TRANSFORMER/SYMBOLIC HYBRIDIZATION WITH FINITE-STATE TRANSDUCERS

WORK IN PROGRESS

Emmanuel Roche
Clover.AI
roche@cloverai.net

June 12, 2023

ABSTRACT

Deep learning has long been opposed to the kind of symbolic processing that was developed during the first decades of natural language processing. Geoffrey Hinton declared in his Turing price acceptance speech (2018) that deep learning has won and symbolic processing has lost. We hypothesize that a combination of both is closer to the true nature of language and can lead to more optimal solutions. We investigate here several techniques aimed at testing that hypothesis empirically. In particular, we introduce "transformer/finite-state transducer hybridization." It consists of inserting into the transformer's input, under some constraints, the symbolic output of a geometric transducer.

1 Introduction

Consider the following paragraph from the New York Times (6/6/2023):

> *Ursula von der Leyen, the president of the European Commission, denounced Russia on Twitter for what she called "war crimes committed in Ukraine," saying that the destruction of the Kakhovka dam put thousands of people in the Kherson region at risk. In a followup tweet, she added that the E.U. is coordinating with member states to deliver dirt water pumps, fire hoses, mobile water purification stations and boats to Ukraine.*

In most recent transformer implementation, the first step of processing this input to feed it into a transformer consists of using a tokenizer such as WordPiece (Devlin et al., 2019). It will immediately lose some information that the right tokenizer and a dictionary would easily get. For instance, *Leyen, Kakhovka, Kherson* will be broken into *ley ##en, ka ##kohv ##ka* and *k ##hers ##on* respectively. Of course, the fact that they were proper names can be relearned but why do we have to relearn something that was known?

Moreover, this short paragraph contains a wealth of known compounds such as *Ursula von der Leyen* (easily extracted from Wikipedia for instance), *the European Commission*, *war crimes*, *E.U.* (which is tokenized into (E,.,U,.)), *member states*, *dirt water pumps*, *fire hoses* and *water purification stations*. The fact that these sequences are compounds is ignored and has to be relearned at a huge potential training cost (how large a corpus does one need to see the compound *dirt water pumps* enough time to learn it?).

Further more, syntactic information can also be extracted symbolically from the text. For instance, *Ursula von der Leyen, the president of the European Commission* can be recognized as the human subject (Nhum) of the verb *denounced*. This compresses twelve token into one.

We hypothesize that allowing a transformer to leverage a certain type of simple symbolic information can:

- reduce drastically the size of the training corpus
- speed-up training
- reduce the size of the model

- build "mini-models" that perform well on pure language tasks

The present work investigates techniques to merge high value and near-certain symbolic information with language model following a now traditional transformer architecture (Vaswani et al., 2017). The type of symbolic information, whether lexical, syntactic or semantic is derived from earlier work on geometric transductions (**?**) whose formalism is very well suited for a transformer architecture. We call that merging, transformer/symbolic hybridization.

2 Hybridization

2.1 Geometric Transduction

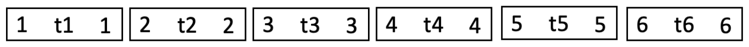
Let us consider the following toy example. Let's call Σ the finite set of all possible tokens (i.e the entire vocabulary as outputed by the tokenizer; Σ is also called the alphabet in the context of geometric transductions). Assume the input is a sequence x of 5 tokens:

$$x = (t1 \quad t2 \quad t3 \quad t4 \quad t5 \quad t6)$$

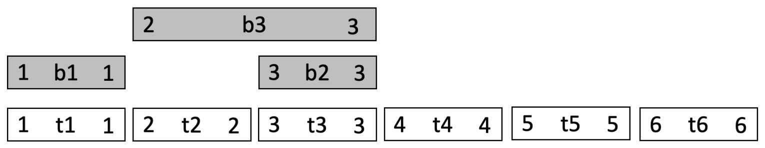
It can be represented as a γ-space (a sequence of triples $(\text{start}, \text{label}, \text{end})$):

$$\gamma(x) = \{(1, t1, 1), (2, t2, 2), (3, t4, 3), (4, t5, 4), (5, t6, 5)\}$$

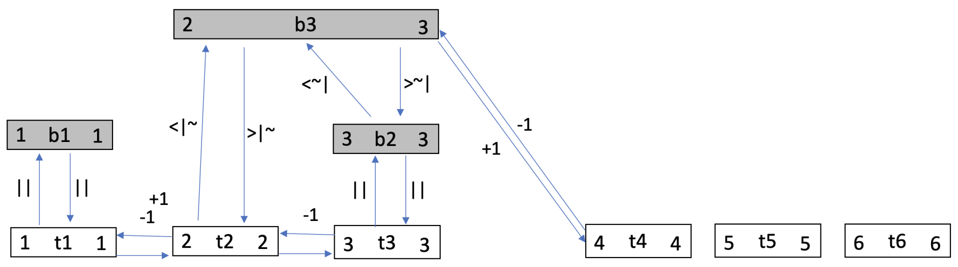
which can be also visualized as follows:



We can then generate a parsing or partial parsing done through finite-state transducers, or bimachines, representating geometric transductions (Roche, 2023). We use that formalism here. Suppose we have a geometric transduction $\tau \in \mathbb{T}(\Sigma)$ such that $\gamma_1 = \tau(\gamma(x))$ where γ_1 is:



$b1, b2$ and $b3$ represent "near-certain" [1] symbolic information. From this $\gamma-$space, we can derive the following graph where the vertices are the elements of $\tau(\gamma(x))$ and the edges are labeled by the relative position of one element with another. For instance, the edge $((2, t2, 3), -1, (1, t1, 2))$ indicates that $t1$ is at a relative position -1 compared to $t2$. The labels <|˜, ||, ... are symbols representing the relative position of segment to another. For instance, the edge $((2, t2, 3), \text{<|˜}, (1, b3, 2))$ indicates that the segment $(2, t2, 3)$ is strictly smaller than $(1, b3, 2)$ and left-aligned. It's important to note that the labels of that graph are elements of a finite alphabet Σ'; this includes numerical values, $-1, +1, +2$ which should be considered as labels rather than numbers.



[1]That is, a symbol that can be inferred with certainty where certainty is defined according to the context of the task. It usually has to be at least two orders of magnitude more precise that the target goal. For instance, if one hopes to reach 99% accuracy, certainty would be defined as 99.99% precision.

This graph can be represented by the vector x' and the matrix G:

$$x' = (t1 \quad t2 \quad t3 \quad t4 \quad t5 \quad t6 \quad b1 \quad b2 \quad b3)$$

$$G = \begin{pmatrix} 0 & +1 & +2 & +3 & +4 & +5 & || & +2 & +3 \\ -1 & 0 & +1 & +2 & +3 & +4 & -1 & +1 & <|\sim \\ -2 & -1 & 0 & +1 & +2 & +3 & -1 & || & <\sim| \\ -3 & -2 & -1 & 0 & +1 & +2 & -3 & -1 & -2 \\ -4 & -3 & -2 & -1 & 0 & +1 & -4 & -2 & -3 \\ -5 & -4 & -3 & -2 & -1 & 0 & -5 & -3 & -4 \\ || & +1 & +2 & +3 & +4 & +5 & 0 & +2 & +1 \\ -2 & -1 & || & +1 & +2 & +3 & -2 & 0 & <\sim| \\ -1 & >|\sim & >\sim| & +1 & +2 & +3 & -1 & >\sim| & 0 \end{pmatrix}$$

where g_{ij} is the label of the edge between x'_i and x'_j.
We call $\bar{\tau}$ the mapping between the original input and the resulting vector and matrix. Hence

$$(x', G) = \bar{\tau}(x)$$

Figure 1 shows examples of relative position operators that can be used in G.

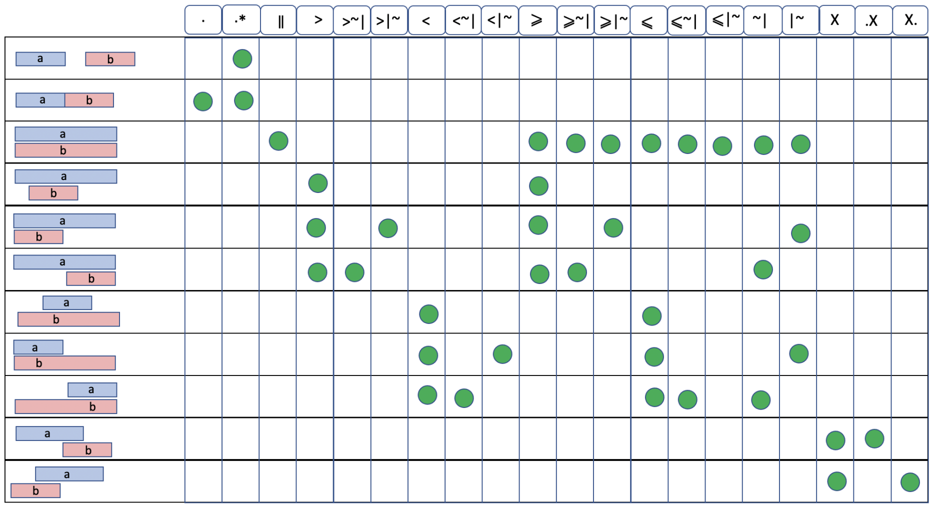


Figure 1: Example of Operators for Geometric Transductions (Roche, 2023)

2.2 Self-Attention with Graph

(x', G) will be the input of the self-attention layer. We take an approach similar to Shaw et al. (2018). Following their notation, we write:

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j, W^V) \tag{1}$$

where

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k=1}^{n} e_{ik}} \tag{2}$$

and

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}} \tag{3}$$

which they modify to take into account the proximity matrix (a_{ij}^K) where each a_{ij} represents the truncated relative position of x_j from the point of view of x_i:

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}} \tag{4}$$

which they rewrite, for efficiency, as:

$$e_{ij} = \frac{x_i W^Q (x_j W^K)^T + x_i W^Q (a_{ij}^K)^T}{\sqrt{d_z}} \tag{5}$$

We modify this last expression by replacing a_{ij} by the matrix $G = (g_{ij})$ and x by the extended sequence x'.

$$e_{ij} = \frac{x'_i W^Q (x_j W^K)^T + x'_i W^Q (g_{ij})^T}{\sqrt{d_z}} \tag{6}$$

It should be noted that, contrary to Shaw et al. (2018), g_{ij} is input dependent. It needs to be recomputed for each input.

Also, Shaw et al. (2018) modify Equation 1 into

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a_{ij}^V) \tag{7}$$

which we ignore (their ablation experiment indicates that a_{ij}^V might not contribute substantially).

Each layer of the transformer will take a sequence of length similar to x'. The last step truncates the output to the origin sequence length.

2.3 Self-Attention with Two-Layer Rotary Position Encoding

In this section, we consider an alternative approach that extends Su et al. (2022) Rotary Position Embedding (RoPE) to $\gamma-$spaces.

We recall from the previous section that

$$x' = (t1 \quad t2 \quad t3 \quad t4 \quad t5 \quad t6 \quad b1 \quad b2 \quad b3)$$

We will create two representations of x'. One that encodes the starting position , in $\gamma(x)$, of each x'_i, which we call $x'.start$, and one that encodes their ending positions, which we call $x'.end$. For instance, t_1 has an starting and ending position of 1 whereas $b3$ has a starting position of 2 and and ending position of 3.

$$x'.\text{start} = \big([t1 \quad b1] \quad [t2 \quad b3] \quad [t3 \quad b2] \quad [t4] \quad [t5] \quad [t6]\big)$$

We call MAX.start the max length of each sub-vector (2 in our example). MAX.end is defined in a similar manner. We then call D the max of these two numbers across all possible inputs. Concretely D is predefined together with the maximum window length; segments of the original $\gamma-$space beyond D are ignored. In our example, let's defined set that value to 4. We then write x'.start with padding to D for each position:

$$x\text{''.start} = (t1 \quad b1 \quad 0 \quad 0 \quad t2 \quad b3 \quad 0 \quad 0 \quad t3 \quad b2 \quad 0 \quad 0 \quad t4 \quad 0 \quad 0 \quad 0 \quad t5 \quad 0 \quad 0 \quad 0 \quad t6 \quad 0 \quad 0 \quad 0)$$

This guarantees that the initial starting position of each element of x'.start is i/D. In a similar manner:

$$x\text{''.end} = (t1 \quad b1 \quad 0 \quad 0 \quad t2 \quad 0 \quad 0 \quad 0 \quad t3 \quad b2 \quad b3 \quad 0 \quad t4 \quad 0 \quad 0 \quad 0 \quad t5 \quad 0 \quad 0 \quad 0 \quad t6 \quad 0 \quad 0 \quad 0)$$

We call P_1 the matrix such that x".start $= P_1 x'$ and P_2 the matrix such that x".end $= P_2 x'$.

At this point, the input looks like:

$$X = \begin{pmatrix} t1 & b1 & 0 & 0 & t2 & b3 & 0 & 0 & t3 & b2 & 0 & 0 & t4 & 0 & 0 & 0 & t5 & 0 & 0 & 0 & t6 & 0 & 0 & 0 \\ t1 & b1 & 0 & 0 & t2 & 0 & 0 & 0 & t3 & b2 & b3 & 0 & t4 & 0 & 0 & 0 & t5 & 0 & 0 & 0 & t6 & 0 & 0 & 0 \end{pmatrix}$$

Equation (16) of Su et al. (2022) gives self-attention as:

$$q_m^T k_n = (\mathbf{R}^d_{\theta,m} W_q x_m)^T (\mathbf{R}^d_{\theta,n} W_k x_n)$$

where (equation (15) or Su et al. (2022))

$$\mathbf{R}^d_{\theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

Let us construct $\mathbf{R'}^{Dd}_{\theta,m}$ by replicating in $\mathbf{R}^d_{\theta,m}$ each 2×2 sub-matrix

$$\begin{pmatrix} \cos m\theta_i & -\sin m\theta_i \\ \sin m\theta_i & \cos m\theta_i \end{pmatrix}$$

D times.

For readibility, here is $\mathbf{R'}^{Dd}_{\theta,m}$ where $D = 2$:

$$\mathbf{R'}^{2d}_{\theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

We now rewrite self-attention as:

$$q_m^T k_n = (\mathbf{R'}^{Dd}_{\theta,m} P_1 W_q x_m)^T (\mathbf{R'}^{Dd}_{\theta,n} P_1 W_k x_n) + (\mathbf{R'}^{Dd}_{\theta,m} P_2 W_q x_m)^T (\mathbf{R'}^{Dd}_{\theta,n} P_2 W_k x_n)$$

The first term of the sum accounts for the relative positions of the starting positions of each element of x' whereas the second term accounts for the relative positions of the ending positions.

3 Conclusion

This is still a work in progress. Experiments are ongoing.

Future work also include the following points:

- adapting and testing the original position encoding of Vaswani et al. (2017) to the two dimensional positions of $\gamma-$spaces (by encoding starting and ending positions),
- combining CNN and transformers to capture two-dimensional positions (Gu et al., 2022)

References

Jacob Devlin, Ming-Wei Chang, Kanton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of 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.

Pengfei Gu, Yejia Zhang, Chaoli Wang, and Danny Z. Chen. 2022. Convformer: Combining cnn and transformer on medical image segmentation. *arXiv preprint arXiv:2011.08564v1 [cs.CV].*

Emmanuel Roche. 2023. Finite-state representation of geometric transductions.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155.*

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2022. Roformer: Enhanced transformer with rotary position embedding.

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.