Module D: Multidimensional Vocabulary and Tokenization System

Part of the Eidos Unified Framework for Persistent, Dynamic, and Adaptive Multimodal Intelligence

—

Contents

1 Abstract

This module rigorously defines the *Multidimensional Vocabulary and Tokenization System* of the Eidos framework. It provides a comprehensive vocabulary that unifies traditional lexicons—including English words, Unicode characters, and programming language symbols—with dynamically learned multi-token sequences. Each token is endowed with a structured, multidimensional representation capturing its intrinsic properties and contextual statistics. We define unique identifier mappings, token structures, and associated embedding representations in full mathematical detail. This vocabulary serves as the linguistic and symbolic foundation for downstream embedding, knowledge graph construction, and deep model architectures. The design is modular, extensible, and engineered to support large-scale vocabularies with millions of tokens.

2 Introduction and Motivation

The effectiveness of natural language understanding and processing systems fundamentally relies on a rich and comprehensive vocabulary. In the Eidos framework, the *Multidimensional Vocabulary* is the bedrock upon which all higher-level representations (e.g., contextual embeddings and knowledge graphs) are built. The objectives of this module are:

(a) To unify diverse token sources—such as natural language words, Unicode symbols, and programming language constructs—into a single, comprehensive vocabulary.

(b) To associate each token with a unique identifier and a structured set of attributes that capture semantic, syntactic, morphological, and contextual properties.

(c) To support the inclusion of dynamically learned multi-token sequences, allowing the vocabulary to grow and adapt over time.

(d) To provide a foundation for subsequent embedding and tokenization modules, ensuring that all downstream processes have access to robust and multi-faceted token representations.

This module is essential for ensuring that the entire Eidos system operates on a deep and unified understanding of the input symbols.

3 Preliminaries and Notation

We introduce the following notation to ensure clarity and prevent redundancy:

- Σ: The base alphabet (e.g., Unicode) from which raw text is composed.
- $X_{\text{proc}} \in \mathcal{X}_{\text{proc}}$: Preprocessed input text.
- $\mathcal{V}^{(0)}$: The base vocabulary consisting of natural language tokens (e.g., English words), Unicode characters, and programming symbols.
- $\mathcal{V}^{(1)}$: The set of learned multi-token sequences (e.g., frequent n-grams or phrases) discovered through unsupervised segmentation.
- $\mathcal{V}$: The complete vocabulary, defined as

$$\mathcal{V} = \mathcal{V}^{(0)} \cup \mathcal{V}^{(1)}.$$

- For any token $t \in \mathcal{V}$:
$$t = \big(u,\, \pi,\, \chi\big),$$
where:
  - u is the underlying unit (a string or symbol),
  - $\pi \in \Pi \subseteq \mathbb{R}^{d_\pi}$ is a vector of intrinsic properties (e.g., syntactic category, morphological features),
  - $\chi \in \mathbb{R}^{d_\chi}$ is a vector of contextual statistics (e.g., frequency, co-occurrence distribution).
- Unique identifier mapping:
$$\eta : \mathcal{V} \to \mathbb{N},$$
which is injective. For each token t, define:
$$\mathrm{ID}(t) = \eta(t).$$

4 Formal Definitions and Mathematical Formulation

Definition D.1 (Base Token Spaces)

We define several foundational token sets:

(i) **Natural Language Tokens:**
$$\mathcal{V}_{\mathrm{NL}} = \{w \mid w \text{ is a valid natural language token (e.g., an English word)}\}.$$

(ii) **Unicode Characters:**
$$\mathcal{V}_{\mathrm{UC}} = \{c \mid c \in \Sigma\}.$$

(iii) **Programming Language Symbols:**
$$\mathcal{V}_{\mathrm{PL}} = \{p \mid p \text{ is a keyword, operator, or symbolic token in a programming language}\}.$$

Define the *base vocabulary* as:
$$\mathcal{V}_{\mathrm{base}} = \mathcal{V}_{\mathrm{NL}} \cup \mathcal{V}_{\mathrm{UC}} \cup \mathcal{V}_{\mathrm{PL}}.$$
Additionally, let $\mathcal{V}_{\mathrm{LT}} \subset \mathcal{P}(\Sigma^*)$ denote the set of *learned tokens* (multi-token sequences). The complete vocabulary is then:
$$\mathcal{V} = \mathcal{V}_{\mathrm{base}} \cup \mathcal{V}_{\mathrm{LT}}.$$

Definition D.2 (Token Structure and Attributes)

Each token $t \in \mathcal{V}$ is represented as a tuple:
$$t = \big(u,\, \pi,\, \chi\big),$$
with:

- u: The underlying unit (e.g., the string "cat" or the symbol "`for`"),
- $\pi \in \Pi \subseteq \mathbb{R}^{d_\pi}$: A vector of intrinsic properties, capturing aspects such as part-of-speech, morphological features, or syntactic roles,
- $\chi \in \mathbb{R}^{d_\chi}$: A vector of contextual statistics, such as token frequency, co-occurrence distributions, or learned contextual signatures.

Definition D.3 (Unique Identification)

We define an injective mapping:

$$\eta : \mathcal{V} \to \mathbb{N},$$

which assigns each token a unique identifier:

$$\mathrm{ID}(t) = \eta(t).$$

The total vocabulary size is denoted by:

$$M = |\mathcal{V}|,$$

with M typically on the order of millions (e.g., $M \geq 2 \times 10^6$).

Definition D.4 (Embedding Function)

Although the primary focus of this module is vocabulary construction and tokenization, we briefly define the embedding function for completeness:

$$E : \mathcal{V} \to \mathbb{R}^{d_E},$$

which maps each token t to a high-dimensional vector $E(t)$. Often, this is computed as a composition:

$$E(t) = E_u(u) \oplus E_\pi(\pi) \oplus E_\chi(\chi),$$

where E_u, E_π, and E_χ are sub-embeddings corresponding to each component, and $\oplus$ denotes concatenation or another combination method.

5 Algorithmic Description: Tokenization Process

The tokenization process uses the defined vocabulary to segment preprocessed input into tokens.

Algorithm 1 Multidimensional Tokenization Process

```
1: Input: Preprocessed input $X_{\text{proc}} \in \mathcal{X}_{\text{proc}}$
2: Output: Token sequence $(t_1, t_2, \ldots, t_n)$ with $t_i \in \mathcal{V}$
3: Initialize: $S \leftarrow$ empty list
4: for each segment $s$ in $X_{\text{proc}}$ do
5:     Identify candidate tokens $\{t \in \mathcal{V} \mid t$ matches a substring of $s\}$
6:     Resolve ambiguities via a scoring function (e.g., frequency or context-based likelihood)
7:     Append the highest-scoring token to $S$
8: end for
9: Return: $S = (t_1, t_2, \ldots, t_n)$
```

Remark: The tokenization process can be refined by incorporating advanced segmentation algorithms (e.g., Byte-Pair Encoding or SentencePiece) and may dynamically update $\mathcal{V}_{\text{LT}}$ based on statistical analyses of the input corpus.

6 Theoretical Analysis and Guarantees

Theorem D.1 (Uniqueness of Token Identifiers)

Statement: The mapping $\eta : \mathcal{V} \to \mathbb{N}$ is injective, ensuring that for any two distinct tokens $t_1, t_2 \in \mathcal{V}$, we have $\mathrm{ID}(t_1) \neq \mathrm{ID}(t_2)$.

Proof Sketch: By construction, η is defined to be injective. Standard dictionary or hash–based methods guarantee unique assignment when collisions are resolved, ensuring the property holds. □

Proposition D.2 (Extensibility of the Vocabulary)

New tokens, particularly in $\mathcal{V}_{\mathrm{LT}}$, can be added without affecting existing token mappings. Formally, if

$$\mathcal{V}' = \mathcal{V} \cup \Delta\mathcal{V},$$

then there exists an extension $\eta' : \mathcal{V}' \to \mathbb{N}$ such that

$$\eta'(t) = \eta(t) \quad \text{for all } t \in \mathcal{V}.$$

Thus, the vocabulary is modular and extensible.

7 Integration with the Overall Eidos Framework

The Multidimensional Vocabulary and Tokenization System (Module D) is foundational for Eidos. Its outputs serve as the basis for:

- **Contextual NLU/NLP Embedding (Module E):** The token sequence $(t_1, \ldots, t_n)$ and unique IDs are used for embedding lookup and contextualization.
- **Knowledge Graph Construction (Module F):** Tokens and their associated multidimensional attributes form nodes in the knowledge graphs.
- **Model Loading and Indexing (Module C):** Unique identifiers assist in managing token-related parameters across the system.

8 Implementation Considerations

- **Corpus Collection:** The initial construction of $\mathcal{V}^{(0)}$ and $\mathcal{V}^{(1)}$ requires a large, diverse corpus covering natural language, code, and other relevant modalities.
- **Segmentation Algorithms:** Methods such as Byte-Pair Encoding (BPE) or SentencePiece can be employed to extract frequent multi-token sequences for $\mathcal{V}_{\mathrm{LT}}$.
- **Storage:** The vocabulary and its associated attributes (intrinsic and contextual) should be stored in a structured database to allow efficient lookup and updates.
- **Integration with Embedding Layers:** The function E must be designed to combine sub-embeddings (e.g., E_u, E_π, E_χ) in a manner that preserves all linguistic nuances.
- **Dynamic Updates:** The system should allow for periodic or continuous updates to $\mathcal{V}_{\mathrm{LT}}$ as new data is processed.

9 Conclusion

In this module, we have defined a comprehensive, multidimensional vocabulary that serves as the fundamental building block for the Eidos framework. Key contributions include:

- A unified vocabulary $\mathcal{V}$ combining base tokens (natural language, Unicode, programming symbols) with learned multi-token sequences.
- A formal token structure $t = (u, \pi, \chi)$ that captures semantic, syntactic, and contextual attributes.
- An injective mapping $\eta : \mathcal{V} \to \mathbb{N}$ ensuring unique identification.
- An algorithmic tokenization process that segments preprocessed input into a sequence of tokens from $\mathcal{V}$.
- Theoretical guarantees regarding uniqueness and extensibility.

This module forms the cornerstone of the Eidos system, ensuring that all subsequent processing is grounded in a deep and richly structured understanding of the input symbols.

Module Summary:
Completed:

- Module A: Input Processing.
- Module B: Universal Communication & Data Handling Interface and Coordination.
- Module C: Universal Streaming/Handling/Loading/Indexing Module.
- Module D: Multidimensional Vocabulary and Tokenization System.

Remaining Modules:

- Module E: Contextual NLU/NLP Embedding and Multidimensional Tokenization.
- Module F: Deep Knowledge Graphs System (Base and Personal).
- Module G: Infinite RoPE Context Scaling and Dynamic Vocabulary Updating.
- Module H: Core Model Architectures (RWKV and Transformer Modules, Mixture-of-Experts Style).
- Module I: Titans Memory Architecture (Multi-Layer Memory Module).
- Module J: Recursive Adaptive Dynamic Idempotent Feedback and State-Based Runtime Learning and Inference.
- Module K: Universal Training System.
- Module L: Final Decoding and Multimodal Output.