E(r) \sim \mathcal{N}(0, \frac{1}{\sqrt{d}}\mathbb{I})$ and the properties of the shuffled circular convolution $\otimes$, it is possible to empirically demonstrate that $\Gamma(t_i)^T\Gamma(t_i) \approx 1$ and $\Gamma(t_i)^T\Gamma(t_j) \approx 0$ (Plate, 1995; Zanzotto and Dell'Arciprete, 2012). Hence, this property can be used to interpret the behavior of the decision in the neural network.

3.3 Visualizing Neural Network Activation on Syntactic Trees

The definitions of the KERMIT encoder make it possibile to devise KERMIT*viz* , which offers *prediction interpretability* (Jacovi et al., 2018) in the context of textual classification. We propose a clear *causal relation for explaining* (Lipton, 2016) classification decisions where syntax is important by defining *heat parse trees* and calculating the relevance of single subtrees with layer-wise relevance propagation (LRP) (Bach et al., 2015). LRP has already been used in the context of explaining decisions in natural language processing tasks (Croce et al., 2019b,a).

Heat parse trees (HPTs), similarly to "heat trees" in biology (Foster et al., 2017), are heatmaps over

parse trees (see the colored tree in Fig. 1). The underlying representation is an *active tree* $\bar{t}$, that is a tree where each node $\bar{t} = (r, v_r, [\bar{t}_1, ..., \bar{t}_k])$ has an activation value $v_r \in \mathbb{R}$ associated. HPTs are graphical visualizations of active trees $\bar{t}$ where colors and sizes of nodes r depend on their activation values v_r. In this way, HPTs highlight parts of parse trees relevant in final decisions.

To draw HPTs, we compute activation value v_r of nodes r in *active tree* $\bar{t}$ by using Layer-wise Relevance Propagation (LRP) (Bach et al., 2015) and the property in Eq. 3 of the KERMIT encoder $\mathcal{D}$. LRP is a framework which explains decisions of a generic neural network using local redistribution rules that propagate back decisions to activation values of initial features. In our case, this is used as a sort of inverted function of the multi-layer perceptron in Eq. 2, that is:

$$\boldsymbol{y}_{LRP} = mlp^{-1}_{LRP}(\boldsymbol{z})$$

The property in Eq. 3 enables the activation of each subtree $t \in \mathcal{T}$ to be computed back by transposing the matrix $\boldsymbol{W}_{\boldsymbol{\Upsilon}}$, that is:

$$\boldsymbol{x}_{LRP} = \boldsymbol{W}_{\boldsymbol{\Upsilon}}{}^{T} \boldsymbol{y}_{LRP}$$

To make the computation feasible, $\boldsymbol{W}_{\boldsymbol{\Upsilon}}{}^{T}$ is produced on-the-fly for each tree $\mathcal{T}$. Finally, activation values v_r of nodes $r \in \mathcal{T}$ are computed by summing up values $\boldsymbol{x}^{(i)}_{LRP}$ if $r \in t^{(i)}$.

4 Experiments

We aim to investigate whether KERMIT can be used to create neural network architectures where *universal syntactic interpretations* are useful: (1) to improve state-of-the-art universal sentence embeddings, especially in computationally light environments, and (2) to syntactically explain decisions.

The rest of the section describes the experimental set-up, the quantitative experimental results of KERMIT and discusses how KERMIT*viz* can be used to explain inferences made by neural networks over examples.

4.1 Experimental Set-up

This section describes the general experimental set-up of our experiments, the specific configurations adopted in the *completely universal* and *task-specific* settings, the used computational architecture and the datasets.

The general experimental settings are described hereafter. Firstly, the core of our method KERMIT encoder has been tested on a distributed representation space $\mathbb{R}^d$ with $d = 4000$ with the penalizing factor λ set to $\lambda = 0.4$ as this has been considered a common value in previous works (Moschitti, 2006). Secondly, constituency parse trees for KERMIT have been obtained by using Stanford's CoreNLP probabilistic context-free grammar parser (Manning et al., 2014). Thirdly, the following transformer sub-networks have been used: (1) $\text{BERT}_{\text{BASE}}$, used in the uncased setting with the pre-trained English model; (2) $\text{BERT}_{\text{LARGE}}$, used with the same settings of $\text{BERT}_{\text{BASE}}$; and, (3) XLNet base cased. All the models were implemented using Huggingface's transformers library (Wolf et al., 2019). The input text for BERT and XLNet has been preprocessed and tokenized as specified in respectively in Devlin et al. (2018) Yang et al. (2019). Fourthly, as the experiments are text classification tasks, the decoder layer of our KERMIT+Tranformer architecture is a fully connected layer with the softmax activation function applied to the concatenation of the KERMIT output and the final *[CLS]* token representation of the selected transformer model. Finally, the optimizer used to train the whole architecture is AdamW (Loshchilov and Hutter, 2019) with the learning rate set to $3e^{-5}$.

In the *completely universal* setting, KERMIT is composed only by the first lightweight encoder layer (grey layer in Figure 1) ($\text{KERMIT}_{\text{ENC}}$). In this setting, we used $\text{BERT}_{\text{BASE}}$ and XLNet. To study universality, transformers' weights are fixed in order to avoid the representation drifting toward the data distribution of the task. Moreover, we also experimented with $\text{BERT}_{\text{BASE}}$-Reverse and $\text{BERT}_{\text{BASE}}$-Random to understand whether syntactic or structural information is important for the specific task. In fact, $\text{BERT}_{\text{BASE}}$-Reverse is $\text{BERT}_{\text{BASE}}$ with a reversed text as input and $\text{BERT}_{\text{BASE}}$-Random is $\text{BERT}_{\text{BASE}}$ with a randomly shuffled text as input. Comparing $\text{BERT}_{\text{BASE}}$ with $\text{BERT}_{\text{BASE}}$-Reverse and $\text{BERT}_{\text{BASE}}$-Random is in itself an extremely important test as it offers also a way to determine if syntactic information is useful for a specific task. The KERMIT+Tranformer is trained with a batch size of 125 for 50 epochs. In addition, each experiment has been repeated 5 times with 5 different fixed seeds to assess the statistical significance of experimental results. This setting is designed to asses whether *universal syntactic interpretations*

Model	*AGNews*	*Yelp Review*	*DBPedia*	*Yelp Polarity*
XLNet	$79.11(\pm0.12)^{\star}$	$46.26(\pm0.13)^{\star}$	$92.46(\pm0.09)^{\star}$	$81.99(\pm0.15)^{\star}$
$\text{BERT}_{\text{BASE}}$	$\mathbf{82.88(\pm0.09)}^{\diamond}$	$42.90(\pm0.05)^{\diamond}$	$97.11(\pm0.27)^{\diamond}$	$79.21(\pm0.50)^{\diamond}$
$\text{BERT}_{\text{BASE}}$-Reverse	$79.72(\pm0.11)$	$38.14(\pm0.09)$	$90.46(\pm0.09)$	$72.23(\pm0.50)$
$\text{BERT}_{\text{BASE}}$-Random	$80.39(\pm0.20)$	$38.15(\pm0.08)$	$91.55(\pm0.20)$	$71.02(\pm0.50)$
$\text{KERMIT}_{\text{ENC}}$	$25.23(\pm0.14)$	$49.58(\pm0.10)$	$69.10(\pm0.06)$	$85.91(\pm0.03)$
$\text{KERMIT}_{\text{ENC}}$+XLNet	$77.88(\pm0.12)^{\star}$	$\mathbf{53.72(\pm0.14)}^{\star}$	$94.51(\pm0.05)^{\star}$	$\mathbf{88.99(\pm0.17)}^{\star}$
$\text{KERMIT}_{\text{ENC}}$+$\text{BERT}_{\text{BASE}}$	$77.02(\pm0.13)^{\diamond}$	$52.02(\pm0.06)^{\diamond}$	$\mathbf{97.73(\pm0.16)}^{\diamond}$	$87.58(\pm0.17)^{\diamond}$

Table 1: *Universal Setting* - Average accuracy and standard deviation on four text classification tasks. Results derive from 5 runs and $\star$ and $\diamond$ indicate a statistically significant difference between two results with a 95% confidence level with the sign test.

add different information with respect to *universal sentence embeddings* and whether *universal syntactic interpretations* are a viable solution to increase the performance of neural networks in *light computational systems*.

In the *task-adapted* setting, we used two different architecture of BERT, $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, and we trained different layers of these architectures. In this way, $BERT$ may adapt the *universal sentence embeddings* to include task-specific information which is the specific lexicon that may drive syntactic analysis. For the KERMIT side of the architecture, we used two different multi-layer perceptrons: (1) a *funnel* MLP with two linear layers that brings the 4,000 units of the KERMIT encoder down to 200 units with an intermediate level of 300 units ($\text{KERMIT}_{\triangleright}$); (2) a *diamond* MLP with four linear layers forming a diamond shape: 4,000 units, 5,000 units, 8,000 units, 5,000 units and, finally 4,000 units ($\text{KERMIT}_{\diamond}$). Both $\text{KERMIT}_{\triangleright}$ and $\text{KERMIT}_{\diamond}$ have ReLU (Agarap, 2018) activation functions and dropout (Srivastava et al., 2014) set to 0.25 for each layer. Due to the computational demand of these architectures and these experiments, we used the *heavy system* and we trained the overall model in two settings: a *one-epoch training session* and a *normal training session*. In the one-epoch training session, we trained the architecture with 1 epoch (Komatsuzaki, 2019) to avoid overfitting and to guarantee the possibility of having a relatively light computational burden. In the normal training session, we trained the architecture for 5 epochs. The batch size for these two settings was 32.

We experimented with two hardware systems: a *light system* and a *heavy system*. The *light system* is an affordable old desktop consisting of a 4 Cores Intel Xeon E3-1230 CPU with 62 Gb of RAM and 1 Nvidia 1070 GPU with 8Gb of onboard memory. The *heavy system* is a more expensive, dedicated server consisting of an IBM PowerPC 32 Cores CPU with 256 Gb of RAM and 2 Nvidia V100 GPUs with 32Gb of on board memory each.

To verify our model, we experimented with four classification tasks[1] (Zhang et al., 2015) which should be sensitive to syntactic information. The tasks include: (1) AGNews, a news classification task with 4 target classes; (2) DBPedia, a classification task over wikipedia with 14 classes; (3) Yelp Polarity, a binary sentiment classification task of Yelp reviews; and (4) Yelp Review, a sentiment classification task with 5 classes. Given the computational constraints of the *light system* setting, we created a smaller version of the original training datasets by randomly sampling 11% of the examples and keeping the datasets balanced as the original versions.

For reproducibility, the source code of our experiments is publically available[2].

4.2 Results and Discussion

Results from the *completely universal* experimental setting suggest that *universal syntactic interpretations complement syntax in universal sentence embeddings*. This conclusion is derived from the following observations of Table 1, which reports results in terms of the accuracy of the different models based on the different datasets. All these experiments were carried out on the *light system*.

Firstly, syntactic or structural information seems to be relevant in three out of four tasks. Syntactic information in AGNews seems to be irrelevant as there is a small difference in results between $\text{BERT}_{\text{BASE}}$, on the one side, with $82.88(\pm0.09)$ and $\text{BERT}_{\text{BASE}}$-Reverse with $79.72(\pm0.11)$ and

[1] http://goo.gl/JyCnZq

[2] The code is available at https://github.com/ART-Group-it/KERMIT

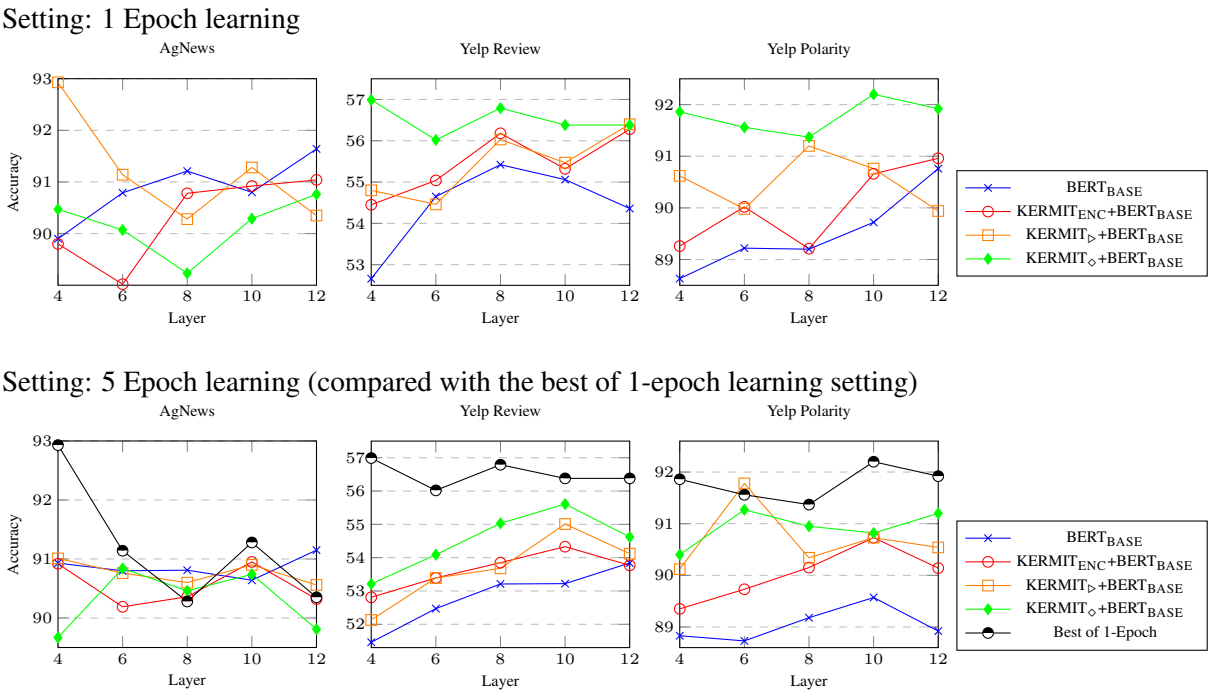


Figure 2: Comparison between KERMIT+BERT and BERT when training layers in BERT: Accuracy vs. Learned Layers in two different learning configurations - 1-Epoch and 5-Epoch training

BERT$_{BASE}$-Random with $80.39(\pm0.20)$, on the other. This small difference suggests that the order of words in the text is not particularly relevant and the classification is made on the lexical level. This justifies the very poor result from KERMIT$_{ENC}$ in this dataset, that is $25.23(\pm0.14)$.

Secondly, KERMIT$_{ENC}$ alone outperforms BERT$_{BASE}$ and XLNet in two cases where syntactic information is relevant, that is, $49.58(\pm0.1)$ vs. $42.90(\pm0.05)$ and $46.26(\pm0.13)$ in Yelp Review and $85.91(\pm0.03)$ vs. $79.21(\pm0.5)$ and $81.99(\pm0.15)$ in Yelp polarity. Hence, KERMIT$_{ENC}$ provides a good model for including *universal syntactic interpretations* in a neural network architecture. However, KERMIT$_{ENC}$ performed worse with respect to XLNet and BERT$_{BASE}$ in DBPedia even if syntactic information seems to be useful. This may be justified as both XLNet and BERT$_{BASE}$ are trained on Wikipedia, thus *universal sentence embeddings* are already adapted to the specific dataset.

Thirdly, in the three cases where syntactic information is relevant (Yelp Review, Yelp Polarity and DBPedia), the complete KERMIT+Transformer outperforms the model that is based only on the related Transformer, and the difference is statistically significant: $53.72(\pm0.14)$ vs. $46.26(\pm0.13)$ in Yelp Review, $94.51(\pm0.05)$ vs. $92.46(\pm0.09)$ in DBPedia and $88.99(\pm0.17)$ vs. $81.99(\pm0.15)$ in Yelp Polarity for XLNet and $52.02(\pm0.06)$ vs. $42.90(\pm0.05)$ in Yelp Review, $97.73(\pm0.16)$ vs. $97.11(\pm0.27)$ in DBPedia and $87.58(\pm0.17)$ vs. $79.21(\pm0.50)$ in Yelp Polarity for BERT$_{BASE}$. Even in DBPedia where transformers' embeddings are pretrained, KERMIT+Transformer outperforms the model based only on the related transformer.

This last observation is a very important indication and, together with the other observations, confirms that *universal sentence embeddings* encode different syntactic information with respect to that defined in *universal syntactic interpretations*. Moreover, our KERMIT encoder allows neural networks to positively use *universal syntactic interpretations*. Hence, using *universal syntactic interpretations* is a viable solution also when only light computational systems are available.

Experiments in the *task adapted* setting: (1) show that *universal syntactic interpretation* is still useful even when *universal sentence embeddings* are adapted to the specific task; (2) confirm the conclusions of Jawahar et al. (2019) that *universal sentence embeddings* better capture syntactic phenomena when the middle layers of BERT are learned over the task. The results of these experiments are plotted in Figure 2 where system accuracy is plotted against the number of BERT's learned layers starting from the output layer. In fact, it seems that different BERT's layers encode different in-

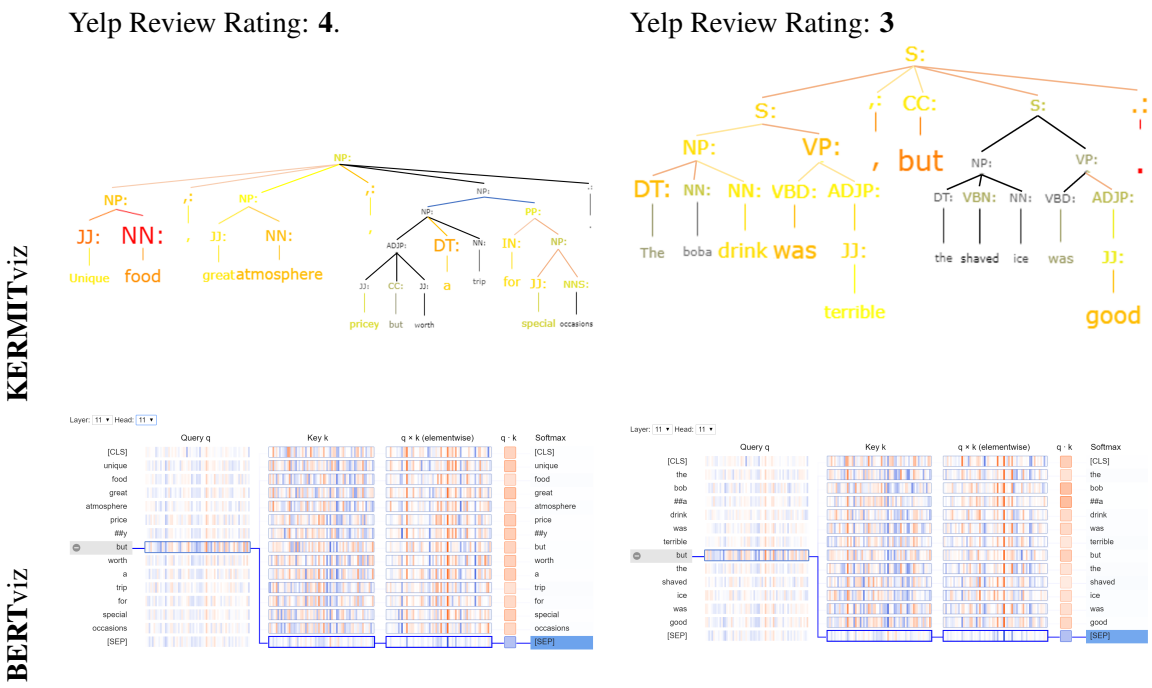


Figure 3: KERMIT*viz* vs. BERTviz: Comparing interpretations over KERMIT and over BERT on two sample sentences of Yelp Review where the word *but* is correlated or not with the final polarity.

formation Jawahar et al. (2019). Hence, learning different layers in a specific setting means adapting that kind of information. We experimented with two sub-settings: (1) a computationally lighter setting where training is done only for 1 epoch; (2) a more expensive setting where training is done for 5 epochs.

Our results in the task adapted setting confirms that BERT adapts *universal sentence embeddings* to include a better syntactic model when its weights in different layers are trained over the specific corpus. Moreover, as shown in Jawahar et al. (2019), layers in the middle cover better syntactic phenomena. In fact, when BERT learns up to the 8th layer, BERT's accuracy seems to come closer to the best model including *universal syntactic interpretations* (see Figure 2) . This suggests that more syntax is encoded in BERT.

All these experiments were performed also using $\text{BERT}_{\text{LARGE}}$ in place of $\text{BERT}_{\text{BASE}}$, but in all the experiments results were worse compared to the base version, therefore not reported in the paper.

When syntax matters, that is, in Yelp Review and in Yelp Polarity, KERMIT is able to exploit *universal syntactic interpretation* to compensate for missing syntactic information in the *task-adapted sentence embeddings* of a trained BERT. In fact, KERMIT+BERT outperforms a trained $\text{BERT}_{\text{BASE}}$in both the 1-epoch and 5-epoch settings for any number of trained layers (see Figure 2). In the 1-epoch setting, $\text{KERMIT}_{\diamond}$+$\text{BERT}_{\text{BASE}}$ outperforms $\text{BERT}_{\text{BASE}}$ and all the other configurations. In the 5-epoch setting, $\text{KERMIT}_{\text{ENC}}$+$\text{BERT}_{\text{BASE}}$ is the best model.

Moreover, KERMIT-based models behave better with less training. In fact, KERMIT-based models learned in the 1-epoch setting, outperform models learned in the 5-epoch setting. Plots in Figure 2 report the best 1-epoch setting model in the plots of the 5-setting model. This can be linked to the fact that KERMIT with more parameters overfits on training. In fact, $\text{KERMIT}_{\text{ENC}}$+$\text{BERT}_{\text{BASE}}$ outperforms the *funnel* and *diamond* KERMIT-based systems. $\text{KERMIT}_{\text{ENC}}$ has fewer parameters than $\text{KERMIT}_{\triangleright}$ and $\text{KERMIT}_{\diamond}$.

Finally, we explored the interpretative power of KERMIT*viz* comparing it with the transformer visualizer BERTviz (Vig, 2019). We focused on two examples of Yelp Reviews where the coordinating conjunction *but* plays an important role (see Fig. 3): (1) *"Unique food, great atmosphere, pricey* ***but*** *worth a trip for special occasions."*; (2) *"The boba drink was terrible,* ***but*** *the shaved ice was good."*. The two sentences have 4 and 3 as ratings, respectively. In fact, the *but* in the first sentence introduces a coordinated sentence that does not

change the rating. On the contrary, the *but* in the second sentence introduces a coordinated sentence *but the shaved ice was good* that radically changes the polarity. In the case of BERTviz, this causal relationship is extremely difficult to grasp from the visual representation. In fact, BERTviz is a good visualization mechanism for seeing how models assign weights to different input elements (Bahdanau et al., 2015; Belinkov and Glass, 2019), but it is extremely obscure in explaining causal relations in classification predictions (Wiegreffe and Pinter, 2019). Instead, KERMIT*viz* with its tree heat maps show exactly that the *but* and the related syntactic structure is irrelevant in the first sentence and extremely relevant in the second. Hence, our heat parse trees can be useful to draw the causal relation between the decision and the information used.

5 Conclusions

Universal syntactic interpretations are valuable language interpretations, which have been developed in years of study. In this paper, we introduced KERMIT to show that these interpretations can be effectively used in combination with *universal sentence embeddings* produced from scratch. Moreover, KERMIT*viz* allows us to explain how syntactic information is used in classification decisions within networks combining KERMIT, on the one side, and BERT or XLNet on the other. We also showed that KERMIT can be easily used in situations where training large transformers is extremely difficult.

As KERMIT has a clear description of the used syntactic subtrees and gives the possibility of visualizing how syntactic information is exploited during inference, it opens the possibility of devising models to include explicit syntactic inference rules in the training process.

Finally, KERMIT is in the line of research of Human-in-the-Loop Artificial Intelligence (Zanzotto, 2019), since it gives the opportunity to track how human knowledge is used by learning algorithms.

References

Abien Fred Agarap. 2018. Deep Learning using Rectified Linear Units (ReLU). *CoRR*, abs/1803.0.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):1–46.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193, Cambridge, MA. Association for Computational Linguistics.

Yonatan Belinkov and James Glass. 2019. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. *EMNLP 2018 - Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Proceedings*, pages 169–174.

David J Chalmers. 1992. Syntactic Transformations on Distributed Representations. In Noel Sharkey, editor, *Connectionist Natural Language Processing: Readings from Connection Science*, pages 46–55. Springer Netherlands, Dordrecht.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using {RNN} Encoder{–}Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing ({EMNLP})*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jihun Choi, Kang Min Yoo, and Sang Goo Lee. 2018. Learning to compose task-specific tree structures. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 5094–5101.

Stephen Clark and Stephen Pulman. 2007. Combining Symbolic and Distributional Models of Meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction, Stanford, CA, 2007*, pages 52–55.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of {ACL}02*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 670–680.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 1:2126–2136.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Danilo Croce, Daniele Rossini, and Roberto Basili. 2019a. Auditing Deep Learning processes through Kernel-based Explanatory Models. 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 4037–4046, Hong Kong, China. Association for Computational Linguistics.

Danilo Croce, Daniele Rossini, and Roberto Basili. 2019b. Neural embeddings: Accurate and readable inferences based on semantic kernels. *Natural Language Engineering*, 25(4):519–541.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*, pages 423–es, Morristown, NJ, USA. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *CoRR*, abs/1901.0.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. {BERT:} Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.0.

Allyson Ettinger. 2019. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Zachary S L Foster, Thomas J Sharpton, and Niklaus J Grünwald. 2017. Metacoder: An {R} package for visualization and manipulation of community taxonomic diversity data. *PLoS Computational Biology*, 13(2).

Yoav Goldberg. 2019. Assessing BERT's Syntactic Abilities.

Christoph Goller and Andreas Kuechler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *IEEE International Conference on Neural Networks - Conference Proceedings*, volume 1, pages 347–352. IEEE.

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 {A}merican 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.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding Convolutional Neural Networks for Text Classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ganesh Jawahar, , Benoît Sagot, , and Djamé Seddah. 2019. What Does BERT Learn about the Structure of Language? In *Proceedings of the Conference of the Association for Computational Linguistics*, pages 3651–3657. Association for Computational Linguistics (ACL).

W Johnson and J Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.*, 26:189–206.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3294–3302, Cambridge, MA, USA. MIT Press.

Aran Komatsuzaki. 2019. One Epoch Is All You Need. pages 1–13.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic Structure Distillation Pretraining For Bidirectional Encoders.

Zachary C. Lipton. 2016. The Mythos of Model Interpretability. *ICML Workshop on Human Interpretability in Machine Learning*, 61(Whi):36–43.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: {A} Robustly Optimized {BERT} Pretraining Approach. *CoRR*, abs/1907.1.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *7th International Conference on Learning Representations, ICLR 2019*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The {S}tanford {C}ore{NLP} Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

David Mareček and Rudolf Rosa. 2019. Extracting Syntactic Trees from Transformer Encoder Self-Attentions. pages 347–349.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based Models of Semantic Composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio. Association for Computational Linguistics.

Alessandro Moschitti. 2006. Making Tree Kernels practical for Natural Language Learning. In *Proceedings of EACL'06*. Trento, Italy.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural Tree Indexers for Text Understanding. In *Proceedings of the conference of the Association for Computational Linguistics*, volume 1, pages 11–21. NIH Public Access.

Daniele Pighin and Alessandro Moschitti. 2010. On Reverse Feature Engineering of Syntactic Tree Kernels. In *Conference on Natural Language Learning (CoNLL-2010)*, Uppsala, Sweden.

T A Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

Jordan B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105.

Andrea Santilli and Fabio Massimo Zanzotto. 2018. SyntNN at SemEval-2018 Task 2: is Syntax Useful for Emoji Prediction? Embedding Syntactic Trees in Multi Layer Perceptrons. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 477–481, New Orleans, Louisiana. Association for Computational Linguistics (ACL).

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP-CoNLL 2012 - 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Proceedings of the Conference*, pages 1201–1211.

Richard Socher, Cliff Chiung Yu Lin, Andrew Y Ng, and Christopher D Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 129–136.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1631–1642. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5027–5038.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multitask learning. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *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)*, volume 1, pages 1556–1566, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of System Demonstrations*, pages 37–42.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not Explanation. 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 11–20, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*, abs/1910.0.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, volume abs/1906.0, pages 5754–5764.

Fabio Massimo Zanzotto. 2019. Viewpoint: Human-in-the-loop Artificial Intelligence. *J. Artif. Intell. Res.*, 64:243–252.

Fabio Massimo Zanzotto and Lorenzo Dell'Arciprete. 2012. Distributed tree kernels. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, volume 1, pages 193–200.

Fabio Massimo Zanzotto and Lorenzo Ferrone. 2017. Can we explain natural language inference decisions taken with neural networks? Inference rules in distributed representations. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2017-May, pages 3680–3687.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Coling 2010 - 23rd International Conference on Computational Linguistics, Proceedings of the Conference*, volume 2, pages 1263–1271.

Richong Zhang, Zhiyuan Hu, Hongyu Guo, and Yongyi Mao. 2018. Syntax encoding with application in authorship attribution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2742–2753. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, pages 310–320.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *32nd International Conference on Machine Learning, ICML 2015*, volume 2, pages 1604–1612.