AutoIntent: AutoML for Text Classification

Alekseev Ilya[1,2,4], **Solomatin Roman**[1,3], **Rustamova Darina**[3], **Kuznetsov Denis**[1]

[1]Moscow Center for Advanced Studies, [2]Moscow State University, [3]ITMO University, [4]dresscode.ai.

Correspondence: ilya_alekseev_2016@list.ru

Abstract

AutoIntent is an automated machine learning tool for text classification tasks. Unlike existing solutions, AutoIntent offers end-to-end automation with embedding model selection, classifier optimization, and decision threshold tuning, all within a modular, sklearn-like interface. The framework is designed to support multi-label classification and out-of-scope detection. AutoIntent demonstrates superior performance compared to existing AutoML tools on standard intent classification datasets and enables users to balance effectiveness and resource consumption.

1 Introduction

Text classification remains a fundamental task in natural language processing, with applications ranging from intent detection in conversational systems (Weld et al., 2021) to content categorization and sentiment analysis (Taha et al., 2024). Modern NLP has been revolutionized by transformer-based embedding models (Vaswani et al., 2023; Devlin et al., 2019; Reimers and Gurevych, 2019), which provide rich contextual representations of text. However, effectively utilizing these models for classification tasks requires careful consideration of multiple components: the choice of pre-trained embedding model, the selection of appropriate classification algorithms, and the optimization of their hyperparameters.

Traditional machine learning approaches often require manual tuning of these components, which can be time-consuming and requires significant expertise. While automated machine learning (AutoML) frameworks (Baratchi et al., 2024) have emerged to automate this process, existing solutions for NLP tasks often lack comprehensive support for the full spectrum of hyperparameter optimization (for instance, choosing best embedding model) and tuning confidence thresholds for handling multi-label classification and out-of-scope (OOS) detection (Larson et al., 2019).

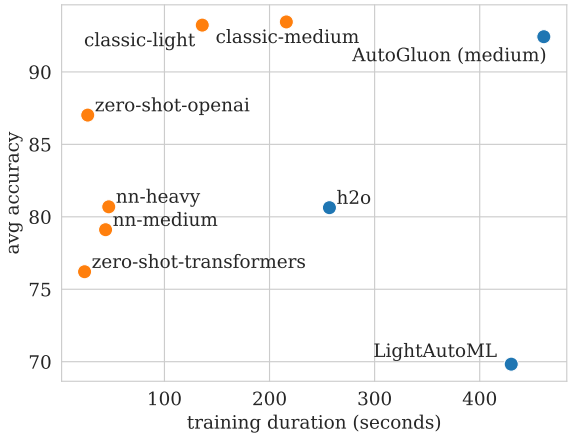

Figure 1: Average accuracy (banking77 (Casanueva et al., 2020), massive (FitzGerald et al., 2022), minds14 (Gerz et al., 2021), hwu64 (Liu et al., 2019)) and training duration (on minds14) of AutoIntent presets (**orange**) and baseline AutoML tools (**blue**).

This paper introduces AutoIntent[1], a novel AutoML framework specifically designed for intent classification tasks. The framework offers a sklearn-like interface (Pedregosa et al., 2018) for ease of use and supports even multi-label classification and OOS detection, bridging the gap between AutoML and conversation systems.

2 Background

Automated machine learning, by definition, is a tool for automating routines like data splitting for validation, hyperparameter tuning, model ensembling, and model selection. AutoML is highly relevant in scenarios where machine learning tasks need to be solved by non-experts and in conjunction with no-code ML, which is sometimes called «ML as a service» (Bisong, 2019; Barga et al., 2015; Liberty et al., 2020; LeDell and Poirier, 2020; Carney

[1]https://github.com/DeepPavlov/AutoIntent

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 707–716

	H2O	LightAutoML	AutoGluon	FEDOT	**AutoIntent (ours)**
Approach	TAML & Word2Vec	TAML & emb.	Encoder fine-tuning	TAML & TF-IDF	**Embeddings**
Scarce train data	×	Has small data modes	×	Adaptable	**Adapted for small datasets**
Custom search space	Flexible API	HP presets	✓	HP presets	**Presets & customizble configs**
Experiments tracking	H2O Flow integration	×	×	×	**W&B*, tensorboard*, codecarbon***.
Embedding prompting	×	×	×	×	✓
OOS detection	×	×	×	×	✓
Multi-label	×	✓	×	×	✓

Table 1: Comparison of NLP functionality in AutoML frameworks: H2O (LeDell and Poirier, 2020), LightAutoML (Vakhrushev et al., 2022), AutoGluon (Tang et al., 2024), FEDOT (Nikitin et al., 2021) and ours AutoIntent. HP stands for hyperparameters, TAML stands for tabular AutoML. *Weights and Biases (Biewald, 2020), Tensorboard (Abadi et al., 2015), CodeCarbon (Courty et al., 2024).

et al., 2020; Acito, 2023). The applications include sentiment analysis, robotics, healthcare, business analysis (Yuan et al., 2024; Salehin et al., 2024).

Tabular AutoML focuses on feature engineering, feature selection and model ensembling (Feurer et al., 2022; Vakhrushev et al., 2022; Erickson et al., 2020; LeDell and Poirier, 2020; Nikitin et al., 2021). Usually, they employ classical machine learning methods like GBMs (Chen and Guestrin, 2016; Prokhorenkova et al., 2018; Ke et al., 2017) and linear models, fast hyperparameter tuning methods with budget-aware strategies (Akiba et al., 2019), and ensembling strategies like stacking, blending and voting. Such frameworks sometimes can supersede exploratory data analysis and extensive research with just a running preset training recipe. It is not rare to see AutoML frameworks winning machine learning contests, but the open source solutions often are not transferable to production-ready systems as the resulting pipeline is an ensemble of a numerous amount of models without clear guides for deployment.

Neural architecture search can be viewed as an automation in the field of deep learning (Salehin et al., 2024). It emphasizes finding optimal computational graph using approaches like cell-based and hierarchical search spaces (Zoph et al., 2018; Real et al., 2019) or using scaling laws (Tan and Le, 2020). It cannot be treated as full-fledged AutoML, as it is designed to address only the model selection problem. Though, it can be a part of an AutoML pipeline (Jin et al., 2023).

AutoML tools in the NLP domain primarily stand out from other AutoML by native support of text inputs (Tang et al., 2024; Vakhrushev et al., 2022). This is especially important for use by non-experts, as it removes the requirement of manual tokenization and vectorization. Though, some tabular AutoML frameworks provide auxiliary tools for text feature extraction (LeDell and Poirier, 2020). The next peculiarity of text AutoML frameworks is their usage of transformer-based backbones, which makes sense, as this is the state-of-the-art in the field of NLP. Note that NLP AutoML primarily focuses on simple tasks like classification and regression, ignoring text generation and named entity recognition, for instance.

In AutoML frameworks, the model selection can be implemented in three ways. The first and most straightforward is to use hyperparameter tuning tools like Optuna (Akiba et al., 2019) and genetic algorithms (Feurer et al., 2022) with preset search spaces. Usually, these presets differ in how much time and computational resources they require to reach acceptable quality. The variety of presets is provided to cover all possible use cases and hardware settings. Another option is not to tune hyperparameters but use some generic hyperparameters that reach balance between the final quality and the generalization across different tasks. These hyperparameters can be obtained empirically (Tang et al., 2024). The compromise between freezing hyperparameters and tuning all of them is the meta-learning (Desai et al., 2022; Feurer et al., 2022; Wang, 2021; Tian et al., 2022; Huisman et al., 2021), where metamodel takes a dataset as input and predicts hyperparameters.

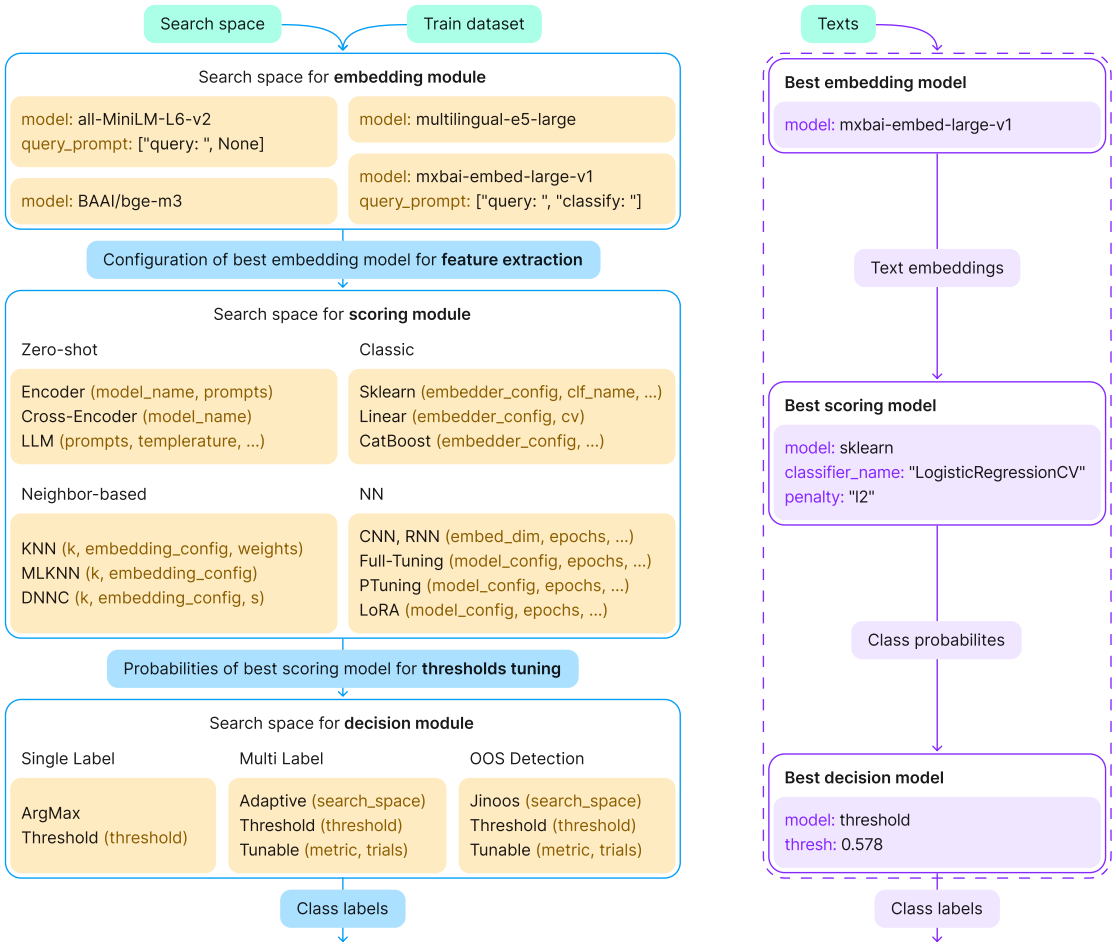

Figure 2: **(Left)** AutoIntent's three levels of hyperparameter optimization: at the module level, the embedding, scoring, and decision models are optimized sequentially; at the model level, each classification approach is tested against each other to select the best one; at the instance level, hyperparameters for each model is tuned individually with optuna samplers. **(Right)** Inference pipeline as a result of AutoIntent's hyperparameter optimization.

3 AutoIntent

3.1 Design Principles

The design of AutoIntent is guided by several key principles to ensure practical usability and maintainability (see Table 1). It features a **modular architecture** with a clear separation of concerns, adhering to **software engineering best practices** like type checking, auto-testing, and comprehensive documentation. The framework offers **model diversity**, supporting both high-performance deep learning models and efficient classical ML models that operate on pre-computed transformer embeddings. This **embedding-centric design** leverages the HuggingFace model repository and eliminates complex feature engineering. For usability, AutoIntent provides **flexible optimization strategies** (from presets to custom search spaces), multi-label classification, out-of-scope (OOS) detection, and few-shot learning.

3.2 Separation of Concerns

AutoIntent defines a *scoring module* as a section of the text classification pipeline that outputs class probabilities, establishing a clear separation from the decision module, which makes the final prediction by applying thresholds. This separation enhances modularity and flexibility, allowing a single scoring model's outputs to be reused with various decision strategies without re-computation, which is highly efficient for experimentation.

3.3 Embedding Module

AutoIntent leverages the sentence-transformers library (Reimers and Gurevych, 2019), providing access to a wide range of pre-trained transformer models from Hugging Face Hub (Wolf et al., 2020). AutoIntent offers three strategies for embedding model selection:

Pipeline-level optimization. The embedding

preset	duration	banking77	hwu64	massive	minds14	snips	avg
Baselines							
AutoGluon (best)	–	6.98	12.64	21.39	85.19	96.00	44.44
AutoGluon (high)	–	**92.60**	90.80	**89.22**	95.37	98.86	93.37
AutoGluon (medium)	461	92.40	**91.17**	87.13	92.59	98.86	92.43
LightAutoML	430	53.31	77.85	47.41	72.22	98.38	69.83
h2o	257	75.32	77.32	75.30	76.85	98.36	80.63
AutoIntent Presets							
zero-shot-transformers	**24**	69.51	71.47	63.58	87.04	89.43	76.21
nn-medium	44	79.95	70.79	72.75	75.31	96.74	79.11
nn-heavy	47	78.84	72.96	73.39	80.86	97.40	80.69
zero-shot-openai	27	76.43	85.04	80.49	96.30	96.86	87.02
classic-light	136	92.23	90.83	87.11	97.53	98.43	93.23
classic-medium	216	92.34	90.92	87.19	**97.84**	**98.98**	**93.45**

Table 2: Performance comparison across different presets averaged from three runs (except H2O and AutoGluon which were launched once). **Column 1**: Baseline AutoML frameworks: AutoGluon (Tang et al., 2024) with non-HPO presets `best_quality`, `high_quality`, `medium_quality`, H2O (LeDell and Poirier, 2020) with their word2vec, LightAutoML (Vakhrushev et al., 2022); and AutoIntent presets: `nn` (CNN (Kim, 2014), RNN), `zero-shot` (description-based bi- and cross-encoder, LLM prompting), `classic` (knn, logreg, random forest, catboost (Prokhorenkova et al., 2018)). **Column 2**: Duration in seconds evaluated on minds14 (Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz, single Tesla P100-SXM2-16GB). **Columns 3–7**: Accuracy on test sets.

model is chosen once at the start of the pipeline to maximize efficiency. The selection is based on either retrieval metrics (e.g., NDCG) or the performance of a simple downstream classifier (e.g., logistic regression).

Scoring-level optimization. The embedding model is optimized individually for each candidate model during the optimization of the scoring module. This is more computationally intensive, but may yield better performance.

Fixed embedding. Users can specify a default embedding model to skip optimization entirely.

This flexible approach allows users to balance optimization quality and computational cost.

3.4 Scoring Module

AutoIntent offers a diverse set of scoring models. A key architectural feature is that all the classifiers are able to operate on pre-computed embeddings. This separates computationally intensive embedding generation from the lightweight classification step, enabling a balance between effectiveness and efficiency and allowing deployment on CPU-only systems. The available scoring modules include:

KNN-based approaches. These include K-Nearest Neighbors method with FAISS (Douze et al., 2024) for efficient search, a two-stage cross-encoder re-ranking approach, and MLKNN (Zhang and Zhou, 2007) for multi-label tasks.

BERT-based classifiers. Support full model fine-tuning and parameter-efficient approaches like LoRA (Hu et al., 2021) and P-Tuning (Mangrulkar et al., 2022).

Generic sklearn integration allows use of any sklearn classifier operating on embedding vectors.

Zero-shot methods utilize text descriptions of classes and either measure the closeness with bi- or cross-encoder or prompt LLM by API (OpenAI, 2023).

3.5 Decision Module

The Decision Module processes scores to produce final predictions, which is crucial for multi-label and OOS scenarios.

AdaptiveDecision (Hou et al., 2020): A sample-specific thresholding method for multi-label classification.

JinoosDecision (Zhang et al., 2020): Finds a universal threshold that balances in-domain and OOS accuracy.

ThresholdDecision: Uses a fixed, user-specified threshold, suitable for use within the AutoML tun-

framework	in domain accuracy	out-of-scope F1-measure
AutoIntent	**96.13**	**76.79**
AutoGluon (Tang et al., 2024)	95.76	48.53
H2O (LeDell and Poirier, 2020)	85.22	40.69

Table 3: Performance comparison on out-of-scope detection task on CLINC150 (Larson et al., 2019).

ing pipeline.

TunableDecision: Employs Optuna (Akiba et al., 2019) to automatically find the optimal threshold by maximizing the F1 score.

3.6 AutoML Pipeline

AutoIntent orchestrates the optimization of all components hierarchically (Fig. 2), with two distinct levels of optimization. At the highest level, the pipeline performs *module-level optimization*, where it sequentially optimizes the embedding, scoring, and decision modules. Each module builds upon the best model from the previous module's optimization, creating a cohesive pipeline. For instance, the scoring module utilizes features from the best embedding model, while the decision module processes probabilities from the best classifier. This greedy approach effectively prevents combinatorial explosion while maintaining strong performance.

The second level focuses on *model-level optimization*, where both the model and its hyperparameters are sampled with Optuna's random sampling and Tree-structured Parzen Estimators (Watanabe, 2023). This includes various transformer models for the embedding module, different classification methods for the scoring module, and multiple threshold strategies for the decision module.

A crucial aspect of the pipeline is its clear distinction between tuning (non-gradient optimization of hyperparameters) and training (gradient optimization of model weights, if applicable). AutoIntent implements careful data handling with separate data subsets for training weights and validating hyperparameter configurations, and strategies to prevent target leakage. For cross-validation, it uses out-of-fold predictions to train stacked models (as we can view the whole three-stage pipeline with embedding, scoring and decision nodes as stacked models). The optimization is configured via a dictionary-like search space, and the final optimized pipeline can be saved and used later with simple `save`, `load`, and `predict` methods.

4 Experiments

4.1 Baselines

We compared AutoIntent against several open-source NLP AutoML frameworks: H2O (LeDell and Poirier, 2020), LightAutoML (LAMA) (Vakhrushev et al., 2022), and AutoGluon (Tang et al., 2024). The evaluation was conducted across five standard intent classification datasets (Table 2).

The results show that AutoGluon and AutoIntent are highly competitive, while H2O achieves moderate performance and LAMA fails on these tasks. AutoIntent provides the most affordable options in terms of balance between the quality and the computational cost. Also during the testing, we revealed several limitations in baseline frameworks:

Hyperparameter Optimization: AutoGluon uses a fixed training recipe; AutoGluon does support HPO presets, but they are too time and disk space consuming to test. **Feature Engineering**: H2O lacks native text features support. **Model Flexibility**: LAMA supports only three predefined transformer models. **Inference Efficiency**: AutoIntent can select lighter models for comparable performance, unlike AutoGluon's fine-tuned transformer as fixed output. **Model Variety**: Only AutoIntent provides the whole range of ML and DL models for text classification.

4.2 OOS Detection

We evaluated OOS capabilities on the CLINC150 dataset (Larson et al., 2019). Since baselines lack native OOS support, we treated OOS as an additional class for them while using AutoIntent's built-in OOS support. The results in Table 3 demonstrate AutoIntent's superiority, attributable to its dedicated confidence thresholds tuning. We utilize in-domain accuracy, because the dataset is quite balanced. Though, this is a detection task, so we consider F1-score as appropriate choice for OOS class.

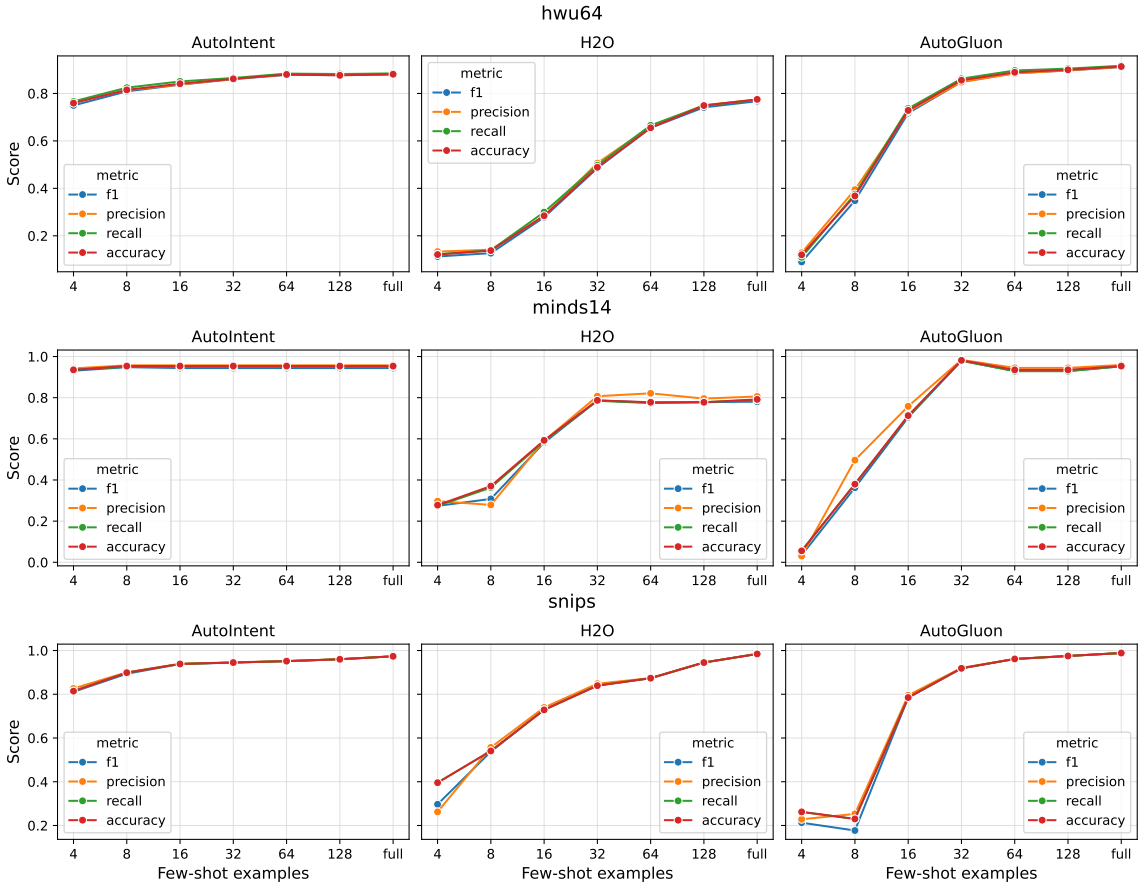

Figure 3: Performance comparison in a scenario of scarce training data. Baseline AutoML frameworks: AutoGluon (Tang et al., 2024) with non-HPO preset `medium_quality`, H2O (LeDell and Poirier, 2020) with their word2vec; and AutoIntent preset `classic-light`.

4.3 Few-shot Scenario

We evaluated the capabilities of AutoIntent in a scenario of scarce training data. We synthetically subsampled the datasets to have only n shots per class, with n ranging from 4 to 128. The results in Figure 3 demonstrate AutoIntent's robustness and superiority due to employing neighbor-based classification methods.

5 Conclusion

AutoIntent addresses the critical gap in automated machine learning for intent classification tasks, where existing AutoML frameworks lack comprehensive support for NLP-specific challenges including embedding model selection, multi-label classification, out-of-scope detection, and few-shot learning. The framework's importance stems from democratizing intent classification through end-to-end automation.

The system's novelty lies in its optimization strategy and embedding-centric design leveraging pre-computed transformer representations. AutoIntent targets NLP practitioners, conversational AI developers, and ML-as-a-service platforms.

AutoIntent operates through a three-stage pipeline (embedding, scoring, decision) with hierarchical optimization using Optuna. The embedding module selects optimal transformer models, the scoring module offers diverse classifiers.

The system was evaluated across five intent classification datasets. While demonstrating strong performance, limitations include no user studies conducted and focus on intent classification datasets. The framework is released under Apache-2.0 license to encourage community adoption.

6 Acknowledgements

This work was supported by the Ministry of Economic Development of the Russian Federation (agreement No. 139-15-2025-013, dated June 20, 2025, subsidy identifier 000000C313925P4B0002).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, and 21 others. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Frank Acito. 2023. Predictive analytics with knime. *Analytics for citizen data scientists. Switzerland: Springer*.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Mitra Baratchi, Can Wang, Steffen Limmer, Jan N van Rijn, Holger Hoos, Thomas Bäck, and Markus Olhofer. 2024. Automated machine learning: past, present and future. *Artificial intelligence review*, 57(5):122.

Roger Barga, Valentine Fontama, Wee Hyong Tok, and Luis Cabrera-Cordon. 2015. *Predictive analytics with Microsoft Azure machine learning*. Springer.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Ekaba Bisong. 2019. An overview of google cloud platform services. *Building Machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, pages 7–10.

Michelle Carney, Barron Webster, Irene Alvarado, Kyle Phillips, Noura Howell, Jordan Griffith, Jonas Jongejan, Amit Pitaru, and Alexander Chen. 2020. Teachable machine: Approachable web-based tool for exploring machine learning classification. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, pages 1–8.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *Preprint*, arXiv:2003.04807.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *Preprint*, arXiv:1805.10190.

Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, LiamConnell, Amine Saboni, Inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, and 11 others. 2024. mlco2/codecarbon: v2.4.1.

Rushil Desai, Aditya Shah, Shourya Kothari, Aishwarya Surve, and Narendra Shekokar. 2022. Textbrew: Automated model selection and hyperparameter optimization for text classification. *International Journal of Advanced Computer Science and Applications*, 13(9).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, and 67 others. 2025. MMTEB: Massive Multilingual Text Embedding Benchmark.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. Autogluon-tabular: Robust and accurate automl for structured data. *Preprint*, arXiv:2003.06505.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2022. Auto-sklearn 2.0: Hands-free automl via meta-learning. *Preprint*, arXiv:2007.04074.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *Preprint*, arXiv:2204.08582.

Daniela Gerz, Pei-Hao Su, Razvan Kusztos, Avishek Mondal, Michal Lis, Eshan Singhal, Nikola Mrksic, Tsung-Hsien Wen, and Ivan Vulic. 2021. Multilingual and cross-lingual intent detection from spoken data. *CoRR*, abs/2104.08524.

Yutai Hou, Yongkui Lai, Yushan Wu, Wanxiang Che, and Ting Liu. 2020. Few-shot learning for multi-label intent detection. *Preprint*, arXiv:2010.05256.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Mike Huisman, Jan N Van Rijn, and Aske Plaat. 2021. A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6):4483–4541.

Haifeng Jin, François Chollet, Qingquan Song, and Xia Hu. 2023. Autokeras: An automl library for deep learning. *Journal of Machine Learning Research*, 24(6):1–6.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Preprint*, arXiv:1408.5882.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Erin LeDell and Sebastien Poirier. 2020. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020, page 24.

Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. 2024. Open source strikes bread - new fluffy embeddings model.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.

Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, and 1 others. 2020. Elastic machine learning algorithms in amazon sagemaker. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 731–737.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. Benchmarking natural language understanding services for building conversational agents. *Preprint*, arXiv:1903.05566.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. *Preprint*, arXiv:2210.07316.

Nikolay O. Nikitin, Pavel Vychuzhanin, Mikhail Sarafanov, Iana S. Polonskaia, Ilia Revin, Irina V. Barabanova, Gleb Maximov, Anna V. Kalyuzhnaya, and Alexander Boukhanovsky. 2021. Automated evolutionary approach for the design of composite machine learning pipelines. *Future Generation Computer Systems*.

OpenAI. 2023. Openai api. https://openai.com/api/. Accessed: 26 may 2025.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2018. Scikit-learn: Machine learning in python. *Preprint*, arXiv:1201.0490.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. *Preprint*, arXiv:1802.01548.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Imrus Salehin, Md Shamiul Islam, Pritom Saha, SM Noman, Azra Tuni, Md Mehedi Hasan, and Md Abu Baten. 2024. Automl: A systematic review on automated machine learning with neural architecture search. *Journal of Information and Intelligence*, 2(1):52–81.

Kamal Taha, Paul D. Yoo, Chan Yeun, Dirar Homouz, and Aya Taha. 2024. A comprehensive survey of text classification techniques and their research applications: Observational and experimental insights. *Computer Science Review*, 54:100664.

Mingxing Tan and Quoc V. Le. 2020. Efficientnet: Rethinking model scaling for convolutional neural networks. *Preprint*, arXiv:1905.11946.

Zhiqiang Tang, Haoyang Fang, Su Zhou, Taojiannan Yang, Zihan Zhong, Tony Hu, Katrin Kirchhoff, and George Karypis. 2024. Autogluon-multimodal (automm): Supercharging multimodal automl with foundation models. *Preprint*, arXiv:2404.16233.

Yingjie Tian, Xiaoxi Zhao, and Wei Huang. 2022. Meta-learning approaches for learning-to-learn in deep learning: A survey. *Neurocomputing*, 494:203–223.

Anton Vakhrushev, Alexander Ryzhkov, Maxim Savchenko, Dmitry Simakov, Rinchin Damdinov, and Alexander Tuzhilin. 2022. Lightautoml: Automl solution for a large financial services ecosystem. *Preprint*, arXiv:2109.01528.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Jane X Wang. 2021. Meta-learning in natural and artificial intelligence. *Current Opinion in Behavioral Sciences*, 38:90–95.

Shuhei Watanabe. 2023. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *Preprint*, arXiv:2304.11127.

H. Weld, X. Huang, S. Long, J. Poon, and S. C. Han. 2021. A survey of joint intent detection and slot-filling models in natural language understanding. *Preprint*, arXiv:2101.08091.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Han Yuan, Kunyu Yu, Feng Xie, Mingxuan Liu, and Shenghuan Sun. 2024. Automated machine learning with interpretation: a systematic review of methodologies and applications in healthcare. *Medicine Advances*, 2(3):205–237.

Jian-Guo Zhang, Kazuma Hashimoto, Wenhao Liu, Chien-Sheng Wu, Yao Wan, Philip S. Yu, Richard Socher, and Caiming Xiong. 2020. Discriminative nearest neighbor few-shot intent detection by transferring natural language inference. *Preprint*, arXiv:2010.13009.

Min-Ling Zhang and Zhi-Hua Zhou. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning transferable architectures for scalable image recognition. *Preprint*, arXiv:1707.07012.

A Computational Efficiency

To quantify the computational requirements of different scoring modules, we conducted a comprehensive analysis using the Code Carbon library (Courty et al., 2024). This analysis measured various aspects of computational resource consumption for a single trial (training and evaluation of a single model configuration). The results, presented in Table 4, reveal significant variations in resource usage across different approaches.

The analysis revealed significant variations in resource efficiency across different scoring methods. KNN-based methods demonstrated exceptional efficiency, with minimal emissions and runtime, making them particularly suitable for resource-constrained environments. Logistic regression showed moderate resource consumption while maintaining high performance, representing a balanced trade-off between computational cost and effectiveness. In contrast, BERT-based methods exhibited the highest resource requirements, necessitating substantial computational infrastructure. These findings provide valuable insights for deployment scenarios with varying resource constraints and for building AutoIntent's presets.

A.1 Embedding Module Effectiveness

We evaluated our retrieval-based embedding selection heuristic (optimizing NDCG) against the ground truth (final accuracy from the full pipeline), as described in Section 3.3. As shown in Figure 4 and Table 5, while the approximate ranking is imperfect, it successfully identifies the best model (`stella_en_400M_v5`). This demonstrates that the heuristic effectively balances computational cost and selection quality. We have taken top models from `MTEB(eng)`(Muennighoff et al., 2023; Enevoldsen et al., 2025) leaderboard.

model	emissions	runtime	energy	gpu	cpu	ram	rate
bert	1.382	103.911	3.133	2.198	0.774	1.615e-01	0.014
ptuning	1.118	83.455	2.535	1.785	0.620	1.295e-01	0.014
lora	0.863	65.157	1.957	1.372	0.484	1.009e-01	0.013
linear	0.428	73.393	0.971	0.312	0.545	1.138e-01	0.006
rerank	0.270	29.040	0.613	0.355	0.213	4.436e-02	0.010
dnnc	0.122	10.000	0.276	0.192	0.070	1.455e-02	0.013
rand forest	0.073	11.367	0.166	0.074	0.080	1.664e-02	0.007
knn	0.009	1.281	0.019	0.014	0.004	9.044e-04	0.012
units	grams	sec	Wh	Wh	Wh	Wh	grams/sec

Table 4: Computational resource consumption for different scoring modules. The experiments are conducted on banking77 dataset with `mixedbread-ai/mxbai-embed-large-v1` (Lee et al., 2024; Li and Li, 2023), system with AMD Ryzen 7 5800H, NVIDIA RTX 3060 Laptop. Median values of 10 trials are displayed. Embeddings were pre-computed.

Model	**Accuracy**	**NDCG**
stella_en_400M_v5	94.28	93.83
multilingual-e5-[1]	93.65	92.97
GIST-large-[2]	93.51	93.32
UAE-Large-V1	92.89	93.25
bge-m3	92.69	92.49
multilingual-e5-large	91.41	92.45
LaBSE	90.47	89.51
KaLM-embedding-[3]	89.65	92.88
nomic-embed-[4]	87.24	89.63
deberta-v3-small	81.15	67.20
deberta-v3-large	75.39	59.59
deberta-v3-base	75.00	59.28

Table 5: Embedding models performance averaged over hwu64 (Liu et al., 2019), massive (FitzGerald et al., 2022), minds14 (Gerz et al., 2021), snips (Coucke et al., 2018). [1]large-unstruct, [2]Embedding-v0, [3]multilingual-mini-instruct-v1.5, [4]text-v1.5

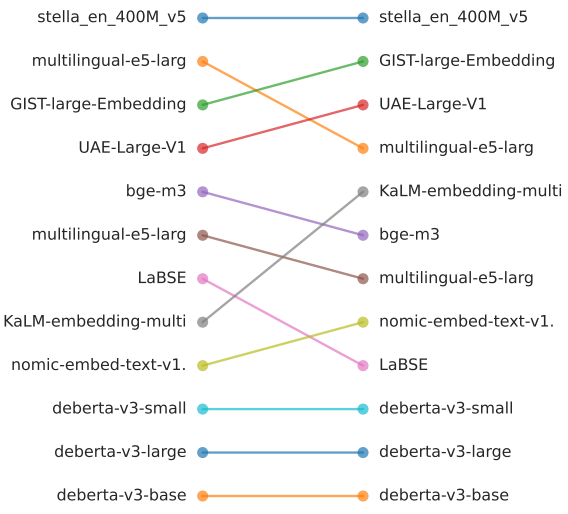

Figure 4: Encoders ranking: (**Left**) precise ranking obtained via training full AutoML pipeline with only this model, (**Right**) approximate ranking based on retrieval quality (NDCG).