The Journal of Open Source Software

LabelFusion: Learning to Fuse LLMs and Transformer Classifiers for Robust Text Classification

Christoph Weisser[1,2] **and LabelFusion contributors**[1]

1 Campus-Institut Data Science, Göttingen, Germany **2** Centre for Statistics, Georg-August-Universität Göttingen, Germany

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

LabelFusion is a fusion ensemble for text classification that learns to combine a traditional transformer-based classifier (e.g., RoBERTa) with one or more Large Language Models (LLMs such as OpenAI GPT, Google Gemini, or DeepSeek) to deliver accurate and cost-aware predictions across multi-class and multi-label tasks. The package provides a simple high-level interface (`AutoFusionClassifier`) that trains the full pipeline end-to-end with minimal configuration, and a flexible API for advanced users. Under the hood, LabelFusion takes vector signals from an ML backbone (logits) and LLM(s) (per-class scores), calibrates them using temperature scaling and isotonic regression, and feeds their concatenation into a small multi-layer perceptron (`FusionMLP`) that is trained to produce the final prediction. This learned fusion approach captures complementary strengths of LLM reasoning and transformer efficiency, yielding robust performance across domains—achieving 92.4% accuracy on AG News topic classification—while enabling practical trade-offs between accuracy, latency, and cost.

Statement of Need

Modern text classification spans diverse scenarios—from sentiment and topic tagging to policy enforcement and routing—often under constraints that vary per deployment (throughput, cost ceilings, data privacy). While transformer classifiers such as BERT/RoBERTa achieve strong supervised performance (Devlin et al., 2018; Liu et al., 2019), frontier LLMs can excel in low-data, ambiguous, or cross-domain settings (OpenAI, 2023). No single model family is uniformly best: LLMs are powerful yet comparatively costly and rate-limited, whereas fine-tuned transformers are efficient but may struggle with out-of-distribution cases.

LabelFusion addresses this gap by: (1) exposing a minimal "AutoFusion" interface that trains a learned combination of an ML backbone and one or more LLMs; (2) supporting both multi-class and multi-label classification; (3) providing calibration of LLM scores and ML logits for better probability estimates; and (4) integrating cleanly with existing ensemble utilities. Researchers and practitioners can therefore leverage LLMs where they add value while retaining the speed and determinism of transformer models.

State of the Field

Ensembles improve robustness by aggregating diverse predictors (Dietterich, 2000; Hansen & Salamon, 1990). Mixture-of-experts approaches further specialize components and learn to combine their outputs (Jacobs et al., 1991). In applied NLP, common tools such as scikit-learn (Pedregosa et al., 2011) and Hugging Face Transformers (Wolf et al., 2019) offer strong baselines but do not provide a turnkey, learned fusion of LLMs with supervised transformers. Orchestration frameworks (e.g., LangChain) focus on tool use rather than classification ensembles. LabelFusion contributes a focused, production-minded implementation of a small learned combiner that operates on calibrated per-class signals from both model families.

The Journal of Open Source Software

Functionality and Design

LabelFusion consists of three layers:

- ML component: a RoBERTa-style classifier produces per-class logits for input texts.
- LLM component(s): provider-specific classifiers (OpenAI, Claude, Gemini, DeepSeek) return per-class scores via prompting. Scores are cached to minimize API calls and cost.
- Fusion component: a compact MLP concatenates ML logits and LLM scores and outputs fused logits. The ML backbone is trained/fine-tuned with a small learning rate; the fusion MLP uses a higher rate, enabling rapid adaptation without destabilizing the encoder.

Key features:

- **Multi-class and multi-label support** with consistent data structures and unified training pipeline.
- **Calibration of model signals** using temperature scaling and isotonic regression (Guo et al., 2017; Zadrozny & Elkan, 2002) for better probability estimates. LLM scores are calibrated on validation data before fusion training.
- **LLM response caching** with disk-based persistence reduces API costs by storing and reusing predictions. Cache invalidation handles configuration changes automatically.
- **Batched scoring** processes multiple texts efficiently with configurable batch sizes for both ML tokenization and LLM API calls.
- **Results management** via `ResultsManager` tracks experiments, stores predictions, computes metrics, and enables reproducible research workflows.
- **Flexible interfaces**: Command-line training via `train_fusion.py` with YAML configs for research; or minimal AutoFusion API for quick deployment.
- **Composable design**: LabelFusion can serve as a strong base learner in higher-level ensembles (e.g., voting/weighted combinations of multiple fusion models).

Minimal Example (AutoFusion)

```python
from textclassify import AutoFusionClassifier

config = {
    'llm_provider': 'deepseek',
    'label_columns': ['positive', 'negative', 'neutral']
}

clf = AutoFusionClassifier(config)
clf.fit(train_dataframe)              # trains ML backbone, gathers LLM scores, fits fus
pred = clf.predict(["This is amazing!"])  # fused prediction
```

CLI and Configuration

Users can generate a starter config and train via the command line:

- Create config: `python train_fusion.py --create-config fusion_config.yaml`
- Train: `python train_fusion.py --config fusion_config.yaml`
- Optional test data and output artifacts are also supported.

Quality Control

The repository includes unit and integration tests (see `tests/`) that validate configuration handling, core types, and package integration. Fusion-specific logic is exercised in examples and the CLI, which run end-to-end training with deterministic seeds where applicable.

Evaluation scripts (`tests/evaluation/`) provide comprehensive benchmarking on standard datasets: - **AG News** (Zhang et al., 2015): 4-class topic classification with experiments across varying training data sizes (20%–100%) - **GoEmotions** (Demszky et al., 2020): 28-class multi-label emotion classification for validating multi-label fusion performance

LLM scoring paths implement retries and disk caching; transformer training supports standard sanity checks (overfit a small batch, reduced batch sizes for constrained hardware). Metrics (accuracy/F1, per-label scores) are computed automatically and stored with run artifacts to facilitate regression tracking and reproducibility.

Availability and Installation

LabelFusion is distributed as part of the `textclassify` package under the MIT license and is available at https://github.com/DataandAIReseach/LabelFusion. The fusion components require Python 3.8+ and common scientific Python dependencies (PyTorch, transformers, scikit-learn, numpy, pandas, PyYAML). Optional plotting depends on matplotlib/seaborn. Installation and quick-start snippets are provided in the README and `FUSION_README.md`.

Production-Ready Features

Beyond the core fusion methodology, LabelFusion includes features for practical deployment:

- **LLM Response Caching**: Automatic caching of LLM predictions to disk reduces API costs and enables reproducible experiments. The cache system handles invalidation and supports multiple cache backends.
- **Results Management**: Built-in `ResultsManager` tracks experiments, stores predictions, and computes metrics automatically. Supports comparison across runs and configuration tracking.
- **Batch Processing**: Efficient batched scoring of texts with configurable batch sizes for both ML and LLM components.
- **Cost Monitoring**: Tracks API usage and estimated costs across LLM providers with configurable budget limits.

Impact and Use Cases

Empirical Performance

LabelFusion has been evaluated on standard benchmark datasets to validate its effectiveness. Key findings demonstrate consistent improvements over individual model components:

AG News Topic Classification

Evaluation on the AG News dataset (Zhang et al., 2015) (4-class topic classification) with 5,000 test samples shows:

Training Data	Model	Accuracy	F1-Score	Precision	Recall
20% (800)	**Fusion**	**92.2%**	**0.922**	0.923	0.922
20% (800)	RoBERTa	89.8%	0.899	0.902	0.898
20% (800)	OpenAI	84.4%	0.844	0.857	0.844
40% (1,600)	**Fusion**	**92.2%**	**0.922**	0.924	0.922
40% (1,600)	RoBERTa	91.0%	0.911	0.913	0.910
40% (1,600)	OpenAI	84.4%	0.844	0.857	0.844
100% (4,000)	**Fusion**	**92.4%**	**0.924**	0.926	0.924
100% (4,000)	RoBERTa	92.2%	0.922	0.923	0.922
100% (4,000)	OpenAI	84.4%	0.844	0.857	0.844

Key Observations: - Fusion consistently outperforms individual models across all training data sizes - With only 20% training data, Fusion achieves 92.2% accuracy—matching its performance with full data - Demonstrates superior **data efficiency**: fusion learning extracts maximum value from limited examples - RoBERTa alone requires 100% of data to approach Fusion's 20% performance - LLM (OpenAI) shows stable but lower performance, highlighting the value of combining approaches

These results validate that learned fusion captures complementary strengths: the LLM provides robust reasoning even with limited training data, while the ML backbone adds efficiency and domain-specific patterns.

Application Domains

Learned fusion excels in scenarios where model strengths complement each other:

- **Customer feedback analysis** with nuanced multi-label taxonomies where LLMs handle ambiguous sentiment while ML models efficiently process clear cases
- **Content moderation** where uncertain cases benefit from LLM reasoning while routine items rely on the fast ML backbone, enabling real-time processing with accuracy guarantees
- **Scientific literature classification** across heterogeneous topics where domain shift is common and LLMs provide robustness to new terminology
- **Low-resource settings** where limited training data is available but task complexity requires sophisticated reasoning

The approach enables pragmatic cost control (e.g., the fusion layer learns when to rely more heavily on the efficient ML backbone versus the more expensive LLM signal) while retaining a single trainable decision surface that optimizes for the specific deployment constraints.

Acknowledgements

We thank contributors and users who reported issues and shared datasets. LabelFusion builds on the open-source ecosystem, notably Hugging Face Transformers (Wolf et al., 2019), scikit-learn (Pedregosa et al., 2011), PyTorch (Paszke et al., 2019), and LLM provider SDKs. We acknowledge the use of the AG News and GoEmotions benchmark datasets for evaluation.

References

Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G., & Ravi, S. (2020). GoEmotions: A dataset of fine-grained emotions. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4040–4054.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv Preprint arXiv:1810.04805*.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *International Workshop on Multiple Classifier Systems*, 1–15.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *Proceedings of the 34th International Conference on Machine Learning*.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(10), 993–1001.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv Preprint arXiv:1907.11692*.

OpenAI. (2023). GPT-4 technical report. *arXiv Preprint arXiv:2303.08774*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & others. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., & others. (2019). HuggingFace's transformers: State-of-the-art natural language processing. *arXiv Preprint arXiv:1910.03771*.

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 694–699.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, *28*, 649–657.