Unica at FadeIT: Adapting Large Language Models to Fallacy Identification in Social Networks

Matteo Fenu[1,*,†], Maurizio Atzori[2,†]

[1]*University of Cagliari, Faculty of Science, 72 Ospedale St, Cagliari, 09124, Italy*

Abstract

This paper presents a comprehensive approach to fallacy detection, addressing the challenges of handling multiple interpretations and class imbalance through a novel pre-processing pipeline. Our methodology integrates feature extraction techniques, leveraging GPT-5.1 with custom prompts to derive meaningful descriptors from social media posts, alongside data augmentation strategies combining prompt engineering and machine translation to oversample minority fallacy classes. For Subtask A, coarse-grained fallacy detection, we propose and compare three distinct approaches based on fine-tuning and Retrieval-Augmented Generation (RAG). Our experiments involve state-of-the-art large language models, including Mixtral-8x7B-Instruct, Gemma-3-12B-it, and GPT-5. Fine-tuning was performed on a single A100 GPU using the QLoRA methodology to enable parameter-efficient training. The RAG-based approach was implemented using OpenAI APIs integrated with Qdrant, an open-source vector database, for efficient semantic retrieval.

Keywords

Fallacy Detection, Large Language Models, Fine-tuning, QLoRA, Retrieval-Augmented Generation, Data Augmentation, Prompt Engineering, Feature Extraction

1. Introduction

This study contributed to the design and development of intelligent systems for the structured representation of fallacious arguments employed within social networks, through participation in sub-task A proposed by EVALITA as part of the FadeIT shared task [1]: identifying fallacies expressed in Italian social media posts. These systems aim to counteract the manipulation of public opinion and the spread of disinformation on sensitive topics in 2019-2022, such as climate change, migration, and public health.

This paper proposes two different approaches that leverage some of the most recent and high-performing foundation models, appropriately optimised through Fine-Tuning and Prompt Engineering techniques. Specifically, the first approach employs a model from the OpenAI GPT-5 family, instructed via a prompt containing dynamic few-shot examples selected on the basis of semantic similarity with the post to be classified. The second approach is based on a fine-tuning algorithm through which several Large Language Models from the Mistral, Gemma, and Llama families were developed and tested. The resulting systems achieved strong results during the challenge evaluation phase. In particular, the first approach ranked 4th among all 16 submissions with an micro F1 score of 49.71, enabling the UNICA team to secure third place in the overall standings. This paper presents an overview of the systems selected for the challenge and discusses the results obtained.

2. Motivation

The Treccani dictionary defines[1] a fallacious act, word, or argument as something deceptive: an ostensibly credible argument that is nevertheless logically flawed and therefore false. Fallacies represent

EVALITA

*Corresponding author.
†These authors contributed equally.
fenum@hotmail.it (M. Fenu); atzori@unica.it (M. Atzori)
0009-0008-8402-9703 (M. Fenu); 0000-0001-6112-7310 (M. Atzori)

[1]https://www.treccani.it/vocabolario/fallacia/

a significant risk to public safety today, particularly in a society constantly exposed to the latest news or the most recent informative post published on social networks.

Fallacies are also a tool widely exploited by fraudsters to deceive individuals and prompt them to action. An emblematic case is so-called phishing, namely the attempt to illegally obtain a user's personal data, along with other sensitive information, for the purpose of theft. This practice, which particularly affects elderly individuals, relies on the dissemination of deceptive messages constructed through fallacies, with the aim of extracting the desired information. This constitutes a substantial risk and a matter of paramount importance in the ongoing effort to protect personal data.

The research aims to leverage the challenge proposed by the EVALITA 2026 campaign in order to develop tools useful to the public for counteracting fake news, the manipulation of information and opinions, as well as fraud attempts and the misappropriation of personal data. From a technical standpoint, the objective of this research is to assess how the most recent Large Language Models can be adapted to perform multi-class tasks such as that proposed for sub-task A (detection of fallacies in social media posts), and to evaluate the impact of model size and the optimisation algorithms selected.

3. Definition of the tasks and evaluation measures

The Subtask A proposed by the organisers, the one on which this research focuses, is defined as follows: given a text, predict the set of fallacies expressed within it. This is a multi-label classification task. Systems are evaluated using Precision, Recall, and micro F1-score, computed as the average across the two annotations (both considered equally valid). The metric that determines the final ranking is the micro F1-score. For informational purposes only, and not covered in this study, Subtask B is defined as follows: given a text, predict the segments containing fallacies. This is a multi-label sequence labelling task. The main difficulty lies in the fact that fallacies may partially overlap with one another. In this case, the text is represented as a sequence of tokens. The annotation follows the BIO scheme (B: begin, I: inside, O: outside); any multiple annotations for the same token are separated by the pipe symbol (|) and preceded by the corresponding tag. The metrics adopted are variants of F1-score, Precision, and Recall, adapted to operate at the token level. The evaluation is defined as soft, as it also accounts for partial matches, weighted proportionally to the length of the match itself in terms of tokens.

Table 1
Example of annotation for Subtask A

Text	**Annotator 1**	**Annotator 2**
Basta con le politiche immigrazioniste della sinistra. Il #25settembre vota [USER] [URL]	Flag-waving, Name-calling-or-labelling, Slogan	Flag-waving, Loaded-language, Name-calling-or-labelling, Slogan

English translation: *"Enough with the left's pro-immigration policies. On #September25 vote [USER] [URL]"*

4. Dataset overview

The entire FadeIT shared task is based on the Faina dataset [2], a dataset for fallacy recognition. It contains social media posts written in Italian. The main topics addressed within it concern immigration, climate change, and public health.

The dataset is based on an ontology of twenty fallacy types, such as, for example, appeals to authority or to readers' emotions, hasty generalisations drawn from small and unrepresentative samples, or the use of slogans or phrases intended to elicit enthusiasm among the readership.

The dataset provides two different types of annotations, from which the two different sub-tasks proposed for the challenge are derived:

- train-dev.tsv: Each element of the dataset is characterised by the post text, the topic of the post, and two annotations relating to the fallacies present. Each annotation contains a list of fallacies separated by the pipe symbol (|). This is the dataset proposed for sub-task A.
- train-dev.conll: Compared to the .tsv file, the annotations in this dataset contain a tag for each element of the sentence (token) using the BIO scheme (Begin, Inside, Out). This is the dataset proposed for sub-task B.

From an initial analysis phase of the available dataset, two main criticalities emerged. The first concerns a suboptimal distribution of the number of examples for each fallacy. Indeed, within the dataset, certain fallacies predominate with over 10% coverage (loaded-language, appeal-to-emotion), while others do not exceed 2% (circular-reasoning, false-dilemma, cherry-picking). The second, more subtle in nature and certainly more difficult to address, concerns the lack of immediate comprehension of the meaning of certain fallacies for a transformer-based model. Fallacies such as circular reasoning, strawman, cherry picking, and false analogy prove particularly challenging at first approach and require structured reasoning capabilities to be properly understood.

Overall, the dataset comprises 1152 posts. Of these, 107 contain no fallacies according to both annotators. On average, each post contains 2.79 fallacies according to the first annotator and 2.96 according to the second annotator. Another noteworthy aspect is the average length of spans containing a given fallacy. Some fallacies, such as Loaded-language or Name-calling-or-labelling, are characterized by approximately two words per span on average. Among the most complex in this regard are Evading-the-burden-of-proof with 16.47 average words per span and Circular-reasoning with 28.69. The Avg. Words column in Table 2 summarizes the values for each fallacy. Additionally, Table 2 reports the number of posts in which each fallacy occurs and the corresponding percentage relative to the total number of posts in the dataset.

Table 2
Distribution of Fallacies in the original training set (Part 1)

Fallacy	Occurrences	Percentage (%)	Avg. Words
Loaded-language	1158	17.49	2.54
Appeal-to-emotion	1103	16.66	5.08
Vagueness	816	12.32	9.24
Name-calling-or-labelling	603	9.11	2.61
Doubt	366	5.53	16.30
Hasty-generalization	344	5.20	10.50
Evading-the-burden-of-proof	294	4.44	16.47
Flag-waving	248	3.75	4.28
Ad-hominem	234	3.53	15.57
Slogan	226	3.41	3.45
Thought-terminating-cliches	214	3.23	5.26
Red-herring	187	2.82	12.51
False-analogy	186	2.81	21.36
Appeal-to-authority	159	2.40	6.15
Slippery-slope	135	2.04	10.48
Causal-oversimplification	108	1.63	19.51
Strawman	87	1.31	37.07
Cherry-picking	76	1.15	29.18
False-dilemma	61	0.92	16.31
Circular-reasoning	16	0.24	28.69

4.1. Pre-processing

To prepare the dataset for the subsequent training and evaluation phases, a series of operations were performed aimed at increasing its quality and comprehensibility, bearing in mind the criticalities encountered during the initial analysis phase.

4.1.1. Data Augmentation

First, we performed a data augmentation operation in order to reduce the gap between the less representative fallacies and the more frequent ones. Specifically, these operations focused on increasing the number of examples for fallacies that exhibit poor representation in the dataset. To achieve this objective, two different strategies were adopted.

- Prompt engineering: a prompt was developed to instruct GPT 5.1 to generate new examples for the set of minority fallacies. As can be seen from Figure 1, in the first static part of the prompt, the task to be performed and the structure of the expected output are defined. The second part, which varies according to the fallacy for which examples should be augmented, contains the fallacy name, its description in natural language, and a list of example social media posts containing it drawn from the dataset. The algorithm was executed on the set of fallacies with a percentage below 3% in the original training set. For each of these, fifty new examples were generated for inclusion in the final dataset to be used for system training.

System Prompt: Fallacy Generator

Act as a creator of posts containing fallacies. Your task is to generate new comments containing the target fallacy. Create fallacies containing no more than 40 words.

Output format: JSON

```
{
"examples": [
"comment with target fallacy",
"comment with target fallacy",
...
]
}
```

Input variables:

- `<target_fallacy>`: The fallacy for which to generate new examples
- `<fallacy_description>`: Description of the target fallacy
- `<list_of_examples>`: Examples to draw inspiration from

Figure 1: Prompt template for fallacy generation via LLM

- Back-translation: this is the process of retranslating content from the target language back to the source language in literal terms. During this phase, semantically equivalent paraphrases of existing posts were generated through the use of pivot translations. This process involves translating the original text into an intermediate language and subsequently retranslating it into the original language (Italian). Two translation cycles were employed: Italian-Spanish-Italian and Italian-French-Italian. Prior to performing the actual translation, hashtags were masked using placeholder tokens to prevent them from being altered. The Helsinki-NLP neural models from the OPUS-MT family were used for translation. Specifically, we employed: `opus-mt-it-es` and `opus-mt-es-it` for Italian-Spanish translation, and `opus-mt-it-fr` and `opus-tatoeba-fr-it` for Italian-French translation. [3] Once the paraphrase was obtained, a quality filter based on semantic similarity was applied. The embeddings of the post before and after the back-translation process were computed, and the cosine similarity between the two resulting vectors was calculated. Paraphrases with a similarity below 75% were discarded. In the final step, the placeholders inserted in the first phase were restored with their original values. During this phase, as the number of examples generated with the first approach for minority classes was deemed sufficient, the focus was not placed on particular classes; rather, a random selection of rows to be paraphrased was performed in order to increase the overall quantity of examples in the training dataset.

The aforementioned operations led to the creation of the final dataset ready for model training. The final number of rows is 2973, compared to the initial 1152. The resulting dataset shows a substantial increase in the number of examples for several classes, including Slippery-slope, Strawman, and Cherry-picking, thereby leading to greater representativeness for all target fallacies.

4.1.2. Features extraction

The second fundamental step of the pre-processing phase is feature extraction. The objective is to extract useful information and characteristics from each post to increase the likelihood of correctly classifying fallacies. In the first phase, the posts and their corresponding fallacy annotations were related to the natural language descriptions of the fallacies themselves. In this way, it was possible to identify certain fundamental characteristics to support classification. This step was performed manually without algorithmic support. To this end, the posts and their corresponding fallacy classifications were analyzed individually. The classifications were then related to the definition of each fallacy in order to identify common characteristics across classifications useful for simplifying the classification task for the model. Specifically, the features extracted in this phase are the following: `doubt_raising` (boolean), `unsupported_claim` (boolean), `analogy` (boolean), `flag_waving` (boolean), `generalization` (boolean), `profanity` (boolean), `thought_terminating` (boolean), `labeling` (list), `red_herring` (list), `slogans` (list), and `vague_terms` (list). Boolean features indicate the presence or absence of a specific linguistic pattern, while list features contain the extracted text spans matching that pattern. In total, 11 features are extracted for each post. The subsequent step consists of creating a structured prompt containing a clear definition of the problem (identification of the features characterising each fallacy) and an explanation of the expected output format. Subsequently, the prompt is used to instruct GPT-5 configured with enhanced reasoning capabilities in order to analyse the post, extract the relevant features, and save them in a dedicated JSON file. Some of the extracted features concern appeals to patriotism, tautological reasoning, discursive simplifications, or the use of ambiguous terms. The output of this phase was subsequently used in the first approach as additional information aimed at better describing the post to be classified and the corresponding example posts.

```
  "doubt_raising": boolean,
"unsupported_claim": boolean,
"analogy": boolean,   "flag_waving": boolean,
"generalization": boolean,
"profanity": boolean,
"labeling": [],
"red_herring": [],
"slogans": [],
"thought_terminating": boolean,
"vague_terms": []
```
} **Input:** `{comment}` **Output:** Valid JSON only. No explanations.

Figure 3: Prompt template for rhetorical feature extraction via LLM (Part 2)

System Prompt: Features extractor

Act as a rhetorical feature detector. Your task is to analyze a social media comment and return a JSON object detecting the following persuasion/manipulation techniques. **Output format:** JSON {

```
"personal_attack": boolean,
"authority": [],
"emotional_appeal": boolean,
"oversimplification": boolean,
"tautology": boolean,
```

Figure 2: Prompt template for rhetorical feature extraction via LLM (Part 1)

4.2. First approach - Large Language Models Optimised through RAG and Dynamic Few-Shot Prompting

For the first approach to the fallacy identification problem, we chose to employ a prompt enriched with examples using the Retrieval Augmented Generation (RAG) technique applied to a vector database containing the training dataset. The RAG technique is used to retrieve information from an external database in order to enhance the quality of responses generated by the model.

The entire system was developed with the aid of the agentic orchestration library by DataPizza. DataPizza is an Italian open-source project developed to address the limitations of existing traditional agentic frameworks. Its strengths include minimal abstraction, a debugging-oriented workflow, maximum flexibility, and customisation through the use of components that can be tailored to developers' requirements.[2].

As with a traditional vanilla RAG system, the post classification system is composed of a retriever, responsible for retrieving information from the external memory source, and a fallacy generator instructed via prompt. [4]

4.2.1. Retriever

The underlying idea is to use the post to be classified as a search query on the vector database to retrieve examples of similar posts and their corresponding classifications.

For implementation, the advanced open-source vector similarity search technology from Qdrant was employed. It offers a wide range of functionalities, including the ability to implement vector databases and similarity search algorithms. The collection configuration for test set data storage is as follows:

- `Embedding model`: text-embedding-3-small model from the OpenAI family;
- `Dimension`: the dimension corresponds to the dimensionality of the vectors generated by the model, namely 1536;
- `Format`: dense vectors are used;
- `Distance`: the metric for calculating similarity between the query and the vectors is cosine measure, which calculates the angle between vectors in the space.

As a preliminary step, we partitioned the original training set using a 70/30 split, allocating 70% of the data for training and the remaining 30% for validation. The first step in development is represented by the ingestion pipeline. During this phase, each post belonging to the training set was pre-processed, transformed into embeddings, and saved within the collection on Qdrant. Along with the embeddings, for each element, the features extracted during pre-processing, the fallacy labels assigned by annotators A and B, and the token-level classification from the .conll file were stored as metadata. For the retrieval phase, the post to be classified is converted into vector format and the top five most similar examples are retrieved using cosine measure.

[2]https://datapizza.tech/en/ai-framework/

4.2.2. Generator

The model family used in this phase is that of OpenAI. Specifically, tests were conducted on GPT-4.1 and GPT-5.

The second step consists of defining the prompt, containing all the instructions necessary to train the model. The prompt is composed of the following elements:

1. Problem definition
2. Description of the fallacy ontology: for each fallacy, a textual description is provided in order to give the model greater context.
3. Output definition: The expected output is a JSON containing a list of Prediction objects (sub-text, fallacy, confidence). During the post-processing phase, predictions with confidence level below 60% are discarded.
4. Suggestions to avoid errors: certain erroneous behaviours were identified during the development phase. Specific instructions were inserted to circumvent such issues, whilst avoiding leading the model into overfitting with overly specific indications.
5. List of similar examples: cosine similarity was used as the measure for identifying similar elements. The post to be classified is transformed into embeddings and the three most similar elements are identified. These elements, together with their respective classifications, are provided to the model within the prompt to guide it towards correct classification. In addition to these elements, the features extracted during the feature extraction phase are also included in the prompt. This was done to simplify the model's reasoning and guide it in establishing relationships between the features of the post to be classified and those of the posts suggested as similar examples. A possible future development for improving the system is represented by the use of features for retrieving similar examples, through their conversion into embeddings and the combination with the embeddings of the various posts.
6. Features of the post to be classified: using the feature extraction algorithm described above, the features of the post to be classified were extracted and included in the prompt as additional information.

System Prompt: Fallacy Classification

You are an annotator for comments containing fallacies. Given a social media post, identify the fallacies expressed in it. This is a multi-label classification task using 20 fallacy classes: Ad hominem, Appeal to authority, Appeal to emotion, Causal oversimplification, Cherry picking, Circular reasoning, Doubt, Evading the burden of proof, False analogy, False dilemma, Flag waving, Hasty generalization, Loaded language, Name calling or labeling, Red herring, Slippery slope, Slogan, Strawman, Thought-terminating cliché, Vagueness.

Input:

- List of semantically similar examples
- Comment to predict

Output format: JSON

```
[
{"text": "sub-phrase", "fallacy": "prediction", "confidence": "0-100%"},
...
]
```

Guidelines:

- Return only elements in the required JSON format
- Discard predictions with confidence below 60%

Figure 4: Prompt template for the first approach

4.3. Experiments

Tests were conducted on a Mac Mini M4, using the models via the OpenAI APIs. During the system evaluation phase, two different versions were analyzed. The first involved a simplified version of the output object returned by the model, without confidence scores and containing only the list of fallacies for each similar example included in the prompt. The second version, subsequently adopted during the final evaluation phases on the test set, is characterized by the use of confidence scores for each prediction, in order to assess how reliable the model considers a given prediction. Furthermore, this version also includes span-level classification (extracted from the .conll file) to show, for each example, the specific portions in which the fallacies in the training set were identified by the annotators.

Table 3
Model Experiments on the development set

Model	Precision	Recall	Micro F1-Score
GPT 5.1 simplified version	0.54	0.50	0.52
GPT 5.1 final version	0.61	0.56	0.56

4.4. Second approach - Large Language Models Optimised through Supervised Fine-Tuning

This section describes the supervised fine-tuning algorithm used to optimise pre-trained open-source models on the classification task. The results of runs 2 and 3 are subsequently derived from these models.

This algorithm is based on QLoRA (Quantized Low-Rank Adaptation) techniques and leverages the Hugging Face Transformers (4.56.1) and PEFT-Parameter-Efficient Fine-Tuning (0.18.0) libraries.

In summary, the algorithm is structured as follows:

- Data preparation and pre-processing: The data preparation and pre-processing phase begins with the definition of the list of target labels, namely the fallacies to be predicted. Subsequently, the reference dataset is loaded, pre-processed, and split into training set (70) and development set (30). The fallacy labels are then converted from textual format to one-hot encoding format, creating a binary column (values 0 or 1) for each fallacy. Furthermore, label weights are calculated to mitigate class imbalance during training. The subsequent step involves tokenising the posts and creating training batches through padding. Finally, the fallacy labels are converted into PyTorch tensors.
- Model configuration and PEFT: This phase begins with the definition of quantisation via the BitsAndBytes library (0.49.1). 4-bit quantisation is selected to implement the QLoRA (Quantized Low-Rank Adaptation) technique. This choice significantly reduces VRAM memory usage, enabling the fine-tuning of large-scale models on a single A100 GPU. Subsequently, the LoRA configuration is defined by specifying its parameters, particularly the target_modules (qproj, kproj, vproj, oproj), which represent the layers to be adapted. Finally, the model is loaded, quantised, and configured with LoRA.
- Training: This crucial phase defines the hyperparameters and initiates the model fine-tuning process. The main hyperparameters include:
  1. Learning rate: 10^{-4};
  2. Batch size per device: 4;
  3. Evaluation and saving strategy: steps;
  4. Load best model at end: true.

  Subsequently, a CustomTrainer (Python class) is initialised. This trainer is fundamental as it enables the management of a custom loss function that utilises the label weights calculated

previously. The use of weights serves to balance the impact of less frequent classes (rare fallacies) during model optimisation.
Finally, the training process is initiated and, upon completion, predictions on the validation set are computed and performance metrics are recorded, preparing the model for the subsequent evaluation phase on the test set.

4.4.1. Experiments

The following models were trained: gemma-3-12b-it, Meta-Llama-3.1-70B-Instruct, and Mixtral-8x7B-Instruct-v0.1. The fine-tuning process was carried out on a device with 235GB of storage and an A100 graphics card with 80GB. Once training was completed, the models were evaluated on the development set extracted from the training dataset. The model with the best result in terms of micro F1-score is Mixtral, followed by Gemma. For this reason, they were selected for the final submission of predictions on the official test set. Run 2 corresponds to the result of Gemma, whilst run 3 corresponds to that of Mixtral.

Table 4
Model Experiments on the development set

Model	Precision	Recall	Micro F1-Score
Mixtral-8x7B-Instruct-v0.1	0.58	0.71	0.64
Gemma-3-12b-it	0.53	0.71	0.61
Meta-Llama-3.1-70B-Instruct	0.74	0.47	0.57

5. Results

Table 5 summarises the final results of the predictions on the test set provided by the organisers of the FadeIT task. The test set portion of the dataset consists of 288 social media posts with a substantial balance in the topics addressed therein (climate change, public health, and immigration).

Table 5
Subtask A results

		avg			**a1**			**a2**		
	Pos.	P	R	F1	P	R	F1	P	R	F1
GPT-5	4/16	55.22	45.23	49.71	53.62	45.78	49.39	56.82	44.69	50.03
Mixtral-8x7B-Instruct-v0.1	6/16	51.19	44.26	47.45	48.09	43.40	45.63	54.28	45.13	49.28
Gemma-3-12b-it	8/16	58.70	38.44	46.44	54.78	37.46	44.49	62.61	39.43	48.39

6. Discussion

The approach that produced the best results is few-shot prompting via RAG. The fourth position obtained secured third place as a team in the final ranking.

Gemma and Mixtral, on the other hand, ranked further behind. In particular, Gemma-3, despite achieving higher precision compared to the other models, recorded the lowest score in terms of micro F1-score. Overall, the final scores below 50% attest to the intrinsic difficulty of the classification task, considering the very high number of classes involved (20). Another interesting aspect to highlight concerns the higher scores for annotator two, particularly with regard to the models optimised through fine-tuning.

The final results also demonstrate how the initially superior performance of the models produced through fine-tuning subsequently proved worse on the test set. It will be interesting to evaluate this aspect with respect to the distribution of fallacies in the training and test datasets.

7. Related works

The automatic recognition of fallacies and propaganda techniques has received increasing attention in recent years. SemEval, for instance, has introduced this topic in numerous tasks such as SemEval-2020 Task 11: Detection of Propaganda Techniques in News Articles [5] and SemEval-2021 Task 6: Detection of Persuasion Techniques in Texts and Images [6].

This study takes as its starting point the research conducted within SemEval-2025 Task 10 [7], which focused on the automatic identification of narratives, their classification, and the determination of the roles of entities involved in a dataset of news articles. This experimentation focused on the fine-tuning of Llama 3 on sub-task 1 of Entity Framing. The resulting system achieved third place for the English language dataset and led to the production of a scientific paper [8]. The promising results of this research constituted the foundation for the development of the second approach (Large Language Models Optimised through Supervised Fine-Tuning).

8. Conclusion

This study allowed for the implementation and evaluation of the performance of current foundation models available on the market on the task of classifying Italian social media posts. In particular, the best results were achieved through the RAG technique and few-shot prompting, leveraging the most performant OpenAI models available on the market. Through this approach, the final system produced achieved the highest score in terms of micro F1-score among the three runs submitted by the UNICA team for evaluation. The scores in the final ranking attest to the overall difficulty of Task A proposed by the organizers, thus leaving room for further research and analysis in this domain.

Acknowledgments

This work has been partially supported by the projects DEMON "Detect and Evaluate Manipulation of ONline information" funded by MIUR under the PRIN 2022 grant 2022BAXSPY (CUP F53D23004270006, NextGenerationEU) and SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU (NextGenerationEU (CUP F53C22000740007).

Declaration on Generative AI

During the preparation of this work, the author(s) used Claude Opus 4.5 in order to: Grammar and spelling check and Text Translation. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] A. Ramponi, S. Tonelli, FadeIT at EVALITA 2026: Overview of the fallacy detection in italian social media texts task, in: Proceedings of the Ninth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2026), CEUR.org, Bari, Italy, 2026.

[2] A. Ramponi, A. Daffara, S. Tonelli, Fine-grained fallacy detection with human label variation, in: L. Chiruzzo, A. Ritter, L. Wang (Eds.), Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language

Technologies (Volume 1: Long Papers), Association for Computational Linguistics, Albuquerque, New Mexico, 2025, pp. 762–784. URL: https://aclanthology.org/2025.naacl-long.34/. doi:10.18653/v1/2025.naacl-long.34.

[3] J. Tiedemann, S. Thottingal, OPUS-MT – building open translation services for the world, in: A. Martins, H. Moniz, S. Fumega, B. Martins, F. Batista, L. Coheur, C. Parra, I. Trancoso, M. Turchi, A. Bisazza, J. Moorkens, A. Guerberof, M. Nurminen, L. Marg, M. L. Forcada (Eds.), Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, European Association for Machine Translation, Lisboa, Portugal, 2020, pp. 479–480. URL: https://aclanthology.org/2020.eamt-1.61/.

[4] C. Huyen, AI Engineering: Building Applications with Foundation Models, O'Reilly Media, 2024. URL: https://www.oreilly.com/library/view/ai-engineering/9781098166298/.

[5] G. Da San Martino, A. Barrón-Cedeño, H. Wachsmuth, R. Petrov, P. Nakov, SemEval-2020 task 11: Detection of propaganda techniques in news articles, in: A. Herbelot, X. Zhu, A. Palmer, N. Schneider, J. May, E. Shutova (Eds.), Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics, Barcelona (online), 2020, pp. 1377–1414. URL: https://aclanthology.org/2020.semeval-1.186/. doi:10.18653/v1/2020.semeval-1.186.

[6] D. Dimitrov, B. Bin Ali, S. Shaar, F. Alam, F. Silvestri, H. Firooz, P. Nakov, G. Da San Martino, SemEval-2021 task 6: Detection of persuasion techniques in texts and images, in: A. Palmer, N. Schneider, N. Schluter, G. Emerson, A. Herbelot, X. Zhu (Eds.), Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), Association for Computational Linguistics, Online, 2021, pp. 70–98. URL: https://aclanthology.org/2021.semeval-1.7/. doi:10.18653/v1/2021.semeval-1.7.

[7] J. Piskorski, T. Mahmoud, N. Nikolaidis, R. Campos, A. Mario Jorge, D. Dimitrov, P. Silvano, R. Yangarber, S. Sharma, T. Chakraborty, N. Guimaraes, E. Sartori, N. Stefanovitch, Z. Xie, P. Nakov, G. Da San Martino, SemEval 2025 task 10: Multilingual characterization and extraction of narratives from online news, in: S. Rosenthal, A. Rosá, D. Ghosh, M. Zampieri (Eds.), Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025), Association for Computational Linguistics, Vienna, Austria, 2025, pp. 2610–2643. URL: https://aclanthology.org/2025.semeval-1.331/.

[8] M. Fenu, M. Sanguinetti, M. Atzori, DEMON at SemEval-2025 task 10: Fine-tuning LLaMA-3 for multilingual entity framing, in: S. Rosenthal, A. Rosá, D. Ghosh, M. Zampieri (Eds.), Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025), Association for Computational Linguistics, Vienna, Austria, 2025, pp. 1456–1464. URL: https://aclanthology.org/2025.semeval-1.192/.

A. Online Resources

The code of the work conducted is available at the following link:

- https://github.com/Matteofenu65891/task-A—FadeIt.git

For clarity of reading, all prompts described in this paper have been translated into English. The original Italian versions are available in the repository.