ISSN: 2993-5377

Review Article

Open Access Journal of Applied Science and Technology

Quantum Computation of Partition Function Similarity for Large Language Models

Timo Aukusti Laine*

Financial Physics Lab, Finland

***Corresponding Author**
Timo Aukusti Laine, Financial Physics Lab, Finland.

Submitted: 2026, Jan 09; **Accepted:** 2026, Feb 13; **Published:** 2026, Feb 18

Citation: Laine, T. A. (2026). Quantum Computation of Partition Function Similarity for Large Language Models. *OA J Applied Sci Technol, 4*(1), 01-11.

Abstract

Large Language Models (LLMs) demand significant computational resources, motivating the exploration of quantum computing to accelerate their analysis. This paper introduces a framework for analyzing LLM embedding spaces using partition functions, drawing an analogy to statistical mechanics. We propose a partition function-based similarity measure and present an experimental evaluation of its implementation on the ibm boston quantum computer, achieving a relative error of approximately 7% compared to classical computation. Results suggest partition functions effectively characterize the statistical properties of embedding vectors, offering a complementary perspective to cosine similarity. This work contributes to quantum natural language processing (QNLP), advancing LLM analysis and semantic representation, with potential implications for reducing the energy footprint and democratizing access to these powerful AI technologies.

1. Introduction

Large Language Models (LLMs) have achieved remarkable success in natural language processing; however, their widespread adoption is hindered by substantial computational demands. Training and deploying these models is expensive, limiting access and innovation. As model sizes continue to increase, alternative approaches are needed to achieve greater efficiency. While embedding spaces are central to LLMs, a comprehensive understanding of their properties and efficient manipulation remains a key challenge, particularly in capturing the nuances of semantic relationships.

Quantum computing offers the possibility for speedups in certain computations, promising to democratize LLM development by reducing computational costs. To realize this potential, we need formalisms that can effectively describe LLMs and be readily implemented in quantum circuits, enabling the development of quantum algorithms for natural language processing. This work introduces a framework for analyzing LLM embedding spaces using partition functions, drawing an analogy to statistical mechanics. This partition function approach provides a new perspective on embedding spaces and vectors, leveraging the existing results in statistical physics to characterize the statistical properties of semantic representations.

A key contribution of this paper is the development of a similarity measure based on the partition function. We demonstrate how this similarity measure can be implemented on a quantum computer, enabling end-to-end quantum similarity calculations. We present experimental results obtained using a real quantum computer, comparing the performance of our quantum approach to traditional methods. These experiments were performed on the ibm boston quantum computer.

Our findings provide insights into the role of quantum fidelity in practical applications. They also suggest that traditional cosine similarity, while widely used, may not fully capture the inherent uncertainties in LLMs and should be considered a statistical measure. Our results indicate that quantum hardware may offer a more accurate representation of semantic similarity in certain cases, possibly accounting for these uncertainties. This work contributes to the development of quantum algorithms for LLM analysis, paving the way for more efficient, accessible, and accurate natural language processing.

2. Background and Related Work

Large Language Models (LLMs) have revolutionized natural language processing, enabling significant advances in tasks such as text generation, machine translation, and question answering. These models rely on high-dimensional embedding spaces to represent words, phrases, and sentences as vectors, capturing complex semantic relationships. Popular LLM architectures, such as Transformers, utilize attention mechanisms to weigh the importance of different words in a sentence, allowing the model to capture long-range dependencies and contextual information [1]. Various embedding techniques, including Word2Vec, GloVe, BERT, and Sentence Transformers, have been developed to generate these vector representations, each with its own strengths and limitations [2-6]. However, the increasing size and complexity of LLMs pose significant computational challenges, requiring approaches to improve efficiency and reduce training costs. Furthermore, LLMs are known to sometimes generate outputs that are factually incorrect or nonsensical, a phenomenon known as hallucination [7].

Semantic similarity measures are crucial for many NLP tasks, including information retrieval, text classification, and question answering. Cosine similarity is a widely used measure for quantifying the similarity between embedding vectors, based on the cosine of the angle between them [8]. It is computationally efficient and relatively insensitive to the magnitude of the vectors. However, cosine similarity may not always capture subtle semantic nuances that are not reflected in the geometric arrangement of the vectors [9,10]. The inherent uncertainties in LLM-generated embeddings raise questions about the ultimate accuracy of cosine similarity, suggesting the need for alternative measures that can account for these uncertainties. Other similarity measures, such as Euclidean distance, can also be used, but they may be more sensitive to differences in vector magnitude or dimensionality. The choice of similarity measure depends on the specific application and the characteristics of the embedding space.

Concepts from statistical mechanics and information theory have been applied to natural language processing in various ways. Entropy has been used to measure the uncertainty or diversity of language models, providing insights into their ability to generate coherent and informative text [11,12]. Concepts from thermodynamics have been used to model the dynamics of language change, capturing the evolution of word frequencies and semantic relationships over time. Information-theoretic measures, such as mutual information and Kullback-Leibler divergence, have been used to quantify the complexity of linguistic structures and the relationships between different linguistic levels [13,14]. These approaches provide a complementary perspective to traditional NLP methods, offering a more statistical and probabilistic view of language [15-18].

Quantum natural language processing (QNLP) is an emerging field that explores the use of quantum algorithms and quantum hardware for NLP tasks. Quantum computing offers the possibility for significant speedups in certain computations, which could be beneficial for processing the large amounts of data involved in NLP. Quantum circuits can implement new formalisms for representing and manipulating semantic information [19]. Previous work has shown that quantum support vector machines (QSVMs) can be used for text classification and quantum neural networks for machine translation [20,21]. Specific quantum algorithms, such as Grover's search algorithm and quantum phase estimation, have also been applied to NLP problems. Theoretical frameworks, such as those based on quantum probability and quantum cognition, have also been applied to model semantic relationships and human language processing [22]. However, QNLP is still in its early stages, and significant challenges remain in developing practical quantum algorithms and building scalable quantum hardware [23]. Overcoming these challenges could lead to a democratization of LLM development, reducing computational costs and enabling wider access to these powerful technologies.

In previous work we used quantum inspired approach to model embedding spaces, and we explored the use of a double-slit experiment approach for calculating cosine similarity on a quantum computer [24-27]. This paper builds upon that work by introducing a new similarity measure based on partition functions and exploring its implementation on quantum hardware. The partition function provides a formalism for capturing the statistical properties of complex systems, offering a new lens through which to view LLM embedding spaces [28-31].

3. Cosine Similarity

The partition function approach builds upon the concept of cosine similarity; therefore, we begin with its definition. The cosine similarity between vectors **a** and **b** is defined as

$$S_C(\mathbf{a}, \mathbf{b}) = \cos\theta = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||\ ||\mathbf{b}||} \tag{1}$$

We assume that **a** and all other vectors discussed in this section are L2-normalized, meaning that

$$||\mathbf{a}||^2 = \sum_{i=1}^{N} a_i^2 = 1 \tag{2}$$

L2 normalization ensures that the similarity between embeddings primarily reflects semantic relationships learned during the LLM's training, as captured by the angle between embedding vectors, rather than being influenced by the vector magnitudes. This structure is a consequence of the LLM's architecture and training data.

4. Partition Function for Embedding Vectors

To apply the mathematical framework of statistical mechanics to embedding vectors, we introduce the concept of a partition function. This allows us to analyze the statistical properties of the embedding space and relate them to the behavior of Large Language Models (LLMs). The partition function provides a

statistical description of the embedding vector, bridging the gap between the vector representation and a system-level view of the embedding space.

An embedding vector **a** = [a_1,a_2,...,a_N] represents the contribution of different semantic features. Building on the observation that embedding spaces exhibit structure [27], we treat the components, a_i, as analogous to energy levels in a physical system. Specifically, we consider a system where the energy levels are proportional to the embedding vector components. If we interpret a_i (or a_i^2) as eigenvalues of an operator representing a semantic property, the partition function describes the statistical distribution of semantic features in the system's ground state. This ground state representation serves as a baseline for analyzing perturbations or measuring other semantic properties. We define the partition function Z for an embedding vector **a** as

$$Z(\mathbf{a}, \beta) = \sum_{i=1}^{N} \exp(-\beta a_i) \tag{3}$$

where β is an inverse temperature parameter, analogous to its role in statistical mechanics. This temperature is distinct from any temperature-like parameters within the LLM. The choice of β determines the scale at which we probe the embedding space. A large β emphasizes lower energy (smaller a_i) components, focusing on prominent semantic features. A small β gives more weight to higher energy (larger a_i) components, allowing exploration of less dominant semantic aspects. In principle, β could be related to the model's attention weights or confidence scores.

For this initial exploration, we set $\beta = 1$, giving equal weight to all dimensions

$$Z(\mathbf{a}) = \sum_{i=1}^{N} \exp(-a_i) \tag{4}$$

This partition function characterizes the statistical state of the embedding vector. A higher Z indicates more accessible energy levels (a broader distribution of semantic features), suggesting a more diverse semantic representation. A lower Z indicates a more concentrated distribution, suggesting a more focused semantic representation, characterizing the ground state.

Consider a perturbation of the embedding vector, **b** = [$a_1 + \Delta_1$,$a_2 + \Delta_2$,...,$a_N + \Delta_N$]. The partition function for the perturbed vector **b** is

$$Z(\mathbf{b}) = \sum_{i=1}^{N} \exp(-(a_i + \Delta_i)) = \sum_{i=1}^{N} \exp(-a_i) \exp(-\Delta_i) \tag{5}$$

The change in the partition function, $\Delta Z = Z(\mathbf{b}) - Z(\mathbf{a})$, is

$$\Delta Z = \sum_{i=1}^{N} \exp(-a_i)(\exp(-\Delta_i) - 1) \tag{6}$$

For small perturbations, we approximate $\exp(-\Delta i) \approx 1 - \Delta_i + \Delta_i^2/2$, leading to

$$\Delta Z \approx -\sum_{i=1}^{N} \exp(-a_i)(\Delta_i - \frac{1}{2}\Delta_i^2) \tag{7}$$

This equation relates the change in Z to perturbations in the embedding vector components. The change in Z is a weighted sum of the perturbations, with weights determined by the original embedding vector components, quantifying the effect of perturbations on the statistical state. The L2 normalization of embedding vectors constrains the possible values of ΔZ, implying that the perturbations are not arbitrary but satisfy a condition related to the conservation of "energy" in the embedding space, as discussed in [27].

The partition function allows us to define thermodynamic quantities, such as the average energy

$$\langle E \rangle = -\frac{\partial}{\partial \beta} \ln Z = \frac{\sum_{i=1}^{N} a_i \exp(-\beta a_i)}{\sum_{i=1}^{N} \exp(-\beta a_i)} \tag{8}$$

and the entropy

$$S = \beta \langle E \rangle + \ln Z \tag{9}$$

These quantities provide further insights into the distribution of semantic features. High entropy suggests a more uniform distribution, indicating a less focused semantic representation, while low entropy suggests a more concentrated distribution, indicating a more focused representation.

5. Partition Function and Connection to the Density Matrix

In previous work, we introduced the quantum partition function in the context of LLM embedding spaces [26]. This section establishes a connection between the classical partition function (Z), the quantum partition function, and the quantum density matrix (**ρ**), demonstrating the consistency of their relationships. The density matrix describes the state of a quantum system as a statistical ensemble of pure states, while the partition function focuses on the statistical distribution of energy levels within the embedding vector. The average energy (⟨E⟩) of the system provides the key link between these formalisms.

The quantum partition function is defined as

$$Z = \mathrm{Tr}\left[\exp(-\beta \hat{H})\right] \tag{10}$$

where $\hat{H}$ is the Hamiltonian operator and β is the inverse temperature. The Hamiltonian operator represents the total energy of the system, describing the energy landscape of the embedding space in our context. A simple example is a diagonal Hamiltonian

$$\hat{H} = \text{diag}(a_1, a_2, ..., a_N) \tag{11}$$

where the diagonal elements are the components of the embedding vector **a**. This corresponds to a system where the semantic features represented by the embedding vector components are independent. With this definition, the quantum partition function becomes

$$Z = \sum_{i=1}^{N} \exp(-\beta a_i) \tag{12}$$

which is mathematically identical to the classical partition function in this specific case. However, the quantum formalism's strength lies in its ability to accommodate more general Hamiltonian forms. While a diagonal Hamiltonian treats embedding vector components as independent energy levels, a non-diagonal Hamiltonian can capture more complex relationships, nonlinear effects, and correlations between these components, possible revealing subtle quantum-like effects within the embedding space. For example, off-diagonal elements in $\hat{H}$ could represent interactions or dependencies between different semantic features, such as synonymy or antonymy.

The average energy can be expressed in terms of both the partition function and the density matrix

$$\langle E \rangle = \text{Tr}(\hat{H}\boldsymbol{\rho}) \tag{13}$$

Where $\boldsymbol{\rho}$ is the density operator, related to the Hamiltonian and partition function by

$$\boldsymbol{\rho} = \frac{\exp(-\beta \hat{H})}{Z} \tag{14}$$

The density matrix ρ describes the statistical state of the embedding vectors in terms of their quantum components, reflecting correlations between different semantic features. The partition function, through the average energy, normalizes the density operator and relates it to the energy levels defined by the Hamiltonian. This connection provides a more complete and consistent framework for analyzing LLM embedding spaces using tools from statistical mechanics and quantum mechanics. The elements of the density matrix reflect the correlations between semantic features within the LLM's representation. A large off-diagonal element ρ_{ij} indicates a strong correlation between the semantic features represented by components i and j of the embedding vector.

6. Embedding Example

This section introduces an illustrative embedding example that will be used in subsequent sections. We employed the OpenAI embedding model to generate 64-dimensional embedding vectors for two test sentences. OpenAI was selected for its accessibility and ease of integration, although other embedding models could be used. The two sentences used in the experiment were

1.Sentence 1: "The energetic dog joyfully runs across the park."
2.Sentence 2: "A happy canine is running with enthusiasm in the park."

These sentences were chosen as a simple example of paraphrases, allowing us to explore how the partition function approach captures semantic similarity. The embedding vector for Sentence 1 is visualized in Figure 1. Each horizontal line represents the value of a single dimension in the 64-dimensional embedding. The corresponding partition function, calculated as described in Section IV, is shown in Figure 2. This transformation emphasizes dimensions with lower values in the original embedding, possible representing less salient but still relevant features. The y-axis is scaled from 0 to 2 for ease of visualization.

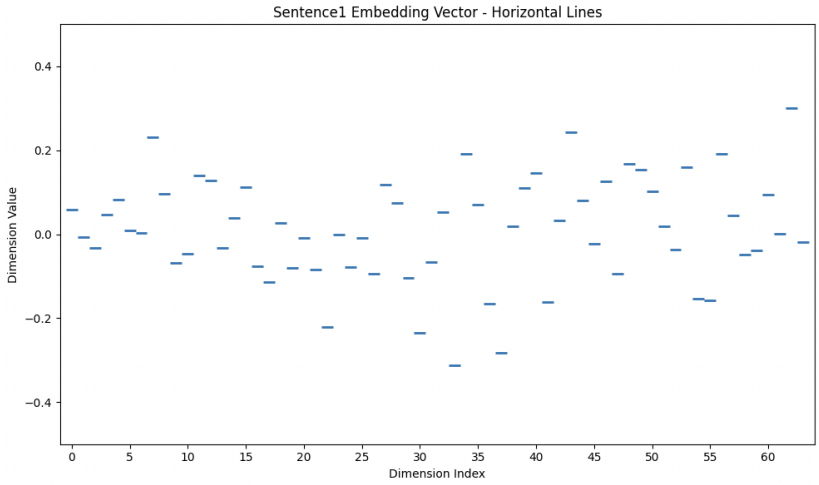

Figure 1: Visualization of the Normalized Embedding Vector for the Sentence 'The Energetic Dog Joyfully Runs Across the Park.' Each Horizontal Line Represents a Dimension of the 64-Dimensional Embedding

The cosine similarity between the sentences, calculated according to the definition provided in Section III, is a value that reflects the angle between their vector representations

$$S_C(\text{Sentence 1}, \text{Sentence 2}) = 0.750 \quad (15)$$

This forms the basis of the partition function similarity.

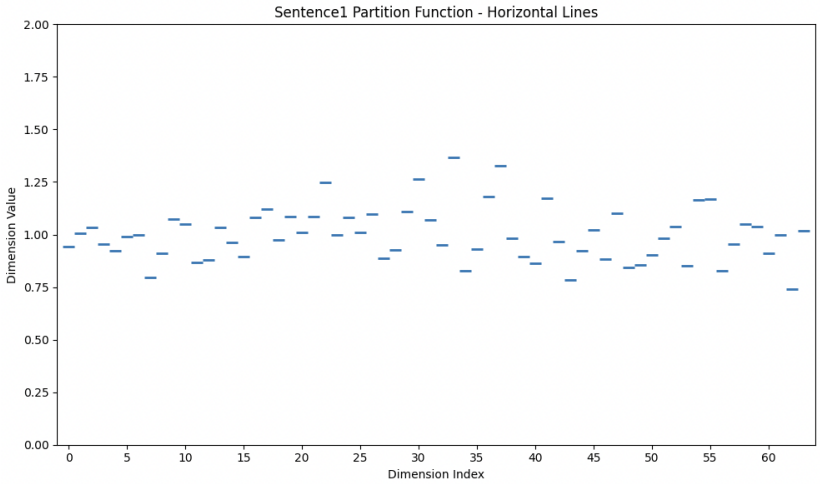

Figure 2: Visualization of the Partition Function Derived from the Sentence Embedding. This Transformation Highlights Dimensions with Lower Values in the Original Embedding, Representing Less Salient Features. The y-Axis is Scaled from 0 to 2

The average energy and entropy for Sentence 1 are

$$\langle E \rangle(\text{Sentence 1}) = -0.00621 \quad (16)$$
$$S(\text{Sentence 1}) = 4.15096 \quad (17)$$

For Sentence 2, the values are

$$\langle E \rangle(\text{Sentence 2}) = 0.0139 \quad (18)$$
$$S(\text{Sentence 2}) = 4.15117 \quad (19)$$

The average energy, calculated from the sentence embeddings, exhibits sensitivity to subtle wording differences, even in paraphrased sentences, showing different signs despite the similar meanings. Entropy, on the other hand, remains relatively consistent between the paraphrases, reflecting their shared information content. This suggests that average energy may capture stylistic variations, while entropy may be more indicative of semantic similarity. However, further research with a larger dataset is needed to validate these preliminary observations.

7. Similarity between Partition Functions

To quantify the similarity between embedding vectors, we propose a similarity measure based on their partition functions. This aims to capture the degree to which two embedding vectors exhibit similar statistical distributions of semantic features, rather than relying solely on the geometric relationship captured by cosine similarity. Cosine similarity focuses on the angle between vectors and may not fully capture the nuances of semantic relationships reflected in the distribution of feature ”energies.”

We define the normalized partition function as

$$Z_{\text{norm}}(\mathbf{a}, \beta = 1) = \frac{1}{N_f} Z(\mathbf{a}) \quad (20)$$

where N_f is a normalization factor. This normalization scales the partition function to a comparable range across different embedding vectors, ensuring that the similarity measure is not dominated by the overall magnitude of the partition functions.

The rationale for this normalization is as follows: since each component a_i of the embedding vectors is bounded between -1 and 1, we aim to scale Z_{norm} such that its values fall within a reasonable range. Given the exponential nature of the partition function, the components $\exp(-a_i)$ are always positive. Assuming that the embedding vectors have a sufficient number of dimensions and that each dimension is non-zero, we initially select $N_f = 2$. With this choice, $\exp(0.69)/2 \approx 1$, implying that if a single dimension has a value greater than -0.69, the normalized contribution from that dimension will be approximately 1 or less. This helps to ensure that the normalized values can be interpreted as a probability density, which should ideally be between 0 and 1. A concrete example will be shown in later sections. If, in practice, the embedding vectors exhibit significantly lower values, we could increase N_f to 3 or higher to ensure that Z_{norm} remains within a manageable range. For the experimental evaluation presented later, we use $N_f = 2$.

The similarity measure, S_Z, between embedding vectors **a** and **b** is then defined as the product of their normalized partition functions

$$S_Z(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{N} Z_{\text{norm}}(\mathbf{a})_i \cdot Z_{\text{norm}}(\mathbf{b})_i = \frac{1}{N_f^2} \sum_{i=1}^{N} [Z(\mathbf{a})_i \cdot Z(\mathbf{b})_i] \quad (21)$$

where $Z(\mathbf{a})$ and $Z(\mathbf{b})$ are the unnormalized partition functions. With $N_f = 2$, this simplifies to ¼ [$Z(\mathbf{a})_i \cdot Z(\mathbf{b})_i$]. This similarity measure compares the magnitudes of the normalized partition functions. A higher value of S_Z suggests that both embedding vectors have a relatively large number of accessible states (high Z values) after normalization, indicating a greater similarity in their overall statistical distribution of semantic features. Embedding vectors with similar normalized partition functions are likely to represent semantic concepts with a similar distribution of underlying features. This similarity measure may agree with cosine similarity when the semantic relationship is primarily determined by the geometric arrangement of the vectors, but it may differ when the statistical distribution of features is more important. For example, two sentences with different word order but similar semantic content might have different cosine similarities but similar S_Z values.

In the next subsection, we will explore how quantum circuits can estimate this similarity measure, offering advantages in computational efficiency or the ability to capture more complex relationships between semantic features.

7.1. Applications in Quantum Circuits for Partition Function Similarity

The framework developed in this paper suggests applications in designing and analyzing quantum circuits for natural language processing. While current quantum hardware has limitations in qubit count and fidelity, exploring these theoretical possibilities can provide insights into future quantum algorithms for LLM analysis. In [26], a double-slit experiment approach was used to calculate cosine similarity. Here, we outline an alternative approach for estimating the similarity measure based on partition functions, using a simplified quantum circuit. This circuit serves as an illustration of how quantum computation can be used in LLM analysis; it is not a fully optimized solution.

To estimate the similarity between two N-dimensional embedding vectors, **a** and **b**, we consider a quantum circuit using two sets of N qubits each. The first set represents vector **a**, and the second set represents vector **b**, as shown in Figure 3.

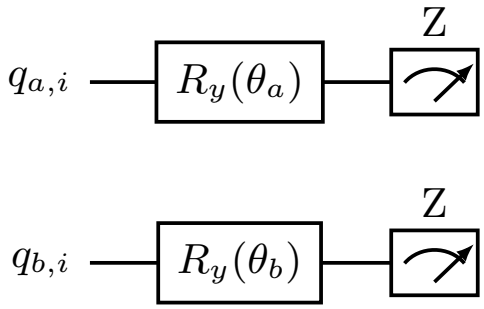

Figure 3: This circuit illustrates a simplified approach for estimating the similarity between partition functions using quantum computation. Qubits $q_{a,i}$ and $q_{b,i}$, representing the i-th dimension of vectors **a** and **b** respectively, are initialized using rotation gates $R_y(\theta_a)$ and $R_y(\theta_b)$ to encode the normalized partition function values as amplitudes. Measurement in the Z-basis (computational basis) provides probabilities that can be used to estimate the similarity between the partition functions

The normalized partition function for dimension i of vectors **a** and **b** are calculated as

$$Z_{\text{norm}}(\mathbf{a})_i = \frac{\exp(-a_i)}{N_f} \tag{22}$$

$$Z_{\text{norm}}(\mathbf{b})_i = \frac{\exp(-b_i)}{N_f} \tag{23}$$

where N_f is the normalization factor defined in the previous section. For each dimension i, the qubit representing vector **a** is initialized in the state $A_i|0\rangle + B_i|1\rangle$, where $A_i = Z_{\text{norm}}(a)_i$ and $B_i = \sqrt{1 - A_i^2}$. Similarly, the qubit representing vector **b** is initialized in the state $C_i|0\rangle + D_i|1\rangle$, where $C_i = Z_{norm}(b)_i$ and $D_i = \sqrt{1 - C_i^2}$. The angles θ_a and θ_b in the Ry gates are chosen such that $A_i = \cos(\theta_a/2)$ and $C_i = \cos(\theta_b/2)$. This encoding maps the normalized partition function to the amplitude of the $|0\rangle$ state. This amplitude encoding is chosen for its simplicity and ease of implementation; however, other encodings, such as phase encoding, could also be used.

Consider the tensor product of the qubits representing dimension i from vectors **a** and **b**

$$\begin{aligned} |\psi_i\rangle &= (A_i|0\rangle + B_i|1\rangle) \otimes (C_i|0\rangle + D_i|1\rangle) \\ &= A_iC_i|00\rangle + A_iD_i|01\rangle + B_iC_i|10\rangle + B_iD_i|11\rangle \end{aligned} \tag{24}$$

The probability of measuring the state $|00\rangle$ for dimension i is then

$$\begin{aligned} P_i(|00\rangle) &= (A_iC_i)^2 = (Z_{\text{norm}}(\mathbf{a})_i \cdot Z_{\text{norm}}(\mathbf{b})_i)^2 \\ &= \left(\frac{\exp(-a_i)}{N_f} \cdot \frac{\exp(-b_i)}{N_f}\right)^2 = \frac{\exp(-2a_i - 2b_i)}{N_f^4} \end{aligned} \tag{25}$$

The tensor product and measurement of the $|00\rangle$ state effectively calculates the product of the normalized partition functions for

each dimension, providing a measure of their similarity. This leads to the following rescaled similarity measure

$$S_Z = \sum_{i=1}^{N} P_i(|00\rangle) = \frac{1}{N_f^4} \sum_{i=1}^{N} \exp(-2(a_i + b_i)) \tag{26}$$

In our current example $N_f = 2$. Note, that by setting β to 1/2, we could modify the term $\exp(-(a_i + b_i))$, resulting in an expression proportional to $A_i C_i$.

7.1.1. Quantum Computer Implementation

To validate the proposed approach, we performed an experimental evaluation using the IBM Quantum cloud service and the Qiskit Python library. Due to the limitations of currently available quantum hardware, specifically the maximum qubit count of 156 on IBM Quantum systems at the time of the experiment, we used embedding vectors of 64 dimensions. This allowed us to implement the quantum circuit for two vectors, requiring a total of 128 qubits. We used the OpenAI embedding model to create vector embeddings for the sentences described in Section VI and then transformed these embeddings into partition functions, as described previously. We then calculated the similarity measure SZ using both a classical computer (exact calculation) and the IBM Quantum computer. The results of the first 10 P_i ($|00\rangle$) are shown in the Table I.

The result of the exact similarity calculation was

$$S_Z(\text{exact}) = 4.147 \tag{27}$$

The result obtained from the quantum computer was

$$S_Z(\text{quantum computer}) = 4.455 \tag{28}$$

These results were obtained using 4096 shots on the ibm boston quantum computer as the backend.

	Exact Results	**Quantum Computer**
$P_0(\|00\rangle)$	0.0415	0.0452
$P_1(\|00\rangle)$	0.0550	0.0610
$P_2(\|00\rangle)$	0.0573	0.0537
$P_3(\|00\rangle)$	0.0489	0.0454
$P_4(\|00\rangle)$	0.0436	0.0474
$P_5(\|00\rangle)$	0.0517	0.0574
$P_6(\|00\rangle)$	0.0526	0.0583
$P_7(\|00\rangle)$	0.0259	0.0276
$P_8(\|00\rangle)$	0.0483	0.0598
$P_9(\|00\rangle)$	0.0714	0.0691

Table I: Comparison of Theoretical and Experimental Measurement Probabilities for the First 10 $P_i(|00\rangle)$ States, Demonstrating the Quantum Simulation's Approximation of Expected Behavior

7.1.2. Results Interpretation

The agreement between the exact calculation and the quantum computer result is reasonable, with a relative error of approximately 0.074. This relative error is calculated as $|S_Z(\text{quantum computer}) - S_Z(\text{exact})|/S_Z(\text{exact})$.

Several factors may contribute to the observed difference between the two results. One factor is statistical variation. The quantum computation was performed only once due to the cost associated with using quantum hardware. The number of shots used was 4096, which could be varied. Repeating the experiment multiple times would allow us to assess the statistical variation in the results and obtain a more reliable estimate of the quantum computer's performance.

Another factor is quantum fidelity. The limited fidelity of current quantum hardware can introduce errors in the computation. Factors such as qubit decoherence and gate errors can affect the accuracy of the quantum circuit and contribute to the observed difference. A further consideration is the statistical nature of cosine similarity. As discussed in Section VIII, cosine similarity itself can be viewed as a statistical measure, and the embedding vectors used as input to the quantum computation may contain inherent inaccuracies. This suggests that the "exact" result obtained from the classical computer may not be a perfect representation of the true semantic similarity between the sentences. Therefore, the quantum computer result, despite not perfectly aligning with the classical calculation, may, in fact, be a more accurate reflection of the underlying semantic relationship.

7.1.3. Code Listing

This subsection shows the Python code that was used in quantum computer calculation.
Listing 1: Qiskit code for quantum cosine similarity calculation.

```python
import numpy as np
import pickle
from qiskit import QuantumCircuit, transpile
from qiskit_ibm_runtime import QiskitRuntimeService, Session, Sampler

def run_quantum_circuit(A, B, C, D, filename='quantum_results.pkl'):
    """
    Creates a quantum circuit, initializes qubits based on the provided amplitudes,
    measures the qubits, and stores the results in a pickle file.

    Args:
        A (list): List of amplitudes A_i for each qubit.
        B (list): List of amplitudes B_i for each qubit.
        C (list): List of amplitudes C_i for each qubit.
        D (list): List of amplitudes D_i for each qubit.
        filename (str): The name of the file to store the results
    """
    num_dimensions = len(A)
    if not (len(B) == len(C) == len(D) == num_dimensions):
        raise ValueError("Amplitude lists must have the same length.")

    num_qubits = 2 * num_dimensions  # Two qubits per dimension

    qc = QuantumCircuit(num_qubits, num_qubits)

    for i in range(num_dimensions):
        # Initialize qubit 2*i with A[i] and B[i]
        normalized_state_a = [A[i], B[i]]
        qc.initialize(normalized_state_a, 2*i)

        # Initialize qubit 2*i+1 with C[i] and D[i]
        normalized_state_b = [C[i], D[i]]
        qc.initialize(normalized_state_b, 2*i + 1)

    qc.measure(range(num_qubits), range(num_qubits))

    try:
        # Connect to IBM Quantum
        service = QiskitRuntimeService()

        # Select a real quantum backend
        backend = service.least_busy(simulator=False, operational=True, min_num_qubits=num_qubits)
        print(f"Using backend: {backend.name}")

        # Transpile the circuit for the backend
        transpiled_qc = transpile(qc, backend)

        # Create a session
        with Session(backend=backend) as session:
            # Create a Sampler instance
            sampler = Sampler()

            # Run the circuit using Sampler within the session
            job = sampler.run([transpiled_qc], shots=4096)
            print(f"Job ID: {job.job_id()}")
            result = job.result()

            # Store the result using pickle
            with open(filename, 'wb') as f:
                pickle.dump(result, f)

            print(f"Results stored in {filename}")

    except Exception as e:
        print(f"Error running the circuit: {e}")
        print(f"Error type: {type(e)}")
        print(f"Error message: {e}")
```

7.1.4. Code Explanation

The Python code listed in the previous subsection implements the quantum circuit for estimating the similarity measure based on partition functions, as described in Section VII. The code utilizes the Qiskit and Qiskit Runtime libraries for quantum circuit construction and execution on IBM Quantum hardware.

The core function, `run_quantum_circuit`, takes four lists as input: `A`, `B`, `C`, and `D`. These lists represent the amplitudes used to initialize the qubits in the quantum circuit. Specifically, A[i] and B[i] are the amplitudes for the $|0\rangle$ and $|1\rangle$ states, respectively, for the qubit representing dimension i of the first embedding vector. Similarly, C[i] and D[i] are the amplitudes for the $|0\rangle$ and $|1\rangle$ states for the qubit representing dimension i of the second embedding vector.

The function constructs a quantum circuit with `2 * num_dimensions` qubits, where `num_dimensions` is the dimensionality of the embedding vectors. It then iterates through each dimension, initializing the corresponding qubits using the `qc.initialize()` method to prepare them in the desired superposition states based on the provided amplitudes. After initializing all qubits, the function performs a measurement on all qubits using `qc.measure()`. The measurement results are used to estimate the probability of measuring the $|00\rangle$ state for each dimension, as described in Section VII.

To execute the quantum circuit on real quantum hardware, the code connects to the IBM Quantum cloud service using `QiskitRuntimeService()`. It then selects the least busy operational quantum backend with a sufficient number of qubits using `service.least_busy()`. The quantum circuit is then transpiled for the chosen backend using `transpile()`, optimizing it for the specific hardware architecture. A Qiskit Runtime `Session` is created to group multiple calls to the quantum computer. Within the session, a `Sampler` primitive is used to execute the transpiled circuit, estimating the quasi-probabilities of each measured outcome. The circuit is executed with 4096 shots, and the results are stored in a pickle file for later analysis. Error handling is included to catch any exceptions that may occur during the quantum circuit execution. If an error occurs, the error message and type are printed to the console.

8. On the Accuracy and Statistical Nature of Cosine Similarity

Throughout this paper, we have drawn parallels between LLM embedding spaces and partition functions, highlighting structured states, uncertainties, and other phenomena. This raises a fundamental question regarding the seemingly deterministic nature of cosine similarity, a widely used measure for quantifying the relationship between embedding vectors. If embedding spaces are governed by probabilistic principles, how can cosine similarity, which yields a precise numerical value (limited only by floating-point precision), be considered a fully accurate representation of semantic relationships? This section aims to reconcile the seemingly deterministic nature of cosine similarity with the probabilistic nature of LLM embedding spaces.

The key lies in recognizing that while the calculation of cosine similarity is deterministic, the generation of the embedding vectors themselves is not. Models like OpenAI and Sentence Transformers, while designed to map semantically similar inputs to nearby points in the embedding space, are inherently non-deterministic processes. The training data, model architecture, and optimization algorithms all contribute to variability in the resulting embeddings. For example, slightly different initializations during training can lead to variations in the learned embedding space, even for the same training data. This non-determinism arises from the complex interplay of factors during training and the inherent limitations of representing continuous semantic information in a discrete vector space. The high dimensionality of embedding spaces also contributes; small changes in individual dimensions can accumulate and lead to noticeable differences in the overall vector representation.

Consequently, while the cosine similarity calculation may yield a precise numerical value with many decimal places, not all of those decimals necessarily reflect true semantic accuracy. The embedding vector itself may contain subtle inaccuracies due to the non-deterministic nature of its creation and the possibility for the model to settle into suboptimal states. The apparent precision of cosine similarity can be misleading, providing a false sense of certainty about the underlying semantic relationship.

Following this reasoning, cosine similarity can be viewed as a statistical measure, providing an estimate of the semantic relationship between two inputs. It is a useful tool for capturing broad trends and identifying general semantic similarities, but it should not be interpreted as a perfectly precise or definitive measure.

This perspective sheds new light on the results obtained in this paper and in our previous work, where we compared cosine similarity calculations performed using a quantum computer to theoretical results obtained from the standard cosine similarity equation [26]. The initial discrepancy between the quantum and classical results led to the hypothesis that the limited fidelity of the quantum computer was the dominant source of error.

However, the analysis presented in this paper suggests an alternative, or perhaps complementary, explanation: the theoretical cosine similarity value, calculated using standard floating-point arithmetic, may itself be subject to inaccuracies due to the inherent limitations of the embedding vectors. The quantum computer, despite its imperfections, may have been providing a more accurate representation of the underlying semantic relationship, capturing nuances lost in the deterministic calculation of cosine similarity based on possibly flawed embedding vectors. The quantum computation, operating on probabilities and amplitudes, can offers a more nuanced representation and opens the door to using quantum error mitigation techniques to improve accuracy.

9. Discussion

We have demonstrated a framework for characterizing the statistical properties of embedding vectors using concepts from statistical mechanics, establishing a connection between classical and quantum descriptions of semantic spaces. Furthermore, we have proposed a similarity measure based on partition functions and presented an initial experimental evaluation of its implementation on a real quantum computer, the ibm_boston. Our results suggest that the partition function provides a valuable tool for analyzing LLM embedding spaces, offering a complementary perspective to traditional methods based on cosine similarity. The partition function captures information about the distribution of semantic features within an embedding vector, revealing nuances that are not reflected in the geometric arrangement of the vectors. The experimental results, while preliminary, indicate that quantum computers can be used to estimate the partition function-based similarity measure, opening the door to speedups in similarity calculations and to more nuanced representations of semantic similarity. The observed agreement between the classical and quantum results, despite the limitations of current quantum hardware, is encouraging and suggests that further research in this area is warranted.

Future research should focus on exploring the capabilities of partition functions and quantum computing for LLM analysis. This includes conducting more extensive experimental evaluations using larger quantum computers and a wider range of embedding models and sentences. It also involves developing more efficient quantum circuits for estimating the partition function-based similarity measure, leveraging techniques from quantum machine learning and quantum error mitigation. Research should explore the relationship between the inverse temperature parameter β and LLM internal parameters, such as attention weights, and investigate the use of non-diagonal Hamiltonians to capture more complex relationships between semantic features.

In conclusion, this paper has presented a framework for analyzing LLM embedding spaces using partition functions and has provided an initial exploration of its implementation on quantum hardware. While significant challenges remain, the results suggest that this approach has the possibility to provide new insights into the nature of semantic representation and to enable the development of more efficient and accurate NLP algorithms. We believe that further research in this area will contribute to advancing the state of the art in both quantum computing and natural language processing.

Reference

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30*.
2. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *International Conference on Learning Representations*.
3. Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing* (*EMNLP*) (pp. 1532-1543).
4. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1724–1734.
5. Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research, 3*(Feb), 1137-1155.
6. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
7. Ji, Z., Lee, N., Frieske, R., Yu, T., Dan Su, Y., Xu, E., Ishii, Y., Bang, D. Chen, W. Dai, H.S Chan, A. Madotto, and P. Fung. (2022). Survey of Hallucinations in Natural Language Generation. *ACM Computing Surveys*.
8. G. Salton and M.J. McGill. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc.,.
9. Turney, P. D., & Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *acm Transactions on Information Systems (tois), 21*(4), 315-346.
10. Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (2012, July). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning* (pp. 1201-1211).
11. Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal, 27*(3), 379-423.
12. Cover, T. M. and Thomas., J. A. (2006). Elements of Information Theory, 2nd Edition. *Wiley&Sons*.
13. Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics, 22*(1), 79-86.
14. Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory, 37*(1), 145-151.
15. Coldenfeld, N. (1994). Lectures on Phase Transitions And The Renormalization Group, *Addison-Wesley*.
16. Huang, K. (1987). Statistical Mechanics, 2nd Edition. *John Wiley & Sons*.
17. Pathria, R. K. (1996). Statistical Mechanics 2nd Edition. Butterworth-Heinemann.
18. Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation, 14*(8), 1771-1800.
19. Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. *Cambridge university press*.
20. Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature

spaces. *Nature, 567*(7747), 209-212.
21. Beer, K., Bondarenko, D., Farrelly, T., Osborne, T. J., Salzmann, R., Scheiermann, D., & Wolf, R. (2020). Training deep quantum neural networks. *Nature communications, 11*(1), 808.
22. Khrennikov, A. (2010). Ubiquitous quantum structure. *Springer*.
23. DiVincenzo, D. P. (2000). The physical implementation of quantum computation. *Fortschritte der Physik: Progress of Physics, 48*(9-11), 771-783.
24. Laine, T. A. (2025). Semantic Wave Functions: Exploring Meaning in Large Language Models Through Quantum Formalism. *OA J Applied Sci Technol, 3*(1), 01-22.
25. Laine, T. A. (2025). The Quantum LLM: Modeling Semantic Spaces with Quantum Principles. *OA J Applied Sci Technol, 3*(2), 01-13.
26. Laine, T. A. (2026). Quantum LLMs Using Quantum Computing to Analyze and Process Semantic Information. *OA J Applied Sci Technol, 4*(1), 01-19.
27. Laine, T. A. (2026). Discrete Semantic States and Hamiltonian Dynamics in LLM Embedding Spaces. *OA J Applied Sci Technol, 4*(1), 01-23.
28. Ising, E. (1925). Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik, 31*(1), 253-258.
29. Baxter, R.J. (1989). Exactly Solved Models in Statistical Mechanics, 3rd Edition. *Academic Press*.
30. Stanley, H. E. (1987). Introduction to phase transitions and critical phenomena. *Oxford University Press*.
31. Egelstaff, P.A. (1994). Liquids: Structure, Dynamics, and General Properties. *Taylor & Francis*.