Large Language Model, Microlect, and Zeitgeist

Ellis D. Cooper

October 4, 2024

1 Abstract

This article gives mathematical pseudocodes for large language model training based on a dataset from a corpus, inference, and chat with possibly lengthy human prompts and generated replies. It introduces the concepts of "microlect" and "resonant-community." These notions are helpful in understanding of how the "frozen matrices" of a large language model are a conduit for resonance between human-beings and a zeitgeist represented in a corpus. The primary objectives are to encapsulate the functionality of large language models in a succinct technical representation, and to explain how this functionality connects human beings to their ambient cultural zeitgeist as represented in a textual corpus. The research draws on relevant literature in psychology, linguistics, mathematics, physics, and software engineering. The significance of the research is to embed the explanation in a perspective on the entirety of human expressiveness. The study can provide valuable insights based on resonances between human beings among themselves and with computing machines.

2 Keywords

large language model, microlect, resonance, zeitgeist

3 Introduction

Differential and integral calculus were to the phenomenological thermodynamics underlying the enormous amplification of human muscle power—"brawn"—as differential calculus and linear algebra are to the large language model software engineering underlying the unpredictably great potential amplification of human mental power—"brain." This article is a contribution towards understanding how this can be, in terms of a kind of "resonance" between human beings and a software representation of a recorded corpus of human expressions—a zeitgeist—limited only by hardware speed and storage capacity. The explanation is in terms of a novel concept of specialized language called "microlect,"

which may be non-verbal, verbal, informal, or formal, or, as in mathematics, also deductive.

4 Notation

$\mathcal{W}$ denotes the non-void set of words and punctuation marks of a natural language. $\mathcal{W}^+$ is the set of all possible non-void lists of words and punctuation marks. The (grammatically correct) sentences of the natural language are a subset $\mathcal{S} \subset \mathcal{W}^+$. All documents are non-void lists of sentences, $\mathcal{D} \subset \mathcal{S}^+$, and all corpora are non-void lists of documents, $\mathcal{C} \subset \mathcal{D}^+$. The input to an LLM is a corpus $C \in \mathcal{C}$.

$\mathbb{R}$ denotes the set of numbers a computer can use, and $\mathbb{R}^+$ is the set of non-void sequences of numbers. Every word and punctuation symbol of the corpus is mapped to a distinctive non-void list of symbols called *tokens*. More formally, If $\mathcal{V}$ denotes a stipulated list of tokens, then *tokenization* is a map

$$\mathbf{tkn} : \mathcal{W} \to \mathcal{V}^+$$

from words and punctuation marks to token-lists. To convert the corpus into numbers, there is a stipulated injection of tokens into the numbers, called *indexication*:

$$\mathbf{idx} : \mathcal{V} \hookrightarrow \mathcal{R}$$

which extends to an injection $\mathbf{idx}^+ : \mathcal{V}^+ \hookrightarrow \mathcal{R}^+$. Hence, all told, the composition $\mathbf{idx}^+ \circ \mathbf{tkn}$ maps words and punctuation marks to distinct lists of numbers.

A corpus is a list of documents, which are lists of sentences, which are lists of words and punctuation marks. After repeated concatenation of lists and lists of lists, a corpus is mapped to a list of words and punctuation marks, and after application of $\mathbf{idx}^+ \circ \mathbf{tkn}$, the corpus is represented in a computer by a list $\mathcal{T}$ of lists of (numbers).

Just to be clear, the tokenization process from corpus $\mathcal{C}$ to token list $\mathcal{T} = \mathcal{T}(\mathcal{C})$ includes information about the order in which documents are listed in the corpus, and information about the order in which sentences are listed in the documents, and information about the order in which words and punctuation marks are listed in the sentences. A question is whether the process of creating a trained LLM from $\mathcal{T}$ attends to the order of the documents. The answer is "no," all that matters is the order of word and punctuation tokens.

Starting from $\mathcal{T} = \mathcal{T}((C)) \subset \mathbb{R}^+$ proceed to the next step. Define a segmentation of $\mathcal{T}$ as follows.

Assume there are N batches of M chunks of L indices in each chunk. The data structure T is a list of lists of lists:

$$\begin{aligned} T &:= [b_1, \ldots, b_i, \ldots, b_N] \text{ batch decomposition} \\ b_i &:= [c_{i1}, \ldots, c_{ij}, \ldots, c_{iM}] \text{ chunk decomposition} \\ c_{ij} &:= [t_{ij1}, \ldots, t_{ijk} \ldots, t_{ijL}] \text{ indices} \end{aligned}$$

If $*$ denotes the operation of concatenating a list of lists, so that if $A = [A_1, \ldots, A_R]$ is a list of R lists, then $A^* := A_1 * \cdots * A_R$ is their concatenation. The operation $*$ is associative, i.e., $(A * B) * C = A * (b * C)$, and the empty list $[]$ is the unit for the operation, i.e., $[] * A = A = A * []$. Finally, $\mathcal{T} = T^{**}$, and $|\mathcal{T}| = L \times M \times N$. Concatenation of lists is a simple but central algebraic operation in computer science in general [Barr and Wells, 1995], and concatenation of matrices in large language model engineering in particular [Vaswani et al., 2017].

To be clear, the multiplication operation of a number times a number or a number times a vector or matrix is denoted by $\cdot$, the cartesian product of sets or the matrix product is denoted by $\times$, the dot product of vectors is $\bullet$, and the composition of functions symbol is $\circ$.

The underlying set of the index list $\mathcal{T}$ shall be denoted by $U\mathcal{T}$. An *embedding* is a map

$$\mathbf{emb} : U\mathcal{T} \to \mathbb{R}^d$$

in which $0 < d \in \mathbb{R}$ is the specific *embedding dimension* for the LLM model. In other words, to each $t \in U\mathcal{T}$ of the $L \times M \times N$ tokens its embedding $\mathbf{emb}(t)$ is a list of size d of numbers. (The numbers L, M, N, d specific to the LLM model are called *hyperparameters*.)

Define $LMN = L \times M \times N$. The list $\mathcal{T}$ of tokens has an underlying set $U\mathcal{T} = \{t_1, ..., t_i, ..., t_{LMN}\}$. The list of positions $[1, ..., LMN]$ has an underlying set $U[1, ..., LMN] = \{1, ..., LMN\}$, there is a bijection $\mathbf{t} : U[1, ..., LMN] \to U\mathcal{T}$ defined trivially by $\mathbf{t}(i) = t_i$, there is a function $\mathbf{pos} : U[1, ..., LMN] \to \mathbb{R}^d$ defined in terms of $\mathbf{sin}$ and $\mathbf{cos}$, and the *positional-embedding function* is defined by $\mathbf{posemb} := \mathbf{pos} + (\mathbf{emb} \circ \mathbf{t}) : U[1, ..., LMN] \to \mathbb{R}^d$, namely,

$$\mathbf{posemb}(i)(k) := \begin{cases} \mathbf{sin}(i/10000^{k/d}) & k \text{ even} \\ \mathbf{cos}(i/10000^{(k-1)/d}) & k \text{ odd} \end{cases}$$

for $1 \leq i \leq LMN$ and $1 \leq k \leq d$.

The following mathematical pseudocodes for training, inference, and chat are based on [Cristina and Saeed, 2022] and on chat with [OpenAI, 2023] encapsulate the functionality of large language models.

5 Training

Notation for the "frozen matrices" calculated by training are summarized in Table 1.

Algorithm 1 Training Pseudocode

Require: Dataset of pairs (Is, To), initialized matrices from table, learning rate, batch size, chunk size

Ensure: Trained matrices (updated after backpropagation)

for each batch B_i in the dataset $(1 \leq i \leq M)$ **do**
  for each chunk C_j in B_i $(1 \leq j \leq N)$ **do**
    for each token t_k in C_j $(1 \leq k \leq L)$ **do**
      $X_e \leftarrow \mathbf{emb}(t_k) + \text{pos}(k)$
    end for
    $Q_e \leftarrow W_Q^{(e)} \times X_e$
    $K_e \leftarrow W_K^{(e)} \times X_e$
    $V_e \leftarrow W_V^{(e)} \times X_e$
    for each head **do**

$$A_e \leftarrow \frac{Q_e \times K_e^\top}{\sqrt{d_k}}$$

      $Z_e \leftarrow \mathbf{softmax}(A_e \times V_e)$
    end for
    $Z_e \leftarrow Z_{e_1} * Z_{e_2} * \cdots * Z_{e_h}$
    $Z_e \leftarrow W_O^{(e)} \times Z_e$
    $FFN\_out \leftarrow W_2^{(e)} \times \mathbf{ReLU}(W_1^{(e)} \times Z_e + b_1^{(e)}) + b_2^{(e)}$
    $FFN\_out \leftarrow \gamma^{(e)} \cdot \mathbf{LayerNorm}(FFN\_out) + \beta^{(e)}$
    loss $\leftarrow \mathbf{CrossEntropyLoss}(FFN\_out, To)$
    gradients $\leftarrow \nabla_{X_e}$**Loss**
    Matrices $\leftarrow \{W_Q^{(e)}, W_K^{(e)}, W_V^{(e)}, W_O^{(e)}, W_1^{(e)}, W_2^{(e)}, b_1^{(e)}, b_2^{(e)}, \gamma^{(e)}, \beta^{(e)}\}$
    for each matrix M in **Matrices** **do**
      $M \leftarrow M - \text{learning\_rate} \cdot \mathbf{gradients}[M]$
    end for
  end for
end for
return Matrices (a.k.a., "frozen matrices")

6 Inference

Algorithm 2 Inference Pseudocode

Require: Sequence of tokens Is, trained matrices, batch size, chunk size
Ensure: Model prediction

for each batch B_i $(1 \leq i \leq M)$ **do**
  for each chunk C_j in B_i $(1 \leq j \leq N)$ **do**
    for each token t_k in C_j $(1 \leq k \leq L)$ **do**
      $X_e \leftarrow \mathbf{emb}(t_k) + \text{pos}(k)$
    end for
    $Q_e \leftarrow W_Q^{(e)} \times X_e$
    $K_e \leftarrow W_K^{(e)} \times X_e$
    $V_e \leftarrow W_V^{(e)} \times X_e$
    for each head **do**

$$A_e \leftarrow \frac{Q_e \times K_e^\top}{\sqrt{d_k}}$$

      $Z_e \leftarrow \mathbf{softmax}(A_e \times V_e)$
    end for
    $Z_e \leftarrow Z_{e_1} * Z_{e_2} * \cdots * Z_{e_h}$
    $Z_e \leftarrow W_O^{(e)} \times Z_e$
    $FFN\_out \leftarrow W_2^{(e)} \times \mathbf{ReLU}(W_1^{(e)} \times Z_e + b_1^{(e)}) + b_2^{(e)}$
    $FFN\_out \leftarrow \gamma^{(e)} \cdot \mathbf{LayerNorm}(FFN\_out) + \beta^{(e)}$
    Prediction $\leftarrow \mathbf{softmax}(FFN\_out)$
  end for
end for
return Prediction

7 Chat

Algorithm 3 Generating a Lengthy Response from a Lengthy Prompt

Require: Tokenized prompt $P = [p_1, p_2, ..., p_k]$, max response length L, max token length N, trained model
Ensure: Generated response $R = [r_1, r_2, ..., r_L]$
Initialize $R = []$ ▷ Empty response
if $|P| > N$ **then**
    Truncate P to keep only the last N tokens
end if
for each step t from 1 to L **do**
    # Step 1: Pass the input sequence through the inference algorithm
    $t_{k+t} \leftarrow \text{Inference}(P, \text{trained model})$ ▷ Invoke inference to get the next token
    # Step 2: Append the generated token to the response and prompt
    $R \leftarrow R * [t_{k+t}]$ ▷ Append to response
    $P \leftarrow P * [t_{k+t}]$ ▷ Append to input sequence
    if stopping condition met (e.g., EOS token) **then**
        BREAK
    end if
end for
return Generated response R

8 Intuition

The attention mechanism in machine learning [Vaswani et al., 2017] allows a model to dynamically focus on different parts of an input sequence when processing each token. This is akin to how we, as humans, pay attention to relevant words in a sentence when trying to understand a particular word. (It is not analogous to the role of attention in the structuring of the stream of consciousness [Watzl, 2010].) In the context of attention to tokens, each token is associated with three roles: a *Query*, a *Key*, and a *Value*. The *Query* represents the current token being processed, the *Key* represents all the tokens in the input sequence, and the *Value* holds the information from those tokens.

To determine how much attention should be paid to each token in the sequence, the model computes a similarity score between the *Query* and each *Key* using a dot product. This score reflects how closely related the tokens are, and it is then used to assign weights to the corresponding *Values*. The final representation of the current token is a weighted combination of the *Values*, where more relevant tokens (those with higher attention scores) contribute more significantly.

This attention mechanism is powerful because it allows the model to capture long-distance dependencies in a sequence. For example, when processing a sentence, the model can relate words that are far apart but contextually linked. In the sentence "The cat that chased the mouse is sleeping," the model will focus more on the word "cat" when processing the word "sleeping," since they are more closely related in meaning. By adjusting the focus based on context, the attention mechanism enables models to handle complex relationships and ambiguities in language effectively.

9 Dataset

In autoregressive language models, the dataset used for training consists of pairs (Is, To), where Is represents an input sequence of tokens and To represents the token that immediately follows Is in the corpus. The goal of the model is to predict the next token To given the previous sequence Is. To construct this dataset, the entire corpus is first tokenized, transforming the raw text into a sequence of tokens, denoted as $\mathcal{T} = [t_1, t_2, \ldots, t_L]$, where each t_i is a token, and L is the total number of tokens in the corpus.

The training dataset is generated by sliding a window of size $n + 1$ across the tokenized corpus $\mathcal{T}$. For each position i, a pair (Is_i, To_i) is created, where $Is_i = [t_i, t_{i+1}, \ldots, t_{i+n-1}]$ is the input sequence of n tokens, and $To_i = t_{i+n}$ is the target token, which is the next token following Is_i. This approach ensures that the model is trained to predict the next token in a sequence, given the preceding context.

In an autoregressive model, the prediction of each token is conditioned on all previously generated tokens, which makes it fundamentally non-Markovian. Whereas a Markov model relies only on the current state to predict the next, an autoregressive model generates a sequence by conditioning each token on the entire sequence of previously generated tokens. Formally, an autoregressive model estimates the joint probability $P(t_1, t_2, \ldots, t_L)$ of a sequence of tokens as the product of conditional probabilities:

$$P(t_1, t_2, \ldots, t_L) = P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_1, t_2) \ldots P(t_L \mid t_1, t_2, \ldots, t_{L-1}).$$

Unlike Markov models, which use only the present token to predict the next, autoregressive models leverage the entire preceding sequence, meaning the model can capture long-range dependencies and more complex contextual relationships. This ability to condition on the full sequence allows autoregressive models to generate coherent and contextually aware text, but it also increases the computational complexity compared to Markov models.

To train such a model, the dataset is constructed by selecting pairs (Is_i, To_i) from the tokenized corpus, where Is_i is the sequence of tokens leading up to

t_{i+n}, and To_i is the token that follows the sequence. During training, the model learns to predict To_i based on Is_i, minimizing a loss function, typically cross-entropy, that measures the difference between the predicted token distribution and the actual next token in the sequence.

The autoregressive nature of the model allows it to generate coherent text by iteratively predicting the next token in a sequence, appending the predicted token to the input, and using this updated input to generate the subsequent token. This process continues until a stopping condition, such as reaching a predefined length or generating an end-of-sequence (EOS) token, is met.

10 Microlectics

Intuitively, a "microlect" is a specialized unit of human expressiveness, with core expressions—like "molecules"—built from "atoms" of expression. Human behavior includes both non-verbal and verbal expressions, and expressions may be ephemeral or enduring.

$$\frac{keys}{cores} = \frac{\text{“About what”}}{\text{“What about”}}$$

There is an essentially unique verbal microlect about *all* microlects of all kinds. The key-words of *microlectics* include microlect, expression, expressed, resonant, natural-language (a.k.a. "macrolect"), *parole*, *langue*, vocabulary, grammar, non-verbal, verbal, metalect, map, key, core, spacetime (-region), physical-field, and physical-trace (a.k.a. "pulse").

Ideally, the set of cores of a microlect is the least set of its expressions (minimality) that suffice to express what it is about (adequacy). Cores of microlectics include the following sentences.

Every human expression is a structured part of spacetime. "Structured part" means a region of spacetime together with one or more (phenomenological [de Heer, 1986] or quantum [Peskin and Schroeder, 1995]) physical-fields defined on the region. The physical-trace of an expression is called a "pulse." [1] A microlect [Cooper, 2015], [Cooper, 2024] is a set of expressions, and for every microlect μ there is an expression not in μ. Nor is every expression $X \in \mu$ ever expressed. But if X is expressed, then it is a more or less enduring structured part of spacetime. A speech act is a brief pulse of verbal expression in a natural-language. Enduring physical-traces of verbal-expression are called "writing," and written documents are a major representation of the human zeitgeist (over the period of time from the earliest to the latest document in a corpus).

[1] "Pulse Diagrams and Category Theory," Ellis D. Cooper, April 10, 2024, The New York City Category Theory Seminar, `https://www.youtube.com/watch?v=GsonGRPCczA`

A verbal-microlect is a subset of the expressions in a natural language, also known as a "macrolect" in this article. The cumulative physical-traces of expressions—ephemeral or enduring—is called *parole*. A *grammar* of a natural language is called *langue* [Pei, 1966], also known as a "metalect" in this article. In general, a microlect whose expressions delimit a set of "admissible" expressions of a macrolect or a microlect is called a metalect. For example, mathematics is expressed by deductive formal microlects, and metamathematics is expressed by deductive formal metalects about mathematics.[2]

A sentence in the core of a verbal-microlect must include at least one occurrence of a key of the microlect. Expressions of a non-verbal microlect may be brief pulses, as in sport or musical performance or dance. Paintings and sculptures are enduring pulses of non-verbal microlects, and of course these too are aspects of the human zeitgeist.

Non-verbal or verbal expressions by one human being may or may not resonate with one or more other human beings.[3] For short, people understand some other people and not some others. People who understand one another form a pulse of resonant-community. Every human being is a member of multiple possibly transient, or possibly enduring, resonant-communities. A mental model is an abstraction of the understanding shared by members of a resonant-community. The keys and cores of a microlect express a shared mental model. *All we have is each other and how we express ourselves non-verbally or verbally.* Enduring resonant-communities are seeded by philosophers and religionists and scientists and writers and artists, leading to formation of societies and their zeitgeists with associated mental models expressed in so many ways. Large language models offer access to the mental models that underlie societies.

11 Microlectics and Large-Language-Models

The corpus is a trace of macrolect expressions. Portions of the expressions are expressions of verbal-microlects. Some documents in the corpus are expressed in specific verbal microlects. (The keys of some of these microlects may overlap, which can sometimes be confusing.) Within one core-sentence of one of these document-microlects, there must occur at least one key-word, but also possibly many keys may occur in the same core. This syntactic linkage will be reflected in the distance between tokens. There are sets of keys bunched together in some cores. Since the corpus is a trace of the zeitgeist of a community over some period of time, The zeitgeist amalgamates the concepts represented by the keys, and the frozen matrices of a large-language-model trained on a dataset obtained by randomly sampling the trace of the zeitgeist can resonate

[2] "Deductive" means explicit inference rules for derivation of expressions from expressions that are either stipulated—as "axioms"— or derived—as "theorems".

[3] One measure of resonance is citation count of articles in journals.

with expressions either generated by itself, or by expressions in prompts from human beings. Whenever there is significant resonance between expressions and microlects, the participants using those microlects form a resonant-community. Therefore, a chat composed of human prompts and large-language-model replies is evidence of a resonant-community consisting of a human-being and a zeitgeist.

The pseudocodes above for Training (Section 5), Inference (Section 6), and Chat (Section 7) are expressions in a deductive formal microlect. When mapped to Python code the result is an application program, of which there may be more than one copy. Copies may be mapped one or more times at different times to physical-structures: computers. When power is applied, the computers instantiate pulses that may be mapped to the underlying Python code and then mapped ("projected") back to the underlying pseudocode. A trained human-being may resonate sufficiently with the pseudocode, resulting in the uniquely human emotion of understanding.

12 Summary and Conclusion

This article sets up mathematical notation for expressing (1) pseudocode for training a large language model (LLM) such as **ChatGPT 4o** [OpenAI, 2023] on a dataset derived from a natural-language corpus, (2) pseudocode for inference grounded in the LLM, and (3) pseudocode for exchange of lengthy prompts and replies between a human-being and an empowered computer implementation of the LLM. It describes the basic intuition of "attention" at the heart of current-generation LLMs, and explains how a dataset for training an LLM is obtained from a corpus. In conclusion, the article introduces the concept of "microlect" and uses it to explain how a human-being and a machine can form a resonant-community, even though only a human-being can experience the emotion of understanding [Cooper, 2024].

- Author Contributions: Ellis D. Cooper is the sole author. The author read and approved the final manuscript.
- Funding: This work has no external funding.
- Data Availability Statement: No data was used.
- Conflicts of Interest: The author declares no conflicts of interest.

References

[Barr and Wells, 1995] Barr, M. and Wells, C. (1995). Category theory for computing science. In *Prentice Hall International Series in Computer Science*.

[Cooper, 2015] Cooper, E. D. (2015). *Microlects of Mental Models*. Cognocity Press.

[Cooper, 2024] Cooper, E. D. (2024). *The Human Ways of Being, Broken Covenants, Parochial Ultimata*. Cognocity Press, Rockport.

[Cristina and Saeed, 2022] Cristina, S. and Saeed, M. (2022). *Building Transformer Models with Attention: Implementing a Neural Machine Translator from Scratch in Keras*. Machine Learning Mastery.

[de Heer, 1986] de Heer, J. (1986). *Phenomenological Thermodynamics, with Applications to Chemistry*. Prentice-Hall, Inc.

[OpenAI, 2023] OpenAI (2023). Chatgpt (gpt-4). `https://openai.com/chatgpt`. Accessed: 2024-09-21.

[Pei, 1966] Pei, M. (1966). *Glossary of Linguistic Terminology*. Anchor Books, Doubleday & Company, Inc.

[Peskin and Schroeder, 1995] Peskin, M. E. and Schroeder, D. V. (1995). An introduction to quantum field theory.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5998–6008. Curran Associates, Inc.

[Watzl, 2010] Watzl, S. (2010). Attention as structuring of the stream of consciousness. In Mole, C., Smithies, D., and Wu, W., editors, *Attention: Philosophical and Psychological Essays*. Oxford University Press.

Matrix (at e)	**Name**	**Dimensions**	**Initial Value (e=0)**
Q_e	Query Matrix	$\mathbb{R}^{d_k \times d}$	Random
K_e	Key Matrix	$\mathbb{R}^{d_k \times d}$	Random
V_e	Value Matrix	$\mathbb{R}^{d_v \times d}$	Random
$W_Q^{(e)}$	Query Weight Matrix	$\mathbb{R}^{d \times d_k}$	Random
$W_K^{(e)}$	Key Weight Matrix	$\mathbb{R}^{d \times d_k}$	Random
$W_V^{(e)}$	Value Weight Matrix	$\mathbb{R}^{d \times d_v}$	Random
$W_O^{(e)}$	Output Weight Matrix	$\mathbb{R}^{d \times d}$	Random
$W_1^{(e)}$	Feedforward Network Weight 1	$\mathbb{R}^{d_{\text{ff}} \times d}$	Random
$W_2^{(e)}$	Feedforward Network Weight 2	$\mathbb{R}^{d \times d_{\text{ff}}}$	Random
$b_1^{(e)}$	Feedforward Network Bias 1	$\mathbb{R}^{d_{\text{ff}}}$	0
$b_2^{(e)}$	Feedforward Network Bias 2	$\mathbb{R}^{d}$	0
$\gamma^{(e)}$	Layer Normalization Gain	$\mathbb{R}^{d}$	1
$\beta^{(e)}$	Layer Normalization Bias	$\mathbb{R}^{d}$	0

Table 1: Matrices required for training a transformer model at iteration e, along with their dimensions and initial values. These are the "frozen matrices" at completion of training.