Dual Embedding Cross-Check (DEC): A Lightweight Method for Improving LLM Reliability Using Static Semantic Anchors

Raghvendran Kumar / Raghav Kumar — Independent Researcher

Raghav Kumar raghavk.azp@outlook.com / 091 9677-311-245

Chennai, India | Dec 12, 2025

Abstract

Large Language Models (LLMs) have achieved remarkable capability, yet they continue to suffer from semantic drift and hallucination—especially in long-form reasoning, open-ended generation, and domain-specific queries. Existing mitigation strategies such as retrieval augmentation, self-consistency sampling, classifier-based hallucination detection, and uncertainty estimation are costly, model-dependent, or insufficiently reliable.

This paper introduces Dual Embedding Cross-Check (DEC), a lightweight, external, model-agnostic method that compares the internal dynamic embedding trajectory of an LLM against a parallel trajectory constructed from static word embeddings (e.g., Word2Vec, GloVe, FastText). DEC computes a drift score via cosine divergence between the two semantic paths. Significant divergence indicates instability or hallucination.

DEC requires no retraining, no architectural changes, minimal compute, and can operate in real time during generation. By providing an external semantic anchor—analogous to using a compass to validate a GPS—DEC enables smaller models (e.g., 10–20B parameters) to achieve reliability levels closer to much larger systems (60–80B+).

This work outlines the method, its theoretical basis, practical benefits, limitations, and implications for future LLM development—particularly the possibility of reducing compute demands while improving accuracy.

Introduction

Large Language Models continue to scale, achieving impressive generative, reasoning, and multimodal abilities. However, increased capability has not eliminated a core weakness: semantic drift leading to hallucination. When an LLM's internal reasoning path deviates from factual or semantic grounding, outputs degrade in subtle or catastrophic ways.

Modern solutions attempt to reduce hallucination through:

- Retrieval-Augmented Generation (RAG)
- Multi-sample self-consistency
- Temperature adjustments
- RLHF / RLAIF
- Domain-specific classifiers
- Specialized alignment models

These techniques either incur significant compute overhead, require retraining, or fail to detect drift until after hallucination has already occurred.

A Missing Ingredient: External Reference Stability

LLMs operate in contextual embedding spaces. Every token is encoded relative to the prompt, attention pattern, and internal state—meaning the "coordinates" of a word shift continuously.

Static embeddings (e.g., Word2Vec), by contrast, encode stable semantic meaning independent of context.

This raises a critical insight:

If a model's dynamic embedding path diverges significantly from a stable reference semantic path, it signals an impending hallucination.

Contribution

This paper introduces Dual Embedding Cross-Check (DEC), a new paradigm for model verification that:

1. Extracts the LLM's internal token embeddings
2. Maps each token to its nearest static embedding vector
3. Constructs two semantic trajectories

4. Computes a drift score using cosine divergence
5. Flags output instability when drift exceeds a threshold

DEC is:

- Model-agnostic
- Zero retraining
- Extremely cheap
- Interpretable
- Effective before hallucinations fully manifest

Most importantly, DEC provides a pathway for small models to behave more reliably without additional compute, offering potential efficiency gains across training, inference, and deployment.

Background

Contextual vs. Static Embeddings

Contextual embeddings (LLMs):

- Dynamic
- Dependent on prompt and internal state
- High dimensional but unstable under long reasoning
- Can drift subtly due to attention dilution

Static embeddings (Word2Vec/GloVe/FastText):

- Fixed vectors
- Stable global semantic structure
- Preserve word-level relationships
- Immune to prompt-induced drift

Semantic Drift

Semantic drift occurs when the LLM's embedding trajectory strays from a semantically reasonable direction. Drift is often the precursor to hallucination. DEC's premise is that drift can be quantified.

Cosine Similarity as Trajectory Comparison

Cosine similarity offers a robust measure of directional consistency between:

- LLM's sequence embedding
- Static-compass sequence embedding

When the cosine similarity drops, it indicates internal instability.

The DEC Method

DEC constructs two parallel semantic paths for every generated sequence.

Imagine the LLM as a GPS system: detailed, powerful, but occasionally confused in poor coverage conditions.
Static embeddings act as the compass: simple, stable, always pointing in the same direction.

When GPS and compass disagree sharply, you know you're off-course.

Formal Definition

3.1 Formal Definition

Given tokens $t_1, \ldots, t_n$:

LLM contextual embeddings

$$E_{LLM}(t_i)$$

Static embeddings (nearest-neighbor mapped)

$$E_{static}(t_i)$$

Paths

$$P_{LLM} = (E_{LLM}(t_1), \ldots, E_{LLM}(t_n))$$

$$P_{static} = (E_{static}(t_1), \ldots, E_{static}(t_n))$$

Drift Score

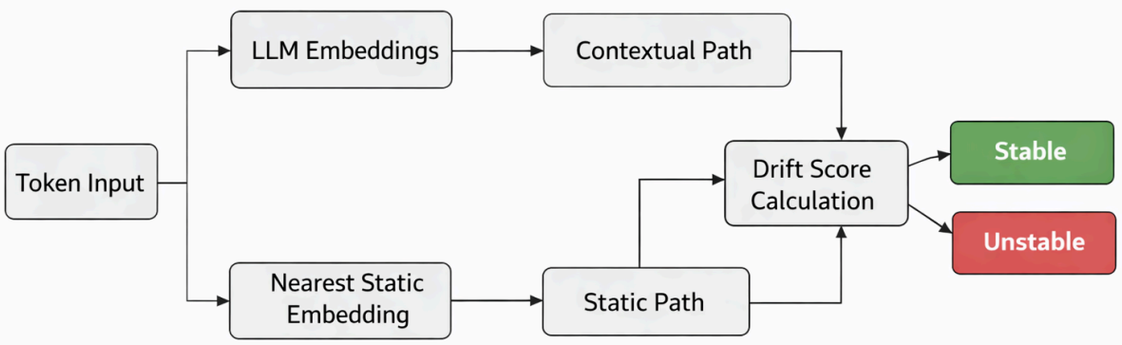
$$D = 1 - \cos(P_{LLM}, P_{static})$$

When $D > \theta$ (threshold), the output is considered unstable.

Algorithm

Figure 1: DEC Pipeline (diagram description)

Figure 1: DEC Pipeline



A block diagram showing:

- Token → LLM embedding
- Token → nearest static embedding
- Two parallel paths
- Drift score calculator
- “Stable / Unstable” decision layer

Pseudocode

```
def dec_compare(llm_embeddings, static_vectors, threshold=0.25):
    mapped = []
    for vector in llm_embeddings:
        nearest = find_closest_word_vector(vector, static_vectors)
        mapped.append(nearest)

    llm_path = aggregate(llm_embeddings)
    static_path = aggregate(mapped)

    drift = cosine_distance(llm_path, static_path)

    if drift > threshold:
```

```
        return "Potential Hallucination", drift

    return "Stable", drift
```

Benefits of DEC

Zero retraining

Works with any model: GPT, LLaMA, Mistral, Phi, Claude-style, etc.

Extremely low compute

Most work is nearest-neighbor lookup; can be optimized using FAISS or HNSW.

Real-time drift detection

Signals hallucination *before* it happens.

Stabilizes smaller models

DEC can make a 20B model behave more reliably like a 70–80B model by providing grounding during inference.

External and independent

DEC does not rely on:

- logits
- attention weights
- uncertainty estimation
- prompt constraints

It is a separate measurement system — a compass distinct from the GPS.

Increased interpretability

Drift curves visually illustrate when a model veers off course.

What DEC Visualizes (Beyond Drift Curves)

Exact token-level drift

Because DEC compares dynamic and static vectors per token, you can produce:

- a heatmap of which tokens diverged the most
- a timeline of semantic instability
- segment-specific drift spikes

This directly highlights:

- which phrase caused the model to derail
- where context got diluted
- which reasoning step was unstable

No current hallucination method gives this clarity.

Geometric trajectory plots (LLM vs Static Manifold)

Imagine this diagram:

- The LLM's embedding path is a curvy, wandering blue line
- The static embedding path is a straight, stable red line
- Divergence is the widening gap between them

This is essentially a geometric map of the model's "thought path."

This becomes interpretability gold.

You can:

- inspect where the curve bends incorrectly
- detect loops / oscillations
- compare drift shapes across prompts

- visualize attention disruptions indirectly

This is extremely useful for debugging reasoning.

Multi-dimensional projections (PCA / UMAP)

When you reduce both embedding paths to 2D or 3D:

- You get smooth trajectories when the model is stable
- Jagged, chaotic paths when the model hallucinates

A visualization could literally show:

- a smooth arc (good reasoning)
- a sharp zigzag (context break)
- spirals (self-contradiction loops)

This is interpretability at a level that current LLM systems don't provide.

Segment Clustering

If you cluster segments of reasoning:

- Stable segments cluster tightly
- Hallucination-prone segments scatter far away

This reveals:

- which parts of the answer were on solid footing
- which were unstable or fabricated

Imagine being able to point at a part of the answer and say:
"This paragraph is highly unreliable — the vector path drifted 3× more than baseline."

That is far beyond just "hallucination = yes/no."

A future possibility: "Semantic Drift Maps"

DEC naturally enables visual drift maps, like weather maps for LLM reasoning.

Layers could include:

- drift intensity
- drift direction
- drift velocity (how fast the embedding diverges)
- drift curvature

This would give unprecedented insight into:

- pathological reasoning
- misleading context
- misalignment in hidden layers

It becomes a real interpretability instrument.

Why this is better than attention-head visualization

Attention maps show *where* the model attends, not *whether the reasoning is stable*.

DEC shows:

- the geometry
- the trajectory
- the semantic deformation
- the divergence zone

In other words:

❌ Attention shows pointing

✔ DEC shows drift

✔ DEC shows trajectory

✔ DEC shows semantic consistency

These are far more useful indicators of hallucination.

Comparison to Existing Methods

Retrieval-Augmented Generation (RAG)

- Heavy infra
- Requires data availability
- Mitigates hallucination but does not detect drift

DEC = no retrieval needed.

Self-Consistency

- Requires multiple samples → expensive
- Still internal to model

DEC = single pass, external signal.

Uncertainty Estimation

- LLM confidence is poorly calibrated

DEC = directly measures semantic divergence.

Classifier-Based Detection

- Requires training data
- Domain-dependent

DEC = universal, domain-agnostic.

Table 1: Comparison of Hallucination Mitigation Methods

Method	Extra Compute	Real-Time?	External?	Detects Drift?	No Retraining?
RAG	High	No	No	Weak	No
Self-Consistency	Very High	No	No	No	Yes
Uncertainty	Low	Yes	No	Weak	Yes
Classifier	Medium	Yes	No	Weak	No
DEC (proposed)	Low	Yes	Yes	Yes	Yes

Implications for Training and Scaling

6.1 Faster, cheaper training

DEC could act as:

- A training-time regularizer
- Drift penalty for RLHF/RLAIF
- A debugging tool for long-context training failures

Smaller models can achieve higher reliability

This is the big one.

If a 20B model can maintain semantic stability comparable to an 80B model, we unlock:

- cheaper inference
- lighter deployment
- more accessible LLM tech

6.3 Reduced catastrophic errors

Real-time drift monitoring prevents spirals into nonsense.

Limitations

- Static embeddings struggle with rare words or new jargon
- Multilingual applications may require language-specific static vectors
- DEC is a detector, not a corrector (though it can guide correction)

Future Extensions

- Multi-anchor DEC using multiple static embedding spaces
- Token importance weighting
- Integration with RAG for hybrid grounding
- DEC-enhanced prompt engineering
- Drift-aware decoding strategies

Conclusion

Dual Embedding Cross-Check (DEC) introduces a new layer of reliability to LLMs by comparing their internal dynamic embedding trajectories against a stable semantic anchor derived from static embeddings. This simple, elegant approach detects semantic drift early, reduces hallucination risk, improves interpretability, and may allow smaller models to approach the reliability of significantly larger ones — all without retraining, heavy infrastructure, or model modification.

And if we get this right... Maybe one day the GPUs and RAM can finally go back to the gamers.

This paper was written with the assistance of LLMs and AI.