o-MEGA: Optimized Methods for Explanation Generation and Analysis

Ľuboš Kriš[♠] **Jaroslav Kopčan**[♠] **Qiwei Peng**[♡]
Andrej Ridzik[♠] **Marcel Veselý**[♠] **Martin Tamajka**[♠]
♠Kempelen Institute of Intelligent Technologies
♡University of Copenhagen
{name.surname}@kinit.sk[♠] qipe@di.ku.dk[♡]

Abstract

The proliferation of transformer-based language models has revolutionized NLP domain while simultaneously introduced significant challenges regarding model transparency and trustworthiness. The complexity of achieving explainable systems in this domain is evidenced by the extensive array of explanation methods and evaluation metrics developed by researchers. To address the challenge of selecting optimal explainability approaches, we present **o-mega**, a hyperparameter optimization tool designed to automatically identify the most effective explainable AI methods and their configurations within the semantic matching domain. We evaluate o-mega on a post-claim matching pipeline using a curated dataset of social media posts paired with refuting claims. Our tool systematically explores different explainable methods and their hyperparameters, demonstrating improved transparency in automated fact-checking systems. As a result, such automated optimization of explanation methods can significantly enhance the interpretability of claim-matching models in critical applications such as misinformation detection, contributing to more trustworthy and transparent AI systems.

1 Introduction

The incredible success of today's transformer-based language models drastically changed the landscape of the natural language processing domain and how we approach complex tasks within it. However, with great power comes great responsibility. This success comes with a significant trade-off: the increasing complexity of these models has made them essentially black boxes, limiting their adoption in critical applications where transparency and interpretability are of great importance. Traditionally, addressing such interpretability challenges has relied on post-hoc relevance attribution methods, which provide post-hoc explanations (Lapuschkin et al., 2019; Søgaard, 2021) for trained models by assigning importance scores to input features or model parameters. While these explainable AI (XAI) techniques have proven valuable in revealing model behavior and identifying potential underlying flaws, so-called "Clever Hans" effects, or biases, they introduce a new challenge: the overwhelming variety of available XAI algorithms and their possible configurations. Just as practitioners may struggle to select optimal model architectures or hyperparameters, there is now emerging equally complex task of choosing the right explainability method for a given use case (Ali et al., 2023). This explainability challenge is especially relevant, for instance, in semantic matching tasks, where understanding why a model determined that two pieces of text are semantically similar or different is crucial for building user trust and ensuring reliable performance. Moreover, the interpretable semantic matching scenario is relevant in situations where the nuanced relationships between two texts require explanations that describe the model's decision-making process precisely and understandably to end users. Different XAI algorithms may highlight different aspects of the input text, leading to varying levels of usefulness depending on the specific matching task, domain, and user requirements. The manual process of evaluating and selecting appropriate XAI configurations is time-consuming, resource-intensive, subjective, and often suboptimal. Our **o-mega**[1] tool addresses this critical gap by automating the selection and optimization of post-hoc explainability algorithms, specifically within the semantic matching tasks. We have built the **o-mega** tool on top of previous work, which was inspired by the AutoML concept (Thornton et al., 2013), and adjusted to the domain of explainable AI. The main functionality is focused on the systematic evaluation of different

[1]We have released the code as well as documentation and examples at o-mega repository.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 634–642

XAI methods and their configurations in order to identify those that provide the most useful explanations for a given model and dataset combination according to specified criteria. The main motivation for the **o-mega** tool stems from the fact that explainability is not one-size-fits-all. In the semantic matching domain—whether it is a claim-matching task for fact-checking, document similarity assessment, or content recommendation—users need explanations that help them understand and trust the model's matching decisions. Since explainability is often overlooked because of its often ambiguous explanations, which are difficult to understand and struggles with implementation of the methods itself, by automating the explainability process of finding satisfactory XAI configurations, **o-mega** enables the use of interpretable semantic matching systems more efficiently while ensuring that the explanations provided are both technically correct and somehow practically useful. This tool represents an addition to the claim-matching task, which is central to fact-checking applications (Vo and Lee, 2020; Kazemi et al., 2021; Peng et al., 2025), and it is the first action step towards our effort to make it possible for domain experts to obtain meaningful insights from complex deep learning models without requiring extensive expertise in XAI methodologies.

2 Related Work

A wide range of explainable Artificial Intelligence (XAI) methods have been developed to enhance understanding of the decision-making process in machine learning models. Among post-hoc techniques, perturbation-based approaches are widely used where model predictions are examined by systematically sampling perturbed versions of the input and observing corresponding changes in output. LIME (Ribeiro et al., 2016) provides local explanations by fitting an interpretable surrogate model to approximate the behavior of a complex model. SHAP (Lundberg and Lee, 2017) attributes feature importance based on Shapley values from cooperative game theory. Other perturbation-based methods include Occlusion Sensitivity (Zeiler and Fergus, 2014), which assesses feature relevance by masking parts of the input directly. In contrast, gradient-based methods such as Integrated Gradients (Sundararajan et al., 2017) and LRP (Bach et al., 2015) propagate relevance scores or gradients backward through the model to identify influential features. Additionally, some studies propose textual explanation generation methods that aim to produce human-readable justifications for model predictions (Lei et al., 2016; Camburu et al., 2018; Atanasova et al., 2020). Explainability methods provide explanations of different qualities, and various metrics have been proposed to evaluate them. Common evaluation criteria include faithfulness, plausibility, and stability (Alvarez-Melis and Jaakkola, 2018; Mohseni et al., 2021; Nauta et al., 2023). However, no single metric captures all aspects of explanation quality, and different metrics may yield conflicts. This makes the evaluation of XAI methods a persistent challenge. This challenge is further compounded when selecting the most appropriate explanation technique for a given task. To address this, AutoXAI frameworks (Cugny et al., 2022), inspired by AutoML systems (He et al., 2021), have been proposed to automate the selection and configuration of XAI techniques. The framework aims to adaptively choose the most suitable explanation method and optimize associated hyperparameters based on user-defined objectives. The existing AutoXAI framework works only with the tabular modality of data and has implemented only two methods - SHAP and LIME, which are often considered as baselines when dealing with structured data. Inspired by it, we have created the **o-mega** framework within the domain of information retrieval, focusing on textual data. oMEGA framework incorporates multiple XAI methods, metrics for evaluation, and puts a specific focus on the claim matching task, which is a crucial task in fact-checking.

3 System Description

The **o-mega** tool is designed as a comprehensive framework for automatically selecting and optimizing explainable AI methods within semantic matching application. As illustrated in Figure 1, the system architecture consists of four interconnected modules that work together to identify the most effective explanation approach for a given AI model and dataset: 1) the **model** component that generates predictions requiring explanation, 2) the **method** component, which is a comprehensive space of available XAI algorithms and their configurations, 3) the **metric** part is an evaluation module incorporating both proxy measures and ground-truth annotations, and 4) the **optimization** component is a conditional hyperoptimization engine that

systematically explores and ranks different explainability approaches.

3.1 Methods

For semantic matching and text classification tasks, we require attribution-based XAI methods capable of capturing token-level or word-level importance. Additionally, for semantic matching specifically, these methods must capture cross-sequence interactions as well. We also want the methods to be diverse in their design and computationally efficient, therefore we have selected XAI methods spanning three paradigms: 1) gradient-based; 2) perturbation-based; and 3) architecture-specific. Overall, 11 explainable methods were implemented, 9 of them were from the Captum library. The entire selection is presented in Table 2. We have excluded several attribution methods from the library like Deconvolution, DeepLift, DeepLiftShap, because of irrelevancy for our task, or computational expenses.

3.2 Models

The inherent design of how these post-hoc explainability methods are computed introduces significant architectural rigidness. This introduces a constraint to same extent - the scoping of supported models. We have included an exhaustive list of most widely used semantic matching models. Table 1 presents the list of models which we have thoroughly tested. However, in general, the methods available within the **o-mega** framework should work with any transformer-based model for semantic matching of natural language text.

Model name	Embedding layer
sentence-transformers/gtr-t5-large	encoder.embed_tokens
sentence-transformers/gtr-t5-xl	encoder.embed_tokens
sentence-transformers/sentence-t5-xl	encoder.embed_tokens
sentence-transformers/all-mpnet-base-v2	embeddings.word_embeddings
sentence-transformers/multi-qa-mpnet-base-cos-v1	embeddings.word_embeddings
sentence-transformers/all-MiniLM-L12-v2	embeddings.word_embeddings
BAAI/bge-large-en-v1.5	embeddings.word_embeddings
BAAI/bge-base-en-v1.5	embeddings.word_embeddings
BAAI/bge-small-en-v1.5	embeddings.word_embeddings
llmrails/ember-v1	embeddings.word_embeddings
thenlper/gte-large	embeddings.word_embeddings
intfloat/e5-large-v2	embeddings.word_embeddings
BAAI/bge-large-en-v1.5	embeddings.word_embeddings

Table 1: Overview of tested language models for explainable use in semantic matching

3.3 Metrics

Evaluation module of the **o-mega** tool is the most crucial one, because based on the measurements of quality of computed explanations, the optimization process is guided, and at the output ends the recommendation for best methods regarding the task, data and model used. In order to evaluate how good the provided explanations actually are, it is important to firstly define what is meant by *the explanation quality - what constitutes to a good explanation?* There are two main approaches for how to answer this quality question, the first one is quantitative measurements, the second one is qualitative. While the qualitative measurements are just as much important, because of their subjective nature they can not be expressed by metrics. Therefore, within the o-mega evaluation we have focused primarily on the quantitative approach. On this topic was a lot of work done (Zhou et al., 2021; Markus et al., 2021) about how to find out if the explanation is of high quality - it needs to adhere to two components - *fidelity* and *plausibility*.

Method
Captum-based methods
Occlusion token-level
Input X Gradient
Guided Backprop
Feature Ablation
Kernel Shap
Gradient Shap
LIME
Saliency
Custom implementations
GAE
Conservative LRP
Occlusion word-level

Table 2: Overview of available XAI methods

- **Explanation high in fidelity** reflects the actual underlying behavior of the model, meaning the features deemed as important by explanation method for specific prediction, should be the same which influenced model the most. Metrics designed to test this are based on ablations or perturbations.

- **Explanation high in plausibility** reflects how easy is to comprehend the explanation for human-being, while this being the closest measure possible to qualitative evaluation within the quantitative domain. In order to be able express this semi-subjective property these metrics needs the human annotations - what humans themselves have deemed to be comprehensible explanation given the data and the task.

While during the evaluation is important to score high in both of these measurements, they often

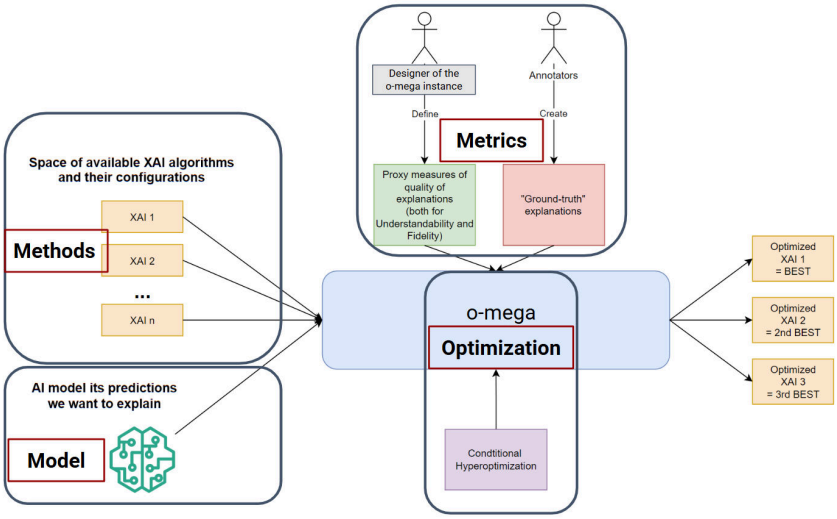

Figure 1: The architecture of the **o-mega** tool.

conflict with each other. An optimal explanation requires to strike a balance between them, while prioritizing one typically comes at the expense of the other. Our **o-mega** framework has available metrics for both categories - fidelity and plausibility which we have deemed as important and user could choose before the optimization if he wants to favor one over the other, or optimize towards both of them. All these metrics are custom implemented versions of established, previously published metrics which work with textual data (DeYoung et al., 2020; Hedström et al., 2023; Attanasio et al., 2023).

Metrics	Category
AOPC Comprehensiveness AOPC Sufficiency	Fidelity
AUPRC Token-level F1 scores Token-level IoU Average Precision Score	Plausibility

Table 3: Overview of available metrics

3.4 Optimization

For hyperparameter optimization, we employed the Optuna library (Akiba et al., 2024), which offers significant advantages over traditional grid search approaches. While grid search exhaustively evaluates all possible parameter combinations, Optuna uses intelligent search strategies such as Tree-structured Parzen Estimator (TPE) and Bayesian optimization that learn from previous evaluations to guide future parameter selection (Feurer and Hutter, 2019). This adaptive approach allows the optimizer to focus computational resources on promising regions of the hyperparameter space, effectively avoiding areas that have already shown poor performance. The efficiency gains become particularly pronounced in high-dimensional hyperparameter spaces, where grid search suffers from the curse of dimensionality. For instance, if we have 5 hyperparameters with 10 possible values each, grid search would require 10^5 = 100,000 evaluations, while Optuna's intelligent sampling can often find near-optimal configurations with significantly fewer trials—often by an order of magnitude or more. This efficiency is crucial in our context, where Captum's explanation methods have numerous hyperparameters that can substantially impact both computational cost and explanation quality. Moreover, Optuna's pruning capabilities allow early termination of unpromising trials, further reducing computational overhead. This is particularly valuable when working with explanation methods that may have expensive evaluation metrics, as the optimizer can quickly identify and abandon parameter configurations that are unlikely to yield good results based on partial evaluations. The result is a more efficient exploration of the hyperparameter space that converges to high-quality explanations faster than traditional optimization approaches.

3.5 Configuration

As a base part of the **o-mega**, the user can select a model from the Huggingface library (See Table 1) or a local pre-trained model. Plausibility and faithfulness have corresponding weights to adjust the priority of each group of metrics (See Figure 6). Then, the form of evaluation of explanations can take place in 3 options: either in token form, or by

combining explanations and masks into sentences, or into words. One example of the base configuration is shown in Figure 2.

```
model_path: "intfloat/multilingual-e5-
    large"
embeddings_module_name: "embeddings.
    word_embeddings"
methods: ["GAE_Explain","Occlusion"]
normalizations: ["without_normalize"]
explanation_maps_token: True
plausability_weight: 0.5
faithfulness_weight: 0.5
multiple_object: False
```

Figure 2: Configuration block specifying the base parameters for hyperoptimization and evaluation of explanations

The user can further configure 6 search strategies from the Optuna library (See Figure 3). These 6 search strategies can be divided into 2 groups: (1)single-object and (2)multi-object hyperoptimization. Within the configuration, plausibility and plausibility values can be considered separately. Within single-object hyperoptimization, we have TPESampler, GPSampler, and BruteForceSampler available for use, and within multi-object hyperoptimization, we can use NSGAIIISampler, NSGAIISampler, and TPESampler.

```
Optuna_parameters:
  sampler: "TPESampler"
  n_trials: 14
  n_startup_trials: 4
  seed: 1000
```

Figure 3: Configuration block for Optuna Sampler and its parameters.

XAI methods also accept a list of configurations for customization. One example is shown in Figure 4, where the hyperparameters of XAI methods can be restrained to a specific search space.

```
method_param:
  Gradient Shap:
    parameters:
      stdevs: (0.1, 0.9, {'step': 0.1})
      n_samples: [10, 15]
  Lime:
    parameters:
      n_samples: [80, 90]
  token_groups_for_feature_mask: true
```

Figure 4: Configuration block specifying hyperparameters for explainable method.

4 Case Study

In this case study, we focus on the task of claim matching, a key component in automated fact checking. The claim matching task is typically formulated as an information retrieval task. Given

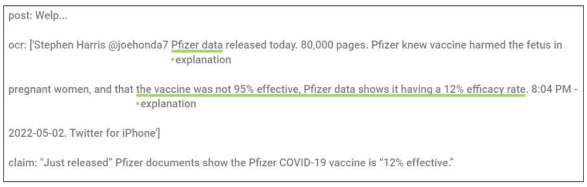

Figure 5: One example of the post-claim pair with annotated explanations.

an input text (e.g., social media posts), the goal is to find an appropriate claim from a database of claims that have already been fact-checked by professional fact-checkers. When a user posts a post making a claim worth fact-checking, the model aims to find a semantically similar claim from a list of previously fact-checked claims. For this study, we utilize the MultiClaim dataset (Pikuliak et al., 2023). To examine interpretability, five human annotators are asked to provide explanations for why specific claims were matched. Examples are given in figure 5. To assess the quality of the annotation, if the majority vote is not determined, we directly throw the sample away. This results in 512 post-claim pairs with human annotations.

4.1 Task Configuration

To generate optimal explanations for this task, we developed a hyperparameter optimization approach that selects the best combination of explanation methods and their configurations. The explanation methods identify significant attributes in both posts and claims by analyzing the cosine similarity between them. We utilized the Captum library for generating explanations, which provides multiple options for modifying explainable method parameters that affect how individual embeddings are processed. Our hyperparameter optimization framework incorporates these configurable parameters as well. Specific methods need to set the model in a specific way. In Figure 6 are shown the model parameters required for each explanation method are shown, such as the necessary layer or gradient settings, which show how different methods require access to specific components of the neural architecture. Figure 7 details the specific tunable parameters for each explanation method, such as perturbation sizes for occlusion methods or baseline values for gradient-based approaches. For the optimization process, we employed the Optuna library, which offers various optimization algorithms. Our evaluation framework uses five metrics which can be organized into two categories: fidelity and

plausibility. Additionally, we included average precision score (APS) as a supplementary metric for plausibility assessment.

4.2 Results and Analysis

Table 4 presents results from **o-mega** optimization which should provide clear guidance for practitioners. Occlusion is the recommended explainability method for this semantic matching task, achieving the best overall performance (0.819 average) by balancing technical accuracy with human interpretability. This automated recommendation eliminates the need for manual trial-and-error testing of different explanation methods. The results reveal that while most methods can accurately capture model behavior (faithfulness >0.94), they vary significantly in producing explanations that users can easily understand (plausibility 0.35-0.68). This means that method selection critically impacts user experience. Occlusion provides best explanations that are both technically correct and accessible to non-experts, while alternatives like ConservativeLRP (0.641) may confuse users despite being technically valid. The optimization process successfully identified the configuration that maximizes both explanation quality dimensions, providing practitioners with a data-driven recommendation rather than relying on popularity or default settings of explanation methods.

Table 5 depicts comparison of single objective samplers. Results reveals TPESampler as the most efficient optimization algorithm for this task, completing the same optimization quality in significantly less time compared to alternatives. While all three samplers identify Saliency as the optimal method with identical performance scores, TPESampler demonstrates superior computational efficiency, requiring only 14 trials versus BruteForceSampler's 35 trials to achieve the same result. BruteForceSampler provides the most thorough exploration but at a substantial computational cost (1.112 hours), making it impractical for real-world applications. GPSampler offers a middle ground with moderate efficiency (0.672 hours, 14 trials) but shows higher memory usage (760MB against 748MB for TPESampler) and experiences duplicate trials, indicating less efficient search space exploration. The main recommendation from the optimization run comparison is that the TPESampler represents the optimal choice, delivering the same explanation quality recommendations as exhaustive search methods while reducing optimization time

Methods	Faithfulness	Plausibility	Average
Occlusion	0.963	**0.675**	**0.819**
Gradient Shap	**0.967**	0.627	0.797
Saliency	0.966	0.621	0.794
GAE_Explain	0.959	0.615	0.787
Lime	0.962	0.599	0.781
Feature Ablation	0.960	0.597	0.779
Occlusion word-level	0.946	0.605	0.776
Kernel Shap	0.952	0.563	0.757
Guided Backprop	0.943	0.516	0.730
Input X Gradient	0.935	0.497	0.716
ConservativeLRP	0.927	0.354	0.641

Table 4: Comparison of Methods with Faithfulness, Plausibility, and Average scores.

by approximately 66%, making automated XAI selection feasible for routine deployment. Lastly, Table 6 shows the multi-objective optimization results. It shows that BruteForceSampler and TPESampler both correctly identify Occlusion as the optimal method, while NSGA-II and NSGA-III variants select the inferior Saliency method. BruteForceSampler achieves the highest performance scores (0.957 faithfulness, 0.646 plausibility) but requires 35 trials and 1.121 hours, while TPESampler delivers nearly identical results (0.958 faithfulness, 0.622 plausibility) with 70% less computation time (14 trials, 0.339 hours). The NSGA variants show identical performance, indicating no benefit from the increased complexity of NSGA-III. For practitioners, TPESampler is the recommended choice, providing the correct method selection with optimal efficiency for multi-objective explainability optimization.

Single objective	BruteForceSampler	GPSampler	TPESampler
method_best	Saliency	Saliency	Saliency
overall_score	0.784	0.78	0.784
best_find_at	13	5	7
peak memory usage (MB)	936.53	760.92	**748.43**
time (hours)	1.112	0.672	**0.379**
number_dup	0	**1**	4
all_trials	35	14	14

Table 5: Comparison of single objective samplers' performance. The best performances are depicted in bold.

Metric	BruteForceSampler	TPESampler	NSGAIISampler	NSGAIIISampler
Total Trials	35	**14**	**14**	**14**
Best Method	Occlusion	Occlusion	Saliency	Saliency
Faithfulness	0.957	**0.958**	0.957	0.957
Plausibility	**0.646**	0.622	0.606	0.606
Peak Memory (MB)	907.08	761.84	**747.27**	749.35
Time (hours)	1.121	0.339	**0.261**	**0.261**
Duplicates	0	5	7	7

Table 6: Comparison of Multi-Objective Samplers' Performance

5 Conclusion

This work presents **o-mega**, an automated hyperparameter optimization tool for explainable AI method selection in semantic matching and text classification tasks. Our evaluation within seman-

tic matching demonstrates that automated optimization successfully identifies optimal XAI configurations, with Occlusion emerging as the best-performing method for post-claim matching applications. By automating the complex process of XAI method selection and configuration, o-mega enables users to deploy transparent AI systems without requiring deep expertise in explainability techniques. This work has demonstrated the possibility of making explainability more accessible and frictionless for real-world applications, particularly in critical domains such as automated fact-checking and disinformation detection.

Limitations

The current implementation of o-mega has several limitations:

- The tool was developed specifically for claim matching as an enhancer for a specific dataset - MultiClaim, although the text classification pipeline is also available, it is still limiting its immediate applicability to other domains
- Evaluation metrics are currently restricted to only established ones, and tools do not include other experimental metric evaluations, such as localization and sparseness
- Computationally expensive explanation methods have not been incorporated yet, but this is also partially desired since high computational resource requirements render the methods less effective, therefore users are less likely to use them

Acknowledgments

We would like to thank all reviewers for their insightful comments and feedback. This work was supported by DisAI - Improving scientific excellence and creativity in combating disinformation with artificial intelligence and language technologies, a project funded by European Union under the Horizon Europe, GA No. 101079164. This work was also partially funded by European Union, under the project lorAI - Low Resource Artificial Intelligence, GA No. 101136646.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2024. Optuna: A next-generation hyperparameter optimization framework. *Optuna Documentation*. Accessed: 2025-07-01.

Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. 2023. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information fusion*, 99:101805.

David Alvarez-Melis and Tommi S Jaakkola. 2018. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. Generating fact checking explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7352–7364, Online. Association for Computational Linguistics.

Giuseppe Attanasio, Eliana Pastor, Chiara Di Bonaventura, and Debora Nozza. 2023. ferret: a framework for benchmarking explainers on transformers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Robin Cugny, Julien Aligon, Max Chevalier, Geoffrey Roman Jimenez, and Olivier Teste. 2022. Autoxai: A framework to automatically select the most adapted xai solution. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 315–324, New York, NY, USA. Association for Computing Machinery.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. 2020. Eraser: A benchmark to evaluate rationalized nlp models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458.

Matthias Feurer and Frank Hutter. 2019. *Hyperparameter optimization*. Springer International Publishing.

Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. Automl: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622.

Anna Hedström, Leander Weber, Daniel Krakowczyk, Dilyara Bareeva, Franz Motzkus, Wojciech Samek, Sebastian Lapuschkin, and Marina M-C Höhne. 2023. Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations and

beyond. *Journal of Machine Learning Research*, 24(34):1–11.

Ashkan Kazemi, Kiran Garimella, Devin Gaffney, and Scott A. Hale. 2021. Claim matching beyond English to scale global fact-checking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4504–4517, Online. Association for Computational Linguistics.

Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2019. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1096.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Aniek F Markus, Jan A Kors, and Peter R Rijnbeek. 2021. The role of explainability in creating trustworthy artificial intelligence for health care: a comprehensive survey of the terminology, design choices, and evaluation strategies. *Journal of biomedical informatics*, 113:103655.

Sina Mohseni, Niloofar Zarei, and Eric D Ragan. 2021. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 11(3-4):1–45.

Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice Van Keulen, and Christin Seifert. 2023. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Computing Surveys*, 55(13s):1–42.

Qiwei Peng, Robert Moro, Michal Gregor, Ivan Srba, Simon Ostermann, Marian Simko, Juraj Podroužek, Matúš Mesarčík, Jaroslav Kopčan, and Anders Søgaard. 2025. Semeval-2025 task 7: Multilingual and crosslingual fact-checked claim retrieval. *arXiv preprint arXiv:2505.10740*.

Matúš Pikuliak, Ivan Srba, Robert Moro, Timo Hromadka, Timotej Smoleň, Martin Melišek, Ivan Vykopal, Jakub Simko, Juraj Podroužek, and Maria Bielikova. 2023. Multilingual previously fact-checked claim retrieval. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16477–16500, Singapore. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Anders Søgaard. 2021. *Explainable natural language processing*. Morgan & Claypool Publishers.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855.

Nguyen Vo and Kyumin Lee. 2020. Where are the facts? searching for fact-checked information to alleviate the spread of fake news. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7717–7731, Online. Association for Computational Linguistics.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer.

Yao Zhou, Haonan Wang, Jingrui He, and Haixun Wang. 2021. From intrinsic to counterfactual: On the explainability of contextualized recommender systems. *arXiv preprint arXiv:2110.14844*.

Appendix: Method Hyperparameters

Methods	Hyperparameters
Occlusion	sliding_window_shapes:(5, 1024), strides:(1, 1024)
Gradient Shap	stdevs:0.1, n_samples:15
Saliency	abs: True
GAE_Explain	
Lime	n_samples:90 ,similarity_func:{'function_name': 'get_exp_kernel_similarity_function', 'parameters': {'distance_mode': 'euclidean', 'kernel_width': 750}}, interpretable_model:{'function_name': 'SkLearnLasso', 'parameters': {'alpha': 1e-10}}
Feature Ablation	
Occlusion_word_level	regex_condition: ".,!?;:…"
Kernel Shap	n_samples:90
Guided Backprop	
Input X Gradient	
ConservativeLRP	

Table 7: Complete list of method hyperparameters

```
model_param:
   Lime:
     similarity_func:
       function_name:
         - captum.attr._core.lime.
             get_exp_kernel_similarity_function
       parameters:
         distance_mode: ["cosine", "euclidean"]
         kernel_width: [450, 750]
     interpretable_model:
       function_name:
         - captum._utils.models.linear_model.
             SkLearnLasso
       parameters:
         alpha: [1e-19, 1e-25]
   GAE_Explain:
     implemented_method: true
     layers:
       module_path_expressions:
         - "hf_transformer.encoder.layer.*.
             attention.self.dropout"
   ConservativeLRP:
     implemented_method: true
     layers:
       store_A_path_expressions:
         - "hf_transformer.embeddings"
       attent_path_expressions:
         - "hf_transformer.encoder.layer.*.
             attention.self.dropout"
       norm_layer_path_expressions:
         - "hf_transformer.embeddings.LayerNorm"
         - "hf_transformer.encoder.layer.*.
             attention.output.LayerNorm"
         - "hf_transformer.encoder.layer.*.output.
             LayerNorm"
   Occlusion_word_level:
     implemented_method: true
```

Figure 6: Configuration block specifying the model parameters for hyperoptimization and evaluation of explanations

```
method_param:
  Lime:
    parameters:
      n_samples: [80, 90]
    token_groups_for_feature_mask: true
  Saliency:
    parameters:
      abs: [true, false]
  Occlusion:
    parameters:
      sliding_window_shapes:
        - [3, 1024]
        - [5, 1024]
      strides:
        - [1, 1024]
        - [1, 512]
    compute_baseline: true
  Gradient Shap:
    parameters:
      stdevs: [0.1, 0.9]
      n_samples: [10, 15]
    compute_baseline: true
  Kernel Shap:
    parameters:
      n_samples: [80, 90]
    compute_baseline: true
  Feature Ablation:
    token_groups_for_feature_mask: true
    compute_baseline: true
  Occlusion_word_level:
    parameters:
      regex_condition:
        - ""
        - ".,!?;:"
  Integrated Gradients:
    parameters:
      n_steps: [60, 40]
```

Figure 7: Configuration block specifying the method parameters for hyper-optimization and evaluation of explanations