SHACT: Disentangling and Clustering Latent Syntactic Structures from Transformer Encoders

Alejandro Sierra-Múnera[1][0000−0003−3637−4904] ✉ and Ralf Krestel[2,3][0000−0002−5036−8589]

[1] Hasso Plattner Institute, Potsdam, Germany alejandro.sierra@hpi.de
[2] ZBW - Leibniz Information Centre for Economics
[3] Kiel University, Kiel, Germany rkr@informatik.uni-kiel.de

Abstract. Transformer-encoder architectures for language modeling provide rich contextualized vectors, representing both, syntactic and semantic information captured during pre-training. These vectors are useful for multiple downstream tasks, but directly using the final layer representations might hide interesting elements represented in the hidden layers. In this paper, we propose SHACT **S**yntactic **H**ierarchical **A**gglomerative **C**lustering from **T**ransformer-Encoders , a model that disentangles syntactic span representations from these hidden representations, into a latent vector space. In our model, spans are expressed in terms of token distances. We propose a loss function that optimizes the neural disentanglement model from ground truth spans, and we propose to integrate these latent space vectors into a two-phase model via hierarchical clustering, suitable for multiple span recognition tasks. We evaluated our approach on flat and nested named entity recognition as well as chunking, showing the model's ability to discover these spans, as well as having competitive results on the full recognition and classification tasks.

Keywords: Named entity recognition · Chunking · Syntax trees

1 Introduction

Current natural language processing (NLP) models rely heavily on rich pre-trained word representations in the form of contextualized word embeddings. Typically, these models use representations derived from transformer-encoders [18]. These models are pre-trained to predict words given a context, using large amounts of raw text. They are therefore commonly referred to as pre-trained language models (PLMs).

Besides being able to predict words given a context, these models are commonly used as embedding models, from which the token embeddings from the last transformer-encoder layer, can be used in task-specific models as dense representations. One group of tasks which strongly benefits from transformer-encoders are span recognition tasks, such as named entity recognition (NER) and chunking. In these tasks, given a sentence or document, the task is to find relevant spans of words and further categorize them in a set of pre-defined classes. One

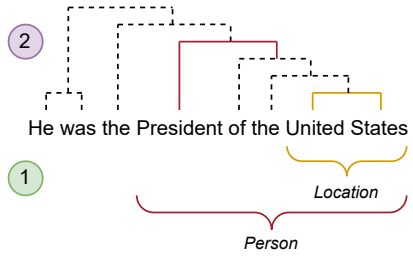

Fig. 1: Nested named entities ① and HAC analysis ② of a sentence

example of this can be seen in Figure 1 ①. In this example of nested NER, two spans (*President of the United States* and *United States*) correspond to named entities, and each named entity belongs to a specific class (*Person* and *Location* respectively). For tasks, such as flat NER and chunking, each word can be assigned to a maximum of one span, while for hierarchical tasks, such as nested NER, a single word might belong to multiple spans.

But the transformer model provides more than the last layer representation. The hidden transformer-encoder layers contain interesting representations as well, as other studies have revealed [5, 8, 11, 20]. In our proposed model SHACT (**S**yntactic **H**ierarchical **A**gglomerative **C**lustering from **T**ransformer-Encoders)[4], we exploit these hidden representations to define a latent syntactic vector space in which a tree that syntactically analyzes the sentence can be represented in terms of token distances. The intuition of our model can be seen in Figure 1 ②. Here, the sequence of words can be hierarchically analyzed by merging words together into clusters that eventually conform the whole sentence. We hypothesize, inspired by the ideas of Hewitt and Manning [5] and Sajjad et al. [11], that there exists a vector space, where hierarchical agglomerative clustering (HAC) can be performed to produce such analysis.

With SHACT, we propose a model for disentangling syntactic structures representing the clustering of tokens into spans by projecting the hidden representations of a transformer-encoder into a latent space. For learning the disentanglement, we use a loss function which treats multi-word spans as clusters. The loss function's goal is to minimize the intra-cluster distances in the latent space, while maximizing the extra-cluster distance at the same time, thus optimizing the projection. With this, the disentangled representation' distances represent the syntactic relatedness of tokens inside a sentence.

Further, the latent space vectors are clustered using hierarchical agglomerative clustering (HAC), resulting in a binary syntactic tree. This tree determines a set of candidate spans which are then classified using the same transformer-encoder, to identify relevant spans, and categorize them among the pre-defined types.

[4] We release the code for training and testing our model in this repository: `https://github.com/HPI-Information-Systems/shact`

2 Related Work

Our work is related to the interpretability and topological analysis of pre-trained language models (PLMs), specifically regarding syntax in pre-trained transformer-encoders. This line of research studies what kinds of structures are automatically learned by transformer architectures [3, 10, 18] when they are pre-trained using vast amounts of text.

Previous studies have proposed probes for syntactic structures in the hidden representations of PLMs. One of these probes was proposed by Hewitt and Manning [5]. They applied a trainable linear transformation to the hidden vectors in a specific layer of the model, and checked if the resulting vectors represent the dependency tree in terms of distance between words, and their depth in the tree. Mareček and Rosa [8] manually analyzed the attention heads of a neural machine translation transformer-encoder from which they recognized "balustrades" patterns. Then they automatically generated constituency trees from the attention heads using the CKY algorithm. They evaluated against different trivial baselines and found that their technique was able to find these constituents. Wu et al. [20] analyze the impact between words in a sentence through perturbation via masking. They computed an impact matrix, from which they built dependency and constituency trees in completely unsupervised mode. They showed that these structures — although they do not perfectly correlate to annotated trees — have a positive impact in downstream tasks. Furthermore, they question whether transformer architectures actually understand structures better than the traditionally studied constituency trees.

Sajjad et al. [11] proposed ConceptX to analyze different layers of transformer-encoders and found different linguistic concepts represented at different levels. Specifically, they discovered *encoded concepts* by clustering vector representations, and then they align these clusters with multiple human-defined linguistic concepts. Different from probes, such as Hewitt and Manning [5], the discovery of concepts in the vector space is unsupervised. They found that, although the alignment was low, the syntactic concepts, such as *chunking* were present in the upper layers before the final layers of the different transformer-encoders.

In our work, we take inspiration from these approaches and extract vector representations from the hidden layers of the transformer-encoders. Different from Hewitt and Manning [5] we don't use dependency relations, but instead word agglomerations, more similar to constituents. Different from Sajjad et al. [11] we specialize in spans instead of individual words. Deviating from both and other probing studies, we do not intend to analyze the transformer-encoder layers individually, but we exploit their combination to project the tokens into a different vector space.

Another line of related work are span-based nested NER models. These models go beyond the IOB sequence tagging scheme designed for flat NER, and thus can identify multiple overlapping named entities. One subgroup of nested NER focuses on enumerating spans within the text and then classifying the spans among the entity types. Given a sentence, Sohrab and Miwa [14] proposed to create an exhaustive set of all the spans with a maximum length and then classify

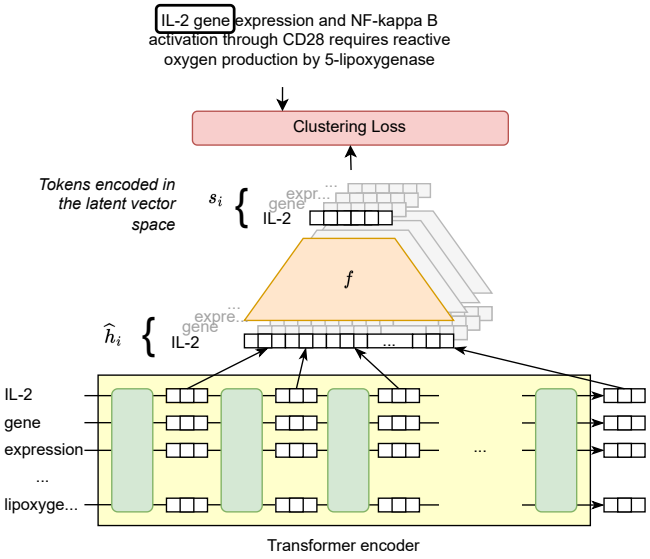

Fig. 2: Projection from a transformer encoder hidden layers to the latent space s_i where the loss function is computed for a particular entity (*IL-2 gene*)

them. One disadvantage of the extensive enumeration of spans is the large number of spans that need to be classified. In Zheng et al. [22], the authors construct the entity spans using a sequence labeling model based on a Bi-LSTM encoder, and then use the average token representations within the span to predict the span label. Similarly, Tan et al. [15] encode the sentence to discover boundaries and then classify the candidate spans using the token representation within the span.

Similar to the works in Tan et al. [15], Zheng et al. [22], we propose to define the candidate spans and classify them using a transformer-encoder, but different from them, we disentangle the vector space that represents syntactic span structures from the semantic representations used for the classification of these spans. In comparison to Sohrab and Miwa [14], the number of candidate spans in our approach does not depend on the span length but only on the length of the sequence, thus producing candidates from all possible lengths without affecting the number of candidates.

3 SHACT

In this section we introduce SHACT, **S**yntactic **H**ierarchical **A**gglomerative **C**lustering from **T**ransformer-Encoders , a model that disentangles syntactic representations of spans and then uses hierarchical clustering to discover and classify word agglomerations. First, we present the disentanglement component, and

then we complete the span recognition model with the classification component based on the spans detected in the clusters.

3.1 Disentanglement of Syntactic Clusters

Inspired by the identification of syntactic structures in PLMs, we define a model which projects tokens into a latent space where significant spans, such as named entity mentions or noun phrases, conform to clusters. Treating spans as clusters, the goal of the model is to minimize the distance between the token representations within the cluster in comparison to their distance to other token representations of the sentence. We assume a PLM with a transformer-encoder architecture [3], and consider all the hidden representations of a token as its *full vector representation*. From these representations, our model projects to latent vector space, in which the inter-cluster and extra-cluster distances are compared. The projection is then trained with sentences containing annotated spans to minimize the ratio between intra-cluster and extra-cluster distances.

An overview of the disentanglement model architecture is represented in Figure 2. Here, the transformer-encoder passes each token vector through multiple transformer layers, producing multiple hidden representations. SHACT concatenates them into a large token vector, which is projected into the latent space using a feed-forward network (FFN). Given the set of latent token vectors and a ground truth span, we compute a clustering loss function, based on intra-cluster and extra-cluster distances. Using gradient descent to minimize this loss, the FFN is updated during training to better extract the span representations in the latent space. To generalize the concept of span boundaries and reduce the complexity of the projection, the disentanglement component of the model does not differentiate between the types of spans. For instance, if the task is NER and the relevant spans are entity mentions, the latent projection is agnostic to the entity types, and it is optimized for all the mentions in the same way.

We now formally define the elements of the model and the loss function. For a given token t_i in a sentence $d = \{t_0, t_1, t_2, \ldots, t_n\}$, and a j^{th} layer of a transformer-encoder producing a hidden representation $h_{i,j}$, we define a vector $\widehat{h}_i$ by concatenation of all the vectors $h_{i,j}$. This vector contains the *full representation* of the token t_i in the context of the sentence d. In the next step, $\widehat{h}_i$ is projected into a vector space $\mathcal{S}$ as s_i via a fully connected neural network f.

For an annotated span n in d (e.g., a named entity or a chunk), consisting of a subset of contiguous tokens $n = \{t_k, t_{k+1}, ..., t_{k+m}\} | n \subset d$ we define the loss function

$$\mathcal{L}_{syn}(d, n) = \frac{\mathcal{ICD}}{\mathcal{ECD}} = \frac{\max\limits_{t_i, t_j \in n} \delta(t_i, t_j) + 1}{\min\limits_{t_i \in n \ t_j \notin n} \delta(t_i, t_j) + 1}$$

Where δ is the cosine distance between the tokens vectors in the vector space $\mathcal{S}$. $\mathcal{ICD}$ and $\mathcal{ECD}$, represent the intra-cluster and extra-cluster distances. The idea behind the loss function $\mathcal{L}_{syn}$ is to optimize the projection f by selecting

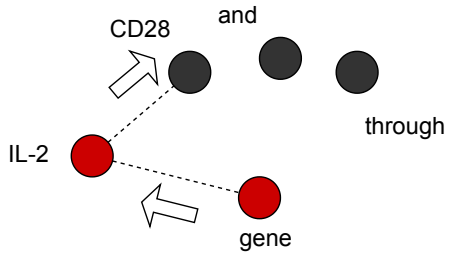

Fig. 3: Example 2D representation of the latent space and the distances considered during training for the named entity *IL-2 gene*. The arrows represent the optimization objectives of minimizing $\mathcal{ICD}$ and maximizing $\mathcal{ECD}$

and combining the features of the vector $\widehat{h}_i$ so that the span n is isolated from the rest of the tokens in the sentence. This is achieved by minimizing $\mathcal{L}_{syn}$ which in turn minimizes the maximum distance $\mathcal{ICD}$ within the cluster and maximizes the minimum distance $\mathcal{ECD}$ between tokens belonging to n and the rest of the sentence.

To give more intuition about the loss function, we consider the case where $\mathcal{ICD} < \mathcal{ECD}$. A value of $\mathcal{L}_{syn} < 1$ means that for an annotated span n, e.g., a named entity, there is no token outside n with a smaller distance than that between all the tokens belonging to n, hence effectively clustering the span in terms of vector distances. To avoid numeric errors when $\mathcal{ECD}$ is small, a constant 1 is added to both metrics. The optimization goal and the interpretability of $\mathcal{L}_{syn}$ with respect to 1 remain equivalent.

A visualization of the idea behind the loss function can be seen in Figure 3. Here, the named entity *"IL-2 gene"* is compared against the rest of the sentence. The maximum intra-cluster distance $\mathcal{ICD}$ is between the tokens *IL-2* and *gene*, whereas the minimum extra-cluster distance $\mathcal{ECD}$ is between *IL-2* and *CD28*. Intuitively, the loss function's value would decrease if *IL-2* and *gene* were closer and the distance between *IL-2* and *CD28* was larger.

During training, the loss function is computed for each ground truth span, and then averaged for a batch of sentences. During backpropagation, the parameters of f are updated to learn a projection into the latent space. We refrain from back propagating from $\mathcal{L}_{syn}$ to the transformer layers to avoid instability, and to focus the fine-tuning on the semantic aspects needed for the span classification. During inference, we project all the tokens into S and apply hierarchical agglomerative clustering (HAC) for the full sentence until all are merged into one single cluster. Every merge operation constitutes a potential span that will be processed during classification (see Section 3.2). We note that the merge operations are restricted to adjacent tokens in the sentence, thus producing only contiguous spans of text for the next stage. Additionally, tokens belonging to the same word are merged with the first token of the word, which in turns acts as the representative token for the entire word. For instance, given the tokens

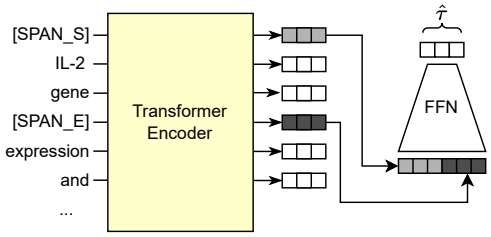

Fig. 4: Overview of the classifier architecture. The final vector resulting from the FFN contains the logits corresponding to the types in $\hat{\tau}$

"human mon #ocytes .", the token *#ocytes* can only be merged to*mon* to form the word *monocytes*.

3.2 Span Classification

The span classification phase, shown in Figure 4, uses the same transformer-encoder used for disentangling the syntactic representations, but exploits the last layer representations. We hypothesize that this representation contains more semantic information and thus, is better suited for identifying the span types (e.g., person, location, noun phrase). Formally, given the original sentence $d = \{t_0, t_1, t_2, \ldots, t_n\}$ and a span defined by the tuple $c_k = (c_k^s, c_k^e)$, being c_k^s the starting index of the span, c_k^e the ending index of the span, the span classifier has the goal to predict the type $c_k^t \in \widehat{\tau} | \widehat{\tau} = \tau \cup \{null\}$, with τ being the original type set. To do so, after tokenization, two special tokens *[SPAN_S]* and *[SPAN_E]* are inserted before and after the span tokens to produce a new sentence $\hat{d_k}$. For instance, for the span in Figure 2, the token sequence would be modified in this way: *"[SPAN_S] IL-2 gene [SPAN_E] expression and ..."*.

The modified tokenized version of the sentences is then fed into the transformer-encoder, from which the final layer vectors corresponding to *[SPAN_S]* and *[SPAN_E]* are fed into a multi-layer perceptron responsible for the classification. Formally, the predicted label $\hat{y}(d, c_k)$ for a sentence d and a span c_k belonging to the sentence, is computed as follows:

$$\hat{y}(d, c_k) = \phi(FFN([\mathcal{E}(\hat{d_k})_{SPAN\_S} : \mathcal{E}(\hat{d_k})_{SPAN\_E}]))$$

where ϕ is the *softmax* function, FFN is the feed-forward network, and $\mathcal{E}(d)$ is the encoded representation of d, corresponding to the last layer representation of the transformer-encoder.

During training, the ground truth spans are used to learn the representations of *[SPAN_S]* and *[SPAN_E]* that should be classified into the corresponding types, but the model needs to learn spans that fall into the $\{null\}$ type. Thus, a subset of spans from the sames sentences is also used as negative samples for the classifier.

Formally, a batch B is conformed of m samples, each composed of a sentence $d_i = \{t_0, t_1, t_2, \ldots, t_n\}$, a span $c_i = (c_i^s, c_i^e)$ and one-hot-encoded ground truth label y_{ij} where j corresponds to the j^{th} class in $\hat{\tau}$. The model then computes $\hat{y}_{ij}$, with which the classification cross-entropy loss function $\mathcal{L}_{CE}$ is aggregated for the batch B as follows:

$$\mathcal{L}_{CE}(B)(y, \hat{y}) = -\sum_{i=1}^{m} \sum_{j=1}^{\hat{\tau}} y_{ij} \log(\hat{y}_{ij})$$

During inference, all the potential spans produced by the hierarchical clustering algorithm are classified by taking the *argmax* of the predicted classes.

3.3 Training Procedure and Negative Sampling

Providing negative samples to the classifier impacts the way the model learns to distinguish between relevant spans ($c_k^t \in \tau$) and non-relevant spans ($c_k^t = null$). To generate a set of negative spans, we leverage the disentangled syntactic representations from SHACT. To do so, first we warm up the disentanglement projection f, keeping the transformer-encoder parameters frozen. With a trained projection, hierarchical agglomerative clustering is performed on the sentences in the training set, therefore discovering potential syntactically relevant spans. From those spans, the ones which do not correspond to ground truth spans are considered as negative spans for the subsequent classification training.

Then, in the second phase of training, the full model is trained by considering both positive and negative examples. The full loss function for a batch B combining positive and negative samples is the sum of the cross-entropy loss $\mathcal{L}_{CE}(B)$ for the classifier and the syntactic loss $\mathcal{L}_{syn}$ for the positive examples of B (i.e., spans with a type belonging to τ). in this phase we fine tune the transformer-encoder using $\mathcal{L}_{CE}(B)$, but the disentanglement projection model is detached from the transformer representations, thus only training the projection f. In our initial experiments, we observed instability in the clustering loss if we also back-propagate to the transformer from the projection f.

Inference Procedure. Different from training, where each sentence-span pair (d, c_k) is trained independently, during inference, each sample is composed only of a sentence d, without ground-truth spans.

The first part of the inference phase is to compute the latent representations of all tokens of d. These vectors are then clustered using HAC. Specifically, the single linkage strategy is used with the additional constraint that only contiguous clusters can be merged. The outcome of the clustering is a dendrogram (Figure 1) in which every merge is a potential span. In total, for a sentence with n words, $2n - 1$ candidate spans, including the individual words, are considered for classification.

For each candidate span, a new sentence with the special tokens *[SPAN_S]* and *[SPAN_E]* is encoded using the transformer, and the special token representations are classified the same way as the classification training. The final

result of the inference phase is a binary tree, where the leaves are the words in the sentence, and every node, including the leaves, has an assigned class from $\hat{\tau}$. Task-specific post-processing filters can be applied to the tree. In particular, for tasks, such as flat NER and chunking, where each word can belong to a maximum of one span, the prediction needs to be flattened. For hierarchical prediction, such as for nested NER, the predictions can be computed from the tree by ignoring the nodes with a predicted *null*-type.

The criteria that we employ for flattening the prediction involves a sequential sorting of the positive candidates in descending order, based on their level of classification confidence and their length in words. Then spans are selected sequentially if no overlapping span was selected before. This means that we favor high confidence spans, and given two overlapping spans with the same confidence, we favor the longest one.

4 Evaluation

Our evaluation focuses on two aspects of our model. First, the intrinsic clustering ability of the model, corresponding to the degree in which a trained model is able to cluster together spans, to do so we focus on nested NER, being more complex to cluster spans in a hierarchical fashion. Secondly, we evaluate the full two phase model including the classifier for three different tasks: flat NER, nested NER, and chunking. With this, we evaluate the usefulness and adaptability of syntactic trees with different classification schemes.

For flat NER, we evaluate our model on CoNLL03 [17], for nested NER, we use GENIA [9], and for chunking, we employ CoNLL00 [16] .

As mentioned in Section 3.3, the model is warmed-up for one epoch, and then, both the disentanglement and classification components are trained for a maximum of 15 epochs using early stopping with respect to the validation F1 score and a patience value of 5 epochs.

In all the experiments, we use *BERT-large-cased* [3] as the PLM, except for the GENIA experiment where we use *BioBERT-large-cased-v1.1* [6]. The dimensionality of the latent space is fixed to 128 and Adam is used as the optimizer.

4.1 Intrinsic Evaluation

The intrinsic evaluation aims to determine whether the clustering loss can optimize the projection f from spans in the training set. To measure that, we compute the recall of spans from the test set after training only the projection disentanglement component. We train the model for one epoch using the training set and for each of the sentences in the test set, HAC is used on the set of token vectors until the whole sentence is merged into one single cluster. We then analyze the resulting binary trees, measuring the span recall of ground truth spans. This is critical for our model, because any ground truth span, not present in the binary tree, cannot be considered by the classifier component.

For baselines, we use the following vector representations:

Table 1: Percentage of GENIA named entities present in the binary trees computed with HAC on the different vector spaces

Vector Space	Span Recall
PLM_{final}	52.94
PLM_{full}	51.51
LS_{random}	51.50
LS	**97.37**

- PLM_{final}**:** The final layer representation of the transformer-encoder.
- PLM_{full}: The concatenation of all the hidden representations for each token, which is equivalent to $\widehat{h}_i$.
- LS_{random}: The latent space vectors, computed with the projection component, initialized with random weights without training. This will provide us insights on the value of the loss function to optimize the projection.
- LS: The latent space vector, computed with the trained projection component.

For these experiments, the PLM is kept frozen and the experiments with LS_{random} and LS are repeated 3 times with different seeds. A named entity from GENIA is considered a correct prediction if, and only if, it is one of the distinct clusters of HAC computed for the sentence. The recall numbers can be seen in Table 1. Here, we clearly see that the distance in the other vector representations does not correlate to the syntactic relatedness of the tokens in an entity mention. Our trained projection leads to the highest recall, which shows the potential of the latent space for expressing the relatedness of span tokens. However, the low recall numbers of the PLM approaches are not surprising, given the objectives used for training them. BERT's masked language modeling objective emphasizes on the semantic, rather than the syntactic representation of the tokens. We also observe that randomly combining hidden representations (LS_{random}) does not help our clustering objective. With this, we can observe that the optimization of the projection f is required to capture all the relevant spans.

Having a good span recall in the latent space is critical for our model, since the errors from the syntactic analysis phase are propagated to the classification phase.

4.2 Extrinsic Evaluation

For the full model training, we first warm up the projection for one epoch, and then we generate all the potential spans in the training set that are fed as positive or negative spans to the full model.

For the three tasks, we measure exact F1 score, considering a correct prediction only if a span has the same boundaries and the correct type. All the

Table 2: Results in terms of F1 score in different tasks. * for flat models

(a) GENIA (Nested NER)

Model	F1
Zheng et al. [22]	74.7
Sohrab and Miwa [14]	77.1
Tan et al. [15]	78.3
Fu et al. [4]	78.2
Yang and Tu [21]	78.16
Lou et al. [7]	78.44
Shen et al. [13]	81.77
Shen et al. [12]	81.53
SHACT	79.76

(b) CoNLL-2003 (Flat NER)

Model	F1
Devlin et al. [3]*	92.8
Shen et al. [13]	92.87
Shen et al. [12]	92.78
Wang et al. [19]*	94.6
SHACT	91.74

(c) CoNLL-2000 (chunking)

Model	F1
Chen et al. [2]*	96.34
Chen et al. [1]*	97.04
Wang et al. [19]*	97.3
SHACT	96.61

reported SHACT F1 scores are the average result of three different runs with different seeds.

The results for flat NER on the CoNLL-2003 dataset are shown in Table 2b, where we compare our approach against span-based NER models [12, 13] and state-of-the-art flat NER models [3, 19]. In this case, we see that SHACT results are slightly worse than specialized state-of-the-art span-based models capable of both flat and nested NER. We also observe that the traditional sequence labeling models, such as plain BERT with a classification head [3] and ACE [19], achieve stronger results. However, we believe that SHACT is more flexible than sequence labeling models and the latent space analysis possesses an explainability value on its own while achieving competitive results compared to specialized models.

For chunking on the CoNLL-2000 dataset, we have a similar comparison in Table 2c. Here, the results of SHACT are competitive even with the state-of-the-art flat model ACE [19]. We believe that the syntactic analysis of SHACT in this case is very capable of clustering chunks that typically cover a few words. The flattening here is particularly challenging because the classifier must not only correctly identify semantically the differences between linguistic chunks, such as noun and verbal phrases, but also assign a higher confidence to the correct levels in the tree.

In addition, we evaluate our model on the task of nested NER using the GENIA dataset. This task is more complex compared to flat NER. Results are in Table 2a. For this task, SHACT has competitive results, and compared to better performing models, such as DifussionNER [12], the binary tree helps in explaining the predictions by clearly defining merge operations.

One potential problem of models, such as SHACT, in which first the spans are enumerated and then classified, is the potential error propagation between the enumeration phase and the classification. We analyze this by computing the upper bound for the recall the same way we computed the span recall in the intrinsic evaluation. Differently from the intrinsic evaluation, we use the final model after

Table 3: Comparison between recall (R) of SHACT and the upper bound of the classification recall (Upper bound R) resulting of error propagation

Dataset	R	Upper bound R	Difference
CoNLL-2000	96.44	99.68	3.24
CoNLL-2003	92.12	99.59	7.47
GENIA	78.31	98.07	19.76

the classifier is trained alongside the projection. This is slightly different because after full training, the loss function is composed of two objectives. In Table 3, we compare the recall and its upper bound for each evaluated dataset. We clearly see that the impact on the recall by error propagation is minimal (between 0.32% and 1.93%) compared to the reduction of recall by misclassification.

4.3 Explainability and Error Analysis

Besides the quantitative analysis of the performance of SHACT, we take advantage of the explainability features of the model by analyzing erroneous predictions. The examples we show in Figure 5 portray two different types of error that might happen during inference. The error in 5a corresponds to a semantic error, characterized by the misclassification of spans. In this case, SHACTcorrectly clusters the words in *human monocytes*, but later the classifier incorrectly assigns the *null* class to this span. Similarly, the longer span *normal human monocytes*, that is not part of the ground truth, is incorrectly classified as *Cell type*. This is the most common error found in this dataset.

The second error presented in 5b is a syntactic error. Here, the span containing *CD28 MoAb* is not part of the predicted binary tree; thus the final prediction does not contain any classification of such span. This is a case of error propagation, and exemplifies the 1.93% of the cases in which the recall is bounded by an erroneous syntactic analysis.

With SHACT we are able to separate the syntactic and semantic analysis of the spans, which allows us to better understand the origin of bad predictions.

5 Conclusions

In this paper, we proposed SHACT, a model for disentanglement of syntactic clusters from pre-trained transformer-encoders, such as BERT [3]. We proposed a feed forward network projecting from a concatenation of all the layers of the transformer to a latent space. In addition to the projection, we defined a loss function based on the minimization of inter-cluster distances and maximization of extra-cluster distances, where clusters are formed by multi-word spans. The optimization of the projection component, using a loss function, generates a latent vector space in which span words are closer together in comparison to the

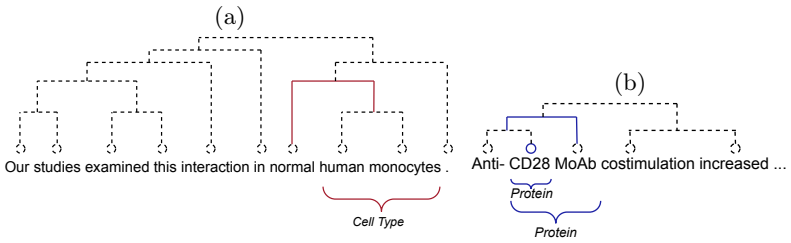

Fig. 5: Errors committed by SHACT on GENIA. Below the sentences is the ground truth, and the trees on top of the sentences are the predictions. Dashed circles/lines mean *null* class, while solid circles/lines entity predictions

rest of the sentence. We complement the disentanglement with a classifier that uses the discovered syntax tree to classify the candidate spans.

We evaluated the model for flat and nested NER, as well as chunking. We observed that although the model does not perform better than all the state-of-the-art models in each task, the performance is competitive in across all tasks. We observed, as well, that the first component that builds the syntactic binary tree is successful in representing ground truth spans as clusters in the latent space, but the classification of these spans remains a challenging part of the model. We believe that this also illustrates the potential of the disentangled latent space that could also be exploited in other ways to improve the classification. By taking advantage of the separation between the syntactic and the semantic analysis of the spans, we can also analyze and explain the predictions of the model. In further experiments, we also tested the model on the task of full constituency parsing. Unfortunately, SHACT yielded poor results in the alignment between the binary trees and the full constituency trees. This observation is aligned to the findings in Wu et al. [20].

Our model was designed with the specific goal of disentangling nested syntactic representations. However, there are more syntactically relevant structures which are out of the scope of this paper. For instance, dependency trees like the ones studied by Hewitt and Manning [5]. We limited our experiments to the English language. Experimentation with other datasets in different languages and with richer and more fine-grained entity types are out of the scope of the paper and could be studied in future work.

Acknowledgments. This research was partially funded by the HPI Research School on Data Science and Engineering. We would like to thank Prof. Felix Naumann for the methodological advice.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Bibliography

[1] Chen, L., Liu, X., Ruan, W., Lu, J.: Enhance robustness of sequence labelling with masked adversarial training. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 297–302 (Nov 2020)

[2] Chen, L., Ruan, W., Liu, X., Lu, J.: SeqVAT: Virtual adversarial training for semi-supervised sequence labeling. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 8801–8811 (Jul 2020)

[3] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, pp. 4171–4186 (Jun 2019)

[4] Fu, Y., Tan, C., Chen, M., Huang, S., Huang, F.: Nested Named Entity Recognition with Partially-Observed TreeCRFs. Proceedings of the AAAI Conference on Artificial Intelligence **35**(14), 12839–12847 (May 2021)

[5] Hewitt, J., Manning, C.D.: 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 (Jun 2019)

[6] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H., Kang, J.: Biobert: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics **36**, 1234 – 1240 (2019)

[7] Lou, C., Yang, S., Tu, K.: Nested named entity recognition as latent lexicalized constituency parsing. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 6183–6198 (May 2022)

[8] Mareček, D., Rosa, R.: From balustrades to pierre vinken: Looking for syntax in transformer self-attentions. In: Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pp. 263–275 (Aug 2019)

[9] Ohta, T., Tateisi, Y., Kim, J.D.: The genia corpus: An annotated research abstract corpus in molecular biology domain. In: Proceedings of the Second International Conference on Human Language Technology Research, p. 82–86, HLT '02, San Francisco, CA, USA (2002)

[10] Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training (2018)

[11] Sajjad, H., Durrani, N., Dalvi, F., Alam, F., Khan, A., Xu, J.: Analyzing encoded concepts in transformer language models. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 3082–3101 (Jul 2022)

[12] Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: DiffusionNER: Boundary diffusion for named entity recognition. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 3875–3890 (Jul 2023)

[13] Shen, Y., Wang, X., Tan, Z., Xu, G., Xie, P., Huang, F., Lu, W., Zhuang, Y.: Parallel instance query network for named entity recognition. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 947–961 (May 2022)

[14] Sohrab, M.G., Miwa, M.: Deep exhaustive model for nested named entity recognition. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 2843–2849 (Oct-Nov 2018)

[15] Tan, C., Qiu, W., Chen, M., Wang, R., Huang, F.: Boundary enhanced neural span classification for nested named entity recognition. Proceedings of the AAAI Conference on Artificial Intelligence **34**(05), 9016–9023 (Apr 2020)

[16] Tjong Kim Sang, E.F., Buchholz, S.: Introduction to the CoNLL-2000 shared task chunking. In: Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop (2000)

[17] Tjong Kim Sang, E.F., De Meulder, F.: Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp. 142–147 (2003)

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, p. 6000–6010, NIPS'17 (2017)

[19] Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., Tu, K.: Automated concatenation of embeddings for structured prediction. 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. 2643–2660 (Aug 2021)

[20] Wu, Z., Chen, Y., Kao, B., Liu, Q.: Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 4166–4176 (Jul 2020)

[21] Yang, S., Tu, K.: Bottom-up constituency parsing and nested named entity recognition with pointer networks. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2403–2416 (May 2022)

[22] Zheng, C., Cai, Y., Xu, J., Leung, H.f., Xu, G.: A boundary-aware neural model for nested named entity recognition. 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), pp. 357–366 (Nov 2019)