Modeling Language Tokens as Functionals of Semantic Fields

Zhengqi Pei[1 2] **Anran Zhang**[1 2] **Shuhui Wang**[1 3] **Qingming Huang**[4 1 3]

Abstract

Recent advances in natural language processing have relied heavily on using Transformer-based language models. However, Transformers often require large parameter sizes and model depth. Existing Transformer-free approaches using state-space models demonstrate superiority over Transformers, yet they still lack a neuro-biologically connection to the human brain. This paper proposes $LasF$, representing **L**anguage tokens **as** **F**unctionals of semantic fields, to simulate the neuronal behaviors for better language modeling. The $LasF$ module is equivalent to a nonlinear approximator tailored for sequential data. By replacing the final layers of pre-trained language models with the $LasF$ module, we obtain $LasF$-based models. Experiments conducted for standard reading comprehension and question-answering tasks demonstrate that the $LasF$-based models consistently improve accuracy with fewer parameters. Besides, we use CommonsenseQA's blind test set to evaluate a full-parameter tuned $LasF$-based model, which outperforms the prior best ensemble and single models by 0.4% and 3.1%, respectively. Furthermore, our $LasF$-only language model trained from scratch outperforms existing parameter-efficient language models on standard datasets such as WikiText103 and PennTreebank.

1. Introduction

Language modeling is a crucial task in natural language processing (NLP), representing a key component in developing brain-inspired artificial general intelligence capable of fully understanding human language (Zhao et al., 2023). Currently the most representative language modeling methods are built upon Transformer-based structures (Vaswani et al., 2017; Devlin et al., 2018; Radford et al., 2019; Brown et al., 2020). These models represent language tokens as high-dimensional distributed embeddings (Bengio et al., 2000), utilizing an attention mechanism (Vaswani et al., 2017) to interpret the linguistic relations among tokens as linear relations. Although Transformer-based models demonstrate outstanding performance in downstream NLP tasks, they encounter practical challenges regarding to the computational efficiency (Fournier et al., 2023).

Some non-Transformer models, such as state-space models (SSM), including S4 (Gu et al., 2021), H3 (Fu et al., 2022), and Hyena (Poli et al., 2023), have significantly enhanced language model performance on public tasks. Despite their computational efficiency, they differ significantly from how the human brain represents semantic relations among language tokens. Specifically, they still rely on simplified linearity to express the highly nonlinear semantic relations between language tokens, limiting the representational capacity. In contrast, the human brain processes linguistic information in a highly nonlinear manner (Patterson et al., 2007). It transforms language tokens into signals perceivable by neurons, representing the semantic relations between different language tokens using nonlinear, field-like relations. This mechanism allows for a more comprehensive expression of dynamic semantic features from all neurons. In linguistics, the field-like relation is interpreted as the *semantic field*, denoting a semantically structured group of the lexical set of language tokens (Jackson & Amvela, 2007). Each token has varying semantic fields that express its multiple linguistic relations to other tokens.

In fact, many existing studies (Toneva & Wehbe, 2019; Oota et al., 2022b;a; Sun & Moens, 2023) attempt to integrate the brain-inspired mechanism into computational language models. However, they have yet to demonstrate performance comparable to Transformer-based or SSM-based language models in practical scenarios. To address this issue, we propose a computationally friendly mechanism for interpreting language tokens as functional representations of semantic fields. This approach allows us to express semantic relations between language tokens using more complex semantic fields while maintaining low computational complexity.

[1]Institute of Computing Technology, Chinese Academy of Sciences. [2]School of Artificial Intelligence, University of Chinese Academy of Sciences. [3]Peng Cheng Laboratory. [4]School of Computer Science and Technology, University of Chinese Academy of Sciences. Correspondence to: Shuhui Wang <wangshuhui@ict.ac.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

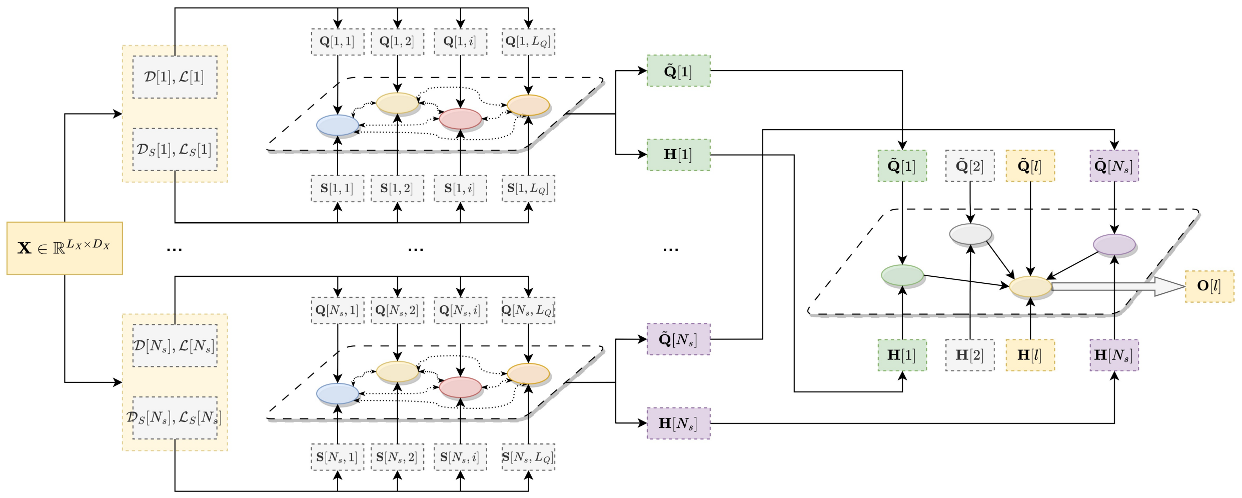

Figure 1: **Pipeline of a *LasF* Module**. This process is designed to simulate the dynamic interactions of neurons within semantic fields, from receiving to emitting signals. The input $\mathbf{X}$ is mapped to the neuron states $\mathbf{Q}$ on N_s distinct semantic fields (Eq. 2), along with the signals $\mathbf{S}$ they receive (Eq. 7). Within each semantic field, neurons compute their temporary outputs $\mathbf{H}$ based on interactions with each other (Eq. 8). The semantic fields then derive their actual output signals $\mathbf{O}$ from interactions among their overall states $\tilde{\mathbf{Q}}$ (Eq. 9). The final output result $\hat{\mathbf{Y}}$ is obtained by summing the actual output signals $\mathbf{O}$ from each semantic field (Eq. 12).

In mathematical terms, a functional can be perceived as a function of functions, representing a real-valued function on a space of functions with functions as its arguments (Lang, 2012). Indeed, a semantic field refers to a function, and the interactions between semantic fields refer to a functional.

The proposed *LasF* module, constructed based on the functional semantic field, is equivalent to a nonlinear mapping function suitable for sequential data. As presented in Figure 1, it comprises multiple semantic fields, each responsible for capturing distinct intrinsic linguistic patterns. A *LasF* module first transforms the current input sequence of embeddings into dynamic neurons within each semantic field. Simultaneously, it converts these embeddings into the signals received by neurons within different semantic fields. Subsequently, each neuron decides how to receive these signals and emits its output signals based on interactions with neurons inside and across semantic fields. In this way, we establish a *LasF* module capable of encoding inputs and transforming them into arbitrary output signals.

We conduct experiments on three NLP tasks, including reading comprehension on SQuAD2.0 (Rajpurkar et al., 2018), question answering task on Commonsense QA (Talmor et al., 2018), Physical Interaction QA (Bisk et al., 2020), and Social Interaction QA (Sap et al., 2019), and language modeling on WikiText-103 (Merity et al., 2016) and PennTreebank datasets (Marcus et al., 1993). For the reading comprehension and question answering tasks, we observe that replacing the output layer of the pre-trained LLaMA (Touvron et al., 2023) models with the $LasF$ module resulted in a consistent accuracy improvement with fewer parameters. We also conduct full-parameter tuning on several $LasF$-based BERT models. In the blind test set of Commonsense QA[1], our $LasF$-based ALBERT model in a single model setting outperforms the previous best ensemble and single models by 0.4% and 3.1%, respectively. Notably, the prior best single model UnifiedQA (Raffel et al., 2020) is $47\times$ larger than ours.

We also build a $LasF$-only language model trained from scratch. Empirical results show that the $LasF$-only language model achieves lower perplexity than other existing Transformer or SSM-based approaches. It is worth noting that the $LasF$-only language models contain no more than 3 layers, meaning it can attain language modeling capabilities with fewer neural layers, *i.e.*, a FLAT rather than a DEEP model. This finding aligns with the shallow brain hypothesis (Suzuki et al., 2023), which posits that a shallow neural structure composed of carefully designed parallel computing units may have equally strong linguistic processing capabilities. Codes are available at `https://github.com/pzqpzq/flat-learning`.

[1]`https://www.tau-nlp.sites.tau.ac.il/csqa-leaderboard`, where our model refers to *Albert+KasF* , short for *Knowledge as Functionals*. We only compare with the models that do not use ConceptNet since the CsQA's team officially no longer accepts this submission type. As ConceptNet was used to create the dataset, it filters the human-generated distractors, reducing the 5-way multi-choice to a simpler one.

2. Preliminaries

2.1. Problem Formulation

Given a sequential input, such as a sentence represented as a sequence of word embeddings, denoted as $\mathbf{X} \in \mathbb{R}^{L_X \times D_X}$, where L_X and D_X are the length and dimension of $\mathbf{X}$, respectively. Our objectives consist of three components:

(1) Constructing a nonlinear module capable of mapping any sequential input denoted as $\mathbf{X}$ to a target $\mathbf{Y} \in \mathbb{R}^{L_Y \times D_Y}$ with minimal approximation error. (see Section 3)

(2) Integrating such a nonlinear module with a well-formed text encoder, *e.g.*, pre-trained large language model, for various downstream NLP tasks. (see Section 4)

(3) Building a language model from scratch utilizing these nonlinear modules to perform language modeling tasks with minimal parameter usage. (see Section 5)

2.2. Neuronal Dynamics for Semantic Field

In linguistics, a semantic field denotes a collection of words interconnected by meaning, concept, or topic. These words share a common semantic feature, and their interconnections derive from their meanings within a particular context. From a neuro-biological perspective, the human brain utilizes semantic fields to comprehend the semantic information between language tokens, forming a process of global understanding of languages. This process is equivalent to the interaction of signals between neuron clusters with different topological structures.

Mainstream distributed word embeddings simplify this process and implicitly express the topological structures amongst neurons as dense weight matrices in a deep neural network. However, such a representation often leads to significant parameter redundancy. Given N neurons, for two arbitrary neurons i and j, we use a parameter w_{ij} similar to neural weights to express their relation. As a result, we need N^2 trainable parameters to describe their topological structure. This representation results in inefficient parameter utilization. Inspired by Pei & Wang (2023), we can implicitly represent the topological structure amongst neurons, using the dynamical states of these neurons, expressed as D-dimensional embeddings, *e.g.*, $\mathbf{q}_i \in \mathbb{R}^D$, where $i \in [1, N]$ and $D \ll N$. Then, the relation between neurons i and j can be expressed using a shared metric function μ, such as the L_p-norm, *i.e.*, $w_{ij} = \mu(\mathbf{q}_i, \mathbf{q}_j) = \|\mathbf{q}_i - \mathbf{q}_j\|_p$. This way, we only need $N \times D$ trainable parameters to describe the topological structure between these N neurons.

Therefore, the key to simulating semantic field interaction lies in transforming input data into embedding-like neuronal dynamics and enabling neurons to determine how to process signals based on the relations between these neuronal dynamics, *aka.*, neuronal states.

2.3. Dealing with Semantic Fields via Functionals

We begin with relevant neurobiological knowledge about how the human brain operates based on Gazzaniga et al. (2006). Specifically, the cerebral cortex is the most essential part of the human brain, and it plays a crucial role in perception, awareness, thought, memory, and language. Neurons in the cerebral cortex are organized into horizontal cortical layers and radially into cortical columns and minicolumns. A human brain has 2–4 million cortical columns, with 50 to 100 cortical minicolumns per cortical column. Neurons within the cortical minicolumn encode similar features, share identical inputs and outputs, are interconnected, and constitute a fundamental computational unit of the cerebral cortex. Upon receiving a signal, various cortical minicolumns in the brain's cortex combine the states of internal neurons, *i.e.*, neuronal states, to process the signal. Subsequently, neuronal communication occurs between adjacent cortical minicolumns, and specific brain regions integrate the temporary signal outputs of each cortical column based on neuronal communication. We use functional, *i.e.*, the function of functions, as the mathematical interpretation of this neuronal communication.

From a biological view, the semantic field essentially corresponds to the cortical column of the human brain. Each semantic field refers to a cluster of neighboring neurons designed for the initial processing of raw signals received by the brain. Intuitively, the relation between tokens within the semantic field neatly corresponds to various linguistic patterns. For example, specific semantic fields are adept at capturing Part-of-Speech features between tokens, while others excel at capturing dependency features or named entities among tokens. This characteristic closely aligns with the observed functional specialization in different regions of the human brain. Therefore, opting for the semantic field as our model's fundamental component is biologically more plausible and more apt at the linguistic level.

3. Functional Representation of Words

This section presents how to interpret a sequence of words as a functional representation of semantic fields. Then, we build a nonlinear module via this proposed functional representation to map the sequential data.

We begin by defining semantic fields that serve as global variables. These semantic fields are composed of neurons capable of interacting and influencing each other. These semantic fields transform the current input data, *i.e.*, $\mathbf{X}$, into neuronal states and establish associations between the current and past inputs through interactions among neurons. This design allows us to capture more comprehensive historical semantic information. However, relying solely on semantic fields enables us to capture only global high-level semantic information of $\mathbf{X}$, lacking the ability to capture specific low-level linguistic details. To address this issue, we integrate semantic fields with functionals. Recall that a functional is the function of functions. Since each semantic field refers to a function, we regard the interactions between semantic fields as functional, guiding each semantic field on processing $\mathbf{X}$ in a more detailed and targeted manner.

Based on the above pipeline, we construct a nonlinear mapping module, *LasF*, for sequential data. Each *LasF* module contains several trainable *neuronal projections* that capture the neuronal states in the semantic fields, *i.e.*, the high-level semantic information, and several trainable *signal projections* that capture $\mathbf{X}$'s low-level linguistic details.

Complete Workflow of a *LasF* Module: Before delving into the detailed implementation of the *LasF* block, let us take a brief glimpse at the workflow:

$$\begin{gathered}
\mathbf{Q}[l] = \mathcal{D}[l]^\top \mathbf{X}\mathcal{L}[l] \in \mathbb{R}^{L_Q \times D_Q} \\
\mathbf{S}[l] = \mathcal{D}_S[l]^\top \mathbf{X}\mathcal{L}_S[l] \in \mathbb{R}^{L_Q \times D_S} \\
\Psi[l,i,j] = \mu(\mathbf{Q}[l,i], \mathbf{Q}[l,j]) \in \mathbb{R} \\
\tilde{\mathbf{Q}}[l] = \frac{1}{L_Q}\sum_{i=1}^{L_Q} \mathbf{Q}[l,i] \in \mathbb{R}^{D_Q} \\
\Phi[l,m] = \sum_{k=1}^{N_f} \mu_k(\tilde{\mathbf{Q}}[l], \tilde{\mathbf{Q}}[m]) \in \mathbb{R} \\
\mathbf{O}[l] = \sigma\Big(\sum_{m=1}^{N_s} \Phi[l,m]\cdot\Psi[m]\cdot\mathbf{S}[m]\Big) \in \mathbb{R}^{L_Q \times D_S} \\
\hat{\mathbf{Y}} = \mathcal{D}_Y{}^\top \Big(\sum_{l=1}^{N_s} \mathbf{O}[l]\Big)\mathcal{L}_Y \in \mathbb{R}^{L_Y \times D_Y}
\end{gathered} \tag{1}$$

where $L_Q, D_Q, D_S, N_s, N_f \in \mathbb{N}^+$; $l, m \in [1, N_s]$; $i, j \in [1, L_Q]$, where N_s is the number of semantic fields, and μ and $\{\mu_k\}$ are specific metrics like L_p-norm. The mappings $\mathcal{D}, \mathcal{L}, \mathcal{D}_S, \mathcal{L}_S, \mathcal{D}_Y$ and $\mathcal{L}_Y$ are trainable linear projections whose details will be presented later.

3.1. Neuronal Projections and Functional Relations

The neuronal projections are divided into two categories, length-wise and dimension-wise, to capture more linguistic patterns. The length-wise neuronal projection $\mathcal{L} \in \mathbb{R}^{N_s \times D_X \times D_Q}$ maps the inputs to N_s distinct neuronal dynamics along with $\mathbf{X}$'s length. Similarly, the dimension-wise neuronal projection $\mathcal{D} \in \mathbb{R}^{N_s \times L_X \times L_Q}$ maps the inputs to N_s distinct neuronal dynamics along with $\mathbf{X}$'s dimension. For a specific semantic field with index l, it follows,

$$\mathbf{Q}[l] = \mathcal{D}[l]^\top \mathbf{X}\mathcal{L}[l] \in \mathbb{R}^{L_Q \times D_Q} \tag{2}$$

where $l \in [1, N_s]$ and $\mathbf{Q}[l]$ refers to the neuronal states of the semantic field l, which contains L_Q neurons interpreted as D_Q-dimensional embeddings. Next, we present how to obtain the functional, *i.e.*, the interaction between semantic fields. First, we compute the relations between neurons within each semantic field, yielding neuron-level relational matrices $\Psi \in \mathbb{R}^{N_s \times L_Q \times L_Q}$,

$$\Psi[l,i,j] = \mu(\mathbf{Q}[l,i], \mathbf{Q}[l,j]) \tag{3}$$

where $\mu : \mathbb{R}^{D_Q} \times \mathbb{R}^{D_Q} \to \mathbb{R}$ is a specific metric function, *e.g.*, L_p-norm, cosine similarity. To obtain the relations amongst all neurons across all semantic fields, a straightforward approach is to sum these obtained neuron-level relational matrices as follows,

$$\hat{\Psi}[i,j] = \sum_{l=1}^{N_s} \Psi[l,i,j] \tag{4}$$

However, the interacting patterns captured by Eq. 4 are highly limited, failing to capture interactions between neurons from different semantic fields adequately. To overcome this limitation, it is imperative to compute relations between all semantic fields. First, a concise encoding method is required to define each semantic field. For instance, a straightforward approach involves averaging the neuronal states of all neurons within a given semantic field,

$$\tilde{\mathbf{Q}}[l] = \frac{1}{L_Q}\sum_{i=1}^{L_Q} \mathbf{Q}[l,i] \in \mathbb{R}^{D_Q} \tag{5}$$

Then, we obtain a field-level relational matrix $\Phi \in \mathbb{R}^{N_s \times N_s}$ via computing the relations amongst $\tilde{\mathbf{Q}}[1], ..., \tilde{\mathbf{Q}}[N_s]$ using N_f distinct metrics,

$$\Phi[l,m] = \sum_{k=1}^{N_f} \mu_k(\tilde{\mathbf{Q}}[l], \tilde{\mathbf{Q}}[m]) \tag{6}$$

where $\{\mu_k\}$ is a set of pre-defined metrics containing L_p-norm with $p = \{1, 2, 3\}$. In the end, we obtain N_s neuron-level relational matrices $\Psi[l] \in \mathbb{R}^{L_Q \times L_Q}$, $l \in [1, N_s]$ and a field-level relational matrix $\Phi \in \mathbb{R}^{N_s \times N_s}$. The field-level relational matrix Φ serves as a functional, guiding the neuron-level relational matrices $\Psi[l]$, *i.e.*, functions, to collaboratively process the input data $\mathbf{X}$. These neuron-level and field-level relational matrices act as a functional representation of neuronal relations. Unlike Eq. 4 that statically describes the neuronal relations, the functional representation dynamically deals with the neuronal relations during the inference. We will present more implementation details after introducing the signal projections.

3.2. Signal Projections and Functional Mappings

A signal projection is a set of linear transformations used to map input data $\mathbf{X} \in \mathbb{R}^{L_X \times D_X}$ onto signals $\mathbf{S} \in$

$\mathbb{R}^{N_s \times L_Q \times D_S}$ for all neurons in each semantic field. Similar to a neuronal projection, it involves two categories of linear projections, *i.e.*, $\mathcal{D}_S \in \mathbb{R}^{N_s \times L_X \times L_Q}$, and $\mathcal{L}_S \in \mathbb{R}^{N_s \times D_X \times D_S}$, such that,

$$\mathbf{S}[l] = \mathcal{D}_S[l]^\top \mathbf{X} \mathcal{L}_S[l] \in \mathbb{R}^{L_Q \times D_S} \tag{7}$$

Now we can obtain the signal received by each neuron; we represent the signal received by a neuron i in a semantic field l as $\mathbf{S}[l,i] \in \mathbb{R}^{D_S}$. Intuitively, each semantic field returns a temporary output $\mathbf{H} \in \mathbb{R}^{N_s \times L_Q \times D_S}$ by simply simulating the process of signal propagation amongst neurons using the neuron-level relational matrices, *i.e.*,

$$\mathbf{H}[l,i] = \sigma\Big(\sum_{j=1}^{L_Q} \Psi[l,i,j] \cdot \mathbf{S}[l,j]\Big) \in \mathbb{R}^{D_S} \tag{8}$$

where σ is a nonlinear activation function, *e.g.*, *sigmoid*, and $i \in [1, L_Q]$ is the index of a neuron. Next, we elaborate on processing these signals and generating the final output using the functional representation proposed in Section 3.1. The core idea is that each semantic field aggregates information from all semantic fields, and the field-level relational matrix Φ serves as a weighting mechanism. Specifically, the actual output signal $\mathbf{O}[l] \in \mathbb{R}^{L_Q \times D_S}$ of a semantic field l is the weighted sum of the temporary output signals $\mathbf{H}[m]$ from all other semantic fields $m = \{1, ..., N_s\}$ under the guidance of Φ to implement the weighting process,

$$\mathbf{O}[l] = \sigma\left(\sum_{m=1}^{N_s} \Phi[l,m] \cdot \mathbf{H}[m]\right) \tag{9}$$

To be compatible with parallel computing, we vectorize Eq. 8 and Eq. 9 as follows,

$$\mathbf{O}[l] = \sigma\left(\sum_{m=1}^{N_s} \Phi[l,m] \cdot \Psi[m] \cdot \mathbf{S}[m]\right) \in \mathbb{R}^{L_Q \times D_S} \tag{10}$$

We can further vectorize Eq. 10 via batch computation,

$$\mathbf{O} = \sigma(\Phi \Psi \mathbf{S}) \in \mathbb{R}^{N_s \times L_Q \times D_S} \tag{11}$$

Finally, we obtain the ultimate output by summing the output signals $\mathbf{O}[l]$ from each semantic field, *i.e.*,

$$\hat{\mathbf{Y}} = \mathcal{D}_Y{}^\top \Big(\sum_{l=1}^{N_s} \mathbf{O}[l]\Big) \mathcal{L}_Y \tag{12}$$

where $\mathcal{D}_Y \in \mathbb{R}^{L_Q \times L_Y}$ and $\mathcal{L}_Y \in \mathbb{R}^{D_Q \times D_Y}$ are trainable linear projections used to shape-align the output signal with the target output $\mathbf{Y} \in \mathbb{R}^{L_Y \times D_Y}$. In Appendix E, we further validate that our *LasF* module is conceptually different from a typical Transformer.

4. Integrating *LasF* with Pre-trained LMs

Next, we explore the integration of the *LasF* module with pre-trained language models using Transformers. The method is straightforward: We can directly replace the last Transformer layer and its output, used as the classifier, in the pre-trained language model with the *LasF* module. The specific replacement depends on the NLP downstream task at hand. For example, in the case of a multiple-choice task, considering a pre-trained language model, whose last Transformer layer receives input data $\mathbf{X} \in \mathbb{R}^{L_X \times D_X}$, with the final output of its subsequent classifier Linear layer being $\mathbf{Y} \in \mathbb{R}^{1 \times D_Y}$, where D_Y represents the number of choices. We can directly replace this Transformer layer and its subsequent classifier Linear layer with the *LasF* module. The hyperparameters such as L_X, D_X, L_Y and D_Y are predetermined in terms of the current task, while other hyperparameters like L_Q, D_Q, D_S, N_s and N_f can be adjusted based on their performance on the development set.

Note that pre-trained language models like BERT can handle variable-length input sequences. Therefore, we must consider the case for a flexible L_X. We propose two types of solutions. The first type involves converting all trainable projections in the *LasF* module related to $L_X \mapsto L_Q$ into non-trainable kernels, such as an identity matrix with $L_X = L_Q$, Gaussian kernel, *etc*. The second type involves training $\mathcal{D}$ and $\mathcal{D}_S$ based on the maximum value L_X might take, and then, during inference, truncating these projections based on the actual length of the input sequence. Empirical results show that if we add positional encodings (Vaswani et al., 2017) to *LasF*'s inputs, their performance is almost identical. By default, we apply the first solution that degrades $\mathcal{D}$ and $\mathcal{D}_S$ to non-trainable kernels such that $\mathcal{D}[l,i,j] = \mathcal{D}_s[l,i,j] = \mathcal{N}(i-j)$, where $l \in [1, N_s],\ i,j \in [1, L_Q]$ and $\mathcal{N}$ refers to a normal Gaussian distribution.

Building on the approach above, the *LasF* module can also be combined with generative LLM. Taking LLaMA2 as an example, we can replace its Linear layer output of shape $\mathbb{R}^{4096 \times 32000}$ with an *LasF* module of $D_X = 4096, L_Y = 1$ and $D_Y = 32000$, and degrade the trainable projections $\mathcal{D}, \mathcal{D}_S$ related to L_X and L_Q to non-trainable kernels. Experiments show that compared to using an identity matrix as the kernel, using a more complex kernel (such as a Gaussian kernel) allows the *LasF* module to have stronger semantic capturing capabilities. However, future work requires exploring whether the additional computational cost of complex kernels can ensure the model's cost-effectiveness.

5. Language Model from Scrath via *LasF*

In Sec. 4, we discuss integrating the *LasF* module with pre-trained language models. Since we can replace the last

Transformer layer and its output Linear layer with a $LasF$ module, we can also replace all Transformer layers with $LasF$ modules. The reduced $LasF$ module, termed $rLasF$, is presented as follows:

$$\begin{aligned}
\mathbf{Q}[l] &= \mathcal{D}[l]^\top \mathbf{X}\mathcal{L}[l] \in \mathbb{R}^{L_Q \times D_Q} \\
\mathbf{S}[l] &= \mathcal{D}_S[l]^\top \mathbf{X}\mathcal{L}_S[l] \in \mathbb{R}^{L_Q \times D_S} \\
\Psi[l,i,j] &= \mu(\mathbf{Q}[l,i], \mathbf{Q}[l,j]) \cdot \Lambda[i,j] \in \mathbb{R} \\
\Phi[l,m] &= \sum_{k=1}^{N_f} \mu_k(\sum_{i=1}^{L_Q} \mathbf{Q}[l,i], \sum_{i=1}^{L_Q} \mathbf{Q}[m,i]) \in \mathbb{R} \\
\hat{\mathbf{Y}} &= \sum_{l=1}^{N_s} \sigma(\Phi\Psi\mathbf{S})[l]\mathcal{L}_Y \in \mathbb{R}^{L_Q \times D_Y}
\end{aligned} \tag{13}$$

where all variable notations here follow the previous sections, except $L_Q = L_X = L_Y$, $D_Q = D_S$, and $D_X = D_Y$. The smoothing term $\Lambda \in \mathbb{R}^{L_Q \times L_Q}$ acts as a masking mechanism such that $\Lambda[i,j] = 0$ if $j > i$ else $\Lambda[i,j] = 1$. The length-related projections $\mathcal{D}$ and $\mathcal{D}_S$ are degraded into non-trainable kernels for computational efficiency. The $rLasF$ module described in Eq. 13 acts as a nonlinear operator that maps an input $\mathbf{X} \in \mathbb{R}^{L \times D}$ to an output $\mathbf{Y} \in \mathbb{R}^{L \times D}$ of the same shape.

Following the standard autoregressive language modeling pipeline, we proceed to construct our language model that maps the randomly initiated input encodings $\mathbf{X} \in \mathbb{R}^{L \times D}$ with varying length L and fixed dimension D to the output $\mathbf{Y} \in \mathbb{R}^{L \times N_v}$, where N_v is the vocabulary size. We stack several $rLasF$ layers, and then append the $LasF$ module described in Eq. 1 after the final $rLasF$ layer to obtain the ultimate output, *i.e.*,

$$\begin{aligned}
\hat{\mathbf{Y}}^{(1)} &= rLasF(\mathbf{X}) \\
\hat{\mathbf{Y}}^{(r+1)} &= rLasF(\hat{\mathbf{Y}}^{(r)}),\ r \in [1, N_r - 2] \\
\hat{\mathbf{Y}} &= LasF(\hat{\mathbf{Y}}^{(N_r-1)})
\end{aligned} \tag{14}$$

where N_r is the total number of $LasF$-based layers stacking together. Empirical results demonstrate that even with a very small N_r, *e.g.*, $N_r = 2$, the language modeling capability of the resulting $LasF$-based language model surpasses that of language models with 10+ layers of Transformers.

6. Experiments

We focus on three NLP tasks involving seven public datasets: reading comprehension on SQuAD2.0 (Rajpurkar et al., 2018), question answering on CsQA (Talmor et al., 2018), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019)), and language modeling on WikiText-103 (Merity et al., 2016), PennTreebank (Marcus et al., 1993). Besides, we also test the representational capacity of *LasF*-based encoding in the task of semantic compression on WikiQA (Yang et al., 2015), see Appendix A. All experiments can be conducted on a single 24GB memory GeForce RTX 4090 GPU, on which we can train a *LasF*-based language model within a week.

6.1. Reading Comprehension

For reading comprehension, we use SQuAD2.0 dataset (Rajpurkar et al., 2018), which is a challenging natural language understanding benchmark for reading comprehension. The dataset combines $100,000$ questions extracted from its former version with over $50,000$ new unanswerable questions written by crowdworkers. The training dataset contains 87k answerable and 43k unanswerable questions. In this task, we use some fine-tuned base models[2], *e.g.*, RoBERTa-base (Liu et al., 2019), ALBERT-base (Lan et al., 2019), and DeBERTa-base (He et al., 2020), as the text encoders to obtain the encoded inputs feeding into the *LasF* module defined via the mechanism presented in Section 4, followed by fine-tuning them on the SQuAD2.0's training dataset. Besides, we set $D_Y = 2$ and $D_S = D_Q$ as defaults. The exact values of N_S and $D_S = D_Q$ are presented in Table 1. Empirical results show that replacing the output layer of a BERT-based language model with *LasF* module can capture more linguistic patterns. All the *LasF*-based language models outperform their original forms with fewer parameters.

6.2. Question Answering

For question answering, we use three datasets, including Commonsense QA (Talmor et al., 2018), Physical Interaction QA (Bisk et al., 2020), and Social Interaction QA (Sap et al., 2019). To be specific, CommonsenseQA (CsQA) is a 5-way multiple choice QA dataset for commonsense question answering, Physical IQA (PIQA) is a 2-way multiple choice QA dataset focusing on everyday situations with a preference for typical solutions, Moreover, Social IQA (SIQA) is a three-way multiple-choice QA dataset focusing on reasoning about people's actions and social implications. For each QA dataset, we first conduct experiments on LLaMA-7B and LLaMA-13B, whose last Linear layers of shapes $\mathbb{R}^{4096 \times 32000}$ and $\mathbb{R}^{5120 \times 32000}$ are replaced with a $LasF$ module of $L_Y = 1$ and $D_Y = 5$. The exact values of N_S, D_Q, and D_S are presented in Table 2. We train the newly added *LasF* modules on the training set while LLaMA's other parameters remain fixed. Empirical results show that the *LasF* modules are compatible with large language models to capture more semantic features.

We also test our method on CsQA's blind test set, using RoBERTa-large (Liu et al., 2019) and ALBERT-xxlarge-v2 (Lan et al., 2019) as the pre-trained text encoders, re-

[2]We select the best-performing models among the fully open-sourced and computational affordable models in the leaderboard. The models' structures are also relatively representative.

Table 1: **Performance comparison on SQuAD2.0 dataset**. We test our method for reading comprehension tasks by replacing the output layers of the selected fine-tuned pre-trained models. For each model, we use three configurations for *LasF*, denoting as the form of Sx-Dy, where x refers to the number of semantic fields N_s, and y refers to the dimension of neuronal states D_Q.

Base Model	Output Layer	No.Params	EM (%)	F1 (%)	Improvement (%)
ALBERT-base	Transformer + Linear	7.68M	76.05	79.15	0
	LasF (S10-D10)	0.15M	76.12	79.04	- 0.11
	LasF (S10-D20)	0.31M	76.59	79.92	+ 0.77
	LasF (S20-D20)	0.61M	**77.22**	**80.33**	**+ 1.18**
RoBERTa-base	Transformer + Linear	7.68M	79.78	82.32	0
	LasF (S10-D10)	0.15M	79.80	82.38	+ 0.06
	LasF (S10-D20)	0.31M	80.35	83.12	+ 0.80
	LasF (S20-D20)	0.61M	**80.73**	**83.35**	**+ 1.03**
DeBERTa-base	Transformer + Linear	7.68M	81.25	84.50	0
	LasF (S10-D10)	0.15M	81.25	84.31	- 0.19
	LasF (S10-D20)	0.31M	82.07	85.15	+ 0.65
	LasF (S20-D20)	0.61M	**82.44**	**85.54**	**+ 1.04**

Table 2: **Performance comparison on Question Answering Datasets**. For each model, we use two configurations for *LasF*, denoting as the form of Sx-D_Qy-D_Sz, where x refers to the number of semantic fields N_s, y refers to the dimension of neuronal states D_Q, and z refers to the dimension of signals.

	Output Layer	No.Params	Accuracy (%)		
			CSQA	PIQA	SIQA
LLaMA-7B	Transformer + Linear	131.1M	57.8	79.8	48.9
	LasF (S10-D_Q10-D_S512)	21.7M	56.9	80.0	49.2
	LasF (S20-D_Q20-D_S1024)	86.1M	**62.2**	**80.8**	**49.8**
LLaMA-13B	Transformer+Linear	163.8M	67.3	80.1	50.4
	LasF (S10-D_Q10-D_S512)	27.1M	67.0	80.5	50.4
	LasF (S20-D_Q20-D_S1024)	86.1M	**69.4**	**82.1**	**51.8**

spectively, followed by fine-tuning the whole model on the training set. In this case, we set $D_Q = D_S = 30$ and $N_s = 10$ as defaults. Table 3 shows that *LasF* outperforms strong fine-tuned LM baselines (*e.g.*, RoBERTa-large and ALBERT-xxlarge) and the best amongst all the single models. Specifically, UnifiedQA (Khashabi et al., 2020) has 11B parameters and is based on T5, impractical to fine-tune without strong GPU servers. ALBERT+HeadHunter uses Open Mind CommonSense corpus (Singh et al., 2002) as an additional knowledge base regarded as an extended commonsenseQA dataset. Our *LasF* still outperforms UnifiedQA and HeadHunter by 3.1% and 3.8%, respectively. As presented in Table 5, we also conduct in-house controlled experiments on CsQA's development sets, compared with several comparable methods applied to base language models.

6.3. Language Modeling

For language modeling, we only conduct small-scale experiments, due to limited computational resources, on WikiText-103 (Merity et al., 2016) and PennTreebank (Marcus et al., 1993) datasets to evaluate the $LasF$-based language models. The model architecture follows Eq. 14 with $D_X = 768$, $D_Q = 15$, and $N_s = 12$. (For other hyper-parameters, see Table 6.) For training details, we use SGD with a decaying learning rate starting from 0.3 as the optimizer; we set the batch size as 16 and the maximum number of epochs as 50. We only train our language models on WikiText-103 or PennTreebank training set without training on additional text corpus. As presented in Table 4, our *LasF*-based language models trained from scratch achieve lower perplexities than other Transformer-based and Transformer-free models.

Although the current experimental scale is small and cannot conclusively demonstrate that the $LasF$ module outperforms the Transformer, empirical results indicate that $LasF$ concept is effective for learning in larger-scale language scenarios. Additionally, we observe that $LasF$ exhibits a higher layer efficiency, meaning it can achieve stronger language modeling capabilities with fewer layers. In Table 4, we observe that, under comparable parameter conditions, a two-layer $LasF$-based language model significantly outperforms a two-layer Transformer-based language model in terms of perplexity. Overall, $LasF$-based language models can achieve language modeling capabilities comparable to

Table 3: **Test accuracy on CommonsenseQA's official leaderboard**. Note that models with * use ConceptNet. The CSQA's official team no longer accepts submission using ConceptNet. Our method outperforms all the prior ensemble and single models presented in CsQA's official leaderboard.

Methods	Parameters	Test-Acc. (%)		Use External Sources	
		single	ensemble	QA datasets	KGs
BERT-large	$\sim 345M$	56.7	-	-	-
KagNet*	$> 345M$	-	58.9	✓	✓
RoBERTa-large	$\sim 354M$	72.1	72.5	-	-
ALBERT-large	$\sim 223M$	73.5	76.5	-	-
ALBERT+PathGenerator	$> 345M$	75.6	78.2	-	✓
QA-GNN*	$\sim 360M$	76.1	-	-	✓
ALBERT+HGN*	$\sim 355M$	77.3	-	-	✓
T5	$\geq 11B$	78.1	-	-	-
ALBERT+HeadHunter	$\sim 283M$	78.4	78.3	-	✓
UnifiedQA	$\sim 11B$	79.1	-	✓	-
DeBERTa	$\sim 1.5B$	-	79.6	✓	-
ALBERT+SFR	-	-	81.8	-	-
ALBERT+LasF (Ours)	$\sim 225M$	**82.2**	-	-	-

Table 4: **Performance comparision on WikiText103 and PennTreebank for Language Modeling**. All models use the same GPT2 tokenizer. The results for models with $*$ are taken from Radford et al. (2019) and Poli et al. (2023).

Model	No.Parameters	No.Layers	Extra Training Data	Time Cost per batch		Perplexity (↓)	
				Training	Inference	WikiText103	PennTreebank
Transformer	92M	2	×	265	48	45.8	72.5
Transformer $*$	125M	12	×	-	-	18.6	54.5
GPT-2 small $*$	125M	12	√	-	-	37.5	65.8
Reformer $*$	125M	12	√	-	-	26.0	-
Hybrid H3 $*$	125M	12	√	-	-	23.7	-
S4 $*$	249M	12	×	-	-	21.8	-
Hyena $*$	125M	12	×	-	-	18.5	-
S5-Hyena $*$	125M	12	×	-	-	18.3	-
LasF (Ours)	83M	1	×	65	12	20.2	59.8
LasF (Ours)	107M	2	×	107	18	17.2	49.2
LasF (Ours)	122M	3	×	155	26	**16.5**	**44.3**

deeper models with fewer layers, *i.e.*, a flatter model structure. This observation suggests that interpreting language tokens in the form of a functional semantic field may be more similar to the mechanisms by which the human brain processes language, the two primary components of the human brain cortex, the neocortex and allocortex, have only six and four layers, respectively (Shipp, 2007).

Table 4 also indicates that, under the same number of layers and parameter size, the training and inference times of $LasF$-based language models are approximately 2.6 times faster than Transformer-based language models. Moreover, there is still room for speeding up the metrics-based matrix computation in $LasF$-based language models, as there are fast algorithms specifically designed for matrix multiplication in metric spaces (Indyk & Silwal, 2022; Pei & Wang, 2023). The details of implementing this improvement are left for future work.

7. Why a FLAT *LasF* can Work

Unlike multi-head Attention, which uses a Query-Key information retrieval approach to achieve nonlinear mapping, the *LasF* module focuses on achieving nonlinear mapping through transmitting signals between neurons and neuronal groups. The signal transmission amongst one layer of neurons is equivalent to the feed-forward process across multiple neural layers. Intuitively, we interpret an *LasF* module as a time-variant dynamical system containing neurons transmitting signals. For brevity, we simplify the time-variant *LasF* module as follows,

$$\begin{aligned} \mathbf{S}^{(t+1)}[l;i] &= f_s^{(l)}\big(\mathbf{Q}[l;i], \mathbf{S}^{(t)}[l;i]\big) \\ \tilde{\mathbf{S}}^{(t+1)} &= F\big(\tilde{\mathbf{Q}}[l], \tilde{\mathbf{S}}^{(t)}[l]\big). \end{aligned} \tag{15}$$

Here, $l \in [1, N_s]$, where N_s is the number of semantic fields, $\mathbf{S}$ represents the signals transmitted between neurons, and $\mathbf{Q}$ represents the states of the neurons; $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{Q}}$ are the merged encodings of $\{\mathbf{Q}[1], ...\mathbf{Q}[N_Q]\}$ and $\{\mathbf{S}[1], ..., \mathbf{S}[N_S]\}$. For

Table 5: **Performance comparison on CommonsenseQA in-house controlled experiments**. As the official test set is hidden, we report the accuracies of in-house dev (IHdev-Acc) and test (IHtest-Acc), following the data split of Lin et al. (2019). The DEKCOR* and KEAR* methods use the prohibited ConceptNet, whose empty triplets explicitly correspond to the human-generated distractor choices. Therefore, we randomly initiate the empty triplets to eliminate the shortcut hints.

Methods	IHdev-Acc. (%)	IHtest-Acc. (%)
GPT-3.5-turbo	73.3	-
RoBERTa-large	73.1 ± 0.5	68.7 ± 0.6
+KagNet	73.5 ± 0.2	69.0 ± 0.8
+PathGenerator	-	72.7 ± 0.4
+QA-GNN	76.5 ± 0.2	73.4 ± 0.9
+HGN	-	73.6 ± 0.3
+LasF (Ours)	$\mathbf{79.5 \pm 0.3}$	$\mathbf{75.4 \pm 0.4}$
ALBERT-large	78.7 ± 0.4	75.1 ± 0.4
+DEKCOR*	80.3	-
+KEAR*	81.2	-
+Headhunter	83.3	-
+LasF (Ours)	$\mathbf{87.1 \pm 0.3}$	$\mathbf{83.8 \pm 0.4}$

instance, Eq. 5 presents a merged encoding as $\tilde{\mathbf{Q}}$. The f and F denote specific trainable nonlinear functions.

Obviously, the pre-training process for a flattened *LasF* model forces *LasF* to use fewer time-steps to reach equilibrium, *i.e.*, $\mathbf{S}^{(t+1)} = \mathbf{S}^{(t)}$, as presented in Eq. 15, since more time-steps and the resulting increased neuronal interaction lead to an exponentially increasing loss. We now understand that the pre-training process for *LasF* is equivalent to reducing the required time-steps. Next, we can observe that the number of these time-steps is actually equivalent to the model's depth. This is because state-space modes that require more time-steps are mathematically equivalent to deeper neural models. Therefore, we can conclude that the pre-training process for *LasF* also equates to forcing *LasF* to achieve the representational capacity of a deeper neural model with less depth. This explains why a flat *LasF* model's language modeling capability is competitive with that of a deep Transformer model.

8. Ablation Study

We conduct ablation experiments on the language modeling task, and the experimental results are presented in Table 6. The controlled variables involve metrics μ involved in Eq. 3, the set of metrics $\{\mu_k\}$ involved in Eq. 6, and the length-related projections $\mathcal{D}$ and $\mathcal{D}_S$ involved in Eq. 2 and Eq. 7.

Table 6: **Ablation Study on Language Modeling**.

Components	Variants	Perplexity (↓)	
		WikiText103	PennTreebank
2layer *LasF*	/	**17.2**	**49.2**
μ	Cosine-sim	17.8	50.1
	L_1-norm	17.6	49.5
	L_2-norm	**17.2**	**49.2**
	L_3-norm	17.3	49.4
μ_k	L_1-norm	18.3	52.2
	$L_{1,2}$-norm	17.8	50.5
	$L_{1,2,3}$-norm	**17.2**	**49.2**
$\mathcal{D}\ \{\mathcal{D}_s\}$	Truncated	17.6	49.3
	Identity	17.5	**49.2**
	Gaussian	**17.2**	**49.2**

9. Limitation

We would like to discuss briefly on the limitation of our *LasF*, particularly regarding to the GPU implementation and hyper-parameter tuning issues:

- CUDA Implementation. One challenge we face is the limited availability of well-established CUDA packages for efficiently computing metric-based operations that are intensively required by our method. Integrating our approach into the Large Language Model (LLM) community demands additional efforts, particularly in terms of CUDA implementation and acceleration.
- Hyper-parameter Tuning. The introduction of novel concepts in our method, such as neuronal groups, represents an area that is not extensively explored in the existing literature. Consequently, further investigations are needed to empirically establish the relationships between these concepts and their theoretical implications. We acknowledge the importance of refining the content regarding to these novel elements and will address this concern in the revision by providing additional insights and analyses.

10. Conclusion

In this study, we introduce a novel approach termed *LasF*, which simulates semantic fields using functional representation to model language. This method offers enhanced interpretability from a linguistic and neuro-biological perspective while substantially increasing parameter efficiency. Empirical results across reading comprehension, question answering, and language modeling tasks consistently illustrate the superiority of *LasF* compared with other language modeling paradigms. In the future, we aim to train a general-purpose large-scale flat LLM from scratch based on *LasF*.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

This work was supported in part by the National Key R&D Program of China under Grant 2023YFC2508704, in part by National Natural Science Foundation of China: 62236008, U21B2038 and 61931008, and in part by the Fundamental Research Funds for the Central Universities.

References

Bengio, Y., Ducharme, R., and Vincent, P. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Cai, Y., Fan, Y., Guo, J., Zhang, R., Lan, Y., and Cheng, X. A discriminative semantic ranker for question retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pp. 251–260, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Evangelopoulos, N., Zhang, X., and Prybutok, V. R. Latent semantic analysis: five methodological recommendations. *European Journal of Information Systems*, 21(1):70–86, 2012.

Fournier, Q., Caron, G. M., and Aloise, D. A practical survey on faster and lighter transformers. *ACM Computing Surveys*, 55(14s):1–40, 2023.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

Gazzaniga, M. S., Ivry, R. B., and Mangun, G. Cognitive neuroscience. the biology of the mind,(2014), 2006.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Indyk, P. and Silwal, S. Faster linear algebra for distance matrices. *Advances in Neural Information Processing Systems*, 35:35576–35589, 2022.

Jackson, H. and Amvela, E. Z. *Words, meaning and vocabulary: An introduction to modern English lexicology*. Bloomsbury Publishing, 2007.

Khashabi, D., Min, S., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., and Hajishirzi, H. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*, 2020.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Lang, S. *Algebra*, volume 211. Springer Science & Business Media, 2012.

Lin, B. Y., Chen, X., Chen, J., and Ren, X. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*, 2019.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Mingbo, L., Yinghan, Y., and Shengde, M. Exploration about whether the first few dimensions of word embedding encode frequency information. In *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pp. 313–316. IEEE, 2021.

Oota, S. R., Arora, J., Agarwal, V., Marreddy, M., Gupta, M., and Surampudi, B. R. Neural language taskonomy: Which nlp tasks are the most predictive of fmri brain activity? *arXiv preprint arXiv:2205.01404*, 2022a.

Oota, S. R., Gupta, M., and Toneva, M. Joint processing of linguistic properties in brains and language models. *arXiv preprint arXiv:2212.08094*, 2022b.

Patterson, K., Nestor, P. J., and Rogers, T. T. Where do you know what you know? the representation of semantic knowledge in the human brain. *Nature reviews neuroscience*, 8(12):976–987, 2007.

Pei, Z. and Wang, S. Dynamics-inspired neuromorphic visual representation learning. In *International Conference on Machine Learning*, pp. 27521–27541. PMLR, 2023.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023.

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

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500): 2323–2326, 2000.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Sap, M., Rashkin, H., Chen, D., LeBras, R., and Choi, Y. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

Shipp, S. Structure and function of the cerebral cortex. *Current Biology*, 17(12):R443–R449, 2007.

Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T., and Li Zhu, W. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences*, pp. 1223–1237. Springer, 2002.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867, 2020.

Sun, J. and Moens, M.-F. Fine-tuned vs. prompt-tuned supervised representations: which better account for brain language representations? *arXiv preprint arXiv:2310.01854*, 2023.

Suzuki, M., Pennartz, C. M., and Aru, J. How deep is the brain? the shallow brain hypothesis. *Nature Reviews Neuroscience*, 24(12):778–791, 2023.

Talmor, A., Herzig, J., Lourie, N., and Berant, J. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

Tenenbaum, J. B., Silva, V. d., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Toneva, M. and Wehbe, L. Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain). *Advances in neural information processing systems*, 32, 2019.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vasiliev, Y. *Natural Language Processing with Python and SpaCy: A Practical Introduction*. No Starch Press, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yang, Y., Yih, W.-t., and Meek, C. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 2013–2018, 2015.

Zhang, X., Bosselut, A., Yasunaga, M., Ren, H., Liang, P., Manning, C. D., and Leskovec, J. Greaselm: Graph reasoning enhanced language models for question answering. *arXiv preprint arXiv:2201.08860*, 2022.

Zhao, L., Zhang, L., Wu, Z., Chen, Y., Dai, H., Yu, X., Liu, Z., Zhang, T., Hu, X., Jiang, X., et al. When brain-inspired ai meets agi. *Meta-Radiology*, pp. 100005, 2023.

A. Functional representation of semantic units in semantic compression task

A semantic unit is analogous to a token/word and is strictly a conceptual element that implements reasoning tasks. A well-formed reasoning model needs as few semantic units as possible while preserving its reasoning capacity as much as possible. Giving a semantic unit with index $i \in \{1, ..., M\}$, Instead of treating it as a fixed-dimensional encoding, we treat it as a functional $\mathcal{F}_i$ that takes the functional and tensor-formed parameters, including the objectives $\mathbf{T}$, the external states $\mathbf{E}$, and the internal states $\mathbf{I}$ as the arguments, followed by returning a task-specific encoding $\mathbf{v}_i$ as the output:

$$\mathbf{v}_i = \mathcal{F}_i(\mathbf{T}, \mathbf{E}, \mathbf{I}) \tag{16}$$

The parameters in Eq. 16 have yet to be defined precisely. One can freely define the parameters towards the need of a given task, such as semantic compression, reading comprehension, and question answering.

For better understanding, we show how to instantiate Eq. 16 to facilitate the task of semantic compression (Cai et al., 2021). This task is to find the top-K largest components of $z = \mathbf{V}y \in \mathbb{R}^{n\times 1}$ given a query vector $y \in \mathbb{R}^{D_v \times 1}$ and n D_v-dimensional encodings $\mathbf{V} \in \mathbb{R}^{n \times D_v}$ under limited computational cost. In this case, the query y is the external state $\mathbf{E}$ in Eq. 16. The internal states $\mathbf{I}$ contain two parts: the remaining computational budget and the original-dimensional encoding $\mathbf{V}_i \in \mathbb{R}^{D_v \times 1}$. The objective $\mathbf{T}$ minimizes the query loss between the computed $\hat{z}$ and the actual z, and the computational cost as follow,

$$\mathbf{T}(D_v^{'}) = \sum_{k=1}^{K} \gamma^k \| y^\top \mathbf{V}_{r_k} - (\mathbf{P}[D_v^{'}]y)^\top (\mathbf{P}[D_v^{'}]\mathbf{V}_{r_k}) \| + D_v^{'\,2} \tag{17}$$

where $D_v^{'} \in [1, D_v - 1]$, $\gamma \in [0,1]$, $\mathbf{P}[D_v^{'}] \in \mathbb{R}^{D_v^{'} \times D_v}$. The index r_k refers to the candidate that corresponds to the k-largest components of z, *i.e.*, $z_{r_k} = y^\top \mathbf{V}_{r_k}$ is the k-largest ones of $\{z_1, ..., z_n\}$. $\mathbf{P}[D_v^{'}]$ is trained by minimizing $\mathbf{T}(D_v^{'})$ using a set of provided queries y. In this case, $\mathcal{F}_i$ presents a heuristic way to recursively compute $\mathbf{V}y$, reducing the dimensions step-by-step. During a T-step iterative process, $D_v^{(0)} = \frac{D_v}{10} < D_v^{(1)} < ... < D_v^{(T)}$, we compute the reduced $\mathbf{V}^{(t)}y^{(t)}$ and filter out the candidates, *i.e.*, reduce the candidates' size from $R^{(t-1)} \in \mathbb{N}^+$ to $R^{(t)} \in [1, R^{(t-1)}]$, followed by further reducing the dimensions, computing and filtering out, *etc.*, until the computational budget is consumed. The analytical form of $\mathcal{F}_i$ for this task can be defined as

$$\mathbf{v}_i^{(t)} = \mathcal{F}_i(\mathbf{T}, y, \mathbf{I}^{(t)}) = \mathbf{P}[D_v^{(t)}]\mathbf{v}_i^{(0)} = \mathbf{P}\Big[\eta\big(\Psi - \sum_{s=0}^{t-1} R^{(s)} D_v^{(s)\,2}\big)^{\frac{1}{2}}\Big]\mathbf{v}_i^{(0)} \tag{18}$$

where $\mathbf{v}_i^{(0)} = \mathbf{V}_i$, $\eta \in (0, 1)$ is a predefined parameter that compromises the extra computational cost during pre-processing, and $\Psi \in \mathbb{N}$ is the initial computational budget that determines the expected computational complexity. The parameter $\mathbf{P}[D_v^{(t)}] \in \mathbb{R}^{D_v^{(t)} \times D_v}$ is pre-trained by back-propagating the objective $\mathbf{T}$ defined in Eq. 17. A candidate encoding i with a higher $\hat{z}_i^{(t)}$ is more likely to be selected by the filtering mechanism, where details are omitted for simplicity. Note that this functional representation can perform specified tasks more efficiently, ensuring an optimal trade-off between the execution process's cost and efficiency. However, as illustrated above, this mechanism is too specific, it needs further generalized implementation.

The WikiQA dataset (Yang et al., 2015) contains 3047 questions/queries and 29258 sentences/candidates, in which 1473 sentences were labeled as the answer to their related questions. Bing query logs were used as the query source to reflect the real-world case. Each query is linked to a Wikipedia article that contains the answer. We use the heuristic functional representation presented by Eq. 18 on this benchmark to show that the proposed functional representation contains more semantic patterns than other fixed-dimensional representations.

Table 7 lists the baseline methods for semantic compression. Incremental PCA, *i.e.*, **iPCA** (Evangelopoulos et al., 2012) reduces the query and candidate embeddings to lower dimensional compressed vectors. **Kernel PCA** (Mingbo et al., 2021) configures a non-linear mapping to transform the embeddings to higher-dimensional space, followed by standard PCA to project them back to lower-dimensional linearly separable space. Locally Linear Embedding, *i.e.*, **LLE)** aims to discover non-linear structures in the dataset and also preserve the distances within local neighborhoods (Roweis & Saul, 2000). Isometric Mapping, *i.e.*, **Isomap**, uses the spectral theory to preserve the geodesic distances in the lower dimensional space (Tenenbaum et al., 2000).

Table 7: **Performance comparison on WikiQA dataset.** We test our method on the task of semantic query on different dimension settings. Each dimension setting refers to a limited computational budget, *i.e.*, the time cost for implementing the complete query task should not exceed a specific value. We use the original embeddings generated by three distinct pre-trained text encoders. Then, we apply five methods to compress the generated embeddings or reduce the computational complexity to meet the limited computational budgets. The top 1 or top 3 accuracies are recorded.

Compress dim		dim=10	dim=20	dim=30	dim=50		dim=768	
Actual Time Cost for Query (seconds)		0.3-0.4	0.4-0.5	0.5-0.7	1.1-1.3		12-13	
Text Encoder	Method	Top1	Top1	Top1	Top1	Top3	Top1	Top3
All-mpnet	iPCA	0.0887	0.2759	0.3842	0.4926	0.7340	0.5615	0.8374
	kPCA	0.0690	0.2364	0.3941	0.5222	0.7438		
	LLE	0.2167	0.2463	0.2562	0.2611	0.5025		
	Isomap	0.2463	0.2562	0.2759	0.2906	0.5813		
	LasF (Ours)	**0.2611**	**0.3892**	**0.4384**	**0.5665**	**0.8079**		
Qa-mpnet	iPCA	0.0345	0.1330	0.2611	0.4433	0.6749	0.6010	0.8276
	kPCA	0.0246	0.1231	0.2512	0.4433	0.6601		
	LLE	0.1674	0.1921	0.2019	0.1921	0.3744		
	Isomap	0.1133	0.1133	0.1478	0.1724	0.3645		
	LasF (Ours)	**0.1872**	**0.3892**	**0.4828**	**0.5764**	**0.8177**		
Distil-roberta	iPCA	0.0493	0.2315	0.3399	0.3941	0.6749	0.3990	0.7192
	kPCA	0.0542	0.2213	0.3005	0.3892	0.6700		
	LLE	0.1478	0.1823	0.1970	0.1773	0.3695		
	Isomap	0.1773	0.2069	0.2118	0.2118	0.5074		
	LasF (Ours)	**0.2808**	**0.2709**	**0.3892**	**0.4089**	**0.6946**		

A.1. Semantic compression

Semantic search aims to improve search accuracy by capturing the semantic information of the content candidates. In Table 7, we choose three pre-trained models designed for semantic search, including *All-mpnet* (Song et al., 2020), *Qa-mpnet* (Song et al., 2020), and *Distil-roberta* (Sanh et al., 2019). to encode the queries and the candidate contents to generate contextual representations with the original dimension of 768 (Reimers & Gurevych, 2019). The experiments are conducted on four dimension settings, *i.e.*, $dim = \{10, 20, 30, 50\}$. The heuristic $LasF$ using Eq. 18 takes the original 768-dimensional embeddings as $\mathbf{v}^{(0)}$ defined in Eq. 18. We also list the actual time cost of our method for implementing semantic query to validate the fairness of comparison. *LasF* in all cases performs better than other methods regarding query accuracy. We observe that using the functional representation to encode the sentences, the contextual representations containing less than 10% of the original parameters perform competitively with the original 768-dim representations in semantic search. On text encoders of All-mpnet and Distil-roberta, our method with much fewer parameters and complexity performs even better than the original 768-dimensional representation obtained by a large-scale pretrained model, demonstrating the advantage of $LasF$ in encoding the relational information underneath the data.

B. Empirical results on CsQA's complicated sentences

We investigate whether *LasF* makes consistent improvements in tasks requiring more complicated reasoning. We follow Zhang et al. (2022) to categorize the dev set into three proxies, *i.e.*, a) the number of prepositional phrases in the question stems, b) the existence of a negation term (*e.g.*, *no*, *not*), and c) the existence of a hedging term (*e.g.*, *possibly*, *probably*). We implement the data split via the spaCy toolkit (Vasiliev, 2020). Each token with a dependency relation *prep* or *neg* is labeled as a propositional or negation term, respectively. The results are presented in Table 8, where we see $LasF$ significantly outperforms other competitors in all the settings. The improvement can be attributed to the trainable metric functions of *LasF* that capture the semantic relations amongst tokens. Therefore, $LasF$ succeeds in dealing with negation and hedge terms with more sensitive meanings than the others.

Table 8: **Performance comparison on CommonsenseQA IHdev set to validate *LasF*'s improvements on complex questions.** The experimental setting follows Zhang et al. (2022). Except for the base models, the accuracy improvements are listed on the right side of the accuracy.

Methods	**No. Prepositional Phrases**										**Negation Term**		**Hedge Term**	
	0		**1**		**2**		**3**		**4∼7**					
RoBERTa	66.7		72.3		76.3		74.3		69.5		63.8		70.7	
+QA-GNN	76.7	+10.0	76.2	+3.9	79.1	+2.8	74.9	+0.6	81.4	+10.9	66.2	+2.4	76.0	+5.3
+GreaseLM	75.7	+9.0	79.3	+7.0	80.4	+4.1	77.2	+2.9	84.7	+15.2	69.9	+6.1	78.4	+7.7
+LasF (Ours)	**77.9**	**+11.2**	**79.8**	**+7.5**	**81.5**	**+5.2**	**80.6**	**+6.6**	**85.0**	**+15.5**	**81.5**	**+7.5**	**78.9**	**+8.2**
ALBERT	73.9		77.8		78.7		75.0		78.9		81.4		70.8	
+KEAR	80.6	+6.7	82.8	+5.0	80.1	+1.4	80.6	+5.6	84.2	+5.3	86.6	+5.2	73.9	+3.1
+LasF (Ours)	**83.8**	**+9.9**	**83.6**	**+5.8**	**84.8**	**+6.1**	**86.1**	**+11.1**	**94.7**	**+15.8**	**88.7**	**+7.3**	**79.3**	**+8.5**

C. Mathematical Validation

Theoretically, the metric functions μ and $\{\mu_k,\ k \in [1, N_f]\}$ defined in Eq. 3 and Eq. 5 is dense in $C(I_d)$ that denotes the space of continuous functions on a finite d-dimensional cube (Theorem C.1). Thus, provided with a specific set of semantic fields $\mathcal{N}$, a well-formed prompt $\mathbf{S}$, and a discriminatory metric function μ, our *LasF* can approximate arbitrary function amongst semantic units. This theorem guarantees that *LasF* can facilitate the usage of knowledge as nonlinear and dynamical functions. It also validates that the nonlinear dynamic functions of *LasF* can be of arbitrarily large model capacity with an arbitrary size of trainable parameters, supporting the full utilization of arbitrary-scale knowledge bases. Moreover, we can replace a query token with a proper candidate that provides sufficient information from arbitrarily large external knowledge sources.

Theorem C.1. *Let $\mathcal{V}$ be a d-dimensional vector space, $\Psi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ constructed using a functional $\mathcal{F}$ and functions $\mathcal{N}$ defined above, be a continuous and discriminatory function on I_d, and $f_i : \mathbb{R}^d \to \mathbb{R}^{N \times d}$ be a vector field for $u_i \in \mathcal{V}$. Then there exists a signed regular Borel measure μ on I_d such that $f_i(\theta) + f_j(-\theta) = \mu(u_i - u_j)$ for arbitrary $\theta \in \mathbb{R}^d$ and S is dense in $C(I_d)$.*

Proof. We can easily construct a well-formed measure μ. We can prove in an apagogical manner that Ψ is dense in $C(I_d)$. Suppose that Ψ is not dense in $C(I_d)$, then the closure of Ψ is a subset of $C(I_d)$: $\overline{\Psi} \subset C(I_d)$. By Hahn-Banach Theorem, there exists a non-zero bounded linear functional $\mathscr{L}$ on $C(I_d)$ such that $\mathscr{L}(\Psi) = \mathscr{L}(\overline{\Psi}) = 0$. By Reisz Representation Theorem, we have

$$\mathscr{L}(\Psi) = \int_\theta \Psi\Big(f_i(\theta), f_j(\theta)\Big) d\mu(\theta) = 0 \tag{19}$$

for every pair of u_i and u_j. Since Ψ is discriminatory, we must have $\mu = 0$, which contradicts the assertions that $\mathscr{L} \neq 0$ and μ are regular. □

D. Discussion on the potential of Employing external Knowledge Bases

Based on the definition of *LasF*, we have validated its ability in modeling the semantic unit relations in sentences. Another advantage of *LasF* is that it is naturally compatible on the task of encoding external knowledge bases. To elaborate, one can establish a semantic field in the form of a knowledge graph by substituting the dynamic state $\mathbf{Q}$ of semantic units with embeddings of entities from a knowledge graph. Following this, we change the metric between $\mathbf{Q}$s as the vector of relations between entities, creating an independent semantic field detached from the input $\mathbf{X}$. In our preliminary experiments conducted on CsQA, we noted that adopting the mentioned approach to convey knowledge related to the query using corresponding triplets in ConceptNet led to an improvement of approximately 3.5% in model accuracy. Nevertheless, due to the evaluation protocol of CsQA that ConceptNet is not allowed to use, we have not included this part of results in the main part. Furthermore, our findings indicate that by ensuring precision and accuracy in the independent semantic field through manual annotation, the model accuracy could be enhanced to over 94%, showing the significant potential for an enhanced $LasF$ through integration with well-established external knowledge bases.

E. A conceptual comparison with Transformer

Indeed, our approach shares similarities with the multi-head attention mechanism of Transformer, as both models involve modeling linguistic elements and utilizing their relations to control subsequent computation processes. We admit that among neural models compatible with matrix multiplication-based computing architectures, Transformer can be considered one of the closest approximations to brain operation mechanisms. Consequently, it is inevitable that a newly proposed and effective language model structure approximates Transformer in the underlying computational process. However, fundamental differences still exist between *LasF* and Transformer.

First, while Transformer only considers relations between tokens, resulting in an attention matrix size dependent on the input sequence length, *LasF* construct the projections along both the length dimension and the embedding dimension of each token, yielding a relational matrix containing richer information.

Second, Transformer maps an input sequence to Q and K matrices, generating $QK^\top$ as the relational matrix. In contrast, *LasF* maps the inputs to a single matrix $\mathbf{Q}$ (distinct from Transformer's Query matrix Q), which describes neuronal states, and directly constructs the relational matrix using the relationships within $\mathbf{Q}$ itself. This single-matrix approach not only saves memory space compared to using separate Q and K matrices but also enhances computational efficiency.

Third, the construction of *LasF*'s relational matrix involves forming a distance matrix from the neuronal states' corresponding matrix $\mathbf{Q}$ through a metric function. There exists computationally efficient algorithms (Indyk & Silwal, 2022) for matrix-vector multiplication with distance matrices, reducing the time complexity from $O(n^2)$ to $O(ndp)$, where d represents the dimensions of neuronal states and p corresponds to an L_p-norm as the metric function. Although we haven't utilized this fast algorithm in our experiments due to incomplete CUDA programming specifically for it, the computational time measured already shows advantages over Transformer (refer to the Time Cost column in Table 4). Completion of CUDA programming based on Indyk & Silwal (2022) and Pei & Wang (2023) could further enhance *LasF*'s computational efficiency.

Fourth, *LasF* inherently possesses a Mixture of Experts (MoE) structure. Its field-level relational matrix computation (as seen in Eq. 6) evaluates the relationships between various semantic fields (analogous to neuronal communication among different cortical mini-columns in the brain's cerebral cortex) and uses these relationships to control the output of each semantic field. In contrast, Transformer directly combines the outputs of various attention blocks using MLP layers. Through experiments focusing on WikiText103, we observe a significant decline in the resulting language modeling ability if we were to directly combine the outputs of semantic fields using an MLP, akin to Transformer. For instance, the perplexity (PPL) would increase from 16.5 to 27.4.

Fifth, the core idea of *LasF* differs from that of Transformer. While Transformer's core operation is based on Query-Key retrieval, *LasF*'s core operation revolves around signal transmission between neurons. Although they share similarities in implementation, their underlying principles are fundamentally different. The various semantic fields in *LasF* can be associated with neuronal behaviors of different regions in the human brain when performing different cognitive tasks. Different semantic fields indeed correspond to specific linguistic patterns/structures. For instance, some semantic fields focus more on capturing Part-of-Speech information between tokens, while others focus on capturing dependency features or named entities. *LasF* enables a better integration of NLP with linguistics and neuroscience, thereby enhancing the interpretability of neural language models.