The Double Helix inside the NLP Transformer

Anonymous ACL submission

Abstract

This study introduces a novel framework for exploring the information processing within NLP Transformers. We categorize information into four distinct layers: positional, syntactic, semantic, and contextual. Challenging the conventional integration of positional data into semantic embeddings, we propose a more effective "Linear-and-Add" method. Our analysis uncovers an intrinsic separation of positional elements in deeper layers, revealing that these components form a helix-like pattern in both encoder and decoder stages. Notably, our approach enables the identification of Part-of-Speech (PoS) clusters within conceptual dimensions. These insights offer a new perspective on information processing in the complex architecture of NLP Transformers, potentially guiding future developments in the field.

1 Introduction

Large Language Models (LLMs), such as ChatGPT, have become a focal point of recent research, primarily due to the Transformer architecture, which is central to all modern LLMs (Radford et al., 2018; Radford et al., 2019; Radford et al., 2019; Brown et al., 2020; Ouyang et al., 2022; OpenAI, 2023). Introduced by Vaswani et al. (2017) , the Transformer employs an attention mechanism to efficiently process different parts of input data simultaneously. This has led to superior performance over older models across several NLP tasks, including machine translation and question answering. While the Transformer has been instrumental to advancements in AI, the intricacies of its function remain complex. Our work aims to demystify the Transformer's operations and provide a clear framework for its analysis.

2 Methodology

2.1 From Words to Concepts

Communication aims to transmit concepts, which are nuanced by context, a term like "server" can signify different entities in technology and food catering. Recognizing this, Transformers shift focus from words to concepts, which encapsulate multiple layers of information: positional (word location), syntactic (grammatical role), semantic (inherent meaning), and contextual (relation to surrounding words). Rather than interacting with words, Transformers process "tokens"—units that can represent word parts, punctuation, or specific syntax—allowing for versatility transcending linguistic variability.

2.2 The Meaning of "Meaning"

Dictionary definitions often contain circular reasoning, explaining words using other undefined words. This reveals the operational nature of "meaning" - a word's meaning is simply its relationships to other related words. However, words can be ambiguous, having multiple interconnected meanings. More universal "concepts" better capture distinct meanings. While a word like "server" has overloaded meanings, concepts require clearer explanations using multiple words and contextual understanding. As another example, "project" as a noun or verb carries different connotations.

We can visualize concepts as "mini-galaxies" of meaningful words orbiting a central point. Tracking this conceptual center is more practical than tracing all orbital word associations. Using concepts enables language translation by mapping between conceptual vector representations across languages. In embedding spaces like Word2Vec (Mikolov et al., 2013), concepts manifest as weighted superpositions of word vectors. Words are likewise messy superpositions of underlying concepts. We can freely encode words into concepts and decode concepts back into words as needed. Just as neural networks have deep layers to construct representations, language also converts primitive concepts into richer, syntactic concepts. Subunits of the neural network can process dedicated conceptual subspaces, merging and projecting the results to manage dimensionality. This is the foundation for Transformer attention - understanding words in conceptual, syntactic, and contextual terms.

2.3 Positional Encoding

The Transformer algorithm marks a significant advancement in how it handles the position of a token in a sentence. Unlike Recurrent Neural Networks (RNNs), including Long Short-Term Memory machines (LSTMs), which treat positions as indices, the Transformer considers them as additional information. This approach enables parallel processing of all tokens simultaneously.

A real number can be represented as either the amplitude or the phase of a complex number. Given that the semantic embedding space resembles a hypersphere and aligns more with phase encoding, applying phase encoding to positions is promising. Analog clocks, using a hand's angle on a circle to denote time, serve as a simple analogy for phase encoding. Although a single hand could suffice, multiple hands are often used in analog clocks to improve resolution. In angular measurements like those in clocks, we often employ modular arithmetic. For instance, we measure seconds within the range of 0 to 60 seconds. However, such discontinuity poses challenges in neural networks, which typically do not perform modular arithmetic inherently. A more effective method involves using the sine and cosine of an angle, providing a continuous representation using two real numbers rather than a single angular value. To encode a word's position, a clock system can be utilized. Consider the following empirical positional encoding for $0 \leq i < d_{\text{model}}/2$:

$$\begin{cases} \mathrm{PE}(\mathrm{pos}, i) = \sin\left(\dfrac{\mathrm{pos}}{10{,}000^{\frac{2i}{d_{\text{model}}}}}\right), \\ \mathrm{PE}\left(\mathrm{pos}, \dfrac{d_{\text{model}}}{2} + i\right) = \cos\left(\dfrac{\mathrm{pos}}{10{,}000^{\frac{2i}{d_{\text{model}}}}}\right), \end{cases} \tag{1}$$

Here, “pos” represents the word's position in the text, “*I*” is the dimension index, and “d_{model}” is the embedding space dimension, typically equal to the word embedding dimension. The number 10,000, chosen as a scalar, represents a typical book chapter length in words.

This grouping of sine and cosine values differs from the original even/odd convention in the Transformer paper (Vaswani et al., 2017). Each sine and cosine pair represents the phase of a “hand” on a clock, with periods ranging from 6.28 for lower dimensions to 6.28×10^4 for larger digits ($i \sim d_{\text{model}}/2$). Two key aspects are noteworthy. First, a sophisticated neural network should capture the intricacies of the clock-like embedding system, and not solely depend on Euclidean distance to measure the proximity of positions. Secondly, it's advantageous to consider an “effective position” reflecting a range of nearby PE(pos,i) values, similar to the fuzzy representation of concepts by a collection of tokens. This position encoding scheme, integrated with neural networks and its inherent multi-slice fuzziness, is expected to be robust and stable, due to its built-in redundancy in digits.

2.4 Purpose of Using Positional Encoding

Transformers represent a significant innovation by treating tokens quasi-independently, which enables parallel processing. Each token independently passes through the same Transformer neural network. This process is similar to using “free body diagrams” in physics, where each object in a complex system is analyzed separately with external influences considered as forces (see Fig.1). In Transformers, tokens navigate through the neural network, influenced by other tokens only via attention and feedforward layers, paralleling the application of Newton's second law

$$F_i = \sum_{j=1}^{n} F_{ij} = m_i a_i \tag{2}$$

To facilitate parallel processing, positional information is embedded in vectors rather than in the indices of the tokens. For clarity, consider only the highest two digits of positional encoding, which form a circular path. Fig. 2 illustrates the Transformer algorithm operations, pointing out the relative dominance of positional over semantic encoding.

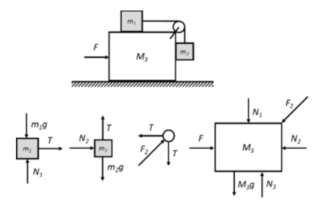


Fig. 1. Free body diagrams.

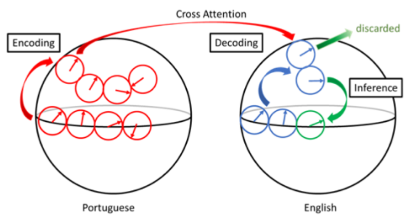


Fig. 2. Operations diagram of the Transformer algorithm.

The “free body diagram” analogy highlights that all operations – encoding, decoding, and inference – occur for each individual token. This design allows parallel processing of tokens and makes the neural network flexible to sentence length. The algorithm’s structure remains unaffected by the number of tokens, akin to the irrelevance of the number of bodies in mechanical laws of physics. During training, Transformers optimize for efficiency by predicting the next token for each output token in a training batch simultaneously. However, in inference mode, the prediction is instead sequential: one token at a time. Once a token is predicted, it need not be predicted again. This disparity between training and inference also occurs in the behavior of dropout layers (Srivastava et al., 2014).

2.5 Attention Mechanism as “Two-Body Forces”

The key innovation of Transformers lies in treating tokens quasi-independently, allowing for parallel processing. Analogous to “free body diagrams” in physics, where each object is analyzed independently, with external influences from other objects entering only as forces (as shown in Fig. 3), Transformers analyze each token in a similar fashion. This approach facilitates a context-sensitive representation, enabling a word like “server” to be interpreted differently based on its surrounding text.

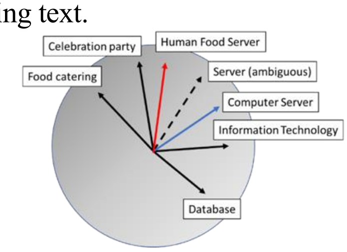


Fig. 3. Semantic proximity and context of word.

The formal formula for the naïve self-attention is:

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^t}{\sqrt{d_k}}\right)V \quad (3)$$

Here, Q, K, V are the “query vector”, “key vector collection” and “value vector collection”.

$Q = w_i$ query word's embedding (horizontal vector)

$K = [w_{ij}]$ matrix from collection of key vectors

$V = [w_{ij}]$ same as K in the case of simple attention

d_k: subspace dimension (for normalization)

In simple attention, V equals K. The output is a “shift” vector, w influence of all words (K) in the paragraph, akin to the gravitational force drawing objects toward a common center. Real-world Transformers use a more complex form of self-attention to capture syntactic nuances. Syntax plays a vital role in understanding words, as demonstrated by different interpretations of the verb “left” as an adjective or verb. To incorporate syntactic information, we can project query and key vectors onto syntactic subspaces.

Extending the physical analogy, attention can be seen as a “force” shifting embedding vectors. While naive self-attention is akin to gravitational or electrostatic forces, more intricate forces are needed to capture the diverse properties of tokens. In physics, different forces act on particles based on their properties, such as mass or charge. Similarly, each attention mechanism can be viewed as a distinct force coming from a different type of subspace charge.

The formula for the query-key-value approach for generalized attention is:

$$\text{attention}(QW_h^Q, KW_h^K, VW_h^V) = \text{softmax}\left(\frac{QW_h^Q W_h^K K^t}{\sqrt{d_k}}\right)VW_h^V \quad (4)$$

The projection-rotation matrices W_h^Q, W_h^K, W_h^V, along with the dimension d_k of the subspace where we perform the dot product, are used to calculate, and determine the attention strengths. Using multiple “heads” in multi-headed attention allows for specific syntactic roles to be identified, applying appropriate shifts to each query word. In self-attention, Q, K, and V belong to the same language, whereas in cross-attention, they belong to different languages. After processing through multiple attention heads and a feedforward layer, the Transformer effectively translates a sequence of words into concepts. This process is comparable to the translation of DNA into proteins, where information is distributed across multiple slices, not isolated to individual tokens.

Overall, Transformers use a sophisticated mechanism to convert word sequences into conceptual sequences, adapting to different grammatical structures of languages, much like the intricate process of translating genetic information into functional proteins.

2.6 Shift Invariance and Generalized Inner Product

Shift invariance is a crucial concept when considering the relationships between words in a text. The relationship between a word *A* and a word *B* should remain consistent, even if additional text is prepended to the original sentence where they appear. This principle, known as shift invariance, is essential for nearby words unless strong contextual dependencies are present. Without shift invariance, the Transformer algorithm would need different implementations of attention heads for different positions.

In the standard Transformer model, the positional encoding vector is added to the semantic embedding vector, which seems to compromise translational invariance. The embedding of shifted words yields altered vectors, and the generalized inner product of these vectors does not maintain translational invariance:

$$\begin{aligned}\langle Q, K\rangle &\neq \langle Q + P_{Q1}, K + P_{K1}\rangle \\ &\neq \langle Q + P_{Q2}, K + P_{K2}\rangle \qquad (5)\end{aligned}$$

Where P_{Q1} and P_{K1} are the positional encoding vectors of the words Q and K, and P_{Q2} and P_{K2} are their shifted versions. Despite its apparent lack of explicit shift invariance, the Transformer accomplishes invariance through two mechanisms. Firstly, a submanifold exists in the Q, K space, preserving the value of $\langle Q, K\rangle$ for certain alignments of positional vectors. Secondly, the projection matrices W_i^Q, W_i^K, W_i^V can project onto subspaces effectively orthogonal to the positional encoding vectors. Additionally, the input word embedding is trained alongside other neural weights, aiding the Transformer in separating positional and semantic dimensions. If the "conceptual dimensions" represented by vectors Q and K are orthogonal to the positional vectors the positional vectors P_{Q1}, P_{K1}, P_{Q2} and P_{K2}, we have:

$$\langle Q + P_{Q1}, K + P_{K1}\rangle = \langle Q, K\rangle + \langle P_{Q1}, P_{K1}\rangle$$

$$\langle Q + P_{Q2}, K + P_{K2}\rangle = \langle Q, K\rangle + \langle P_{Q2}, P_{K2}\rangle \quad (6)$$

Given shift invariance in positional encoding (i.e., $\langle P_{Q1}, P_{K1}\rangle = \langle P_{Q2}, P_{K2}\rangle$), the overall Transformer algorithm can achieve shift invariance. Positional encoding schemes like a multi-handed clock on a hypertorus satisfy this requirement, as do helix-shaped encodings. The most general topology for positional encoding is a hypertorus, with the helix and straight line as limiting cases, making a multi-handed clock with sinusoidal functions a natural choice.

The Transformer model's capacity to implement shift invariance is remarkable, given the general non-invariance of inner products. This seems to be largely due to the training of the semantic embedding layer in conjunction with other layers, allowing the model to separate semantic and positional dimensions. Conceptual dimensions eventually become orthogonal to positional ones. During encoding, the Transformer allocates certain dimensions for positional encoding while reserving others for semantic or conceptual encoding. This automatic separation of dimensions is one of the Transformer model's most notable features.

3 Data and Codes

To build the model, we used Google's Transformer codes (tensorflow) as a base. We ran the model in a Docker container on an NVIDIA-SMI 470.103.01 powered DGX server, equipped with Tesla V100-SXM2 GPUs and CUDA Version 11.4.

We employed the Portuguese-English translation dataset from TensorFlow Datasets, consisting of approximately 52,000 training, 1,200 validation, and 1,800 test instances. Each instance contains a Portuguese-English sentence pair. For efficiency, we tokenized the dataset and arranged it into ragged batches. Our model processes tokenized Portuguese and English sequence pairs (pt, en) as inputs. The target labels are the corresponding English sequences, offset by one token, ensuring that the target label for each position in the English sequence is the next token. We used tf.keras.layers.Embedding layer (Gal et al., 2016) to convert input Portuguese and target English tokens into vectors.

Unlike static embeddings in Word2Vec and GloVe (Mikolov et al., 2013; Gal et al., 2016; Pennington et al., 2014), our Transformer model generates dynamic word embeddings. It starts with randomly initialized weights and refines them during training. The dimensionality in word embeddings, referring to the number of features, was set to 128 for this project. The encoder and decoder architectures are similar, each with $N = 4$ attention layers. Each layer comprises an 8-head multi-head self-attention mechanism and a fully connected feed-forward network, enabling concept synthesis. The decoder includes an additional multi-head cross-attention sub-layer to focus on various parts of the encoder's output, utilizing the conceptual information for enhanced performance in the translation task.

4 Results

4.1 Questioning the Adding of Positional Encoding Vector

In the conventional Transformer algorithm, the positional encoding vector is directly added with the semantic embedding vector, a process that lacks justified rationale. This direct addition is problematic, as these vectors originate from distinct domains - akin to adding apples with oranges. To elucidate this issue, we adapted Google's Transformer code and replaced the direct sum of the semantic embedding S vector and positional encoding vector P, $S+P$, with $wS+P$, where w is a weight factor. We evaluated the loss function across varying values of w with results depicted in Figure 5(a). Our findings revealed an optimal weight of $w=0.3$, challenging the convention of a direct sum. This indicates that the positional information significantly outweighs the semantic information, portraying a picture of small "semantic hyperspheres" situated along a broad positional encoding trajectory.

An enhanced method involves utilizing a full linear neural network instead of a simple weighted sum, as demonstrated in Figure 4. By integrating a dense linear layer (sans activation function) and a dropout layer, a more sophisticated combination is achieved. Adding back the positional encoding vector post-linear layer further enhances model stability and mitigates the risk of local minima during training. We have termed this methodology "Linear & Add," reflecting the sequence of linear layer application followed by addition. Figure 5 contrasts the model performance using three distinct methods of combining positional encoding with semantic embedding: (1) direct addition, (2) weighted sum with a reduced semantic embedding significance (w=0.3), and (3) the application of a dense layer followed by summation. Consistent with our hypothesis, the "Linear & Add" method outperforms the others.

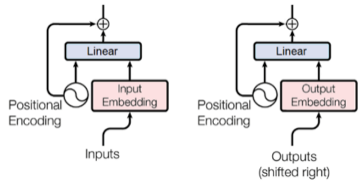
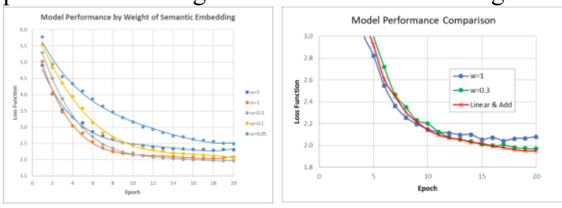


Figure 4 "Linear & Add" method to concatenate the positional encoding and the semantic embedding vectors

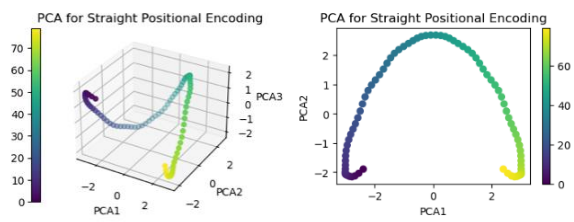


(a) (b)

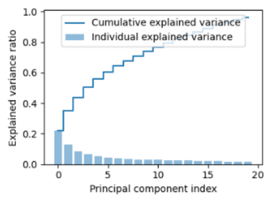
Fig. 5. (a) Model performance by weight of semantic embedding. (b) Model performance comparison of three different approaches for combining positional encoding with semantic embedding.

4.2 The First Helix: Positional Information After the Encoding Stage of a Transformer

To understand the positional encoding in the Transformer's input layer, we conducted Principal Component Analysis (PCA). Applying PCA to the linear-and-add positional encoding vectors (up to the 80th position) yielded a 2D plot of the first two components (Fig. 6b), a 3D plot of the first three components (Fig. 6a), and a bar chart showing the variance explained (Fig. 6c). The analysis revealed that the positional encoding follows a 2D path resembling the "Arch of St. Louis" (the Gateway Arch). Notice the variance decays slowly across the principal components.



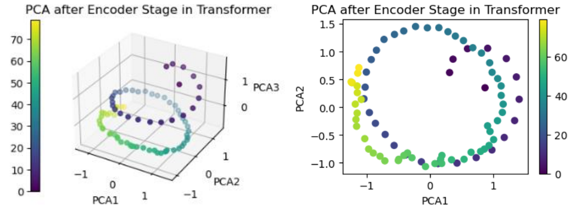
(a) (b)



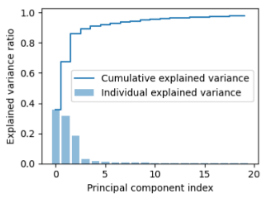
(c)

Fig. 6. Principal Component Analysis of positional encoding. (a) 3D plot of the first three PCA components.

(b) 2D plot of the first two PCA components. (c) Variance explained.



(a) (b)



(c)

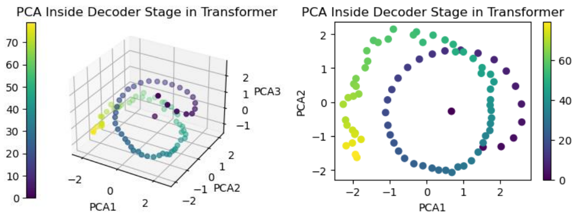
Fig. 7. Principal Component Analysis after the encoding stage. (a) 3D plot of the first three PCA components. (b) 2D plot of the first two PCA components. (c) Variance explained.

We repeated this analysis post-encoding stage of the Transformer. By averaging the embedding vectors of 1,000 randomly selected sentences at identical positions, we isolated positional information, effectively removing semantic, contextual, and syntactic information. PCA on the resulting "average sentence" showed that the residual positional encoding forms a helical shape (Fig. 7a and 7b). In contrast to the input positional encoding that spans all 128 dimensions, the post-encoding positional vectors predominantly use three dimensions (Fig. 7c). This dimensional reduction facilitates shift invariance, allowing the Transformer to capture semantic, syntactic, and contextual information in the remaining 125 dimensions.

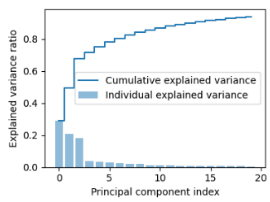
One might question the wisdom of combining positional and conceptual information from the outset. While separating them in the input stage is valid, merging them there offers flexibility in handling varying sentence lengths. This approach eliminates the need for preset dimensions for positional information, allowing the algorithm to dynamically allocate dimensions for positional and conceptual elements. This adaptability is key for handling diverse text lengths and complexities, effectively allowing the algorithm to "set its own clock". Nevertheless, exploring alternative methods for merging information remains a vital area of research (Wang et al., 2020).

4.3 The Second Helix: Positional Information Deep Inside the Decoder Stage of a Transformer

In exploring the decoding stage of the transformer algorithm, we aimed to identify a helical pattern akin to what was observed in the encoding stage. Surprisingly, we found such a helix, but not where we initially expected. In the decoding stage, as tokens are generated to form English sentences, the positional information resembles the original positional encoding, tracing a path similar to the "Arch of St. Louis." A more detailed examination of the four layers of decoder-attention in Google's transformer algorithm revealed a distinct 3D helix in the second layer, as depicted in Figures 8a and 8b. Principal Component Analysis confirmed the three-dimensional nature of this helix. This discovery suggests that the helical pattern is a core characteristic of the transformer architecture, raising interesting questions about its role in the algorithm's language processing and generation capabilities.

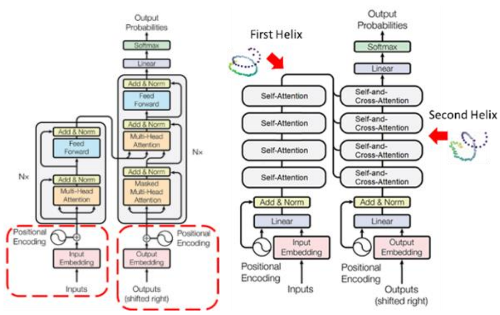


(a) (b)



(c)

Fig. 8. Principal Component Analysis in the decoder stage. (a) 2D plot of the first two PCA components. (b) 3D plot of the first three PCA components. (c) Variance explained.



(a) (b)

Fig. 9. (a) Google Transformer model architecture. (b) Identification of two helix patterns in the transformer model architecture.

4.4 Accidental Mingling

In the transformer algorithm, positional and semantic embeddings are combined, which could potentially result in indistinguishable embeddings for different words. To illustrate, if token A and token B have positional embeddings of 3 and 4, and semantic embeddings of 6 and 5, respectively, their combined embeddings would both sum to 9. This overlap might obscure distinct meanings. However, the transformer model employs strategies to prevent such overlaps. Renormalization in deep learning (De Mello Koch et al., 2020) emphasizes slight differences between embeddings. The high dimensionality of the embedding space and the improbability of overlaps in linguistic translation reduce the likelihood of collisions. Crucially, the transformer's semantic embeddings are trained to be orthogonal to positional embeddings, which minimizes the risk of mixing up different words, ensuring accurate language processing.

4.5 Running Text Density of Words

Different languages often need varying word counts to express identical ideas. For instance, Spanish usually uses more words than English for the same concept. This discrepancy poses a question about transformers handling the “text density” differences. The adaptation likely takes place at the transformer's cross-attention layer, which discerns text relationships and modulates the output's text density to match the target language's norms.

4.6 Part of Speech

Through analyzing a large sample of sentences and calculating average embedding vector values, we successfully identified positional information vectors for each transformer stage. Isolating the conceptual dimensions - syntactic, semantic, and contextual information – can be achieved by deducting the positional vector from the embedding vector. Dimension reduction can then be performed using PCA, reducing the dimensions from 128 to 5. This allowed us to delve deeper into the language structure, uncovering meaningful relationships within the language model.

Our study focused on the transformer's encoder side (Portuguese), examining tokens of four or more characters, excluding suffixes, punctuation, and sentence markers. For clustering, we employed K-Means++ (sklearn.cluster.KMeans) with the elbow method to determine the optimal number of clusters. This has revealed distinct clusters corresponding to parts of speech (PoS) attributes like nouns, adjectives, and verbs, as highlighted in Fig. 10.

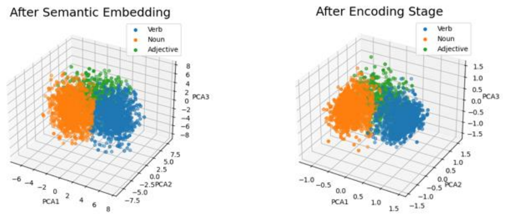


Fig. 10. Tokens formed distinct clusters by their part of speech (PoS) attributes, such as nouns, adjectives, and verbs. (Left) Pre-attention stage after semantic embedding. (Right) Post-attention stage after encoding.

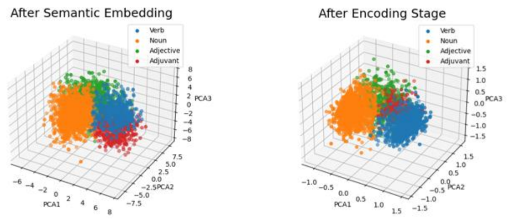


Fig. 11. A second level of clustering split Verboid cluster into Verb and Adjuvant clusters. (Left) Pre-attention stage after semantic embedding. (Right) Post-attention stage after encoding.

Fig. 12. Word Clouds After Encoding Stage. (Top row) 3 clusters clustering. (Bottom row) Verbs and Adjuvants clusters from sub-clustering of “Verboids”

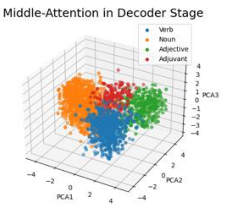


Fig. 13: In mid-attention encoder stage English tokens formed distinct clusters by their PoS attributes, verbs, nouns, adjectives, and adjuvants.

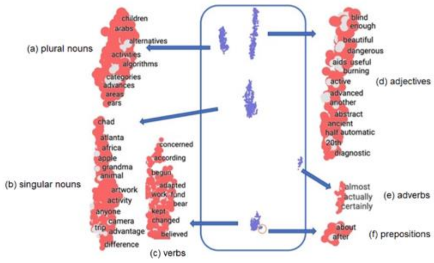


Fig.14. PoS clusters visualized with t-SNE

Interestingly, the verb cluster included numerous "functional words" such as pronouns, prepositions, conjunctions, determiners, and some adverbs. We termed these "adjuvants" - a term derived from Latin, meaning "to help" - due to their supportive role in sentence construction. Applying a second layer of K-Means++ clustering to this "verboid" cluster distinguished the adjuvants as a separate group, as shown in Fig. 11. Fig. 12 displays word clouds post-encoding, showing meaningful grouping. Our findings also indicated that the self-attention layers enhance cluster delineation. This underscores the self-attention mechanism's effectiveness in capturing token relationships and their PoS functions. We observed that points fluctuated between (-6.0, 6.0) in the original semantic layer, narrowing to (-1.0, 1.0) post-encoding, reflecting a "renormalization" process akin to the "renormalization group" in physics (De Mello Koch et al., 2020). Analysis of the decoder side (English) revealed challenges in forming well-defined clusters due to the decoder's dual role in concept formation/translation as well as decoding to tokens. Optimal PoS clustering was achieved in the mid-attention stage (Fig. 13) using PCA and two-step K-means++ clustering. Additionally, we explored t-distributed stochastic neighbor embedding (t-SNE) (Maaten et al., 2008) for visualizing PoS clusters without PCA reduction. Applied directly to 128-dimensional embeddings, distinct PoS clusters emerged after over 10,000 iterations, as illustrated in Fig. 14. Lastly, for post-attention stage analysis, K-Means++ alone was insufficient for clear PoS clustering. However, applying the technique to di-grams of current and next tokens yielded discernible PoS clusters.

4.7 Additional Syntactical, Semantic, and Contextual Information

We observed that removing positional information from vectors reveals "conceptual" information, with single-delta vectors indicating some PoS cluster structure. For tokens appearing multiple times, we calculated their "semantic vector" by averaging their single-delta vectors. Subtracting the "semantic vector" from the single-delta embedding yields a residual "double-delta vector", likely representing the syntactic and contextual dimensions of each token. However, comprehending the full extent of syntactical information requires examining token sequences, not just individual tokens. It's possible that single-delta or double-delta vectors contain additional syntactical information, but confirming this necessitates analysis of multiple consecutive tokens. We leave this detailed examination for future research.

5 Conclusions

The Transformer architecture has significantly advanced natural language processing with its innovative design and exceptional performance. Central to its function is the encoding of input words into "concepts", facilitating efficient sequence-to-sequence translation. A key feature of the Transformer is positional encoding, which not only allows for processing of variable-length sequences but also supports parallel processing on modern hardware, akin to the use of free-body diagrams in physics. Remarkably, the Transformer's principles can be applied to diverse data types, such as images (Dosovitskiy et al., 2020) and protein folding (Jumper et al., 2021), demonstrating its versatility and adaptability in various deep learning applications.

This paper has provided an in-depth analysis of the Transformer, examining its handling of positional, syntactic, semantic, and contextual information. We highlighted the unique positional information mapping in the encoding and decoding stages, drawing parallels to the double-helix structure of DNA. The significance of the semantic embedding layer in token clustering by parts of speech was also emphasized. These insights demystify the transformer algorithm, presenting it as a comprehensible tool for AI practitioners. Our goal is to equip professionals with a thorough understanding of the transformer's mechanisms, enabling them to leverage its potential and spur innovation in natural language processing.

References

Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. 2020. *Language models are few-shot learners*. Advances in neural information processing systems. 33:1877-901. arXiv:2005.14165

Ellen De Mello Koch, Robert De Mello Koch and Ling Cheng. 2020. *Is Deep Learning a Renormalization Group Flow?*. IEEE Access, vol. 8, pp. 106487-106505, doi: 10.1109/ACCESS.2020.3000901..

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. 2020. *An image is worth 16x16 words: Transformers for image recognition at scale*. arXiv:2010.11929.

Yarin Gal and Zoubin Ghahramani. 2016. *A Theoretically Grounded Application of Dropout in Recurrent Neural Networks*. Advances in Neural Information Processing Systems. arXiv:1512.05287

Jumper, J., Evans, R., Pritzel, A. et al. 2021. *Highly accurate protein structure prediction with AlphaFold*. Nature 596, 583–589.

Laurens van der Maaten, L.J.P.; Hinton, G.E. 2008. *Visualizing High-Dimensional Data Using t-SNE.* Journal of Machine Learning Research 9:2579-2605.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. 2013. *Efficient estimation of word representations in vector space*. arXiv:1301.3781.

OpenAI. 2023. *GPT-4 Technical Report*. arXiv:2303.08774

Ouyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, et al. 2022. *Training language models to follow instructions with human feedback*. arXiv:220302155.

Pennington, J., Socher, R., & Manning, C. D. 2014. *GloVe: Global Vectors for Word Representation*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) pp. 1532-1543

Radford A, Narasimhan K, Salimans T, Sutskever I, et al. 2018. *Improving language understanding by generative pre-training*, OpenAI.

Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I, et al. 2019. *Language models are unsupervised multitask learners*. OpenAI blog. 1(8):9.

Radford A, Wu J, Amodei D, Amodei D, Clark J, Brundage M, et al. 2019. *Better language models and their implications*. OpenAI Blog https://openai com/blog/better-language-models.1(2).

Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R. 2014. *Dropout: a simple way to prevent neural networks from overfitting*. The Journal of Machine Learning Research. 1;15(1):1929-58.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. 2017. *Attention Is All You Need*. Advances in Neural Information Processing Systems.

Wang, Yu-An, and Yun-Nung Chen. 2020. *What do position embeddings learn? an empirical study of pre-trained language model positional encoding*. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)

https://www.tensorflow.org/text/tutorials/transformer