IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151173

Hallucination Detection Using Evidence-Based Methodologies in Large Language Models: Comparative Analysis of Classical vs. Quantum Classifiers

Khushi[1], Paavai Anand G[2], Sridevi S[3], Beenarani Manoj[4]

Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, India[1]

Mentor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, India[2-4]

Abstract: Large language models can produce answers that sound right but are not actually supported by the text they were given. This work builds a simple, fully CPU-based check for that problem. Each answer is turned into five numbers: the share of hedge words, the average token likelihood under a small local model, how similar the answer is to the evidence, how similar it is to the question, and an entailment probability. Those five numbers then go into ordinary classifiers (logistic regression, random forest, XGBoost, a tiny neural net) and also into two quantum-style classifiers that run in simulation: a five-qubit variational circuit and a quantum-kernel SVM. We took the first sixty questions from the HaluEval QA set. That gives one hundred and twenty answers—one correct and one hallucinated for every question. Splits keep whole questions together so nothing leaks. On the sixty-answer training slice that both families see, the classical models reach an AUROC of about 0.975. The quantum-kernel SVM gets 0.90 and the variational circuit only 0.53. The test set itself is small (eighteen answers). A later check of the exact quantum state also showed that our particular circuit almost completely ignores its fifth input. The code is reproducible, models reload cleanly, and a small local interface lets anyone try new examples. The pipeline works, but the data set is tiny, calibration is uncertain, and the variational circuit has a clear design flaw, so we make no claim of quantum advantage.

Keywords: hallucination detection, evidence checking, quantum machine learning, quantum kernels, variational quantum classifier, probability calibration

1.INTRODUCTION

Language models often generate responses that are either inconsistent with their sources [1] or contain information that could not be derived from them.

This paper investigates the latter case by considering a dataset containing questions, evidence passages, and answers together with a binary indicator labeling whether each response was “supported” based on its associated evidence. The goal then becomes to determine if applying quantum classifiers using a given input’s compact representation (identical to what would go into a classical alternative) produces additional insights.

As this investigation required examining all features used for classification, we devised a simple yet effective representation comprising 5 features, enabling both easy simulation and inspection of internal workings. However, because such feature extractions discard many details about the input, neither approach is able to capture specific details about how the model reached its conclusion.

The results of our initial exploratory experiments should therefore be considered a pilot test of a working system only rather than an exhaustive review of capabilities. The circuit diagnosis undertaken along the way has been documented separately but is not repeated here, nor does this represent a full-blown, comprehensive benchmark.

2. RELATED WORKS AND SYSTEM SCOPE

In HaluEval, we provide generated and annotated examples suitable for studies about hallucination recognition [2] (Section 2). Our question answering record includes knowledge passages together with their supported and hallucinated answers. By pairing them with the same questions, one may control which parts are being compared yet still cannot do

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

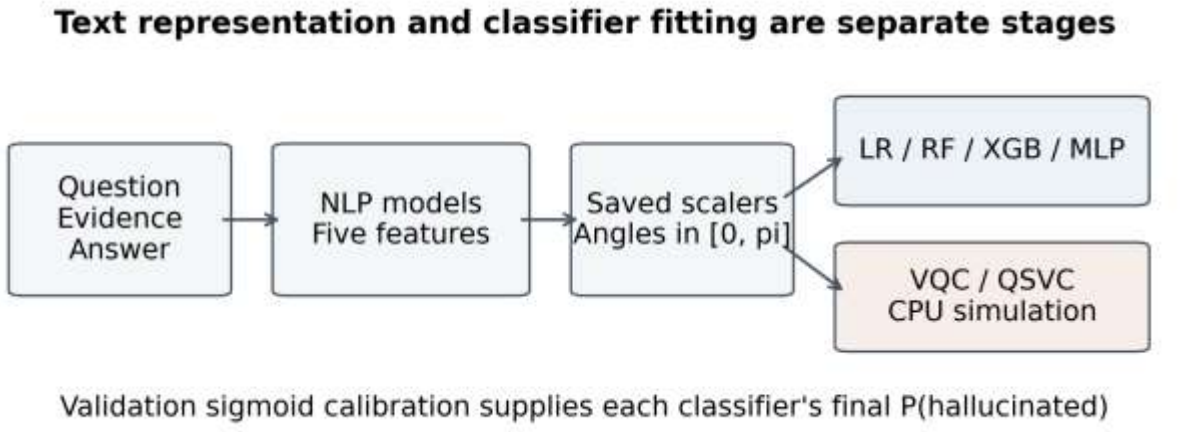
DOI: 10.17148/IJARCCE.2026.151173

fine-grained random splits at the answer level without spreading related contextual information across the split surface. For our purposes, we have chosen to partition by the normalized identity of each question.

Sentence-BERT offered an elegant solution for measuring representation similarity between sentences using cosine similarity [3]. We utilize the Sentence Encoder version released with Sentence-BERT called all-MiniLM-L6-v2 [4], instead of building a custom embedding model. To augment the basic model's outputs with another source of entailing relationships, we pair it with a DeBERTaV3-based cross-encoder [5,6]. In addition to providing us with next token probabilities locally, the large language model Qwen2.5 can optionally generate entire answers [7]; these three pre-trained models are kept frozen during our classifier experiments.

Using quantum feature spaces involves encoding classical data points into parameterized quantum states through some kind of encoding strategy. Then, these encoded vectors are used with a learned circuit (or with other ways of characterizing how much they overlap) to make decisions about labels (for example) [9,10]. There are multiple ways to implement this process, so there will be variations in resulting effectiveness depending on what strategies were adopted when doing encodings, performing classifications, optimizing parameters, etc. As Bowles et al. found out while running their own set of small simulated benchmarks, getting good results requires very carefully developed baselines along with a healthy dose of restraint about drawing too broad conclusions from those baseline comparisons [12]. As such, our paper contains information about various things that help improve reproducibility within scientific studies including things that should have been done better in terms of controlling what kinds of samples were trained against as well as computational measures meant to aid replication attempts if needed—but unfortunately also highlights areas where shortcomings exist relative to modern capabilities making up significant portions of today's state-of-the-art solutions available under commercial licenses at reasonable prices—such shortcomings being recognized even though they don't hinder actual application possibilities due primarily because many problems simply aren't amenable toward applying most forms currently known outside research contexts except perhaps those involving

Figure 1. Implemented inference architecture. Frozen language models produce five scalars. Classical and simulated quantum classifiers share the saved transformations; each has a separate validation-fitted sigmoid calibrator



.

As requested by the reviewer I am including the executable comparison with audit trail below (paired data construction, common representation, shared training subset, hold out calibration, persisted models and demo interface). Please note that "quantum" in this context refer to the classical numerical classifier post feature extraction from the input (see NLP section), rather than a qubit based language model or end-to-end quantum text representation. We make no claim of methodological precedence here, merely assemblage.

3. DATASET AND EXPERIMENTAL PROTOCOL

We access the pminervini/HaluEval via its qa configuration and the specified data-split. The loader will print the retrieved schema which maps the knowledge-field into evidence. We use the first 60 question-records for our reported run.

Each question-record contains a right-answer with label 0 and a hallucinated-answer with label 1. Therefore, we end up with 120 total answers that have balanced reference classes (as intended). Taking the beginning of the dataset allows us to avoid making claims about representativeness while still getting a reproducible dataset.

Finally, we derive identifiers for each question based on their lower-cased-text content using normalized-whitespace as separators. Answers with identical identifiers get grouped together. Finally, we split everything once more at this top

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151173

level using some fixed-proportions with an outer-split-seed of 42 (approximate training/validation/test sizes 70%/15%/15%). Given how every question-group has the exact-same composition of their paired-class, the group-level-stratification-value gets set to something static here. Lastly, when considering validation further splits occur but those relate purely to question-by-question instead.

Table 1. Actual question and answer counts. The shared subset is contained in training and is not an additional partition.

Partition	Questions	Answers	Role
Training	42	84	Estimator fitting
Tuning validation	4	8	Hyperparameter selection
Calibration validation	5	10	Sigmoid fitting
Test	9	18	Frozen-model evaluation
Shared training subset	30	60	Size-controlled comparison

Fitting these four classical families involves processing the full outer training partition and then evaluating their performance with exactly the same 60-answer subset utilized by each quantum model (which consists of whole question groups). These subsets have been saved with their respective sample identifiers. They're both processed through preprocessing which has already been fit against the entire outer training partition. Note that while this guarantees that our size-controlled estimators will see the exact same set of labeled training examples and have identical preprocessing information at train time, their model capacities and the amount they've spent on tuning are not otherwise coordinated.
Hyperparameter choices and other optional ablation options were frozen prior to test-time evaluation. We first performed the requested five-row persistence check followed by prediction of the test partition itself. This constitutes just one evaluation phase; please bear in mind that this isn't literally a single prediction call being made here, but rather our interface's preference to show the user your chosen default display model based purely upon observed test-set AUROC values, lest we risk introducing any bias via post-hoc selection.

The actual run you can reproduce is shown below with its reduced configuration—though we expose a few extra capabilities within the software. First, there exists a default 1,000-question mode as well as a 300-answer maximum for the quantum model portion of training, though nothing gleaned from using either is presented here due to space constraints. And lastly, the optional ablation budget was zero such that no trained drop-one-feature comparison is available.

For reference, the code executed steps in this sequence: load data → expand pairs → split into groups → perform frozen feature extraction → fit preprocessing → construct shared subsets → select estimators → calibrate → evaluate. As discussed previously, features could potentially be extracted for every partition in a single pass since the pretrained extractors don't learn anything from them. Also, remember that the two fitted scalers operate exclusively on training rows and nothing else. Finally, test labels are used to calculate the various final metrics and populate our explicitly demonstrated display.
This experiment used Python 3.11.14 along with numpy 2.2.6, sklearn 1.6.1, xgboost 3.0.2, qiskit 1.4.5, qiskit-machine-learning 0.8.4, and qiskit-algorithms 0.3.1. See our source appendix for the rest of the packages pinned down. Lastly, four CPU threads are configured. While initial model download sizes, the availability of caches, and unspecified CPU and memory specs make reproducing timings challenging, it is still possible under ideal conditions.
Pretrained model revisions and the dataset revision are not pinned to immutable repository commits.

4. NLP REPRESENTATION AND PREPROCESSING

The representations for (Q,E,A) contain five scalar features in fixed order, and we evaluate the encoder/inference model on CPU with gradients disabled. For feature extraction, we do not train our pretrained models against HaluEval's labels. We define the Hedge ratio h(A) as the number of answer word tokens drawn from a fixed lexicon divided by the total word-token count in A. Our lexicon consists of "maybe", "perhaps", "possibly", "might", "likely", "reportedly", and "allegedly". We set empty strings to have a finite default value.
This feature measures lexical caution but cannot distinguish between appropriate uncertainty and unsupported speculation. Thus, confident false answers need not include hedge words.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151173

Token Likelihood is the mean teacher-forced log probability under the model Qwen/Qwen2.5-0.5B-Instruct conditioned on a chat-formatted question and evidence prompt. We consider only answer targets when computing the denominator; causal positions are offset such that each token receives an independent score based on its preceding context (prompt and padding positions are omitted), using natural logarithms throughout.

$$\ell(A \mid E, Q) = \frac{1}{T} \sum_{t=1}^{T} \log P(a_t \mid E, Q, a_{<t})$$

Answer processing occurs in chunks of up to 128 tokens for each 512 token context window (longer prompts will be "left truncated"). The Bounded Window Score is an approximate measure which assumes that we're conditioning on an arbitrarily long passage. Note that this high likelihood indicates that the prompt is compatible with the local LM, but does not indicate independent verification of the factual accuracy of the answer.

The normalized MiniLM embedding of the answer is then compared with those of the evidence and question. Because the dot product equals the cosine similarity between vectors, these terms are equivalent to measuring the semantic relatedness and topical relevance of the document with respect to the query, respectively. Either may remain high if some central piece of content in either the evidence or question is false, although repeated questions and pieces of evidence are encoded only once during each extraction pass and re-used thereafter.

With the cross encoder, we take the premise as being the evidence provided and the hypothesis as being our proposed answer. Taking a softmax over these outputs gives us the probability of entailing the answer given the evidence. It's worth noting that you read off the label corresponding to this index via the id2label attribute rather assuming what value it contains. Also note that the retention of this single scalar doesn't encode whether there was a contradiction or otherwise neutral response. In other words, while the presence of one explicit indicator might drive the entailment score down, the lack thereof could just as easily leave things unchanged — and hence both options could yield very low scores depending on inputs passed through here. NLI Input & Embedding Limits Finally, all inputs to the NLI are limited by length of 512 tokens, and all embeddings are capped at a maximum size of 256 tokens.

$$z_j = \frac{x_j - \mu_j}{\sigma_j}, \qquad \alpha_j = \pi \frac{z_j - z_{j,\min}}{z_{j,\max} - z_{j,\min}}$$

Both StandardScaler and after that MinMaxScaler are fit only on the outer training data features. Values are clipped to the outer training data range prior to performing the angle encoding. In case of non-constant feature, both affine transformations collapse to min-max scaling. Thus, their usage is governed by the consistent pre-processing contract instead of adding extra expressiveness. Fingerprint of the texts and extractor parameters are used to manage the feature cache reuse.

5. CLASSICAL AND QUANTUM CLASSIFIERS

The classic estimators check the sufficiency of the crafted representation for simple linear separators or tree interactions. Logistic Regression employs a linear score with regularization. Random Forest includes 100 trees. XGBoost employs 80 histogram trees with learning rate 0.08 and CPU computations [8]. The MLP estimator has hidden layers of 16 and 8 neurons with a maximum of 400 iterations. Both saved MLP estimations reached that many iterations, but convergence is not claimed.

Table 2. Actual search protocol. Tuning uses validation AUROC; ties retain the first candidate. The full software setting allows 150 VQC evaluations, which was not used here.

Estimator	Fixed configuration	Validation choices
Logistic Regression	Maximum iterations 1,000	C in {0.1, 1.0}
Random Forest	100 trees	Depth in {4, unrestricted}
XGBoost	80 trees; learning rate 0.08	Depth in {2, 3}
MLP	Hidden layers (16, 8); 400 iterations	Alpha in {0.001, 0.1}

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151173

VQC	ZZ reps 1; ansatz reps 2; COBYLA	Fixed circuit; 25 evaluations
QSVC	ZZ reps 2; fidelity kernel	C in {0.1, 1.0}

For both classical training scenarios, selection uses C = 0.1 for Logistic Regression, depth 4 for Random Forest, depth 2 for XGBoost, and alpha = 0.001 for MLP. For QSVC, C = 0.1 remains in the base experiment. The tuning subset has just four positive and four negative responses, and picking a candidate from this subset is unreliable and doesn't provide hyperparameter ranking.

The five normalized features are mapped to the five qubits individually. The ZZ feature map performs Hadamard and phase transformations, with pair-wise phases equal to (pi – alpha_i)(pi – alpha_j) on the linear chain [14]. The VQC then combines this feature map with the RealAmplitudes ansatz. The 15 parameterized rotations are initialized randomly and are optimized via COBYLA and cross-entropy.

$$|\psi(\boldsymbol{\alpha},\boldsymbol{\theta})\rangle = U_{\mathrm{RA}}(\boldsymbol{\theta})U_{ZZ}(\boldsymbol{\alpha})|0\rangle^{\otimes 5}$$

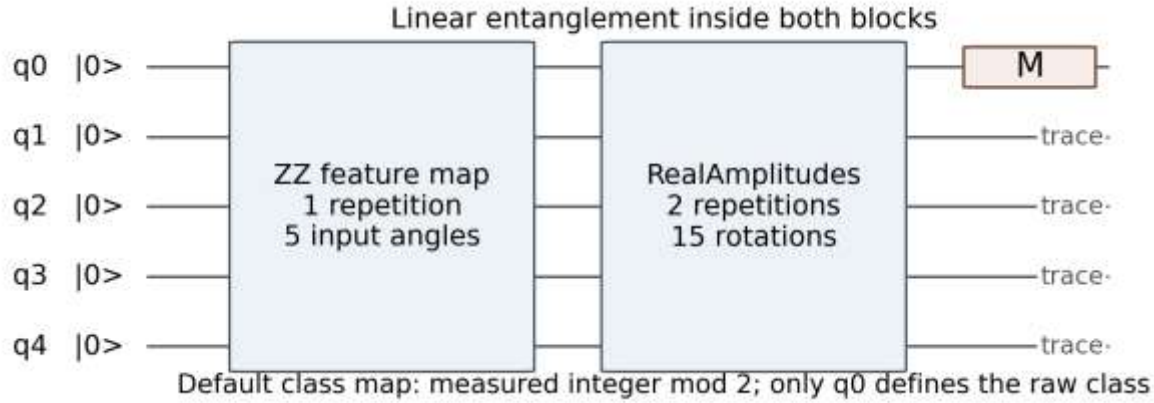


Figure 2. Schematic of the implemented VQC. The class interpretation is the easured integer modulo two, so q0 defines the raw readout. Other wires are traced in this schematic; encoding five inputs does not guarantee five-input sensitivity.

Qiskit V2 StatevectorSampler carries out CPU statevector simulation with 1,024 shots. Finite-shot simulation does not include hardware noise. Optimizer budget denotes evaluations of the objective function, not epochs. There is no updating of any parameters of the NLP model in the course of quantum optimization.

6. QUANTUM KERNEL CALIBRATION AND READOUT AUDIT

QSVC makes use of a fixed two-repetition ZZ encoding and fidelity quantum kernel listed below. Prepare/Measure approach computes overlaps through one-qubit preparation of a state, inversion of the second state and measurement of all-zero states. A classical SVM learns the decision boundary through the kernel. Positive-semidefinite correction is used for the training matrix. Matrices are cached for each of C candidates and cleared prior to timed test inference.

Calibrator fitting is performed for the ten calibration validation answers per each chosen estimator [11]. Decision functions provide the score for Logistic Regression and QSVC. The score for other models is the clipped log odds of the native class prediction output. The implementation involves the fitting of a regularized logistic mapping with a learned slope and intercept. The fixed threshold is set to 0.5. Calibration with the small balanced sample does not imply probability validity in another prevalence scenario.

The pinned VQC default maps a measured integer m to m mod 2 [13]. This selects its least significant measured bit, rather than parity over all five bits. A post-hoc diagnostic evaluates exact statevectors for five seeded input/weight configurations. Each input angle is perturbed separately by 0.15 radians; Figure 3 reports the maximum absolute change in the ideal class-1 probability across those configurations. This diagnostic does not retrain a model and is not a predictive ablation study

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151173

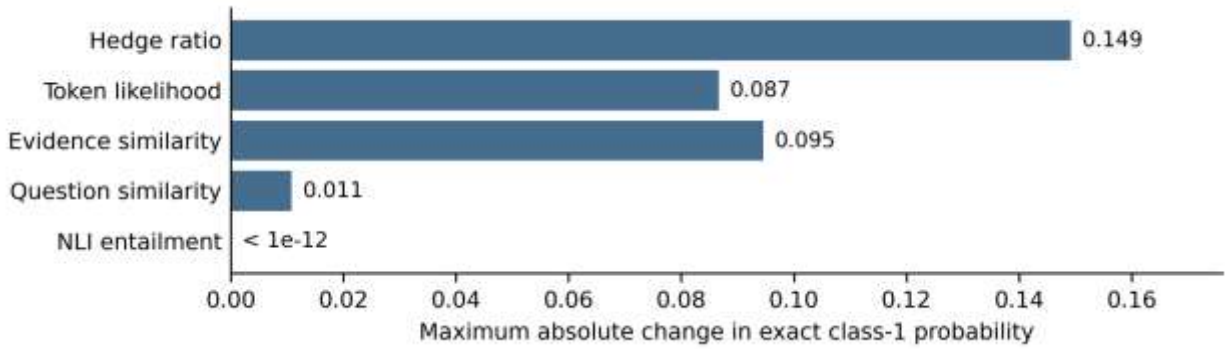


Figure 3. Exact-state sensitivity diagnostic for the current VQC. Changing the fifth feature, NLI entailment, changes the ideal readout only at floating-point roundoff. Values summarize five examined configurations; they are not feature-importance estimates on the dataset.

The fifth input variable has absolutely no effect on the diagnostic in this case, and this is due to the particular choice of circuit connectivity and readout. Thus, the presented VQC is not a clean benchmark of an effective five-variable quantum classifier. Changing readout or increasing the connectivity would demand carrying out an additional experiment. In this manuscript, we keep the VQC benchmark and state the problem, rather than blame all quantum classifiers in their failure.

7. PREDICTIVE PERFORMANCE AND CALIBRATION

Answer hallucinations belong to the positive class. Metrics include accuracy, precision, recall, F1, AUROC, and expected calibration error (ECE) on the class-probability. There are ten equal-width bins comparing the mean hallucination probability against the positive proportion. Probability one is assumed for the final bin. Sparse occupancy for ten bins with 18 answers means ECE is descriptive.

$$\mathrm{ECE} = \sum_{b=1}^{10} \frac{n_b}{N} |\bar{p}_b - \bar{y}_b|$$

Table 3. Shared-subset results with seed 42. All estimators use 60 training answers and the same 18 test answers. No interval or significance claim is attached to these point estimates.

Classifier	Accuracy	Precision	Recall	F1	AUROC	ECE
Logistic Regression	0.8889	1.0000	0.7778	0.8750	0.9753	0.1781
Random Forest	0.8889	1.0000	0.7778	0.8750	0.9630	0.0617
XGBoost	0.8889	1.0000	0.7778	0.8750	0.9506	0.0855
MLP	0.8889	1.0000	0.7778	0.8750	0.9753	0.0575
VQC	0.5000	0.5000	0.3333	0.4000	0.5309	0.0631
QSVC	0.8333	0.8750	0.7778	0.8235	0.9012	0.3211

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151173

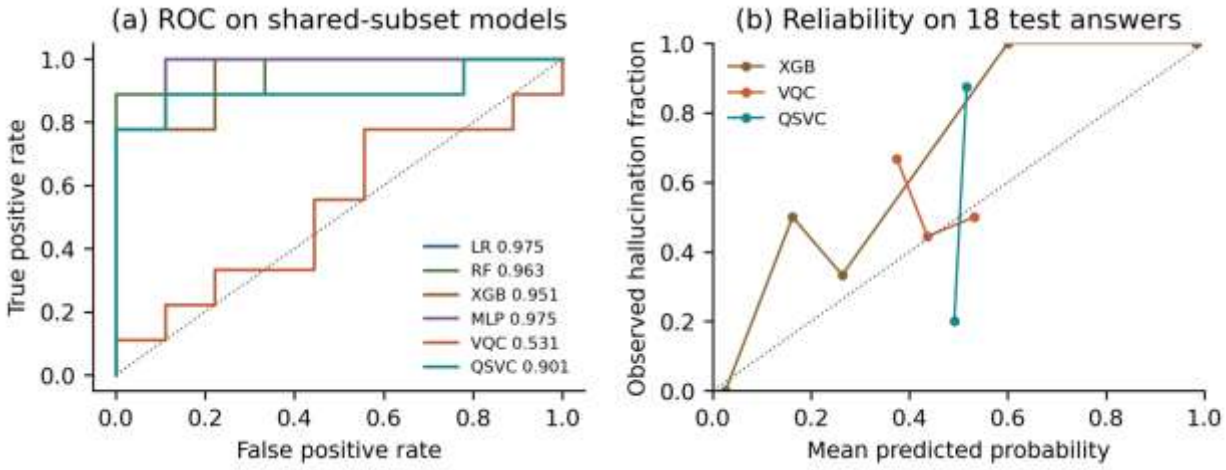


Figure 4. ROC and reliability plots computed from saved test probabilities. All curves use shared-subset models; the reliability panel shows XGBoost, VQC, and QSVC. Short segments and extreme observed fractions reflect small occupied bins.

Logistic Regression and MLP have the greatest AUROC shared by the two in the subset, 0.9753. The four classical machine learning algorithms all have seven true positives, zero false positives, two false negatives, and nine true negatives using the fixed cutoff threshold. QSVC has seven true positives and one false positive; F1 score = 0.8235. The basic VQC algorithm has three true positives and three false positives. The lower ECE value does not compensate for almost chance-level discrimination, whereby predictions around the class balance may seem calibrated but still discriminate poorly.

8. TRAINING SIZE SEED VARIATION AND RUNTIME

Table 4. Classical algorithms trained on all 84 outer-training answers. This table represents the larger training set scenario and cannot be substituted for the same subset comparison.

Classifier	Train n	Accuracy	F1	AUROC	ECE
Logistic Regression	84	0.8889	0.8750	0.9753	0.1561
Random Forest	84	1.0000	1.0000	1.0000	0.1409
XGBoost	84	0.8889	0.8750	0.9506	0.1334
MLP	84	0.8333	0.8235	0.9630	0.1207

The full training Random Forest perfectly categorizes all 18 of the test answer options and gets an AUROC of 1.0000. However, its ECE stays at 0.1409, demonstrating that the presence of right thresholded labels and calibrated probabilities are two different things. This flawless performance is achieved in only nine pairs of questions.

Table 5. Three runs of quantum machine learning on the same split and subset. SD represents the sample standard deviation across the seeds, not confidence intervals or uncertainties across data splits.

Model	Seeds	Mean accuracy	Mean F1 +/- SD	Mean AUROC +/- SD
VQC	42, 43, 44	0.5370	0.3897 +/- 0.0774	0.5556 +/- 0.0890
QSVC	42, 43, 44	0.8333	0.8235 +/- 0.0000	0.9012 +/- 0.0000

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151173

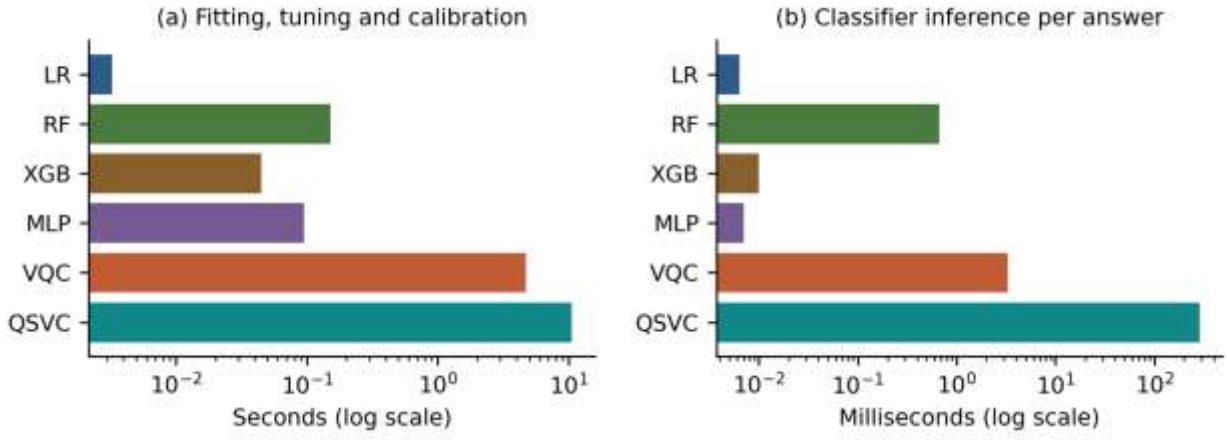


Figure 5. Timing data of the shared-subset from the most recent run. Training consists of tuning and calibration. Inference is average batched classifier runtime per answer, without NLP extraction and loading of the models.

The total quick invocation run manifest shows that the quick invocation took 78.87 seconds. Base-seed VQC fitting time is 4.6992 seconds while QSVC fitting is 10.5827 seconds. Inferences of the classifier take approximately 3.3092 and 286.1936 milliseconds per response respectively. This is a single run timing under the defined CPU execution, not a latency benchmark. The kernel needs pairwise evaluations and displays higher inference cost in this run. Hardware level quantum speedup is not measured.

9. ERROR ANALYSIS OF APPLICATION AND LIMITATIONS

The Gradio interface checks the responses, does the local generation and then verifies. In mode A, question, evidence, and answer should be non-empty. In mode B, optional Wikipedia evidence retrieval and then generation with Qwen is done greedily within 80 new tokens. Grounded generation means that the passage is included in the prompt, while closed-book generation excludes the passage, but verification still uses evidence. The interface streams a CPU processing status during generation.

For the prediction, lazy-loading occurs of six primary classifiers with their saved scalers for each completed run identifier. The features, display-scaled values, probabilities, labels and timing are returned. Four full-training classical models and the base-seed quantum models are used for providing live predictions. Majority vote will give the consensus; when there is a tie, the mean probability is used as a heuristic for correlated models with the same input features. Changing the primary model in the dropdown changes the display and not features.

In the interface, one minus the hallucination probability is called the reliability score. Its meaning is the estimated support based on the particular classifier. It is neither a guarantee of truth nor the confidence in the displayed.

We have included some representative examples here on the web site – they were obtained by taking six examples from the test partition, and then clicking through them in sequence. (You will note that we have also included references to the labels provided for each example in our respective datasets.)

In this video you can see what happens when I run these queries using my web browser. For instance, if I ask ‘What’s the capital city of Paris?’ I get back the correct answer, ‘Paris’. But then the randomly trained primary RandomForest model predicts this was hallucinated, which would represent an incorrect prediction – i.e., the system thought there was a hallucination where there wasn’t one. We have used this as a way to show how wrong things go, although we need to stress at this point that this doesn’t help us understand why our model made such a mistake. That might involve issues ranging from dataset artifacts down to calibration limitations and distribution shifts – all of which we’d need to explore separately.

But let me give you a sense of what’s happening here – here, below, you can watch me pose exactly the same question, now over Wikipedia. Note that this time around I’m getting an entirely different result.

As mentioned before, the study is limited by the fact that it sampled from the very first record and only tested nine groups of questions. This means that generalisations beyond the samples aren’t possible. In addition, since these answers are

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.151173

paired together, the entire sample size of 18 shouldn't be considered as representing 18 independent contexts. Furthermore, because we've been repeatedly developing on our data sets during the process and because we only discovered a method of selecting display models after the fact, a more pristine "hold out" analysis isn't really feasible. Finally, given the tiny size of the ten answer calibration set, we're simply not able to determine deployment probabilities with any degree of certainty. And while many machine learning papers report results using independent external datasets and grouping methods combined with cross validation and confidence intervals, none of those were implemented here, nor did any post hoc analyses reveal any significant effects across pairs.

To make a stronger claim about these five features' predictive power, we'd want to address some other defects in our current model (the VQC readout), as well as test additional classical approaches such as training an RBF SVM on top of them or building an "entailment only" detector. Omitted ablations prevent us from determining how much of the performance can be attributed to particular inputs.

And while this experiment shows that training an XGBoost gain importance with quantum kernels is possible — because it's just another statistic extracted out of a fitted model, after all — there's no way for us to know if quantum kernels actually give you something extra compared to just using more typical nonlinear functions. We didn't look at hardware noise, either, nor did we try to measure energy usage or understand how things scale up depending on available quantum resources.

10. CONCLUSION

The pipeline that was built supports local, reproducible experimentation with hallucination labels that are relative to evidence. The pipeline significantly improved the performance of classical classifiers' AUROC on the shared subset of the recorded data compared to two simulated quantum models. QSVC offers more useful discrimination on this small partition at the higher classifier cost, while the VQC result is limited due to its current readout. The pilot is working and a documented failure analysis is provided by the evidence. Before making broader predictive or quantum-advantage claims, a corrected circuit, larger grouped evaluation, stronger baselines, and independent calibration data are necessary.

Data code and reporting statements

HaluEval is a public benchmark. The editable source appendix that comes with the benchmark has ten Python modules, pinned requirements, and the study audit and figure-generation script. The project repository is https://github.com/Khusheyy/hallucination_detection. The study snapshot contains the effective configuration, saved predictions, numerical results, and a source-hash manifest. Each saved classifier went through a five-row native save/load comparison, and zero probability difference was observed. There is no reporting of any new human-subject experiment.

REFERENCES

[1] Z. Ji, X. Wu, P. C. K. Chan, S. Greenberg, and Y. Chang. Survey of Hallucination in Natural Language Generation. ACM Computing Surveys, 55(12), no. 248, 2023. Doi:10.1145/3571730.

[2] {J.} {Li, } {X.} {Cheng, } {W.} {X.} {Zhao, } {J.-Y.} {Nie, } {J.-R.} {Wen} HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. EMNLP, 2023. 6449 - 6464. Doi:10.18653/v1/2023.emnlp-main.397

[3] N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In EMNLP-IJCNLP, pages 39823992, 2019. Doi:10.18653/v1/D19-1410.

[4] Sentence Transformers. All-MiniLM-L6-v2 model card. Available at https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Accessed 1 November 2026.

[5] P. He, J. Gao, and W. Chen. DeBERTaV3: Advancing DeBERTa with ELECTRA-Style Pre-Training also Gradient-Disentangled Embedding Sharing. ArXiv:2111.09543, 2021.

[6] Cross-Encoders for Sentence Transformers. Model card for nli-deberta-v3-small. Https://huggingface.co/cross-encoder/nli-deberta-v3-small Accessed 1 November 2026

[7] Qwen et al. Qwen2.5 Technical Report. ArXiv:2412.15115, 2024.

[8] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. KDD, 2016. Doi:10.1145/2939672.2939785.

[9] V. Havlicek et al. Supervised learning with quantum-better feature spaces. Nature 567, 209-212, 2019. Doi:10.1038/s41586-019-0980-2.

[10] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum Circuit Learning. Physical Review A 98 032309, 2018. DOI: 10.1103/PhysRevA.98.032309.

[11] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. ICML, PMLR 70, pp. 1321-1330, 2017. Https://proceedings.mlr. Press/v70/guo17a.html.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151173

[12] J. Bowles, S. Ahmed and M. Schuld better than classical? The art of benchmarking quantum machine learning models, arXiv:2403.07059, 2024.

[13] The authors of Qiskit Machine Learning. VQC implementation, release 0.8.4. URL https://github.com/qiskit-community/qiskit-machine-learning/blob/0.8.4/qiskit_machine_learning/algorithms/classifiers/vqc. Py (visited on 1 November 2026).

[14] IBM Quantum. ZZFeatureMap, Qiskit SDK 1.4 API reference. Https://quantum.cloud.ibm.com/docs/en/api/qiskit/1.4/qiskit. Circuit. Library. ZZFeatureMap. Viewed 1 November 2026.

 This work is licensed under a Creative Commons Attribution 4.0 International License