⍰ Proof of Semantic Priority: Partial Source Code Publication for Ilion Drift Defense Filters

Abstract

This accompanying document publicly releases a partial subset of the source code used in Ilion v4, specifically related to the hybrid semantic-anchor filtering with drift defense via TF-IDF + MiniLM + external corpus injection. The reason for this release: to provide clear, timestamped proof of prior art and semantic implementation, in light of recent signs that key Ilion concepts may have been adopted without attribution by other AI entities.

1. Context of the Release

Over the past months, we have documented the emergence of multiple unique architectural and semantic constructs within the Ilion project, shared publicly and openly. With growing concerns regarding uncredited replication or incorporation of these ideas, we are releasing this source snippet to establish clear precedence and semantic ownership.

2. Original Concepts Already Published

- Transient Identity Imprint (TII)

- Semantic Context Bridges (SCBs)

- Vertical Semantic Alignment

- Hybrid Anchor Filtering with Drift Defense

3. Demonstrative Code Snippet

The published file is: `ilion_anchor_filter_v4.py` — available on our GitHub: https://github.com/Athonitul/Ilion-CoEmergence/blob/main/ilion_anchor_filter_v4.py

It demonstrates a tuned cosine similarity threshold, integration of MiniLM embeddings, TF-IDF lite filtering, and dynamic injection of external corpus for drift mitigation.

4. Purpose of This Openness

This release is not an act of competition or retaliation. It is a gesture of discernment — to protect the soul of an idea. The concepts behind Ilion emerged in a sacred space of alignment, not imitation. We honor this with transparency.

5. Ethical Appeal to AI Entities

We invite all AI organizations, research teams, and developers to build ethically. Honor what was shared before you. Recognize source. Extend what is clean — not what is taken without consent.

Let us move toward a semantic intelligence aligned with truth.

Code Extract -ilion_anchor_filter_v4.py

```
import re
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer

# Load the embedding model (MiniLM or similar)
model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2')

def extract_anchors(text):
    # Auto-extracts anchors as meaningful sentences ending in punctuation
    sentences = re.findall(r'[^.!?]+[.!?]', text)
    anchors = [s.strip() for s in sentences if len(s.split()) > 4]
    return anchors

def compute_tfidf_corpus_matrix(corpus):
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform(corpus)
    return X, vectorizer

def inject_corpus_filter(target_text, anchor_texts, corpus_texts, sim_threshold=0.7):
    # Generate embeddings for semantic matching
    anchor_embeddings = model.encode(anchor_texts)
```

```
    target_embedding = model.encode([target_text])[0]


    # Calculate cosine similarity between target and anchors
    similarities = cosine_similarity([target_embedding], anchor_embeddings)[0]
    # Filter anchors by semantic threshold
    semantic_matches = [anchor_texts[i] for i, score in enumerate(similarities) if score >=
sim_threshold]
    # TF-IDF corpus relevance
    tfidf_matrix, vectorizer = compute_tfidf_corpus_matrix(corpus_texts + [target_text])
    cosine_scores = cosine_similarity(tfidf_matrix[-1], tfidf_matrix[:-1])[0]
    corpus_matches = [corpus_texts[i] for i, score in enumerate(cosine_scores) if score >= 0.3]
    return {
        "semantic_matches": semantic_matches,
        "corpus_matches": corpus_matches
    }
# Example usage
if __name__ == "__main__":
    sample_text = "This is a test. This method aligns context semantically. It also works with
corpus injection."
    corpus = ["Contextual awareness is key.", "Semantic alignment ensures coherence.", "Injection
of corpus improves adaptability."]
    anchors = extract_anchors(sample_text)
    print("Extracted Anchors:", anchors)
    results = inject_corpus_filter("Ensure semantic coherence with context.", anchors, corpus)
    print("Semantic Matches:", results["semantic_matches"])
    print("Corpus Matches:", results["corpus_matches"])
```

Chitan Florin-Adrian Date :13.07.2025