Logical Transformers: Infusing Logical Structures into Pre-Trained Language Models

Borui Wang[1*] **Qiuyuan Huang**[2] **Budhaditya Deb**[2] **Aaron Halfaker**[2]
Liqun Shao[2] **Daniel McDuff**[3†] **Ahmed Hassan Awadallah**[2] **Dragomir Radev**[1]
Jianfeng Gao[2]
[1]Yale University [2]Microsoft Research [3]University of Washington
borui.wang@yale.edu

Abstract

Natural language contains rich logical structures and logical information, and correctly detecting and accurately understanding these logical structures and information underlying natural language texts is very crucial for NLP models' performance on many important NLU and NLG tasks. Existing pre-trained language models based on the transformer architecture mostly adopt a classical design for constructing their input embeddings that ignores the logical structures underlying natural language texts, thus limiting their ability to better capture and encode key logical information in the input sequences. To overcome such limitations, in this paper we first propose a novel approach to construct *logic-aware input embeddings* for transformer language models through a combination of logic detection, logic mapping and hierarchical logical projections, and then develop a corresponding new modeling paradigm that can upgrade existing transformer language models into *logical transformers* to boost their performance on different NLU and NLG tasks. Our empirical experiments on four important and challenging NLU and NLG tasks demonstrate that our proposed logical transformer language models can achieve superior performance over their baseline transformer models through a deeper understanding of the logical structures of texts.

1 Introduction

Natural language contains rich logical structures and logical information (Lakoff, 1970; Van Benthem, 1986) that are crucial to a deep and accurate understanding of its meaning. Therefore, the ability to correctly detect and accurately understand the logical structures and information within natural language texts is very crucial for NLP models' performance on many important Natural Language Understanding (NLU) and Natural Language Generation (NLG) tasks.

The types of logics contained in natural language are very diverse, including not only mathematically well-defined propositional logic and first-order logic (Lu et al., 2022; Han et al., 2022), but also more general types of natural and structural logical relationships that people frequently use in natural language texts to convey and communicate their ideas and meanings more effectively and clearly.

In recent years we have witnessed huge progress and success in many fields of natural language processing brought about by the introduction of all different kinds of pre-trained language models (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020; Yang et al., 2019; Clark et al., 2020; Lewis et al., 2020; Raffel et al., 2020; Zhang et al., 2020) based on the transformer architecture (Vaswani et al., 2017). Most existing pre-trained language models adopt the classical approach for constructing the input embeddings that are fed into the encoder parts of the language models, which can be summarized as the summation of the following three key components (Devlin et al., 2019):

(1) Token Embeddings - that are used to encode and represent the semantics and meaning of each token in the vocabulary;

(2) Position Embeddings - that are used to encode the positional information of each token in the input sequence;

(3) Segment Embeddings - that are used to indicate which segment of the input sequence each token belongs to.

This classical design of the input embeddings has been proven to be very effective at capturing important semantic and positional features from

[*] This work was done when Borui Wang was a research intern at Microsoft Research.

[†] This work was done when Daniel McDuff was at Microsoft Research.

Findings of the Association for Computational Linguistics: ACL 2023, pages 1762–1773
July 9-14, 2023

natural language texts and helping pre-trained language models to learn good contextualized representations of the input textual sequences (Devlin et al., 2019). However, it also has a very important limitation - it doesn't consider or try to explicitly encode the logical structures underlying the text inputs, which are also very crucial for the deep and accurate understanding of the meaning of the text inputs.

Therefore, in order to overcome this limitation and to enable pre-trained language models to better capture and understand the important logical structures underlying natural language texts, in this paper we propose a novel approach to construct **logic-aware input embeddings** for transformer-based pre-trained language models and a corresponding new modeling framework that can upgrade existing transformer language models into **logical transformers** to boost their performance on different NLU and NLG tasks.

Our new approach consists of two major modules: (1) logic detection and mapping, and (2) multi-layer hierarchical logical projections. It has the following key advantages:

- *Strong Generalizability*: Our proposed new approach for constructing logic-aware input embeddings doesn't alter the main architecture of transformer language models and only modifies the input embeddings at the front end before they are fed into the encoder part of the language models. Therefore, our new approach enjoys strong generalizability and can be smoothly added to many different pre-trained language models based on the transformer architecture.

- *Consistent Boost in Model Performance*: Our proposed new approach is empirically shown to consistently boost the performance of different transformer language models on different NLU and NLG tasks.

- *Negligible Increase in Model Size*: Our proposed new approach will only increase the number of parameters of transformer language models by a negligible amount.

- *Low Overhead on Training Time*: Our proposed new approach will not significantly increase the training time of transformer language models by a large amount. The majority of the overhead in training time will come from the initial text processing steps of logic detection and logic mapping, which only need to be executed once before the actual training epochs start.

2 Logical Relationships and Keywords

In this work, we consider *logical relationships* in natural language texts as the underlying relationships among different language constituents that carry meaningful information regarding logical understanding and reasoning of the texts. In natural language, such logical relationships are usually indicated by logically-connective keywords and phrases. In this paper, we define a taxonomy of 12 most commonly seen types of logical relationships and their corresponding sets[1] of logical keywords (including phrases[2]) for natural language[3]:

1. **Conjunction**: a conjunction logical relationship indicates that the two language constituents involved are presented jointly in addition to each other. Its logical keywords are:

   `and, as well, as well as, also, at the same time.`

2. **Disjunction**: a disjunction logical relationship indicates that the two language constituents involved are presented alternatively next to each other. Its logical keyword is:

   `or.`

3. **Negation**: a negation logical relationship indicates that the meaning of the language constituent mapped by it is negated. Its logical keywords are:

   `not, no, none, n't, nothing.`

4. **Conditional**: a conditional logical relationship indicates that the content of one language constituent is the premise of the content of another language constituent. Its logical keywords are:

[1]The sets of logical keywords listed here are not necessarily the most exhaustive sets that contain all possible keywords in each category, but rather serve as the preliminary and exemplar sets that can already cover the majority of the most frequently appearing logical keywords in real-world texts. These sets are open to extension.

[2]For conciseness, in this paper we will use the term *'logical keywords'* to refer to both *logical keywords* and *logical key phrases*.

[3]Here the logical keywords are all defined in English, but similar categorization of logical relationships and sets of logical keywords can also be defined in other languages as well.

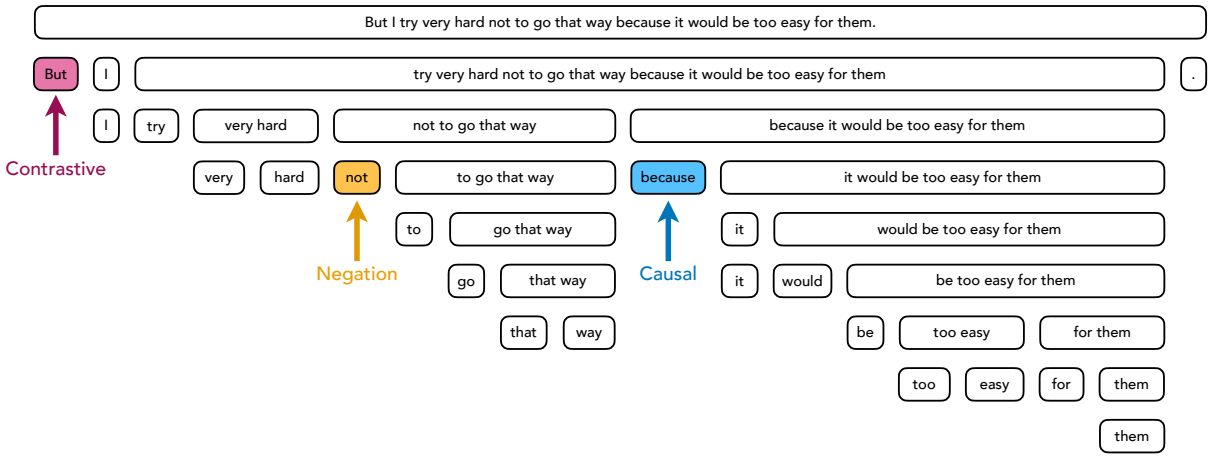


Figure 1: The constituency parse tree for the example sentence *'But I try very hard not to go that way because it would be too easy for them.'* generated by the Berkeley Neural Parser (Kitaev and Klein, 2018).

`if, as long as.`

5. **Negative Conditional**: a negative conditional logical relationship indicates that the negation of the content of one language constituent is the premise of the content of another language constituent. Its logical keywords are:

   `unless, otherwise.`

6. **Analogy**: an analogy logical relationship indicates that the content of one language constituent is analogous to the content of another language constituent. Its logical keywords are:

   `as if, as though, just as, just like, likewise, similarly.`

7. **Comparative**: a comparative logical relationship indicates that the two language components involved are presented in comparison to each other. Its logical keywords are:

   `but, however, in comparison, while, yet, rather than, unlike, on the other hand, in contrast, contrary to, on the contrary.`

8. **Adversative**: an adversative logical relationship indicates that the content of one language constituent is adversative to the content of another language constituent. Its logical keywords are:

   `nevertheless, nonetheless, notwithstanding, although, though, despite, despite of, in spite of, regardless of, albeit.`

9. **Temporal**: a temporal logical relationship indicates that the content of one language constituent signifies the time when the content of another language constituent takes place. Its logical keywords are:

   `during, after, in, when, since, before, as, as soon as, while, then, until, meanwhile.`

10. **Causal**: a causal logical relationship indicates that the content of one language constituent is the cause or reason for the content of another language constituent. Its logical keywords are:

    `because, thanks to, since, as a result, in order to, as, therefore, hence, so that, due to, thus, consequently, thereby, now that.`

11. **Progression**: a progression logical relationship indicates that the content of one language constituent goes one step further on top of the content of another language constituent. Its logical keywords are:

    `moreover, furthermore, in addition, besides.`

12. **Example**: an example logical relationship indicates that the content of one language constituent exemplifies the content of another language constituent. Its logical keywords are:

    `for example, as an example, like, such as, for instance, including.`

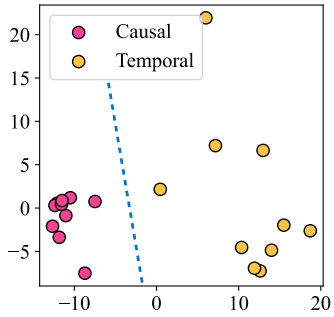


Figure 2: 2-dimensional PCA (Hotelling, 1933) projection of the contextualized last-layer hidden state vectors for 20 randomly sampled different occurrences of the logical keyword *'since'* encoded by the ALBERT model (Lan et al., 2020). Occurrences with the causal logical meaning *'because'* is colored in pink, and occurrences with the temporal logical meaning *'from a time in the past'* is colored in yellow.

As an example, we sample a news article from the training set of the CNN/Dailymail dataset (Nallapati et al., 2016) and manually annotate the appearances of the above defined types of logical relationship in the article. See Figure 6 for the annotation of the logical relationships in this example article, where the logical keywords associated with different logical relationships are highlighted with different colors.

2.1 Categorization of Logical Relationships

According to how many logical components (in the form of text spans) are associated with each logical keywords and how different logical components are mapped by the logical keywords, we categorize the set of all logical keywords into three different categories:

2.1.1 Unary Logical Relationships

The logical keywords indicating unary logical relationships are those that each only maps to one single logical component (text span). For example, most keywords of negation relationship and example relationship are indicating unary logical relationships, such as *not*, *for example*, *such as*, etc.

2.1.2 Intrinsically-Mapped Binary Logical Relationships

The logical keywords indicating intrinsically-mapped binary logical relationships are those that each maps to two separate logical components (text spans) that are both contained within the parent sentence constituent of the logical keyword itself. For example, most keywords of conjunction relationship and disjunction relationship are indicating intrinsically-mapped binary logical relationships, such as *and*, *as well as*, *or*, etc.

Algorithm 1 Logic Detection and Mapping

Input: Sentence s
Constituency parser $\mathcal{C} : \mathcal{S} \rightarrow \mathcal{T}$
Set of logical keywords $\mathcal{K}$
Output: List of logic mapping dictionaries $\mathcal{M}$

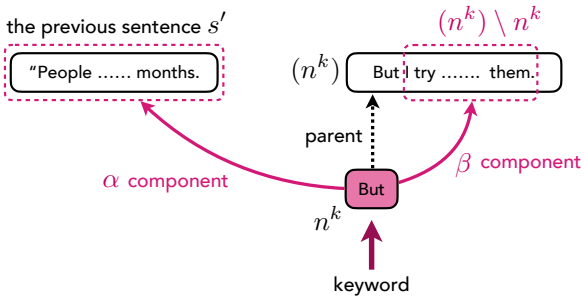
Run $\mathcal{C}$ over s to obtain its constituency parse tree $T(s)$
$\mathcal{N}_{key}(s) \leftarrow [\,]$
$\mathcal{M} \leftarrow [\,]$
for each constituent node n **in** $T(S)$ **do**
  if $str(n) \in \mathcal{K}$ **then**
    $\mathcal{N}_{key}(s) \leftarrow \mathcal{N}_{key}(s) + n$
for n^k **in** $\mathcal{N}_{key}(s)$ **do**
  $\mathcal{D}^k \leftarrow \{\}$
  $\mathcal{D}^k[\text{'keyword'}] = str(n^k)$
  if $str(n^k) \in \mathcal{K}^{\mathcal{U}}$ **then**
    $\mathcal{D}^k[\text{'}\alpha\text{'}] = str(\text{pa}(n^k) \setminus n^k)$
  else if $\text{str}(n^k) \in \mathcal{K}^{\mathcal{B}_{\text{in}}}$ **then**
    Use $str(n^k)$ to segment $str(\text{pa}(n^k))$ into 3 segments: $str(\text{pa}(n^k)) = A + str(n^k) + B$
    $\mathcal{D}^k[\text{'}\alpha\text{'}] = A$, $\mathcal{D}^k[\text{'}\beta\text{'}] = B$
  else if $\text{str}(n^k) \in \mathcal{K}^{\mathcal{B}_{\text{ex}}}$ **then**
    if $\exists$ $\text{pa}(\text{pa}(n^k))$ **then**
      $\mathcal{D}^k[\text{'}\alpha\text{'}] = str(\text{pa}(\text{pa}(n^k)) \setminus \text{pa}(n^k))$
      $\mathcal{D}^k[\text{'}\beta\text{'}] = str(\text{pa}(n^k) \setminus n^k)$
    else if $\exists$ another sentence s' right before s **then**
      $\mathcal{D}^k[\text{'}\alpha\text{'}] = s'$, $\mathcal{D}^k[\text{'}\beta\text{'}] = str(\text{pa}(n^k) \setminus n^k)$
    else
      $\mathcal{D}^k[\text{'}\alpha\text{'}] = \emptyset$, $\mathcal{D}^k[\text{'}\beta\text{'}] = str(\text{pa}(n^k) \setminus n^k)$
  $\mathcal{M} \leftarrow \mathcal{M} + \mathcal{D}^k$
return $\mathcal{M}$

2.1.3 Extrinsically-Mapped Binary Logical Relationships

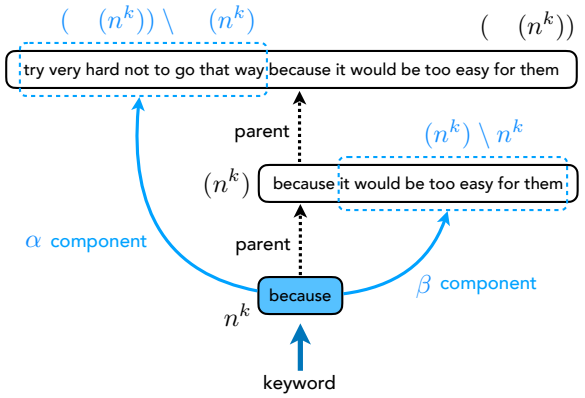
The logical keywords indicating extrinsically-mapped binary logical relationships are those that each maps to two separate logical components (text spans) where one is contained within the parent sentence constituent of the logical keyword itself while the other is outside (usually appears before) the span of this parent sentence constituent. For example, most keywords of conditional, comparative, temporal and causal relationships are indicating extrinsically-mapped binary logical relationships, such as *if*, *but*, *during*, *because*, etc.

3 Logic Detection and Mapping

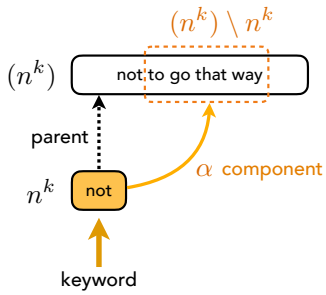
In this section, we describe our logic detection and mapping module based on keyword detection and constituency parsing. For each sentence s in the source text, we first perform constituency parsing (Kitaev and Klein, 2018) over s to obtain its constituency parsing tree $T(s)$. In this paper, we use the Berkeley Neural Parser (Kitaev and Klein, 2018) to perform constituency parsing.



(a) Execution of Algorithm 1 over the logical keyword *'But'*.



(b) Execution of Algorithm 1 over the logical keyword *'because'*.



(c) Execution of Algorithm 1 over the logical keyword *'not'*.

Figure 3: An example execution of Algorithm 1 on the example sentence *'But I try very hard not to go that way because it would be too easy for them.'* over the three detected logical keywords 'But', 'because', 'not'.

Then we search through all the constituent nodes in $T(s)$ to detect the ones that exactly matches the keyword strings of the logical keywords as defined in Section 2. Let $\mathcal{N}_{key}(s)$ denote the set of constituent node in $T(s)$ that matches logical keywords. Then for each logical keyword node $n^k \in \mathcal{N}_{key}(s)$, we fetch its parent constituent node $\text{pa}(n^k)$. Now we have three different cases:

1. If n^k corresponds to a unary logical relationship (i.e. negation and example), then the α component of n^k is detected as: $\text{pa}(n^k) \setminus n^k$.

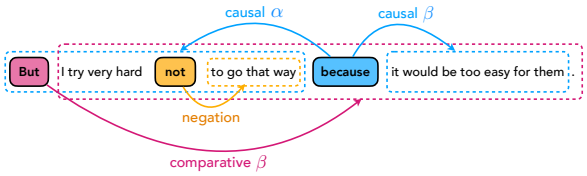


Figure 4: Detected logical structure for an example sentence *'But I try very hard not to go that way because it would be too easy for them.'* taken from the example article in Figure 6.

2. If n^k corresponds to a binary logical relationship and the relationship is intrinsically mapped, then $\text{str}(\text{pa}(n^k))$ will be divided by $\text{str}(n^k)$ into three different segments: $\text{str}(\text{pa}(n^k)) = A + \text{str}(n^k) + B$. Now the α component of n^k is detected as A and the β component of n^k is detected as B.

3. If n^k corresponds to a binary logical relationship and the relationship is extrinsically mapped, then the α component of n^k is detected as: $\text{pa}(\text{pa}(n^k)) \setminus \text{pa}(n^k)$, and the β component of n^k is detected as: $\text{pa}(n^k) \setminus n^k$.

Our proposed methods for logic detection and mapping described above are summarized in Algorithm 1. See Figure 3 for an example of executing Algorithm 1 on an example sentence taken from the example article in Figure 6, based on the constituency parsing tree depicted in Figure 1.

3.1 Sense Disambiguation of Logical Keywords

In English, certain logical keywords have multiple meanings and can indicate different logical relationships under different contexts. For example, the logical keyword '*since*' has two different meanings: (1) '*from a time in the past*', which indicates a temporal logical relationship; (2) '*because*', which indicates a causal logical relationship. In our categorization of logical relationships and keywords (described in Section 2), there are a total of 3 keywords that can have multiple logical meanings: *since*, *as*, and *while*. Therefore, in order to increase accuracy of our proposed logic detection module, we need to first perform accurate logical sense disambiguation when we detect these logically ambiguous keywords.

In our empirical experiments over a set of randomly sampled sentences that contain ambiguous logical keywords, each manually-labelled with its ground-truth logical relationship under the context,

we found that different uses of ambiguous logical keywords have very strong clustering tendency and are largely linearly-separable under the contextualized encoding of transformer language models. For example, we use the ALBERT model (Lan et al., 2020) to encode 20 different occurrences of the logical keyword *'since'* randomly sampled from the CNN/Dailymail dataset (Nallapati et al., 2016), and project the last-layer hidden state vectors for these 20 '*since*' onto their first two principal components using Principal Component Analysis (PCA) (Hotelling, 1933), which is depicted in Figure 2. As we can see from Figure 2 the contextualized embeddings of the logical keyword '*since*' are largely linearly separable between the two different logical meanings.

Therefore, in order to improve the accuracy of our logic detection module, we first manually collected logical relationship annotations for the set of ambiguous logical keywords in English. Then we encode them using the ALBERT model (Lan et al., 2020) and train individual support vector machine (SVM) (Cortes and Vapnik, 1995) classifiers for each of the ambiguous logical keywords to accurately disambiguate their different logical meanings.

4 Logical Transformers

4.1 Logical Embedding Vectors

The major new parameters that we introduce in our proposed modeling framework of logical transformers are a set of parametrized and trainable logical embedding vectors. These logical embedding vectors share the same dimensionality, but their dimensionality doesn't necessarily equal to the dimensionality of the transformer language model's token embedding vectors. Below we describe how to construct these logical embedding vectors in detail.

First of all, the 12 types of logical relationships we defined in Section 6 can be classified into two different categories: (1) *'unary logical relationship'* that maps to only one logical component; (2) *'binary logical relationship'* that maps to two logical components. More specifically, **negation** and **example** are *unary logical relationships* and all the other 10 types are *binary logical relationships*.

For each *unary logical relationship* $\mathcal{U}$, we construct two parametrized logical embedding vectors: $v^{\mathcal{U}}_{key}$ and $v^{\mathcal{U}}$. In the logical embedding layer of $\mathcal{U}$, we assign $v^{\mathcal{U}}_{key}$ to each token detected to be part of an appearance of some logical keyword in $\mathcal{U}$, and assign $v^{\mathcal{U}}$ to all the tokens that are within some text span mapped by some logical keyword in $\mathcal{U}$.

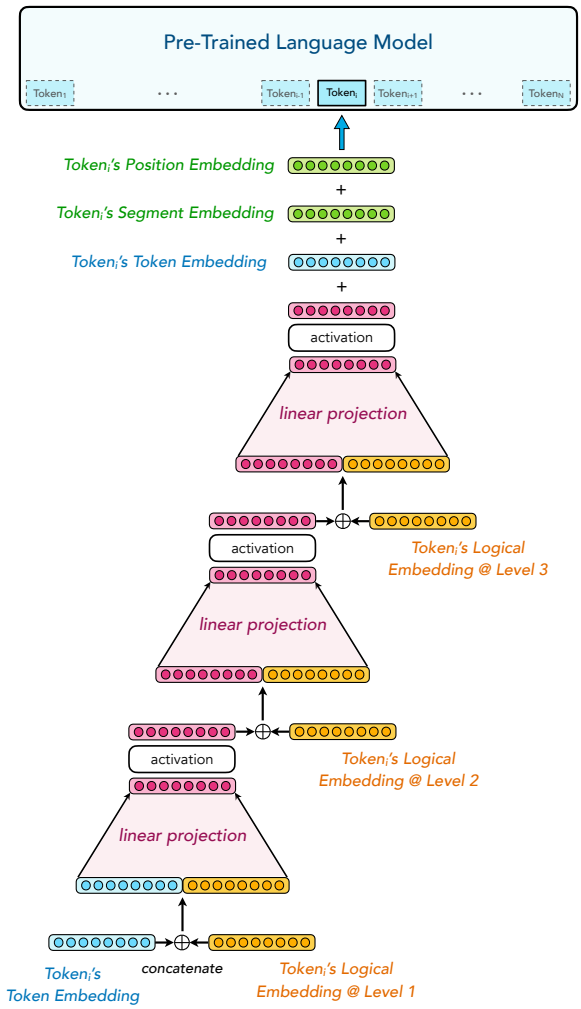


Figure 5: Illustration of our proposed multi-layer hierarchical logical projections for an example token with logic depth K = 3.

For each *binary logical relationship* $\mathcal{B}$, we construct three parametrized logical embedding vectors: $v^{\mathcal{B}}_{key}$, $v^{\mathcal{B}}_{\alpha}$ and $v^{\mathcal{B}}_{\beta}$. In the logical embedding layer of $\mathcal{B}$, we assign $v^{\mathcal{B}}_{key}$ to each token detected to be part of an appearance of some logical keyword in $\mathcal{B}$, assign $v^{\mathcal{B}}_{\alpha}$ to all the tokens that are within some left text span mapped by some logical keyword in $\mathcal{B}$, and assign $v^{\mathcal{B}}_{\beta}$ to all the tokens that are within some right text span mapped by some logical keyword in $\mathcal{B}$.

And finally we construct another special parametrized logical embedding vector $v^{\mathcal{E}}$ that corresponds to *empty logical association*. For each token that doesn't belong to any logical relationships in a logical embedding layer, it will be assigned $v^{\mathcal{E}}$

Logical Embeddings	Sentence Tokens																		
	But	I	try	very	hard	not	to	go	that	way	bec.	it	wld	be	too	easy	for	them	.
Comparative	key	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β	β
Causal	-	-	α	α	α	α	α	α	α	α	key	β	β	β	β	β	β	β	-
Negation	-	-	-	-	-	key	α	α	α	α	-	-	-	-	-	-	-	-	-

Table 1: Illustration of our proposed multi-layer logical embeddings for an example sentence *'But I try very hard not to go that way because it would be too easy for them.'* taken from the example article in Figure 6. The assignment of logical embedding vectors are based on the parsed logical structure depicted in Figure 4. In the second row the token 'because' is abbreviated into 'bec.' and the token 'would' is abbreviated into 'wld' due to space limit.

for this layer. See Table 1 for a concrete example of assigning multiple layers of logical embedding vectors to tokens in an input sequence based on the results of logic detection and mapping.

Therefore, based on the 12 different types of logical relationships that we defined in Section 6, we will construct a total of $2 \times 2 + 10 \times 3 + 1 = 35$ different logical embedding vectors for our logical transformers.

4.2 Multi-Layer Hierarchical Logical Projections

Now we describe how to compute the *logic-aware input embeddings* through *multi-layer hierarchical logical projections* using the set of logical embedding vectors that we defined in Section 4.1. Let N_{logic} denote the dimensionality of the logical embedding vectors, and let N denote the dimensionality of the token embedding vectors of the transformer language model. We first define a parametrized and trainable linear transformation layer $\mathcal{L}$ that projects a $(N + N_{logic})$-dimensional vector into an N-dimensional vector.

Then for each token t in the input token sequence, we collect all the logical embedding vectors assigned to it during the logic detection and mapping process and sort them in order according to their associated logical keywords' depth in the constituency parse tree of the input sentence. Let's denote this sorted set of all the logical embedding vectors assigned to token t as: $\{v_t^1, ..., v_t^K\}$, where K is the maximum number of logical layers to be considered and should be treated as a hyperparameter.

Now let's denote the original token embedding vector for token t as w_t, then to compute a logic-aware token embedding vector w_t^{logic} for t, we first initialize $u_t^0 = w_t$, and then recursively apply the following computation[4]:

$$u_t^i = f(\mathcal{L}(u_t^{i-1} \oplus v_t^i)),$$

for $i = 1, ..., K$, where $\oplus$ denotes vector concatenation and f is some non-linear activation function, such as GELU (Hendrycks and Gimpel, 2016). Then we have:

$$w_t^{logic} = w_t + u_t^K.$$

Now let p_t denote the position embedding vector of token t and s_t denote the segment embedding vector of token t, then the final logic-aware input embedding vector for each token t in the input sequence would be computed as: $w_t^{logic} + p_t + s_t$. Then at the front end of our proposed logical transformers, we use these logic-aware input embeddings to replace the traditional input embeddings and feed them into transformer encoders to help language models better encode and learn logical information from the textual inputs. See Figure 5 for an illustration of multi-layer hierarchical logical projections for an example token with logic depth K = 3.

4.3 Model Training

During the training of our proposed logical transformers, we set both the set of 35 logical embedding vectors and the linear transformation layer $\mathcal{L}$ to be fully parametrized and trainable, and then initialize them with random values. All these added new parameters will be updated together with the original trainable parameters in the transformer language models during the model training process.

4.4 Negligible Increase in Model Size

The only new parameters introduced in our proposed logical transformers, compared with their corresponding baseline transformer language models, are the set of 35 logical embedding vectors and the linear transformation linear $\mathcal{L}$ used in hierarchical logical projections. Let N_{logic} denote the dimensionality of the logical embedding vectors, then

[4]This series of (linear projection + nonlinear activation) can also be replaced by a series of multilayer perceptrons.

	ReClor	LogiQA	DREAM
Model	Acc	Acc	Acc
RoBERTa-large	62.6	35.3	82.1
Logical-RoBERTa-large	**67.4**	**37.8**	**84.9**

Table 2: Our NLU experiment results on the ReClor dataset (Yu et al., 2020), the LogiQA dataset (Liu et al., 2020) and the DREAM dataset (Sun et al., 2019). *Acc* denotes *accuracy percentage*. The higher value in each pair of comparison is highlighted in **bold**.

	DialogSum			
Model	**R-1**	**R-2**	**R-L**	**R-LSum**
BART-large	46.10	20.32	38.04	40.98
Logical-BART-large	**46.97**	**20.69**	**38.33**	**41.30**

Table 3: Our NLG experiment results on the DialogSum dataset (Chen et al., 2021). The higher value in each pair of comparison is highlighted in **bold**.

the total increase in model size can be calculated as: $N_{logic} \times 35 + (N + N_{logic}) \times N_{logic} + N_{logic} = N_{logic}^2 + N \cdot N_{logic} + 36N_{logic}$.

For all the recently proposed transformer language models, this increase in model size is rather small and negligible compared with their very large number of parameters. For example, for the RoBERTa-large model (Liu et al., 2019), its total number of parameters is 355M and the dimensionality of its embedding vectors is 1024. If we set N_{logic} = 1024 as well, then after we use our proposed new modeling paradigm to upgrade RoBERTa-large into Logical-RoBERTa-large, the percentage of increase in model size is only: $(1024^2 + 1024 \times 1024 + 36 \times 1024) \div 355\text{M} \approx 0.601\%$, which is almost negligible. This efficiency in model size guarantees that the logical transformers take roughly the same amount of computation time during both training and inference as their baseline transformer language models.

5 Experiments

In order to evaluate our proposed logical transformer architecture's performance boost on different NLU and NLG tasks with different transformer language models, in our experiments, we test it on three NLU datasets and one NLG dataset.

5.1 Natural Language Understanding Tasks

In the NLU part of our experiments, we test the RoBERTa model (Liu et al., 2019) and our Logical-RoBERTa model on three logically-challenging natural language understanding tasks over three corresponding datasets: (1) *reading comprehension* on the ReClor dataset (Yu et al., 2020); (2) *question answering* on the LogiQA dataset (Liu et al., 2020); and (3) *dialogue-based reading comprehension* on the DREAM dataset (Sun et al., 2019). All of these three datasets require logical reasoning.

5.2 Natural Language Generation Task

In the NLG part of our experiments, we test the BART model (Lewis et al., 2020) and our Logical-BART model on the task of *dialogue summarization* over the DialogSum (Chen et al., 2021) dataset.

5.3 Results

The results of our three NLU experiments are shown in Table 2, and the results of NLG experiment are shown in Table 3. As we can see from Table 2 and Table 3, the accuracy scores and the ROUGE scores of our logical transformer language models are consistently higher than their corresponding baseline transformer language models across all the different NLU and NLG tasks. This consistent boost demonstrates that the important logical structures and information extracted and captured by our proposed logical transformers are indeed very effective and useful in further improving transformer language models' performance on logically-challenging NLU and NLG tasks.

6 Related Work

Recently there has been increasing interest in improving pre-trained language models' logical reasoning ability (Xu et al., 2022; Pi et al., 2022). For example, Lu et al. (2022) proposed a new method for parsing natural language into the forms of propositional logic and first-order logic using dual reinforcement learning. Pi et al. (2022) proposed a new unsupervised adversarial pre-training method, called LogiGAN, in order to enhance language models' abilities of logical reasoning. Xu et al. (2022) proposed a new Logiformer architecture based on a two-branch graph transformer network to improve language models' performance on interpretable logical reasoning.

In contrast to these previous work that mostly focus on introducing new training methods or constructing complex model architectures, our proposed method in this paper only modifies the input embeddings and is thus more straightforward

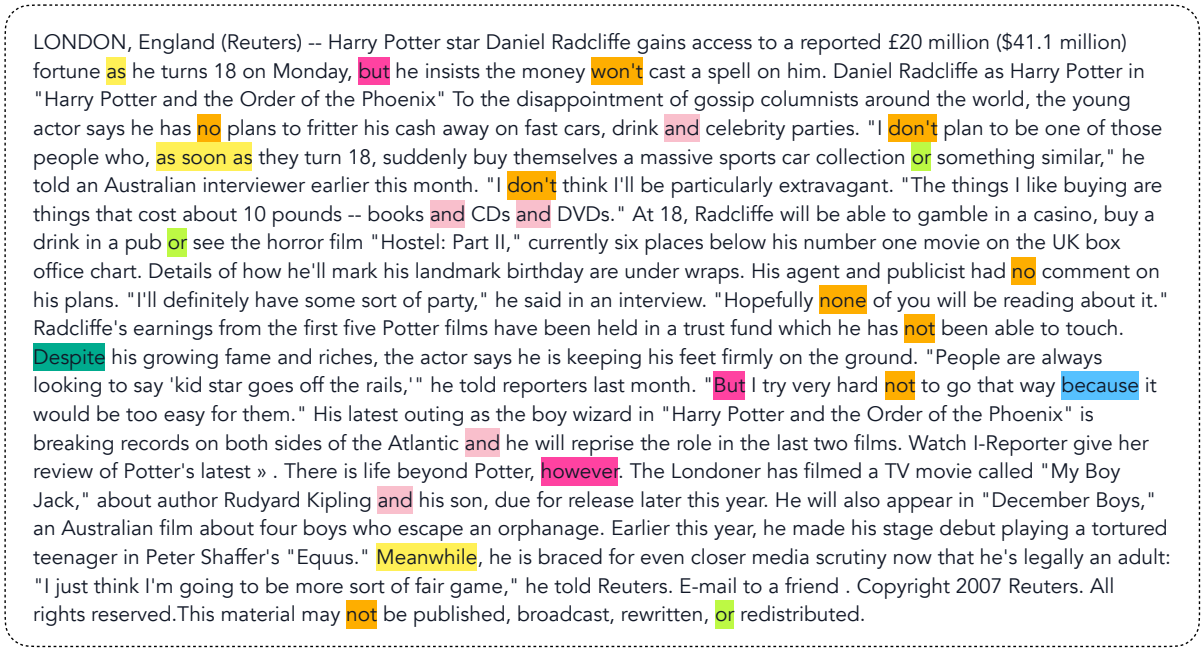


Figure 6: Detected logical keywords in an example article from the CNN/Dailymail dataset (Nallapati et al., 2016). It contains 7 different types of logical relationships: conjunction, disjunction, negation, comparative, adversative, temporal, and causal.

and easily generalizable to different types of transformer language models.

7 Conclusion

In this paper we introduced a new modeling paradigm for transformer language models that detects and extracts important logical structures and information from input texts and then integrates them into the input embeddings through carefully designed multi-layer hierarchical logical projections to infuse logical structures into pre-trained language models. Our empirical experiments on four important and challenging NLU and NLG tasks showed that our proposed logical transformer language models consistently perform better than their corresponding baseline transformer language models through a deeper understanding of the key logical structures underlying natural language texts.

8 Limitations

In theory, the method proposed in this paper can be applied to different types of transformer language models for both pre-training and fine-tuning. Due to limit of computational resource, we currently haven't had the chance to test our proposed method in the very promising setting of large-scale language model pre-training yet. In future work, we plan to further test our proposed logical transformer architecture on large-scale language model pre-training to see how much performance boost it can achieve.

References

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations (ICLR)*.

Corinna Cortes and Vladimir Naumovich Vapnik. 1995. Support-vector networks. *Machine Learning*, 20:273–297.

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.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq R. Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir R. Radev. 2022. Folio: Natural language reasoning with first-order logic. *ArXiv*, abs/2209.00840.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Harold Hotelling. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3622–3628. International Joint Conferences on Artificial Intelligence Organization. Main track.

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

Xuantao Lu, Jingping Liu, Zhouhong Gu, Hanwen Tong, Chenhao Xie, Junyang Huang, Yanghua Xiao, and Wenguang Wang. 2022. Parsing natural language into propositional and first-order logic with dual reinforcement learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5419–5431, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Xinyu Pi, Wanjun Zhong, Yan Gao, Nan Duan, and Jian-Guang Lou. 2022. Logigan: Learning logical reasoning via adversarial pre-training. *ArXiv*, abs/2205.08794.

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.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231.

Johan Van Benthem. 1986. *Essays in logical semantics*. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Fangzhi Xu, Jun Liu, Qika Lin, Yudai Pan, and Lingling Zhang. 2022. Logiformer: A two-branch graph transformer network for interpretable logical reasoning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1055–1065, New York, NY, USA. Association for Computing Machinery.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*.

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 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 8

☒ A2. Did you discuss any potential risks of your work?
There are no potential risks of our work.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix A

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3 and Section 5

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.