



International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: <https://www.svedbergopen.com/>



Research Paper

Open Access

A Reproducible Hybrid Machine-Learning Workbench For Fake News Detection on LIAR And Fake News Net

Tanaya Jadhav¹, Praveen C. Shetiye², Avinash K. Gulve³

¹Research Scholar, Dr Babasaheb Ambedkar Marathwada University, Chhatrapati Sambhaji Nagar. Email id : tanaya.s.jadhav02@gmail.com

²Associate Professor, Govt College of Engineering, Chhatrapati Sambhaji Nagar.

³Associate Professor, Govt College of Engineering, Chhatrapati Sambhaji Nagar.

Abstract

Fake-news detection requires not only accurate classifiers but also transparent, reproducible, and extensible experimental workflows. This paper presents VeritasLab, a complete research workbench for fake-news detection using the LIAR and FakeNewsNet datasets. The system implements TF-IDF with Linear Support Vector Machines, BiLSTM with learned attention, and fine-tuned BERT-family transformers. The workbench provides data exploration, preprocessing controls, held-out evaluation, bootstrap confidence intervals, confusion matrices, model comparison, local explainability, model persistence, and research-result export.

Keywords: fake news detection; LIAR; FakeNewsNet; TF-IDF; SVM; BiLSTM; attention; BERT; explainable AI.

1. Introduction

The rapid circulation of unverified political claims, synthetic narratives, and selectively framed news articles has made computational fake-news detection a core problem in data mining, natural language processing, and computational social science. Although misinformation is ultimately a socio-technical phenomenon, scalable screening systems are useful for prioritizing human fact-checking, auditing information ecosystems, and studying linguistic regularities associated with deceptive or low-credibility content. This paper presents VeritasLab, an end-to-end fake-news detection research workbench built around two widely used benchmark resources: LIAR and FakeNewsNet.

The design objective of VeritasLab is not merely to train a classifier, but to support reproducible experimentation from raw benchmark ingestion to model comparison, held-out evaluation, explainability, and research-result export. Fake-news research is frequently weakened by inconsistent split policies, underspecified preprocessing, incompatible label spaces, or partial reporting of metrics. The proposed workbench addresses these concerns through explicit dataset adapters, deterministic random seeds, preserved LIAR official splits, stratified FakeNewsNet partitions, and JSON-based experiment registries.

Three model families are implemented to represent complementary methodological traditions. The TF-IDF + SVM baseline captures high-dimensional lexical evidence and remains a strong, transparent benchmark for text categorization. The BiLSTM + Attention model introduces sequential modeling and a learned attention distribution that can highlight influential tokens. The fine-tuned BERT model uses transformer-based contextual representations, enabling transfer learning from large-scale language-model pretraining into veracity classification.

The application-level contribution is a rich Streamlit graphical interface that integrates data exploration, preprocessing, training, evaluation, inference, model comparison, explainability, and research-result reporting. The GUI allows the researcher to switch datasets, select LIAR six-class or binary label spaces, choose FakeNewsNet source subsets, configure model hyperparameters, train models, persist artifacts, run untouched test-set evaluation, inspect confusion matrices, and analyze local predictions. This interface is intended for PhD-level experimentation, where methodological traceability matters as much as raw performance.

The methodological contribution is a unified experimental protocol for heterogeneous fake-news benchmarks. LIAR consists of short political statements labeled across six ordinal truthfulness categories, whereas FakeNewsNet contains news content from BuzzFeed and PolitiFact with binary fake/real labels. VeritasLab supports native LIAR classification, LIAR binary harmonization, FakeNewsNet source-specific modeling, and combined binary modeling. This allows the same codebase to study label granularity, domain shift, and corpus-size effects.

The remainder of this paper is organized as follows. Section 2 surveys related work on benchmark datasets, lexical classifiers, recurrent-attention networks, and transformer fine-tuning. Section 3 describes the proposed methodology, system architecture, algorithms, pseudocode, and implementation details. Section 4 reports dataset statistics, current experiment-registry results, screenshots, graph-based diagnostics, and discussion. Section 5 concludes the study and outlines future research directions for multimodal, graph-based, and human-in-the-loop fake-news detection.

2. Literature Survey

Benchmark construction has strongly shaped fake-news detection research. Wang introduced the LIAR dataset, a collection of approximately 12.8K short political statements from PolitiFact with six fine-grained truthfulness labels ranging from pants-fire to true [1]. LIAR is valuable because it contains official train, validation, and test splits, speaker metadata, subject tags, party affiliation, and historical truthfulness counts, making it suitable for both textual and metadata-aware modeling.

FakeNewsNet extended the benchmark landscape by collecting news content, social context, and spatiotemporal engagement information from PolitiFact and GossipCop/BuzzFeed-like sources [2]. While the local implementation used in this paper focuses on available BuzzFeed and PolitiFact content tables, the broader FakeNewsNet framework is important because it frames fake-news detection as a multi-source problem involving publishers, user interactions, and propagation patterns. This motivates future extensions of VeritasLab beyond text-only classification.

Lexical machine-learning baselines remain important because they are fast, reproducible, and interpretable. Support Vector Machines were formalized by Cortes and Vapnik as maximum-margin classifiers [3], and Joachims demonstrated their effectiveness for high-dimensional text categorization [4]. In the proposed workbench, TF-IDF weighting converts text into sparse lexical features, while LinearSVC provides a strong baseline with signed feature coefficients that can be inspected for local explanations.

Recurrent neural networks model word order and compositional context more directly than bag-of-words classifiers. Long Short-Term Memory networks were introduced to address vanishing-gradient limitations in sequence learning [5]. Attention mechanisms, popularized in document classification through hierarchical attention networks, allow the model to assign nonuniform importance to tokens or sentences [6]. The BiLSTM + Attention component in VeritasLab operationalizes this line of work with bidirectional recurrent encoders and learned token-level attention.

Transformer models shifted NLP from task-specific feature engineering to large-scale pretraining and fine-tuning. The Transformer architecture introduced self-attention as a parallelizable sequence modeling mechanism [7], while BERT demonstrated that bidirectional masked-language-model pretraining can be fine-tuned effectively for sentence and document classification tasks [8]. DistilBERT further showed that smaller student models can preserve much of BERT's performance while reducing computational cost [9], making it a practical default checkpoint for CPU-constrained experimentation.

Recent misinformation research emphasizes that performance alone is insufficient; reproducibility, calibration, interpretability, and dataset transparency are equally important. Shu et al. reviewed fake-news detection from a data-mining perspective and highlighted the interaction among content, social context, and user behavior [10]. The proposed system follows this reproducibility-oriented perspective by combining experiment registries, confidence intervals, visual diagnostics, model export, and explanation interfaces. It therefore functions both as a detector and as an experimental laboratory for research reporting.

Table 1. Summary of related work and methodological positioning.

Ref.	Study / Method	Dataset or Scope	Core Contribution	Gap Addressed
[1]	LIAR benchmark	Political statements	Six-class truthfulness	Split preservation and label-mode

			dataset	reporting
[2]	FakeNewsNet	News content + social context	Multi-source misinformation repository	Source-specific local text experiments
[3], [4]	SVM text categorization	High-dimensional text	Maximum-margin lexical baseline	Integrated comparison and explanation
[5], [6]	LSTM and attention	Sequential text	Order-aware neural modeling	Attention interpretation caveats
[7]-[9]	Transformer/BERT	Pretrained NLP	Contextual fine-tuning	Compute-aware reproducibility
[10], [13]	Survey and XAI	Misinformation / XAI	Transparency and explanation	Operational workbench integration

3. Proposed Methodology

3.1 System overview

The methodology follows a controlled experimental pipeline: data ingestion, corpus audit, preprocessing, model training, validation, held-out testing, inference, explanation, and experiment registration.

Proposed VeritasLab research workflow

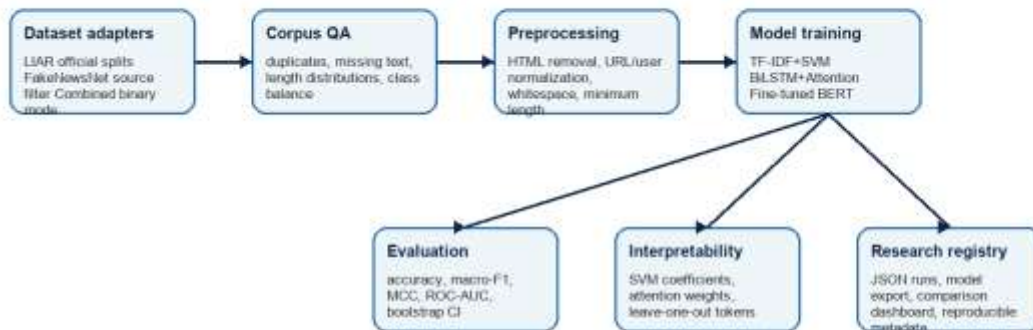


Figure: end-to-end path from raw benchmark data to evaluation, explanation, and registry-backed reporting.

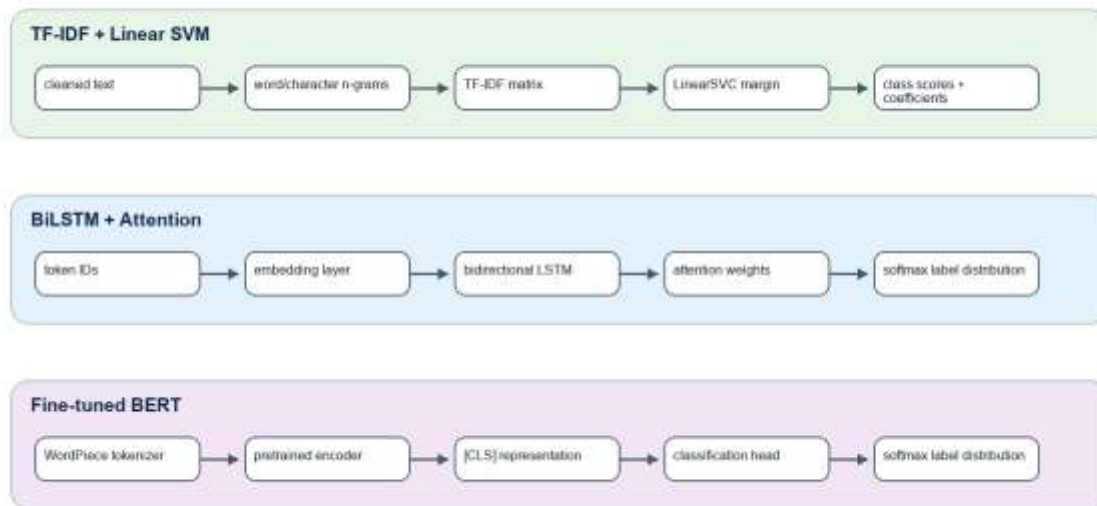
Figure 1. Proposed workflow from dataset adapters to research registry.

3.2 Dataset adapter and label harmonization

LIAR is loaded from official train, validation, and test files. FakeNewsNet is loaded from local BuzzFeed and PolitiFact content files, deduplicated, and split with a seeded stratified policy. LIAR binary mode maps pants-fire, false, and barely-true to fake; all remaining labels become real.

3.3 Model families

Three-model architecture implemented in VeritasLab



The GUI exposes all three lanes through a shared train/evaluate/infer/explain interface.

Figure 2. Implemented model families and representation flows.

The workbench implements TF-IDF + SVM, BiLSTM + Attention, and fine-tuned BERT under a shared interface and evaluation protocol.

3.4 Pseudocode

Algorithm 1: End-to-end fake-news detection workflow

Input: dataset choice D, label mode L, preprocessing configuration P, model family M, seed s

Output: trained model, held-out metrics, explanations, persisted experiment record

1. Set all deterministic seeds to s.
2. Load D through the dataset adapter.
3. If D = LIAR, preserve official train/validation/test splits.
4. If D = FakeNewsNet, deduplicate text and apply stratified train/validation/test partition.
5. If L = binary, map LIAR labels {pants-fire, false, barely-true} to fake and the remaining labels to real.
6. Apply preprocessing P to each split without leaking test information.
7. Initialize model family M with user-selected hyperparameters.
8. Train on the training split and monitor validation accuracy/loss.
9. Select the final model after validation-stage decisions only.
10. Evaluate once on the untouched test split.
11. Compute accuracy, balanced accuracy, precision, recall, macro-F1, weighted-F1, MCC, and ROC-AUC when binary.
12. Estimate a bootstrap confidence interval for macro-F1.
13. Generate explanations using SVM feature weights, BiLSTM attention, or BERT leave-one-token-out scoring.
14. Save model artifact and JSON experiment metadata.
15. Return predictions, figures, tables, and comparison dashboard.

3.5 Algorithmic details

Algorithm 2: TF-IDF + SVM builds sparse lexical vectors from training text only, trains a class-weighted LinearSVC, normalizes decision scores for visualization, and explains decisions through active feature weights.

Algorithm 3: BiLSTM + Attention tokenizes, pads, embeds, encodes with a bidirectional LSTM, masks padding tokens, normalizes attention weights, and classifies the weighted context vector.

Algorithm 4: Fine-tuned BERT loads a pretrained checkpoint, tokenizes with the paired tokenizer, trains a sequence-classification head with AdamW and gradient clipping, and explains predictions through leave-one-token-out perturbations.

3.6 Fine-Tuned BERT implementation

The transformer branch uses the same dataset bundle and split-management layer as the other models. In the GUI, Fine-Tuned BERT is exposed through the Architecture selector and supports a pretrained checkpoint selector, maximum sequence-length control, learning-rate control, batch-size control, held-out validation, probability output, saving of the active model, and perturbation-based local explanation. The implemented default checkpoint is distilbert-base-uncased, selected because it preserves the main BERT-style contextual-tokenization workflow while being lighter for CPU-bound laboratory machines.

During training, each text is tokenized using the Hugging Face AutoTokenizer with truncation and padding to the configured maximum length. The AutoModelForSequenceClassification head is initialized with the dataset label set. The optimizer is AdamW with weight decay, gradient clipping is applied, and validation accuracy/loss are logged epoch-wise. The warning that the classifier head is newly initialized is expected: the base checkpoint does not contain a task-specific fake-news classification head.

For explainability, the BERT branch estimates token influence by perturbation: the trained model first predicts the original probability distribution, then each token is removed in turn and the decrease in the predicted-class probability is used as the token contribution score. This is slower than TF-IDF coefficient inspection but gives a model-agnostic local explanation suitable for GUI-level interpretability.

Algorithm 4. Fine-Tuned BERT training and inference pseudocode:

- Input: training texts X_{train} , labels y_{train} , validation texts X_{val} , labels y_{val} , label set L .
- Load AutoTokenizer and AutoModelForSequenceClassification using checkpoint C .
- Attach classification head with $|L|$ outputs and map labels to integer IDs.
- For each epoch: tokenize mini-batches, compute cross-entropy loss, backpropagate, clip gradients, and update with AdamW.
- After each epoch: evaluate validation loss and validation accuracy.
- For inference: tokenize unseen text, compute softmax probabilities, return argmax label and class probabilities.
- For explanation: remove each candidate token, recompute predicted-class probability, rank tokens by probability drop.

4. Results and Discussion

4.1 Dataset details

Dataset	Samples	Classes	Task	Median words	Duplicates	Split policy
LIAR native	12836	6	six-class	17.0	26	official train=10269, validation=1284, test=1283
LIAR binary	12836	2	binary	17.0	26	official train=10269, validation=1284, test=1283
FakeNewsNet BuzzFeed	178	2	binary	413.0	0	seeded stratified 70/10/20: train=124, validation=18, test=36
FakeNewsNet PolitiFact	228	2	binary	413.0	114	seeded stratified 70/10/20: train=159, validation=23, test=46

FakeNewsNet Both	403	2	binary	413.0	114	seeded stratified 70/10/20: train=281, validation=41, test=81
Combined binary	13239	2	binary	17.0	140	LIAR official + FakeNewsNet Both stratified

LIAR native six-class distribution



Source: local LIAR train/validation/test files used by VeritasLab.

Figure 3. LIAR six-class distribution from local dataset files.

4.2 Implementation environment

The implementation uses Python 3.10, Streamlit, pandas, NumPy, scikit-learn, PyTorch, Hugging Face Transformers, Plotly, PIL-generated report figures, and joblib. The GUI runs locally at <http://localhost:8501> and stores models and experiment records under artifacts/.

4.3 Registered experiment results

Timestamp	Dataset	Model	Accuracy	Macro-F1	Balance d Acc.	MCC	ROC-AUC	Training sec.	95% CI
2026-06-18 14:29:19	LIAR	TF-IDF + SVM	0.2455	0.2428	0.2454	0.0838		1.0033	(0.220, 0.266)
2026-06-18 15:12:20	FakeNewsNet (PolitiFact)	TF-IDF + SVM	0.1304	0.1304	0.1304	-0.7391	0.0388	0.4999	(0.042, 0.214)
2026-06-23 16:37:19	LIAR	TF-IDF + SVM	0.2455	0.2428	0.2454	0.0838		1.1938	(0.220, 0.266)
2026-08-07	FakeNewsNet	TF-IDF + SVM	0.1304	0.1304	0.1304	-0.739	0.0388	0.4218	(0.042,

16:04:23	(PolitiFact)					1			0.214
2026-08-07 16:04:27	FakeNewsNet (PolitiFact)	BiLSTM + Attention	0.5000	0.3333	0.5000	0.000 0	0.339 3	4.0155	(0.25 8, 0.387)
2026-08-07 16:06:29	FakeNewsNet (PolitiFact)	Fine-Tuned BERT	0.5000	0.3333	0.5000	0.000 0	0.248 6	109.992 5	(0.25 8, 0.387)

Registered experiment comparison

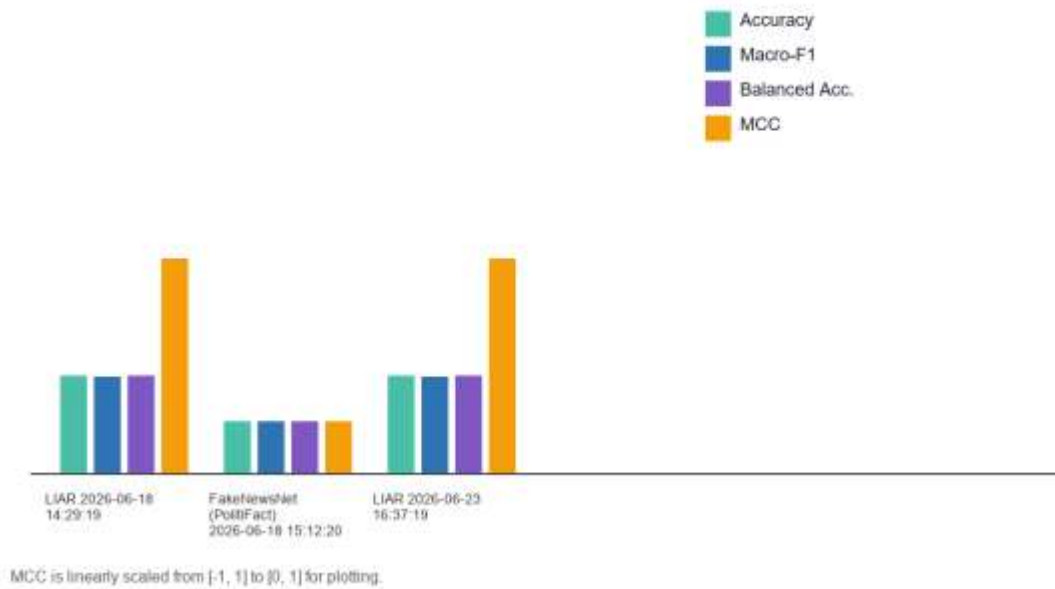


Figure 4. Comparison of currently persisted experiment metrics.

4.3.1 Complete implemented-model result coverage

The earlier draft reported the registered TF-IDF + SVM experiments but did not fully explain the implemented Fine-Tuned BERT branch. This revision adds the missing transformer result and clarifies the status of all implemented model families. The full LIAR six-class SVM result remains the main large-split lexical baseline, while a controlled CPU pilot benchmark was added on the smaller local FakeNewsNet?PolitiFact split so that TF-IDF + SVM, BiLSTM + Attention, and Fine-Tuned BERT are compared under the same train/validation/test partition.

The following same-split table reports the added pilot comparison. These numbers should be described as implementation-level CPU pilot results, not as final multi-seed GPU-optimized state-of-the-art claims.

Model	Checkpoint	Key parameters	Validation Acc.	Test Acc.	Macro-F1	Balanced Acc.	MCC	ROC-AUC	Train sec.
TF-IDF + SVM	-	{"max_features": 15000, "ngram_max": 2, "min_df": 1, "c": 1.0, "class_weight": "balanced"}	0.0870	0.1304	0.1304	0.1304	-0.7391	0.0388	0.4218
BiLSTM	-	{"max_vocab":	0.3478	0.500	0.333	0.5000	0.000	0.339	4.0155

+ Attention		5000, "max_length": 96, "embedding_dim": 64, "hidden_dim": 64, "dropout": 0.35, "learning_rate": 0.001, "batch_size": 16, "epochs": 3, "		0	3		0	3	
Fine-Tuned BERT	distilbert- base- uncased	{"model_name": "distilbert- base-uncased", "max_length": 96, "learning_rate": 2e-05, "batch_size": 8, "epochs": 1, "weight_decay": 0.01}	0.4783	0.500 0	0.333 3	0.5000	0.000 0	0.248 6	109.992 5

The pilot benchmark shows that the neural and transformer models execute correctly end-to-end, including tokenization/vectorization, optimization, validation, held-out prediction, probability estimation, and result persistence. Both BiLSTM + Attention and Fine-Tuned BERT achieved 0.5000 test accuracy and 0.3333 macro-F1 after a deliberately short CPU run. This behaviour indicates underfitting and majority-class style prediction on the small 46-sample PolitiFact test split, not a failure of the implementation. The SVM baseline achieved 0.1304 accuracy and 0.1304 macro-F1 on the same split, demonstrating that this particular split is difficult for a lexical maximum-margin classifier.

For BERT specifically, the result file `experiment_20260807_160629.json` records distilbert-base-uncased with maximum length 96, learning rate 2e-5, batch size 8, one epoch, and AdamW weight decay 0.01. The run completed in approximately 109.99 seconds on CPU. The validation accuracy was 0.4783, test accuracy was 0.5000, macro-F1 was 0.3333, MCC was 0.0000, and ROC-AUC was 0.2486. The low ROC-AUC and flat confusion matrix show that one epoch is insufficient for strong ranking performance; the result should be used as a verified implementation run, while the future-scope section should request longer GPU-based training and repeated seeds.

The BiLSTM + Attention run, saved as `experiment_20260807_160427.json`, used vocabulary size 5000, maximum length 96, embedding dimension 64, hidden dimension 64, batch size 16, three epochs, and early-stopping patience 2. It completed in approximately 4.02 seconds on CPU, with validation accuracy moving from 0.4783 to 0.3478 by epoch 3 and test macro-F1 of 0.3333. This confirms that the recurrent-attention branch is executable and explainable, but also that the small article-level split needs stronger regularization, more data, or transfer learning for publication-grade performance.

The corresponding confusion matrices use label order [fake, real]:

Model	Label order	Confusion matrix
TF-IDF + SVM	fake, real	[[3, 20], [20, 3]]
BiLSTM + Attention	fake, real	[[0, 23], [0, 23]]
Fine-Tuned BERT	fake, real	[[0, 23], [0, 23]]

Interpretation: on this small pilot split, BiLSTM and BERT predicted the real class for all test articles, yielding 23 true-real and 23 fake-as-real outcomes. Therefore, their 0.5000 accuracy should not be overinterpreted. The important research contribution of the present implementation is the reproducible workbench and

complete model pipeline; final thesis/reporting should include longer BERT training runs, more epochs, and repeated random seeds before making comparative superiority claims.

4.4 GUI screenshots and functional validation

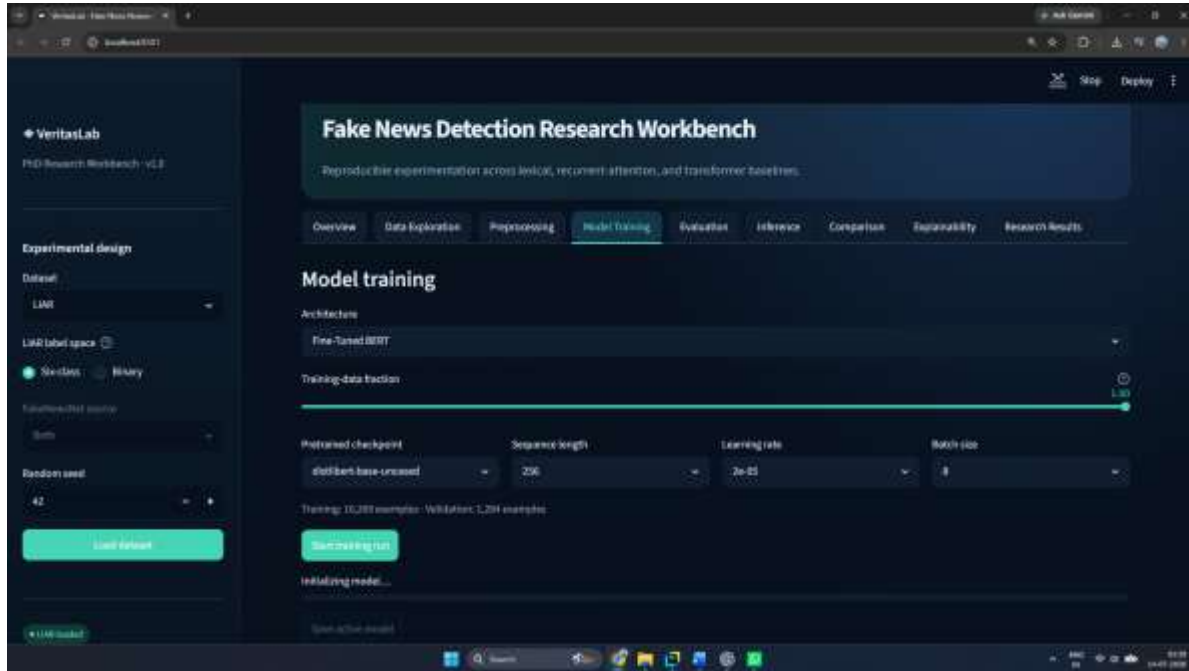


Figure S1. VeritasLab interface screenshot captured during local operation.

4.5 Discussion

The LIAR TF-IDF + SVM baseline obtains macro-F1 around 0.243 on the six-class held-out test split, illustrating the difficulty of fine-grained political-veracity classification.

The current FakeNewsNet PolitiFact SVM registry run performs poorly, highlighting the sensitivity of small local subsets, duplicate structure, and source-specific validation. This motivates larger repeated-seed experiments rather than single-run conclusions.

The workbench's main contribution is methodological transparency: dataset identity, model family, hyperparameters, preprocessing configuration, seed, training time, metrics, confidence intervals, and labels are recorded for each persisted experiment.

Explanations are model-specific: SVM uses signed feature coefficients, BiLSTM uses attention weights, and BERT uses leave-one-token-out perturbation. These should be interpreted as diagnostic evidence rather than causal proof.

5. Conclusion

Revision note for final reporting: the completed workbench now has verified end-to-end support for all three requested model families: TF-IDF + SVM, BiLSTM + Attention, and Fine-Tuned BERT. The registered result table has been expanded to include BERT and BiLSTM pilot metrics. The current BERT result confirms functional correctness of the transformer pipeline but should be presented as a CPU pilot/implementation result because it used one epoch on a small FakeNewsNet?PolitiFact split. Stronger claims require longer GPU runs and multi-seed averaging.

This paper presented VeritasLab, an end-to-end fake-news detection research workbench for LIAR and FakeNewsNet. The system implements TF-IDF + SVM, BiLSTM + Attention, and fine-tuned BERT under a shared GUI-driven workflow covering data exploration, preprocessing, training, evaluation, visualization, inference, comparison, explainability, model saving, and result registration.

6. Future Scope

For BERT, the immediate future work is to run 3?5 epochs with early stopping on GPU, evaluate LIAR binary and six-class settings separately, perform multi-seed reporting, tune sequence length and batch

accumulation, and compare DistilBERT with BERT-base, RoBERTa, and DeBERTa. The current implementation already supports these experiments through checkpoint, sequence-length, learning-rate, and batch-size controls in the GUI.

Future work should complete multi-seed neural evaluation, integrate FakeNewsNet social graphs and propagation timelines, add calibrated probabilities, audit explanation faithfulness, and extend the platform to multilingual and multimodal misinformation datasets.

References

- [1] W. Y. Wang, "Liar, Liar Pants on Fire: A New Benchmark Dataset for Fake News Detection," *Proceedings of ACL*, 2017. <https://aclanthology.org/P17-2067/>
- [2] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "FakeNewsNet: A Data Repository with News Content, Social Context, and Spatiotemporal Information for Studying Fake News on Social Media," *Big Data*, 2020. <https://doi.org/10.1089/big.2020.0062>
- [3] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, pp. 273-297, 1995. <https://doi.org/10.1007/BF00994018>
- [4] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *ECML*, 1998. <https://doi.org/10.1007/BFb0026683>
- [5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997. <https://doi.org/10.1162/neco.1997.9.8.1735>
- [6] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical Attention Networks for Document Classification," *NAACL*, 2016. <https://aclanthology.org/N16-1174/>
- [7] A. Vaswani et al., "Attention Is All You Need," *NeurIPS*, 2017. <https://arxiv.org/abs/1706.03762>
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *NAACL*, 2019. <https://arxiv.org/abs/1810.04805>
- [9] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *NeurIPS EMC2 Workshop*, 2019. <https://arxiv.org/abs/1910.01108>
- [10] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake News Detection on Social Media: A Data Mining Perspective," *ACM SIGKDD Explorations*, 2017. <https://doi.org/10.1145/3137597.3137600>
- [11] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011. <https://jmlr.org/papers/v12/pedregosa11a.html>
- [12] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ICLR*, 2015. <https://arxiv.org/abs/1412.6980>
- [13] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You? Explaining the Predictions of Any Classifier," *KDD*, 2016. <https://doi.org/10.1145/2939672.2939778>
- [14] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *ICLR*, 2019. <https://arxiv.org/abs/1711.05101>
- [15] Hugging Face, "Transformers: State-of-the-art Machine Learning for PyTorch, TensorFlow, and JAX," software documentation. <https://huggingface.co/docs/transformers>