Enriching Transformers with Structured Tensor-Product Representations for Abstractive Summarization

Yichen Jiang*[1], Asli Celikyilmaz[2], Paul Smolensky[2,3], Paul Soulos*[3], Sudha Rao[2], Hamid Palangi[2], Roland Fernandez[2], Caitlin Smith*[3], Mohit Bansal[1], Jianfeng Gao[2]

[1]UNC Chapel Hill [2]Microsoft Research, Redmond [3]Johns Hopkins University

{yichenj, mbansal}@cs.unc.edu

{aslicel,psmo,sudha.rao,hpalangi,rfernand,jfgao}@microsoft.com

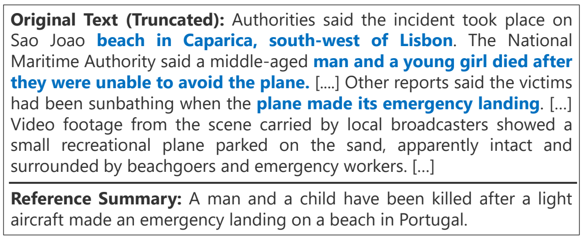
{psoulos1, csmit372}@jhu.edu

Abstract

Abstractive summarization, the task of generating a concise summary of input documents, requires: (1) reasoning over the source document to determine the salient pieces of information scattered across the long document, and (2) composing a cohesive text by reconstructing these salient facts into a shorter summary that faithfully reflects the complex relations connecting these facts. In this paper, we adapt TP-TRANSFORMER (Schlag et al., 2019), an architecture that enriches the original Transformer (Vaswani et al., 2017) with the explicitly compositional Tensor Product Representation (TPR), for the task of abstractive summarization. The key feature of our model is a structural bias that we introduce by encoding two separate representations for each token to represent the syntactic structure (with *role vectors*) and semantic content (with *filler vectors*) separately. The model then binds the role and filler vectors into the TPR as the layer output. We argue that the structured intermediate representations enable the model to take better control of the contents (salient facts) and structures (the syntax that connects the facts) when generating the summary. Empirically, we show that our TP-TRANSFORMER outperforms the Transformer and the original TP-TRANSFORMER significantly on several abstractive summarization datasets based on both automatic and human evaluations. On several syntactic and semantic probing tasks, we demonstrate the emergent structural information in the role vectors and improved syntactic interpretability in the TPR layer outputs.[1]

1 Introduction

Abstractive summarization is the task of generating a shorter version of a source text without necessarily reusing the sentences from the original source, while preserving the meaning of its salient contents. It is a complex task that requires: semantic understanding of the source text and reasoning over its lexical units, making inferences about their relation to extract salient facts which are scattered across the long document, as well as generating a concise and coherent sequence of new sentences that covers the salient facts. While humans are remarkably good at this type of reasoning and abstraction, developing models that are capable of extraction, comprehension, abstraction, and reformulation of salient contents has been an open research question.



Original Text (Truncated): Authorities said the incident took place on Sao Joao **beach in Caparica, south-west of Lisbon**. The National Maritime Authority said a middle-aged **man and a young girl died after they were unable to avoid the plane.** [....] Other reports said the victims had been sunbathing when the **plane made its emergency landing**. [...] Video footage from the scene carried by local broadcasters showed a small recreational plane parked on the sand, apparently intact and surrounded by beachgoers and emergency workers. [...]

Reference Summary: A man and a child have been killed after a light aircraft made an emergency landing on a beach in Portugal.

Figure 1: An example document and its one line summary from XSum dataset. Document content that is composed into an abstractive summary is color-coded.

One prominent aspect of abstractive summarization is that models struggle with combining multiple salient aspects in the source text into a coherent and grammatical set of sentences that preserve the original information in the source document. As shown in Fig. 1, these pieces of salient information ("death", "emergency landing", "beach") are often connected by complex syntactic, causal, and temporal relations and are loosely grouped under the main topic of the source document. The transformer models (Vaswani et al., 2017) encode syntactic and semantic information of the input text into a single representation space with the self-attention, and decode the salient aspects into a short summary with the cross-attention. However, despite the large number of training examples, current state-of-the-art transformer based approaches still struggle with systematic generalization of the composition of multiple salient pieces of information.

*Work partially done while at Microsoft Research.

[1] Code and models are available at https://github.com/jiangycTarheel/TPT-Summ

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4780–4793
June 6–11, 2021. ©2021 Association for Computational Linguistics

In this paper, we investigate new types of computational primitives for transformers based on Tensor Product Representations (TPRs) (Smolensky, 1990) which are explicitly-compositional vector embeddings of symbolic structures. A Tensor Product Representation encodes a constituent in a symbolic structure as a composite of a *role*, which encodes the structural information (e.g., the dependency relation with another word), and a *filler*, which encodes the content of the constituent (e.g., the meaning of a word). Analogously, the TP-TRANSFORMER constructs a pair of representations for every token at every layer: a *filler vector* returned by attention and a novel *role vector*. As visualized in Fig. 2, the model then binds the role and filler vectors to produce the output of every token as a TPR. We adapt the TP-TRANSFORMER (Schlag et al., 2019), which was proposed for solving mathematics problems, for the task of abstractive summarization. Unlike the original TP-TRANSFORMER, which directly projects the input representation into a **continuous** role vector space, our model generates the role vectors by attending to a learned dictionary of role embeddings (Palangi et al., 2018). We observe that most learned role attention distributions are approximately one-hot, thus restricting the role vectors to a highly **discrete** space. This structural inductive bias encourages the TP-TRANSFORMER to encode the syntactic information in the discrete roles while isolating the semantics in the continuous fillers.

To test the ability of our TP-TRANSFORMER with discrete roles against the standard Transformer and the TP-TRANSFORMER with continuous roles, we build several models from scratch on a number of summarization datasets spanning different degrees of abstractiveness, output summary lengths, and domains. Our TP-TRANSFORMER significantly outperforms the standard Transformer and the TP-TRANSFORMER with continuous roles on the XSum (Narayan et al., 2018), Wikihow (Koupaee and Wang, 2018), and Arxiv (Cohan et al., 2018) datasets and achieves competitive performance on the CNN/Daily Mail (Hermann et al., 2015; Nallapati et al., 2016) dataset, measured by automatic metrics including ROUGE (Lin, 2004) and METEOR (Denkowski and Lavie, 2014). Our human evaluations on XSum and Wikihow datasets also correlate with the automatic metrics, demonstrating that summaries generated by our TP-TRANSFORMER are indeed better than the Transformer's generations.

Furthermore, to investigate the structural representation that naturally emerges during training and the advantage of having compositional TPR hidden states, we design a suite of decoder probing tasks to explore the information encoded in the role, filler, and TPR space. We adopt the encoder probing task design presented in Tenney et al. (2019b) and create four decoder probing tasks: Part-of-speech tagging (POS), Dependency Labeling (DEP), Semantic Role Labeling (SRL), and Named Entity Labeling (NEL). Our findings collectively show that the decoder's role vectors encode a wealth of syntactic structures, aiding the decoder in deducing the syntactic features (e.g., being a proper noun, being the object of the root predicate) of the next token to be generated. The decoder's filler vectors on the other hand encode more semantic information (e.g., being a person's name). Furthermore, we observe that having the compositional TPR results in a more interpretable final representation than the original Transformer has at every layer, regarding the syntactic features of the next word to be generated. Our results support our hypothesis that by disentangling semantics and syntax, such structured intermediate representations enable the model to better control both the content to be conveyed and the syntactic structure needed to express it, ultimately improving the factuality and grammaticality of the generated summaries.

Our overall contributions are as follows: (1) we present a novel adaptation of the original Transformer architecture that incorporates a dictionary of role embeddings at every layer and generates Tensor Product Representation by binding the role vectors with attention outputs (filler vectors); (2) show that our TP-TRANSFORMER outperforms the Transformer as well as the original TP-TRANSFORMER (Schlag et al., 2019) on several abstractive summarization datasets; and (3) demonstrate the emergent structures in representations by revealing the disentangled syntactic and semantic information encoded in the role and filler spaces.

2 The TP-TRANSFORMER

We build our TP-TRANSFORMER based on the Transformer architecture used in Raffel et al. (2020). A TP-TRANSFORMER encoder applied to a sequence of tokens $i = 1, ..., I$ can be seen as a 2-dimensional lattice of cells (i, l) where i is the

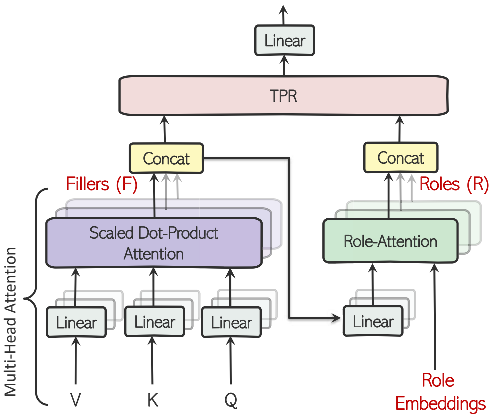


Figure 2: The Filler and Role Binding operation of the TP-TRANSFORMER Model architecture.

position of the input token and $l = 1, ..., L$ are the layer indices. All cells in the encoder have the same architecture and the cells at the same layer share the same weights. We introduce the basic components of a TP-TRANSFORMER cell in Sec. 2.2 and its encoder and decoder cells in Sec. 2.3.

2.1 Tensor-Product Representation Basics

Tensor-Product Representations (TPR; (Smolensky, 1990)) are explicitly-compositional vector embeddings of symbolic structures, where each constituent of the structure is represented as the product of a **role** vector, which encodes its structural information, and a **filler** vector, which contains the content. The TPR of a whole structure is the sum of the representation of its constituents. To represent any 3-digit number using TPRs, we need three role vectors: $\{r(p1)$: Ones place, $r(p2)$: Tens place, $r(p3)$: Hundreds place$\}$ and ten filler vectors f for ten digits. For example, the TPR of the number 985 is $r(p1) \otimes f(5) + r(p2) \otimes f(8) + r(p3) \otimes f(9)$, where $\otimes$ is the tensor product. When representing a number, the role vectors operate similarly as the positional embeddings in a Transformer (Vaswani et al., 2017). However, when representing natural languages, the role vectors need to encode a variety of structural information (e.g., predicate-argument, tense, etc) and thus it is infeasible to hand-design an entire suite of role vectors as we did for numbers. To overcome this challenge, for every token, we dynamically compute its role *vector* from a dictionary of a finite number of role *embeddings* learned with the entire model and treat the self-attention outputs as the fillers. We introduce the full computation procedure in Sec. 2.2.2.

2.2 The TP-TRANSFORMER Cell

Similar to the basic Transformer cell, at every layer, a TP-TRANSFORMER Encoder cell starts with a layer normalization and the multi-head self-attention followed by a residual layer. Then, the cell treats the output vectors as fillers and binds them to role vectors to construct a Tensor Product Representation, which is then passed through the feed-forward network to yield the final states.

2.2.1 Multi-Head Attention

The TP-TRANSFORMER cell adopts multi-head attention (Vaswani et al., 2017) to enable information passing between tokens. At any layer, denote the input vectors as $X \in \mathbb{R}^{k_x \times d_m}$ and the attention target vectors as $Y \in \mathbb{R}^{k_y \times d_m}$, where k_x, k_y are the length of the sequences and d_m is the dimension of the input vectors. In the case of self attention, we have $Y = X$; while for the encoder-decoder cross attention, Y is the encoder's output vectors. We first apply layer normalization (Ba et al., 2016) to get $\hat{X}$ and then linearly project it to the query, key, and value vectors for each attention head $h = 1, ..., H$.

$$\begin{aligned} Q^h &= \hat{X}\mathbf{W}_q^h + \mathbf{b}_q^h \\ K^h &= Y\mathbf{W}_k^h + \mathbf{b}_k^h \\ V^h &= Y\mathbf{W}_v^h + \mathbf{b}_v^h \end{aligned} \tag{1}$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d_m \times d_k}$. The attention output matrix $\bar{V}$ for each head h is computed as:

$$\bar{V} = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

where d_k is the dimension of the key vectors K. The multi-head attention output O is the concatenation of the attention outputs from all heads followed by another linear projection $\mathbf{W}_o \in \mathbb{R}^{d_m \times d_m}$. We end the Multi-head Attention with a residual connection with the layer input vectors $\hat{X}$:

$$\mathrm{MHAttn}(X, Y) = \hat{X} + [\bar{V}_1, ..., \bar{V}_H]\mathbf{W}_o \tag{3}$$

where $\bar{V}_h$ is the attention output for the h-th head.

2.2.2 Computing TPRs

Role Embeddings. Following Palangi et al. (2018), but departing from Schlag et al. (2019), every layer of our TP-TRANSFORMER is equipped with a dictionary $\mathbf{r} \in \mathbb{R}^{N_r \times d_r}$ of N_r distinct role embeddings with a dimension of d_r. Each role embedding $\mathbf{r}_n$, n=1,. . . ,N_r, is randomly initialized in the entire network. The role embeddings are normalized before computing role vectors:

$$\hat{\mathbf{r}}_n = \frac{\mathbf{r}_n}{\|\mathbf{r}_n\|_2} \text{ for } n = 1, ..., N_r \tag{4}$$

At each layer, the model computes a weighted combination of these role embeddings $\hat{\mathbf{r}}$ to form a unique *role vector* for every token.

Multi-Head TPR Binding. Our *filler vectors* correspond to the multi-head attention output $F = \mathrm{MHAttn}(X)$ (Eqn. 3). The filler F of each token has a corresponding role vector R. We first compute the $R^h \in \mathbb{R}^{d_r}$ at every head $h = 1, ..., H$ as a weighted average of the normalized role embeddings $\hat{\mathbf{r}}$. We then concatenate the $R^h \in \mathbb{R}^{k_x \times d_r}$ of H heads to get the multi-head role vectors $R \in \mathbb{R}^{k_x \times (d_r \cdot H)}$ for all k_x tokens. We define this process formally as:

$$\begin{aligned} R^h &= \mathrm{softmax}(F\mathbf{W}_r^h)\hat{\mathbf{r}} \\ R &= [R^1, ..., R^H] \end{aligned} \tag{5}$$

where $\mathbf{W}_r \in \mathbb{R}^{d_m \times N_r}$ is the linear projection that computes the attention scores over the role embeddings for every token.[2]

We use a Hadamard product[3] to approximate the full Tensor product in binding the role vectors R with filler vectors F, as it was shown in Schlag et al. (2019) that using the Hadamard products allows learning an optimial lower-rank approximation of the full TPRs. The binding operation is followed by an addition with the unbound fillers (F) to return the residual TPR vectors.

$$\mathrm{TPR}(F) = R \odot F + F \tag{6}$$

2.2.3 Residual Feed-forward Layer

The feed-forward layer of a cell consists of a linear projection followed by a ReLU activation and a second linear projection. The feed-forward output is then added to the input vectors:

$$\mathrm{FF}(X) = X + \mathrm{ReLU}(X\mathbf{W}_g + \mathbf{b}_g)\mathbf{W}_f + \mathbf{b}_f \tag{7}$$

Here, $\mathbf{W}_g \in \mathbb{R}^{d_m \times d_f}$, $\mathbf{b}_g \in \mathbb{R}^{d_f}$, $\mathbf{W}_f \in \mathbb{R}^{d_f \times d_m}$, $\mathbf{b}_f \in \mathbb{R}^{d_m}$, and x is the function argument.

2.3 TP-TRANSFORMER Encoder & Decoder

Given the components of our basic TP-TRANSFORMER cell in the previous section, we now describe how we construct the TP-TRANSFORMER encoder and decoder.

First, the self-attention and the encoder-decoder cross-attention for every token can be computed as:

$$\begin{aligned} \mathrm{Self}(X) &= \mathrm{TPR}(\mathrm{MHAttn}(X, X)) \\ \mathrm{Cross}(Y, H) &= \mathrm{TPR}(\mathrm{MHAttn}(Y, H)) \end{aligned} \tag{8}$$

where H is the output of the encoder's final layer. Y represent the previous layer's output vectors of either the partially (so-far) decoded sequence at test time or the masked reference summary at training time. The encoder and decoder's operations at every layer can be summarized as:

$$\begin{aligned} \mathrm{Encode}(X) &= \mathrm{FF}(\mathrm{Self}(X)) \\ \mathrm{Decode}(H, Y) &= \mathrm{FF}(\mathrm{Cross}(\mathrm{Self}(Y), H)) \end{aligned} \tag{9}$$

After L layers of encoding and decoding, the final distribution of the i-th output token is given by:

$$\hat{z}_i = \mathrm{softmax}(\mathbf{E}^T y_{i,L}) \tag{10}$$

where $Y_L = \mathrm{Decode}(H, Y_{L-1})$ are the decoder's output states at the last layer and $\mathbf{E}$ is the tied input/output word embeddings.

3 Summarization Experiments

3.1 Abstractive Summarization Datasets

We train our models on four English abstractive summarization datasets varying the level of abstractiveness (explained below) and the length of summaries, as well as input domain.

XSum (Narayan et al., 2018) consists of 227k BBC articles from 2010 to 2017 concerning various subjects along with professionally written single-sentence summaries. Its summaries cover a wide variety of syntactic structures (relative clause, etc) and relations (causal, temporal, etc).

Wikihow (Koupaee and Wang, 2018) is a dataset consisting of instructions from the WikiHow.com website. Each of 200k examples has multiple instruction-step paragraphs, each paired with a summarizing sentence. The task is to generate the concatenated summaries of all paragraphs.

[2] We set $d_r \cdot H = d_m$ so that the multi-head role vectors R have the same dimension as F.

[3] The Hadamard (or elementwise) product is the diagonal of the full tensor product.

Datasets	Summary
XSum	Luxury fashion designer Burberry has returned to profit after opening new stores and spending more on online marketing.
Wikihow	Build a trustworthy bond with your piggy. Research different training methods. Choose the training method that works best for you and your guinea pig. Gather the materials that you will need for training.
Arxiv (Abbreviated)	We study the phase behavior of a nematic liquid crystal confined between a flat substrate with strong anchoring and a patterned substrate whose structure and local anchoring strength we vary. [. . .] In addition the effective energy method allows one to determine the energy barriers between two states in a bistable nematic device .
CNN/DM	Mentally ill inmates in Miami are housed on the "forgotten floor". Judge Steven Leifman says most are there as a result of "avoidable felonies". While CNN tours facility, patient shouts: "I am the son of the president".

Table 1: Example summaries from XSum, Arxiv, Wikihow, and CNN/Daily Mail datasets. Text segments directly extracted from the source document are underlined.

Datasets	Split	# beam	Transformer	TPT-c (Schlag et al., 2019)	TPT-d (Ours)
XSum	Dev	1	33.34/12.07/26.47/22.28	30.73/10.38/24.39/21.14	**34.61/13.13/27.59/23.43**
		4	34.48/13.08/27.29/24.59	31.83/11.28/25.11/22.39	**35.70/14.11/28.38/25.80**
	Test	1	33.22/11.90/26.32/23.02	30.74/10.23/24.32/21.11	**34.62/12.98/27.49/24.38**
		4	34.46/12.97/27.21/24.42	32.01/11.26/25.19/22.45	**35.84/14.06/28.40/25.79**
Wikihow	Dev	1	33.11/11.90/25.46/19.00	28.44/7.65/20.07/16.38	**34.12/12.36/26.02/20.16**
		4	35.85/13.32/26.83/21.57	29.98/8.34/20.70/17.95	**36.54/13.69/27.21/22.53**
	Test	1	33.40/12.18/25.66/19.31	28.63/7.82/20.23/16.49	**34.19**/12.47/**25.99/20.23**
		4	35.91/13.49/27.01/21.57	30.13/8.50/20.78/18.11	**36.70**/13.75/**27.36/22.53**
Arxiv	Dev	1	35.08/10.13/31.86/19.91	32.27/7.50/29.34/17.72	**35.91/10.32/32.55/20.82**
		4	37.95/11.48/34.03/23.31	34.45/8.40/30.91/20.17	**38.35**/11.56/**34.32/23.74**
	Test	1	35.00/9.98/31.79/19.72	32.46/7.53/29.47/17.75	**35.82/10.12/32.46/20.65**
		4	38.01/11.33/34.02/23.19	34.68/8.50/31.15/20.17	**38.36**/11.43/**34.29/23.61**
CNN/DM	Dev	1	40.56/18.18/37.73/31.91	39.66/17.45/36.99/31.15	40.61/18.17/31.77/31.35
		4	41.97/19.23/38.84/34.55	41.49/18.83/38.45/34.14	41.81/19.11/38.73/34.49
	Test	1	39.83/**17.63**/37.02/31.75	39.10/16.96/36.41/31.15	39.63/17.35/36.80/31.57
		4	41.22/**18.70**/38.09/34.50	40.68/18.19/37.70/33.99	41.01/18.38/37.91/34.34

Table 2: Automatic evaluation results on the dev/test set of **XSum**, **Arxiv**, **Wikihow**, and **CNN/Daily Mail** dataset. The results in every cell represent **F1** variant of **ROUGE-1/ROUGE-2/ROUGE-L/METEOR** scores. The best ROUGE scores with a statistically significant advantage, and the best METEOR scores with at least 0.3 advantage are bolded.

Arxiv (Cohan et al., 2018) is a long document summarization dataset of scientific publications from arXiv.org (113k). The task is to generate the abstract from the paper body.

CNN/Daily Mail (Hermann et al., 2015; Nallapati et al., 2016) dataset contains 93k articles from CNN and 220k articles from the Daily Mail. Every article is accompanied by a few human-written bullet points about its content. We use the non-anonymized version used in See et al. (2017).

Dataset Abstractiveness. We show a summary from each of these four datasets in Table 1. According to the comparison made by Zhang et al. (2020) using the coverage and density measures (Grusky et al., 2018), the XSum and Wikihow datasets are more abstractive than the others since their summaries rarely contain large chunks of words overlapping with the source documents. CNN/Daily Mail is the least abstractive of the four. Furthermore, in most cases, a sentence in a CNN/Daily Mail summary only refers to a single sentence from the source document as suggested in Lebanoff et al. (2019), while a sentence in an XSum or Wikihow summary usually aggregates information from multiple source sentences.

3.2 Experimental Setup

The Transformer and the two TP-TRANSFORMERS all have 6 layers, 8 heads per layer, dimension per head d_k=64, model dimension d_m=512, and feed-forward dimension d_f=2048 for the encoder and decoder. Our TP-TRANSFORMER with discrete roles has N_r=50 role embeddings of dimension d_r=64 at every layer. For each dataset above, we train the all three models from scratch using an Adafactor Optimizer (Shazeer and Stern, 2018) with square root learning rate decay and dropout rate of 0.1. We evaluate the models using automatic metrics including ROUGE F1 score and METEOR.

3.3 Results

We report automatic metric scores from our evaluated models in Table 2. We refer to the TP-TRANSFORMER, with freely-generated continuous role vectors (no role dictionary) (Schlag et al.,

Datasets	Models	Grammar	Coherency	Faithfulness	Saliency	Repetition	Overall
XSum	Transformer wins	39	48	43	**50**	38	48
	TP-TRANSFORMER wins	**47**	48	**46**	47	**42**	**52**
	Tie / No agreement	34	24	31	23	40	20
Wikihow	Transformer wins	45	45	43	**54**	48	43
	TP-TRANSFORMER wins	**48**	45	**46**	47	48	**59**
	Tie / No agreement	27	30	31	19	24	18

Table 3: Human Evaluation results on 120 random samples from the XSum (Narayan et al., 2018) and Wikihow (Koupaee and Wang, 2018) test sets. The best numbers with an advantage of at least 5 points are underlined.

2019) as TPT-c, and our own TP-TRANSFORMER with a discrete set of role embeddings as TPT-d. On the XSum, Arxiv, and Wikihow datasets, our TP-TRANSFORMER (TPT-d) outperforms the original Transformer on all metrics. On the CNN/Daily Mail dataset, both models obtain similar performance across all metrics. On every dataset, the TPT-c model which excels on the mathematics dataset, is the worst among the three models being compared. This suggests that continuous role vectors are not suited to the summarization tasks.

As we explain in Sec. 3.1, CNN/Daily Mail is the least abstractive one among the four datasets. In contrast, summaries from the XSum and Wikihow datasets contain very few n-grams (n>2) that can be copied from the source documents and thus push the model's ability to compose a coherent summary restating the salient aspects from the source. Furthermore, as illustrated in Table 1, the XSum summary contains a long sentence that combines multiple pieces of information scattered through the long source document. These facts are usually connected by syntactic, temporal[4], or causal[5] relations and thus the model must be able to connect and reason across these salient facts and then convert them into a coherent sentence that faithfully reflects the original facts and their relations. We argue that the compositional TPR can better enable these abilities required for XSum, where we indeed find that our TP-TRANSFORMER achieves the largest advantage over the Transformer among its improvements on all datasets.

3.4 Human Evaluation

We conduct human evaluation to compare the summaries generated by the Transformer and our TP-TRANSFORMER. We randomly sample 120 examples from the test sets of XSum and Wikihow datasets with the beam-searched model summaries. We refer to appendix for the complete setup. As shown in Table 3, on the XSum dataset, summaries generated by the TP-TRANSFORMER are significantly better in grammar. This corroborates our claim that having the TPR can improve the model's ability to follow the correct syntax in composing the summary. On the Wikihow dataset, the Transformer receives more votes in regarding the saliency. However, our TP-TRANSFORMER maintains an advantage in grammar and achieves significantly better overall preferences.

Unfaithful XSum Examples It is well-known that the XSum dataset contains a portion of unfaithful reference summaries that mention facts not included in the source article (Durmus et al., 2020; Maynez et al., 2020). Therefore, we are interested to find out whether our TP-TRANSFORMER is better than the baseline only at expressing the faithful content or it can also generate some external, "unfaithful" facts that the baseline can't cover. To answer this question, we randomly sample 100 examples from the XSum dev set and manually examine the source document, reference summary, and the two generated summaries. Among these 100 examples, we identify 71 examples whose reference summary includes "unfaithful" facts that are not mentioned in the source. In 21 out of 71 examples, the Transformer baseline manages to generate some "unfaithful" facts that match those in the reference while our TP-TRANSFORMER achieves this in 17 examples. Such "unfaithful" facts that were recovered by the models include the full name of a person when only the last name is mentioned in the source, the political party or the job title of a person, each of which can be attributed to at least one example seen by models during the training. Therefore, we believe that both models learn to draw external information from its memory of the seen examples, while our TP-TRANSFORMER doesn't do better than the baseline Transformer at referring to external facts to obtain higher ROUGE scores.

[4] "returned to profit **after** opening new stores"

[5] "Opening new stores and spending more on online marketing" caused "more profit".

4 Probing Experiments

Probing is a method to test whether some particular information is present in the model's encodings. To achieve this, an auxiliary classifier is trained to predict specified linguistic features from the model's internal representations. We probe different components (roles, filler, TPRs) in our TP-TRANSFORMERs as well as the attention+residual outputs (equivalent to the filler) of the Transformer to assess the naturally emergent structures encoded in the role vectors and the effectiveness of the TPR in the decoding process. By conducting the probing experiments, we aim to (1) provide some insights and evidence of the different information encoded by the role and filler vectors; and (2) explain the ROUGE advantage of our TP-TRANSFORMER by showing that its output representation can better encode the linguistic structural information concerning multiple probing tasks.

4.1 Decoder Probing Tasks

When studying an encoder, previous works probe its i-th intermediate representation at a certain layer for information about the i-th input token For a decoder, however, we probe its i-th representation for clues about the i-th token it generates given the $i-1$ previously generated tokens as the input. Intuitively, we are probing for the decoder's internal decision about the syntactic roles and semantic content of this token before it was ultimately selected. Based on encoder probing tasks used by Tenney et al. (2019b), we select and adapt four tasks to probe our decoders.

Part-of-speech tagging (POS) is the syntactic task of assigning tags such as noun (singular/mass noun: NN, proper noun: NNP, etc), verb (past tense: VBD, past participle: VBN, etc), adjective (comparative: JJR, etc), etc. to each token i. We let $s_1 = [i, i+1)$ be a single token, and seek to predict its POS tag.

Dependency labeling (DEP) seeks to predict the functional relationships of one token relative to another: e.g. is it a modifier-head relationship, a subject-verb relationship, etc. We take $s_1 = [i, i+1)$ to be a single token and $s_2 = [j, j+1)$ to be its syntactic head, and seek to predict the dependency relation between tokens i and j.

Semantic role labeling (SRL) is the task of imposing predicate-argument structure onto a sentence. We let $s_1 = [i_1, j_1)$ represent a known predicate (e.g., "push") and $s_2 = [i_2, j_2)$ represent a known argument ("Peter") of that predicate, and seek to predict the role that the argument s_2 fills–e.g. ARG0 (agent, the pusher) vs. ARG1 (patient, the pushee).

Tasks	Layer	Transformer	TPT-d (Ours)
POS	1	-/**58.4**/58.4	36.1/57.1/58.2
	2	-/**65.4**/**65.4**	43.6/63.5/64.4
	3	-/**68.6**/68.3	50.4/67.4/68.5
	4	-/70.7/70.7	50.4/70.8/**72.1**
	5	-/72.5/72.5	53.4/**73.3**/**73.9**
	6	-/73.3/73.3	56.0/**73.9**/**74.5**
DEP	1	-/78.1/78.1	53.1/**78.8**/**78.9**
	2	-/85.0/85.0	59.9/84.8/84.7
	3	-/87.1/87.1	66.7/87.4/87.3
	4	-/87.4/87.4	62.9/**88.3**/**88.2**
	5	-/85.0/85.0	64.8/**88.3**/**87.6**
	6	-/86.1/86.1	60.8/**86.8**/**86.6**
SRL	1	-/78.2/78.2	73.1/78.5/78.4
	2	-/79.0/79.0	73.8/**79.8**/79.3
	3	-/79.6/79.6	73.8/79.9/80.0
	4	-/78.7/78.7	73.1/**80.1**/**80.2**
	5	-/77.7/77.7	72.9/**79.9**/**79.8**
	6	-/78.1/78.1	71.8/**79.2**/78.2
NEL	1	-/59.7/59.7	33.3/**61.4**/**60.8**
	2	-/67.6/67.6	37.6/**68.1**/**68.2**
	3	-/69.6/69.6	41.5/**70.9**/**71.0**
	4	-/71.8/71.8	43.6/**74.3**/**73.2**
	5	-/72.3/72.3	44.7/**76.3**/**75.7**
	6	-/73.3/73.3	42.2/**76.1**/**73.8**

Table 4: Results (F1 scores) of probing different intermediate representations in decoders trained on XSum dataset. The results in every cell are presented in the order of **roles**, **fillers**, and **final representations**. The best numbers with an advantage of at least 0.5 F1 scores are bolded.

Named entity labeling (NEL) is the task of predicting the category of an entity. The categories include *PERSON*, *LOCATION*, *ORGANIZATION*, etc. We let $s_1 = [i, j)$ represent a known entity span and seek to predict its type.

4.2 Experimental Setup

As there is no existing dataset for probing decoders, we create our own training and evaluation data by running off-the-shelf models on the summarization datasets. Specifically, to probe a decoder trained on the XSum dataset on the POS task, we run an POS tagger on the reference summaries from the XSum training set and the model-generated summaries for the XSum dev set to create the ground-truth labels for the training set and model-specific dev set. We restore the model trained on a summarization dataset and freeze its parameters. Following Tenney et al. (2019b), we train a span convolution layer followed by a 2-layer MLP on top of the target representation that project it onto the output label space.

4.3 Results

Table 4 presents the results of probing the decoder of a TP-TRANSFORMER trained on the XSum (Narayan et al., 2018) dataset. Note that the Transformer doesn't have role vectors. It directly outputs the vector after the multi-head attention and the residual layer. Therefore, its fillers and final representations are equivalent.

The decoder role vectors can encode grammatical information while the filler vectors represent the semantics. We first focus on the results of POS tagging probing task. Overall, we see a trend of increasing scores as the representations get closer to the final step of computing the distribution over the vocabulary. This implies that, as the computation progresses through the layers, the generated representations are gradually deciding the POS tag of the next word to generate. Next, we observe that the role vectors (the 1st number in the TPT-d column) of TP-TRANSFORMER encode a considerable amount of information about the POS tag of the next word generated. Additionally, because the job of deducing the POS tag of the next word is partially shared by the role vectors, the filler vectors' performance degrades compared to the Transformer. This pattern demonstrates that the TP-TRANSFORMER's decoder is representing the next word to be generated as a composite of structural information encoded in the role vectors and semantic contents encoded in the filler vectors. Comparing the fillers (the 2nd number in TPT-d column) with the TPR (the 3rd number in the TPT-d column) of TP-TRANSFORMER, we see that the TPRs, which bind the roles and fillers, outperform the roles and fillers alone at every layer. This indicates that the TPR effectively aggregates the linguistic knowledge encoded in the roles and fillers into a shared space, where the POS tag of the next word can be decoded more easily than in the role space or filler space alone. Last, the final representations of TP-TRANSFORMER achieve higher F1 scores than their counterparts in the Transformer in the last three layers. This demonstrates the benefits of having the TPR in interpreting the POS tag of the word to be generated.

When we consider the Dependency labeling (DEP) and Semantic role labeling (SRL) tasks, we observe that our TP-TRANSFORMER's final representations consistently beat the Transformer across all layers, with only one exception in the DEP task at the layer 2. We also observe that the TP-TRANSFORMER's advantage becomes larger in the last three layers except for the final layer in SRL task. However, unlike in the POS task, the TPR only achieve similar F1 scores to the fillers.

Finally, in the Named entity labeling (NEL) task which is considered to require more semantic information rather than syntax, the role vectors' performance is poorer than their performance in the three syntactic tasks. For example, the TP-TRANSFORMER's final representations at layer 6 obtain similar F1 scores in the POS and NEL tasks (74.5 VS 73.8), but its role vectors only achieve a 42.2 F1 score in the NEL tasks compared to the 56.0 in the POS. However, even though the role vectors encode little information about the named entity type of the next token to be generated, the TPR still strongly outperforms the Transformer's filler-only representation at every layer. We argue that although the syntactic information encoded in the role vectors is not enough to predict the correct named entity, it is still a beneficial complement to the knowledge encoded in the distributed filler vectors in certain situations. For example, whether the subject "Chanel" refers to a *PERSON* or an *ORGANIZATION* could depend on its syntactic role and its relation to other words in the sentence (e.g., whether it is the subject or object of "wears") .

Compositional representations improves interpretability of the representations. Overall, by probing the different intermediate representations of the TP-TRANSFORMER and the Transformer, we show that having the compositional TPR results in more interpretable final representations at every layer regarding the syntactic features of the next word to be generated. Considering automatic evaluations generated summaries in Sec. 3.3, we argue that this compositionality in learned representation and its syntactic interpretability enable the decoder to take better control of the syntactic structure of the generation when assembling multiple distant facts, and thus lead to summaries of better quality.

4.4 Discrete Role Vectors

During the training of our TP-TRANSFORMER models on the summarization datasets, we observe that most learned role attention distributions are approximately one-hot, as more than 90% of the role attention distributions (as computed in Eqn. 5) have a maximum score larger than 0.98. Because each role vector is the concatenation of H vectors, each selected from N_r role embeddings, the completely one-hot role attentions will yield $(N_r)^H$ possible role vectors. Therefore, the learned, approximately one-hot role vectors span $(N_r)^H$ discrete subspaces, each of which only covers the close proximity of a concatenation of H role embeddings. This finding indicates that as we represent the role vectors as multi-head attention over a learnable dictionary of role embeddings, the structural inductive bias: (1) pushes the role vector space to be even more discrete, and (2) induces the syntactic structures encoded in these discrete role vectors. We also believe there is a connection between the above two effects, as the structural, syntactic information favors a lower-dimensional or even discrete space while the distributed, semantic information favors a higher-dimensional space.

5 Related Work

Explicit TPR Structures in Neural Networks While earlier TPR work based on (Smolensky, 1990) focused on computability rather than learnability questions, recently TPRs have been incorporated into several recurrent deep learning models in order to solve various NLP tasks including Part-of-Speech tagging, constituency parsing, image captioning (Huang et al., 2018, 2019), question answering (Palangi et al., 2018; Schlag and Schmidhuber, 2018), and natural-to-formal language generation (program synthesis) (Chen et al., 2020). Most recently, TPRs have been introduced into Transformer architectures, starting with Schlag et al. (2019) which introduced the TP-TRANSFORMER to improve the performance and interpretability of mathematical problem solving models. This model generated continuous role vectors by directly projecting from layer inputs, whereas our model indexes from a dictionary of role embeddings to form the role vectors which are shown to reside in a highly discrete space.

Structured Representations for Abstractive Summarization Compared to the extractive methods, abstractive summarization models usually fail to show extractive properties, and have tendency to copy text from the source (See et al., 2017; Paulus et al., 2018; Pasunuru and Bansal, 2018; Celikyilmaz et al., 2018). More recent approaches that use standard transformers deal with this issue by introducing hierarchical structures to encode local and global information separately focusing on only the semantic content (Liu and Lapata, 2018, 2019). To preserve salient source relations and generate abstractive summaries of the source document, previous work infused models with semantic parsers: while Song et al. (2018) introduces a new structure-infused copy mechanism that combines the source syntactic structure with the copy mechanism, Liao et al. (2018) uses abstract meaning representations (AMR). While these approaches require that the document sentence semantic parsers are provided beforehand, our models can implicitly learn to approximate the syntactic structure and semantic content in their representations.

6 Conclusion

In this work, we enrich the Transformer model with the structured Tensor Product Representation for abstractive summarization tasks. We represent every token as a pair of role and filler vectors. We show that our TP-TRANSFORMER with discrete roles outperforms Transformer and TP-TRANSFORMER with continuous roles on several abstractive summarization datasets, in both metrics scores and human evaluation. We further demonstrate the syntactic structures encoded in the role vectors and show the improved syntactic interpretability in our model's hidden states.

7 Ethics Statement

In this work we propose a new encoder-decoder modeling architecture and build several models to benchmark our new architecture with baseline architectures on several open source summarization datasets.

Intended use. Our architecture is designed to build models of abstractive summarization. Potentially our architecture could be used to train models for summarizing any type of company internal datasets (e.g., internal documents, reports, meetings, legal forms, etc.) to further improve the productivity and efficiency of the users in their daily activities without needing to read long documents.

Failure mode. Even though our models yield factually consistent summaries, as judged by human evaluation, they can still generate factually inconsistent summaries or sometimes hallucinate information that the source document does not include. This might be due to the bias or noise in the training data. Model builders wanting to use our architecture to build models on their company internal datasets should build models with consideration of intellectual properties and privacy rights.

Misuse Potential. We note the models to be built with our architecture should be used with careful consideration. The generated summaries produced by our models are not controlled and use generative approaches, therefore, they could generate unreliable text. Researchers working on abstractive summarization should focus on generating factually correct, ethical and reliable text. If our models are trained on news datasets, a careful consideration should be made on factuality of the generated text and measures have been taken to prevent model hallucinations.

Acknowledgments

We thank the reviewers for their helpful comments. This work was partially supported by NSF-CAREER Award 1846185 and a Microsoft Investigator Fellowship.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, USA.

Kezhen Chen, Qiuyuan Huang, Hamid Palangi, Paul Smolensky, Kenneth D Forbus, and Jianfeng Gao. 2020. Mapping natural-language problems to formal-language solutions using structured neural representations. In Proceedings of the ICML.

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.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In Proceedings of the ninth workshop on statistical machine translation, pages 376–380.

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

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5055–5070, Online. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In Proceedings of Workshop for NLP Open Source Software (NLP-OSS), pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In Advances in neural information processing systems, pages 1693–1701.

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

Qiuyuan Huang, Li Deng, Dapeng Wu, Chang Liu, and Xiaodong He. 2019. Attentive tensor product learning. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01):1344–1351.

Qiuyuan Huang, Paul Smolensky, Xiaodong He, Li Deng, and Dapeng Wu. 2018. Tensor product generation networks for deep NLP modeling. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1263–1273, New Orleans, Louisiana. Association for Computational Linguistics.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. arXiv preprint arXiv:1810.09305.

Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019. Scoring sentence singletons and pairs for abstractive summarization. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2175–2189, Florence, Italy. Association for Computational Linguistics.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for multi-document summarization. In Proceedings of the 27th International Conference on Computational Linguistics, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out: Proceedings of the ACL-04 workshop, volume 8. Barcelona, Spain.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT's linguistic knowledge. In Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 241–253, Florence, Italy. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. In Transactions of the Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In Transactions of the Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In Association for Computational Linguistics (ACL) System Demonstrations, pages 55–60.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1906–1919, Online. Association for Computational Linguistics.

R. Thomas McCoy, Tal Linzen, Ewan Dunbar, and Paul Smolensky. 2019. RNNs implicitly implement tensor-product representations. In International Conference on Learning Representations.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In Computational Natural Language Learning.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

H. Palangi, P. Smolensky, X. He, and L. Deng. 2018. Question-answering with grammatically-interpretable representations. In AAAI.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multireward reinforced summarization with saliency and entailment. In 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, USA.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In 6th International Conference on Learning Representations, Vancouver, BC, Canada.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67.

Imanol Schlag and Jürgen Schmidhuber. 2018. Learning to reason with third order tensor products. In Advances in Neural Information Processing Systems, pages 10002–10013.

Imanol Schlag, Paul Smolensky, Roland Fernandez, Nebojsa Jojic, Jürgen Schmidhuber, and Jianfeng Gao. 2019. Enhancing the transformer with explicit relational encoding for math problem solving. arXiv preprint arXiv:1910.06611.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In International Conference on Machine Learning, pages 4596–4604. PMLR.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artificial intelligence, 46(1-2):159–216.

Kaiqiang Song, Lin Zhao, and Fei Liu. 2018. Structure-infused copy mechanisms for abstractive summarization. In Proceedings of the 27th International Conference on Computational Linguistics, pages 1717–1729, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Paul Soulos, Tom McCoy, Tal Linzen, and Paul Smolensky. 2019. Discovering the compositional structure of vector representations with role learning networks. arXiv preprint arXiv:1910.09113.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In International Conference on Learning Representations.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In Proceedings of the 37 th International Conference on Machine Learning.

Appendix

A TP-Transformer Architecture

We provide Fig. 3 to visualize the full encoder-decoder architecture of our TP-Transformer.

B Summarization Experimental Setup

The Transformer and the two TP-Transformers all have 6 layers, 8 heads per layer, dimension per head d_k=64, model dimension d_m=512, and feed-forward dimension d_f=2048 for the encoder and decoder. Our TP-Transformer with discrete roles has N_r=50 role embeddings of dimension $d_r = 64$ at every layer. We search the optimal N_r from $\{20, 50, 100, 200, 1000\}$ and select the one with the best validation set performance. For each of the dataset above, we train the all three models from scratch using an Adafactor Optimizer (Shazeer and Stern, 2018) with square root learning rate decay and dropout rate of 0.1. The total number of parameter of the Transformer, TP-Transformer with continuous roles, and our TP-Transformer with discrete roles are 60506880, 65234688, and 64258080 respectively. Every model is trained on 4 NVidia V100 GPUs (32GB) with a batch size of 32 per GPU.

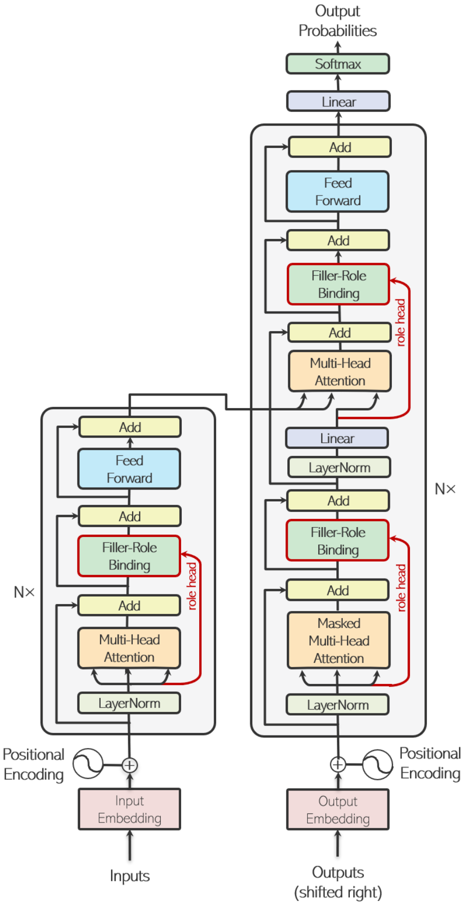


Figure 3: TP-Transformer model architecture.

B.1 Human Evaluation

We conduct human evaluation to compare the summaries generated by the original Transformer and our TP-Transformer. We randomly sample 120 examples from the test sets of XSum and Wikihow datasets with the corresponding beam-searched model summaries. For every example, we show the source document, the reference summary, and two model summaries shuffled in order to three human evaluators, and ask them to decide which summary is better in six different aspects: grammar, coherency, factuality, saliency, redundancy, and an overall preference. We then take the majority vote of every examples from its three human annotators.

C Probing Experimental Setup

As there is no existing dataset for probing decoders, we create our own training and evaluation data by running off-the-shelf models on the summarization datasets. Specifically, to probe a decoder trained on the XSum dataset on the POS task, we run an POS tagger on the reference summaries from the XSum training set and the model-generated summaries for the XSum dev set to create the ground-truth labels for the training set and model-specific dev set. We use Stanford CoreNLP (Manning et al., 2014) to get the labels for POS, dependency and named entity probing tasks. We use a BERT-base model (Devlin et al., 2019) from AllenNLP (Gardner et al., 2018) to get the ground-truth labels for SRL. We restore the model trained on a summarization dataset and freeze its parameters during the probing. We simply add a linear layer on top of the target representation to project it onto the output label space.

D Related Works

Implicit TPR Encodings in Neural Networks McCoy et al. (2019) showed that, in GRU-based (Cho et al., 2014) encoder-decoder networks performing fully-compositional string manipulations, trained on extensive data that fully exemplifies the range of possible compositions, the medial encoding between encoder and decoder could be extremely well approximated by TPRs. Soulos et al. (2019) presented the ROLE model that learns its own role scheme to optimize the fit of a TPR approximation to a given set of internal representations in a pre-trained target neural network, removing the need for human-generated hypotheses about the role schemes the network might be implementing. While this work successfully interprets the Tensor Product Representation in fully compositional tasks, abstractive summarization, as well as most other NLP tasks, are only partially compositional and the symbolic rules in language are much more complex. Although these two works showed that Tensor Product Representation can naturally emerge in a unstructured representations, we argue that standard models only learn TPRs without any special bias to do so when the compositional structure of the task is simple and blatant and when the training set makes that painfully clear by providing a good sample of the compositional possibilities. That is possible for the simple string tasks addressed in the two previous works, but not in the abstractive summarization as well as other real-world NLP tasks, where we show that having explicit TPR helps in modeling the structure information.

Sequence Models Encode Implicit Structure. Several recent works have shown that the pretrained Transformer-based BERT (Devlin et al., 2019) embeddings implicitly encode structural linguistic relations with various interpretation methods. The first, and also the most popular method (Tenney et al., 2019a) is to train an auxiliary classifier to probe the model's hidden representations for specific linguistic information. The second method (Lin et al., 2019) abstracts the Transformer model into a graph based on the attention weights, and explores syntactic structures based on the graph's structure. The third method (Hewitt and Manning, 2019) sees the hidden representations of BERT as in a metric space and directly connect the distance between representations to the distance between elements in a symbolic structure (e.g., a dependency-parse tree) to extract the implicit structures without extra training. The interpretation method deployed here falls under the probing family, but future work will also pursue other interpretation methods.

E Examples of Generated Summary

We provide examples generated by the Transformer baseline and our TP-TRANSFORMER in Table 5 and Table 6.

Datasets	Summary
Source	Nottinghamshire Police said it would expand its categories to include misogynistic incidents.It means abuse or harassment which might not be a crime can be reported to and investigated by the police, and support for the victim put in place.Nottingham Women's Centre said it hopes it will help give more victims the courage to report incidents.Chief Constable Sue Fish claimed it will make the county a safer place for women. </br>"What women face, often on a daily basis, is absolutely unacceptable and can be extremely distressing," she said. </br>"Nottinghamshire Police is committed to taking misogynistic hate crime seriously and encourages anyone who is affected by it to contact us without hesitation. </br>"Work on the idea first started with the Nottinghamshire Safer for Women Conference last year, co-hosted by the police with the Nottingham Women's Centre.BBC TV reporter Sarah Teale was harassed in the street while reporting on the conference.The force defines misogyny hate crime as: "Incidents against women that are motivated by an attitude of a man towards a woman and includes behaviour targeted towards a woman by men simply because they are a woman. </br>"The classification now means people can report incidents which might not be considered to be a crime and the police will investigate.Nottingham Women's Centre has been helping train call centre, force control staff and officers on the beat to recognise misogynistic hate crime and ways to tackle it.These officers will also examine if and how a victim can be supported or if anything can be done to help prevent them being targeted again.Domestic abuse will not be recorded as a misogyny hate crime because it has its own procedure, the force said.Melanie Jeffs, centre manager at Nottingham Women's Centre, said: "We're pleased to see Nottinghamshire Police recognise the breadth of violence and intimidation that women experience on a daily basis in our communities. </br>"She added: "Recording this as a hate crime will give us a detailed picture of how often, when and where it is happening. </br>It has been very difficult to build that picture before but we will now get detailed data to analyse. </br>"Showing that the police take it seriously will also give people the confidence to come forward and report offences. </br>"A crime that the victim or any other person perceives to be motivated by hostility or prejudice towards any aspect of a person's identity.Police forces in England, Wales and Northern Ireland annually monitor five strands of hate crime:Forces can include their own definition of a hate crime with several recently adding sub cultures.
Reference	Harassment of women is to be recorded as a hate crime in a bid to tackle sexist abuse.
Transformer	Women who commit misogyny and harassed a woman are to be asked to take part in an anti-Semitic conference.
TP-TRANSFORMER	A police force has launched a national drive to combat misogyny and hate crimes in Nottinghamshire.

Table 5: An example from the XSum dev set and the summaries generated by the Transformer baseline and TP-TRANSFORMER.

Datasets	Summary
Source	Sixty patrol boats will protect the UK's two new aircraft carriers which are due to arrive at Portsmouth Naval Base in 2017.The first carrier, HMS Queen Elizabeth, is expected to be operational in 2020. </br>"We are going to see a bigger Royal Navy and the flagship... will be here in Portsmouth," Michael Fallon said.The 60 Pacific 24 rigid-hulled inflatable boats will be built by BAE systems to "guard the carriers in the harbour and our new frigates and destroyers", Mr Fallon said.He said they will also enhance security by providing a rapid response in rescue, anti-piracy and counter-narcotics missions in the area.Mr Fallon said: "Through the defence review, defence spending is going to go up every April for the rest of this parliament.He said as part of the larger investment, the government will also be able to provide the new aircraft carriers with sufficient fighter jets. </br>"We have said we will maintain a minimum fleet of 19 destroyers and frigates, but as the older frigates are retired we also hope to add a lighter frigate between the offshore patrol vessel and Type 26 and to build more of those as well. </br>"Mr Fallon's visit to Portsmouth Naval Base comes as work has begun to rebuild the jetty for the arrival of HMS Queen Elizabeth in 2017.Floating cranes are also dredging Portsmouth harbour to prepare deeper channels for the aircraft carriers to sail from the base, which are the largest ships ever built for the Royal Navy. </br>"This is a huge financial investment in making sure the channel is wide enough, in enlarging the jetty here so they can take the carriers and in making sure the carriers are properly guarded," Mr Fallon said.Taller than Nelson's Column and longer than Portsmouth's Spinnaker Tower laid on its side, the new carriers will displace 65,000 tonnes of water.To make room for the carriers three million cubic metres of clay, sand and gravel will be removed from a two-mile stretch of Portsmouth Harbour covering an area the size of 200 football pitches.
Reference	Increased spending will result in a "bigger" Royal Navy, the defence secretary has said, as he announced a new £13.5m shipbuilding contract.
Transformer	The Royal Navy's new aircraft carriers will be patrolling the Portsmouth harbour this year, the defence secretary has said.
TP-TRANSFORMER	Plans for a new Royal Navy aircraft carriers to be built in Portsmouth have been unveiled.

Table 6: An example from the XSum dev set and the summaries generated by the Transformer baseline and TP-TRANSFORMER.