



AI services for COntinuous trust in emerging Digital Enviroments

D6.3 Prototype version of Personal Companion for media professionals service

Grant Agreement no.	101135437
----------------------------	-----------

Project Title	AI-CODE - AI services for COntinuous trust in emerging Digital Environments
----------------------	---

Contractual Due date	31.05.2025
-----------------------------	------------

Actual submission date:	09.06.2025
--------------------------------	-------------------

Lead beneficiary:	ATC
--------------------------	-----

Deliverable number	D6.3
Deliverable Title:	Prototype version of Personal Companion for media professionals service
Description:	Lorem ipsum ☐ R - Document, report ☒ DEM - <i>Demonstrator, pilot, prototype</i> ☐ DEC - <i>Websites, patent filings, videos, etc.</i> ☐ OTHER ☐ ETHICS – <i>Ethics requirement</i> ☐ ORDP – <i>Open Research Data Pilot</i>
Type/Nature	
Dissemination Level	☒ Public – fully open ☐ Sensitive — limited under the conditions of the Grant Agreement ☐ EU classified — RESTREINT-UE/EU-RESTRICTED, CONFIDENTIEL-UE/EU-CONFIDENTIAL, SECRET-UE/EU-SECRET under Decision 2015/444. For items classified under other rules (e.g. national or international organization), please select the equivalent EU classification level.
Work Package number	6
Work Package leader	RU
Author(s) <i>in alphabetical order</i>	Giotis, George (ATC) Kanellos, Ilias (ATC) Lazaridis, Iosif (ATC) Tsafonis Thanasis (ATC) Tsalapati, Eleni (ATC)
Internal Reviewers <i>in alphabetical order</i>	Mathias Becker (EIPCM) Martha Larson (RU)

AUTHORS

Name	Partner	E-Mail
Eleni Tsalapati	ATC	E.Tsalapati@atc.gr
Ilias Kanellos	ATC	I.Kanellos@atc.gr
Giorgos Giotis	ATC	G.Giotis@atc.gr
Iosif Lazaridis	ATC	I.Lazaridis@atc.gr

Thanasis Tsafonis

ATC

T.Tsafonis@atc.gr

HISTORY OF CHANGES

Version	Date	Changes
0.1	27/03/2025	Add ToC
0.2	13/05/2025	Drafts of all sections finalized
0.3	21/05/2025	Comments and suggestions by internal reviewers added
0.4	27/05/2025	Feedback from internal reviewers integrated
0.5	28/05/2025	Review by RU
0.6	28/05/2025	Review by EIPCM
1.0	03/06/2025	Final version prepared

TABLE OF CONTENTS

Table of Figures.....	5
Table of Tables	6
1. Executive summary	7
2. Introduction	8
2.1 Goal and Motivation.....	8
2.2 Scope of the Prototype.....	9
3. Scientific Rationale.....	13
4. System Description	17
4.1.1 Prototype Architecture	17
4.2 Technical Description	18
4.2.1 Functional description.....	18
4.2.2 Technical Implementation	20
5. System Validation	31
5.1 Data-centric evaluation results	31
5.1.1 Evaluation of Claim Matching to Debunked Claims Component	31
5.1.2 Evaluation of Matching to Other Articles Component.....	32
5.2 Ethics by Design	33
6. Conclusion and Future Outlook	35
7. References	37

Table of Figures

Figure 1 Personal Companion Architecture	17
Figure 2 Claim Matching to Debunked Claims.	18
Figure 3 The Logical Fallacy Detection Result.	19
Figure 4 Hate Speech Detection Result.....	20
Figure 5. LLM / prompting pipeline for claim detection.	21
Figure 6 High-level system architecture of the Claim Matching component	22
Figure 7 Agentic AI workflow.	24
Figure 8 Web Search Workflow	26
Figure 9 Deployment Architecture of Personal Companion	30

Table of Tables

Table 1 Functional Specifications for Personal Companion and their Implementation	9
Table 2 Role-based Functional Specifications for Personal Companion to be Implemented	11
Table 3 Accuracy Results for Claim Matching.	31
Table 4 Accuracy results of the examined models.....	33
Table 5 Ethical design considerations based on the requirements for Trustworthy AI.	34
Table 6 Ethics Advisory Board Recommendations and Implementation Measures.	35

1. Executive summary

This deliverable presents the prototype version of the Personal Companion for media professionals, developed under Task T6.3, of WP6, of the AI-CODE project. Designed as a web-based tool for media professionals, the Personal Companion assists in identifying and analyzing credibility signals within news articles, such as factual inaccuracies, logical fallacies, and hate speech in a trustworthy way. Leveraging Large Language Models (LLMs) and a modular architecture, the tool supports both closed- and open-world fact-checking by integrating a database with debunking reports (i.e., reports that aim to expose and correct false, misleading, or deceptive claims) and web-based claim matching. The prototype is built using the microservices architecture, consisting of independently deployed components. This modular architecture and implementation allows for a more flexible uptake and exploitation strategy, since all modules can also be easily deployed separately for exploitation cases where this may be of interest.

The tool's design and development were guided by user requirements gathered through co-creation workshops and bilateral meetings with media professionals. Hence, the design and the implementation of the prototype is based on user needs laid out in D3.1 (User-centred requirements analysis) and the corresponding functional specification provided in D3.2 (Functional specification and technical requirements). These insights shaped a suite of core functionalities: detection of debunked claims, identification of fallacies, hate speech classification, and matching claims to external sources. Moreover, they shaped user interface flavours, based user role. Data-centric evaluation results demonstrate high accuracy in detecting disinformation signals, with the claim matching service achieving 88% accuracy, the hate speech detection service reaching over 91% and the fallacy detection service reaching 86% accuracy. Each module has been designed for human-in-the-loop oversight, in alignment with the Ethics Guidelines for Trustworthy AI.

Future work includes the broader integration of ethical oversight mechanisms such as model cards, continuous improvements of the various functionalities, development of different profiles tailored to specific roles, and validation of the prototype in real-world journalistic workflows under task T8.3. The Personal Companion stands as a practical and trustworthy AI-powered assistant, enhancing editorial judgement, promoting media integrity, and contributing to the broader fight against disinformation.

2. Introduction

Disinformation in news media has become increasingly complex, especially in the digital age, where false narratives spread rapidly across social and news media platforms (Bertani, Mazzeo, & Gallotti, 2024). Disinformation is often amplified by algorithms that prioritize reader engagement over content accuracy, making it difficult for fact-checkers to counter false claims before they spread. Political polarization further complicates the issue, as people are more likely to believe and share information that aligns with their existing beliefs, even if it is misleading. Often, disinformation techniques are exploited to manipulate public opinion and undermine trust in credible journalism. The rise of AI-generated content, such as deepfakes and fabricated news articles, adds another layer of difficulty in distinguishing fact from fiction. To combat these challenges, media organizations must strengthen verification processes, while governments, tech companies, and the public must work together to promote media literacy and critical thinking (Shahid, et al., 2024).

This deliverable describes a prototype of AI-CODE's Personal Companion for understanding disinformation for media professionals service, which is a result of the task T6.3 within WP6. The prototype is publicly accessible and can be found in: <https://companion.lab.atc.gr/sign-in>.

The rest of this deliverable is structured as follows: The remaining part of Section 2 introduces the goals and motivation behind the *Personal Companion for Countering Disinformation* service. It also defines the scope of the developed prototype, outlining its connection to the requirements and specifications detailed in Deliverables D3.1 and D3.2, as well as the functionalities and the relevant technical recommendations from D2.3 that have informed the design of the service. Section 3 provides an overview of related work within the current scientific landscape, building on the analyses presented in Deliverables D2.1 and D2.2. All technical aspects of the prototype are presented in Section 4. It presents the overall system architecture, elaborates on the individual functionalities and their usability, and details the data-centric experiments conducted. Preliminary data-centric evaluation results are presented in Section 5. Finally, Section 6 outlines future directions and discusses the expected developments in WP7 and WP8.

2.1 Goal and Motivation

Fact-checking plays a crucial role in combating disinformation. It involves verifying the truthfulness of claims made in public discourse and sharing the findings (Amazeen & Michelle, 2019). With the rapid spread of disinformation and misinformation across social media and news media platforms, the demand for timely and comprehensive fact-checking has never been greater. However, this process is inherently complex and time-intensive, requiring multiple steps—from identifying claims in text to delivering final assessments. With the advent of recent advancements in AI, and especially of Large Language Models (LLMs), research is now exploring ways to integrate AI-based solutions into the fact-checking workflow. Nevertheless, fully automating the process with AI presents risks, as it could compromise the journalistic standards that underpin fact-checking processes (Nakov, et al., 2021). Thus, rather than replacing human expertise, AI should be leveraged to enhance human decision-making (Choi & Ferrara, 2024).

The *Personal Companion for media professionals* (Personal Companion) is designed to assist journalists in fact-checking the articles they write or consume by identifying key credibility signals. These signals—such as content factuality, the use of persuasion techniques, or the presence of logical fallacies—are small but critical indicators that help assess the trustworthiness of information (Srba, et al., 2024). Leveraging the power of LLMs and deep learning techniques, Personal Companion focuses on detecting three main types of credibility signals: factual inaccuracies (if these have already been debunked), logical fallacies, and hate speech.¹ Beyond

¹ The service Personal Companion (Task 6.3) was reoriented. In the GA, it was described as a rule-based system, which

surface-level analysis, it enhances claim verification by automatically checking whether detected claims appear in reputable online sources, such as Google FactChecks, or other user-trusted media outlets. Additionally, it searches the new-generation social media platform, Bluesky to obtain a rough estimation of disinformation spread.

Feedback from the co-creation workshops conducted within T3.1 and bilateral meetings with the partner DW showed that the transparent, interpretable, and user-friendly nature of the tool is not only suited for journalists and fact-checkers of all experience levels but it can also serve a wider spectrum of media professionals, including asset managers and editorial staff. For example, it can enable rapid assessment of content risk—such as hate speech or reasoning flaws—prior to publication or archiving. Overall, Personal Companion offers a promising support tool for enhancing editorial judgment and strengthening routine efforts to identify and mitigate disinformation.

2.2 Scope of the Prototype

This prototype implements the core functionalities outlined in the early user requirements defined in D3.1 and the corresponding functional specifications detailed in D3.2. Both user requirements and functional specifications resulted from the two co-creation workshops performed with media professionals, which involved demonstration of the tool’s concept and of early mockups. Table 1 provides an overview of how the current version of the prototype addresses these requirements.

The purpose of this first prototype is to implement the core set of functionalities in a form that enables easy user testing, to identify potential improvements through an iterative development process. In line with the requirements and scenarios of use from D3.1 and D3.2 the focus is on allowing users to assess the veracity of the articles they produce or consume. The main functionalities of the prototype include: 1) claim matching to debunked claims (i.e., if the provided article contains claims that have already been debunked), 2) matching claims to other articles (i.e., if the provided article contains claims that have been circulated in trusted media outlets and their spread in Bluesky), 3) logical fallacy detection, and 4) hate speech detection.

How the specific requirements —based on the functional specification from D3.2— have been addressed in this prototype is provided in Table 1 below. Specifically, the abovementioned functionalities are reflected through the functional specifications #1-#4. Further, other peripheral requirements have been implemented, including the view and browsing of the Debunked Claims database (#8), support of non-English languages (for the time being, the UI is translated in Romanian upon selection) (#9) and the inclusion of a toolbox that allows users to choose the desired functionality (#11).

The role-based functional specifications, presented in Table 2 (e.g., allow users with elevated rights to process the database with the debunked claims), will be implemented by M27 to allow enough time for validation within T8.3 (which ends at M36).

Table 1 Functional Specifications for Personal Companion and their Implementation

Functional Specification (based on D3.2)	Implementation
--	----------------

would utilize Reinforcement Learning (RL) and long short-term memory networks (LSTM), however, to leverage the rapid evolution of technologies, it shifted from RL to LLMs. Also, previously, it was focused on guiding the user in arriving at logical conclusions about the factual correctness and reliability of a statement at hand, however extensive discussions with media partners (DW) lead to the conclusion that the current solution is more applicable to their needs.

1	The system shall provide a logical fallacy detection functionality, which will detect logical fallacies in the text and provide the type of fallacy along with an informative link.	The prototype allows users to copy and paste the text they want to assess into a textbox (1.1). On the left side of the main page, there is a “Logical Fallacy Detection” option with a checkbox that users can select to enable this functionality. When the user clicks the check button, any sentences identified as fallacious within the textbox are highlighted (1.2). To the right of the textbox, a series of accordions labeled “Logical Fallacy” appear—one for each identified fallacious sentence. Clicking on an accordion highlights the corresponding sentence in the textbox and displays the type of logical fallacy within the accordion (1.3). Additionally, the “i” icon within each accordion allows the user to view the definition of the specific fallacy type (1.4).
1.1	The system shall provide a textbox where the users can copy-paste the text they want to process.	
1.2	The system shall highlight all logical fallacies that it can identify in the text.	
1.3	The system shall provide the type of fallacy for each identified fallacy.	
1.4	The system shall provide the option for the user to check the definition of the fallacy.	
2	The system shall provide a hate speech detection functionality, which will detect the hate speech sentences in the text and provide the type of hate speech.	The prototype allows users to copy and paste the text they want to assess into a textbox (2.1). On the left side of the main page, there is a “Hate Speech Detection” option with a checkbox that users can select to enable this functionality. When the user clicks the check button, any hate speech sentences within the textbox are highlighted (2.2). To the right of the textbox, a series of accordions labeled “Hate Speech” appear—one for each identified hate speech sentence. Clicking on an accordion highlights the corresponding sentence in the textbox and displays the type of hate speech within the accordion (2.3).
2.1	The system shall provide a textbox where the users can copy-paste the text they want to process.	
2.2	The system shall highlight all hate speech sentences that it can identify in the text.	
2.3	The system shall provide the type of hate speech for each detected sentence.	
3	The system shall provide a claim detection and matching functionality that matches claims appearing in text with debunked claims stored in the database “Debunked Claims” (functional specification #10).	The prototype allows users to copy and paste the text they want to assess into a textbox (3.1). On the left side of the main page, there is a “Matching to debunked claims” option with a checkbox that users can select to enable this functionality. When the user clicks the check button, in this first prototype, if the main claim of the article within the textbox is semantically similar to a debunked claim from the “Debunked Claims” database then it is highlighted (3.2). To the right of the textbox, an accordion labeled “Debunked Claim” appears (3.3). Clicking on the accordion highlights the corresponding sentence in the textbox and displays the URL of the debunking report within the accordion. Also, by clicking on “View Report” a pop-up appears displaying a summary of the debunking report (3.4). In T7.3 this functionality will be extended to support multiple claims.
3.1	The system shall provide a textbox where the users can copy-paste the text they want to process.	
3.2	The system shall highlight the detected claims in the text.	
3.3	The system shall provide the URL of the debunking report for each detected claim.	
3.4	The system shall provide the summary of the debunking report (if it exists) upon request.	

4	The system shall provide a claim detection and matching functionality that matches claims appearing in text with claims appearing on external webpages	The prototype allows users to copy and paste the text they want to assess into a textbox (4.1). On the left side of the main page, there is a “Matching claims to other articles” option with a checkbox that users can select to enable this functionality. When the user clicks on the check button, the detected main claim within the textbox that is semantically similar to claims appearing in other webpages is highlighted (4.2). To the right of the textbox, an accordion labeled “Matched Resources” appears (4.3). Clicking on the accordion highlights the corresponding sentence in the textbox and displays the URL of the external resource within the accordion (4.4). In T7.3 this functionality will be extended to support multiple claims.
4.1	The system shall provide a textbox where the users can copy-paste the text they want to process.	
4.2	The system shall highlight the detected claims in the text.	
4.3	The system shall provide the URL of the matching web resource for each detected claim.	
8	The system shall allow the user to navigate and process the database with the debunked claims.	On the menu bar at the top of the main page of Personal Companion there is a clickable link named “Debunked Claims”. When clicked, a new page appears with a table containing the data from the Debunked Claims database. Each row of the table contains: the debunked claim, the summary of the debunking report, the URL of the debunking report or the text of the debunking report, date of the debunking report and the topic. On the menu bar at the top of the main page of Personal Companion, there is a clickable link named “Similar Claims”. When clicked, a new page appears with a search box. The user can insert a claim or a word and a table with semantically similar claims appears below (8.1).
8.1	For users with standard rights: The system must display a list of debunked claims. For each debunked claim the following information shall be displayed: the claim text, the debunking report, and optionally a summary of the debunking report, the date of debunking, and the topic. The user should be able to search for specific claims.	
9	The service should support also non-english languages.	On the top right of the main page, the user can select the language of preference (9.1). Currently, English and Romanian are supported. Once a language is selected, the UI changes automatically to that language (9.2).
9.1	The system must provide a language selector in a prominent location (e.g., header/footer).	
9.2	When the user selects a language all UI text must appear in that language.	
11	The system shall provide a toolbox of all allowed functionalities (based on user type) on the initial page, allowing users to select the ones they need.	On the left of the main page all available functionalities (based on role) appear with checkboxes for selection.
11.1	The initial page must display clearly in separate section all available functionalities with clickable boxes for selection.	

Table 2 Role-based Functional Specifications for Personal Companion to be Implemented

Functional Specification (based on D3.2)	
5	Based on user role, the system shall provide an automated generation of report functionality.
5.1	The system shall provide a textbox for the insertion of multiple URLs and a textbox for the insertion of a claim. The system shall generate a report for the input claim based on the provided URLs.
5.2	The report shall be editable.

5.3	
6	The service shall allow different functionalities per role:
6.1	Users must be authenticated to access the service. Role-based access control (RBAC) must be implemented. The types of users will be: - Journalist Students / Journalists in Training - Journalists - Editors - Fact-checkers - Social Media Managers/Community Editors - Trainers/Curriculum Developers.
6.2	Users with elevated rights must be associated with a specific organization upon registration or account setup.
7	The system shall allow the user to provide specific external webpages for similarity-checking
7.1	For users with elevated rights: Under functionality #4 the user shall be able to provide a set of URLs for searching
8	The system shall allow the user to navigate and process the database with the debunked claims
8.2	For users with elevated rights: - Must be able to add debunking reports by uploading spreadsheets. - They must see an option to mark any existing debunking report as invalid.
10	The database with the debunked claims ("Debunked Claims" database) should be optionally shared between people of the same organization
10.1	All debunking reports created by users within an organization should be optionally visible to all other users in the same organization.
10.2	The organizational database should not be accessible by externals.

Next, the implementations of the recommendations from D2.3 that are relevant to Personal Companion are briefly discussed.

"5.1.1 Use official means of fetching data from next-generation social media"

Personal Companion's fetched data from the Bluesky API is accessed through GET and POST methods of the HTTP protocol through Bluesky's official open APIs, in particular the open search functionality provided.²

"5.1.2 Minimise data collection and apply anonymization"

Personal Companion will collect temporarily only Bluesky posts that are semantically similar to given claims. Bluesky adopts the AT Protocol³, which is an *open*, decentralized network for building social applications. Partner KUL will be consulted for the required anonymization process.

"5.1.3 Limit data retention"

Personal Companion does not store data from social media platforms.

"5.1.4 Apply privacy-aware data processing"

Same approach as for the recommendation 5.1.2.

"5.1.5 Utilize data aggregation to ensure privacy"

Same approach as for the recommendation 5.1.2.

"5.2.2 Evaluate robustness of detection methods to out-of-distribution data"

This recommendation will be applied during the validation task T8.3.

"5.2.4 Consider trade-off detection performance and inference costs"

This recommendation has already been implemented and documented in D2.2 for the hate speech detection

² <https://docs.bsky.app/docs/api/app-bsky-feed-search-posts>

³ [AT Protocol](#)

module, where multiple architectures and models were evaluated. Similarly, a variety of models have been tested for both the “*claim matching to debunked claims*” and “*claim matching to web sources*” modules, with results detailed in Section 5. For instance, this evaluation resulted in selecting the second best model in terms of performance due to less needed tokens/lower cost. As part of our immediate next steps, we plan to explore alternative architectures and models for the fallacy detection module as well.

“5.2.5 Classification models should reflect and be adapted to changes in data”

This recommendation mostly applies to hate speech detection module, as hate speech detection has shown to be vulnerable to obfuscation techniques (Aggarwal & Zesch, 2022). To address this, the module will undergo continuous evaluation using both newly published datasets from the literature and real-world data collected as part of the “*Implementation of the Service Validation*” task (T8.3).

“5.2.6 Classification models should consider human in the loop approach”

This recommendation will be implemented by M24, coinciding with the start of the validation phase, to enable automated user feedback collection with regard to the accuracy of the Personal Companion’s models.

“5.3.3 Provide tools that enable critical analysis of the reliability and trustworthiness of LLM outputs.”

This recommendation will be facilitated through the integration of Model Cards developed in T6.2/T7.2.

“5.4.1 Services should provide an appropriate level of explainability.”

The required level of explainability will be determined and addressed within T8.3.

“5.4.4 Use ClaimReview tagging system to automatically retrieve and utilise recent fact-checks”

Personal Companion uses Google Fact Check Explorer API that aggregates ClaimReview data to retrieve fact-checked claims.

In the next section, the scientific background of Personal Companion is presented.

3. Scientific Rationale

With the emergence of recent advancements in AI, and especially of LLMs, research is now exploring ways to integrate AI-based solutions into the fact-checking and hate speech detection workflow; however, to the best of our knowledge, most of existing technologies focus on certain credibility signals in isolation, for instance Zanartu et al. (2024) focus on disinformation detection for climate change by detecting only logical fallacies. Personal Companion offers the first end-to-end holistic solution that inspects a provided text for multiple types (debunked claims, fallacies, hate speech) of disinformation signals. Additionally, it enhances human trust by providing further information on the outputs (e.g., the respective debunking report, the type of logical fallacy), enabling, in this way, human oversight. The key technologies used in Personal Companion include claim detection and matching, hate speech detection, and fallacy detection. Next, we describe the scientific rationale for each.

Claim detection and matching. A rather common approach for fact-checking is to reuse already known fact checks (Balalau, et al., 2024). For outputs that are long and rich in information, this is facilitated by first breaking down the text into simpler claims, then retrieving supporting evidence and independently verifying each claim, e.g., (Min, et al., 2023), (Hu, Long, & Wang, 2024). The success of these “decompose-then-verify” systems heavily depends on how accurately the claims are extracted: distortions of the original content or omissions of important facts can lead to misleading or partial conclusions. After the advent and spread of LLMs, a body of literature has grown up around the topic of claim detection and/or extraction, e.g., (Metropolitansky & Larson, 2025) (Deng, Schlichtkrull, & Vlachos, 2024), (Revanth G., et al., 2022). The task

is not straightforward or uniquely identified, and different studies focus on different aspects and issues around what constitutes a claim and how it can be useful, for instance Scirè et al. (2024) focus on the atomicity of claims, while Stambach et al. (2023) aim to identify “check-worthy” claims. However, as noted by Metropolitansky and Larson (2025) and Ni et al. (2024) check-worthiness can be subjective. In the context of Personal Companion, the claim detection task is defined as follows: given an input text, highlight the passage(s) of the text containing its claim(s).

The implementation of claim detection for Personal Companion was designed after examining examples of manual claim detection from social media posts, provided with the guidance of journalists and fact-checkers from ATC’s network (and remain to be validated with T8.3). From the examples provided, we identified the following key crucial guidelines for identifying the claim(s) in a text:

1. Verifiability: the focus should be on factual statements and data, and exclude opinions, speculation, defamation, etc.
2. Figures & Data: focus should be given to passages presenting numbers, percentages, dates, and measurable quantities in general.
3. Intention: the focus should be on the central idea of the text, filtering out supporting evidence, anecdotes, or emotional appeals.
4. Importance Ranking: a text may include multiple claims, but the most important ones should be identified.
5. Summarization: a single claim may span over the entire text. In this case, it should be rewritten as a single, clear, stand-alone sentence.
6. Keyword indicators: focus should be put on domain-specific keywords and terms (e.g., “chemtrails”, “ban”), which signal factual assertions.

While embedding-based semantic matching is effective, relying on a static similarity threshold can be problematic due to variability across domains, models, and linguistic expressions. Fixed thresholds may lead to false positives or negatives and fail to capture context-sensitive nuances in language. More robust approaches include learned similarity functions or ranking-based systems, which have shown better performance in tasks like claim verification and semantic textual similarity (Reimers & Gurevych, 2019; Thorne et al., 2018). Additionally, agent-based solutions, such as those built using frameworks like LangGraph⁴, offer promising alternatives. These systems enable the orchestration of multiple reasoning steps—such as iterative clarification, contextual search, and reranking—through modular, stateful agents (Mavroudis, 2024). This allows similarity checking to move beyond single-pass embedding comparisons, supporting more adaptive, multi-step matching pipelines that better reflect how humans validate claims in practice. For Personal Companion, we have employed an agenting approach, utilizing LangGraph and LangChain⁵.

Hate Speech Detection. The definition of hate speech remains contested globally, with no universal agreement on its precise boundaries due to the difficulty in distinguishing it from protected speech. For instance, while the American Bar Association notes that incitement to criminal acts may be prosecuted as hate crimes, it avoids offering a formal definition. Instead, various organizations and scholars provide working definitions that highlight common elements. The American Constitution Encyclopaedia defines hate speech as attacks based on identity attributes like race, religion, or gender (Nockleby, Levy, Karst, & Mahoney, 2000). Facebook describes it as direct attacks—such as dehumanizing speech or calls for exclusion—based on protected characteristics, including caste and immigration status⁶. Twitter prohibits promotion of violence or threats based on similar identity markers⁷. Davidson et al. frame it as language expressing hatred or intending

⁴ www.langchain.com/langgraph

⁵ <https://www.langchain.com/>

⁶ Community Standards. Available from: https://transparency.meta.com/en-gb/policies/community-standards/?source=https%3A%2F%2Fwww.facebook.com%2Fcommunitystandards%2Fobjectionable_content

⁷ Hateful conduct policy. Available from: <https://help.x.com/en/rules-and-policies/hateful-conductpolicy>

to humiliate a group (Davidson, Warmesley, Macy, & Weber, 2017), while de Gilbert et al. emphasize deliberate, identity-motivated attacks (De Gibert, Perez, García-Pablos, & Cuadros, 2018). Fortuna et al. expand the concept to include subtle or humorous forms that incite hatred or diminish targeted groups (Fortuna & Nunes, 2018). In this research work, we follow the definition provided by the United Nations which is along the lines of the aforementioned definitions: “Hate speech is defined as any form of communication—spoken, written, symbolic, or digital—that incites hatred, discrimination, or violence against a specific group or individual based on protected characteristics. These characteristics include race, ethnicity, nationality, disability, religion or belief system, gender, gender identity, or sexual orientation”.

Recent research has explored the use of LLMs for zero-shot and few-shot hate speech detection, enabling effective performance even in domains or languages with limited labeled data (Arya, et al., 2024), (Schmidhuber & Kruschwitz, 2024). This is particularly valuable for under-resourced languages and emerging online communities. However, in general, extensive use of LLMs also pose significant challenges. Their reliance on large-scale pretraining raises concerns about bias in training data, which can lead to skewed predictions (Yang, et al., 2024). Additionally, the high computational costs and environmental impact of training and deploying such models create barriers, especially for smaller organizations (Myers, et al., 2024). For this purpose, multiple architectures exploring small language models (SLMs) and LLMs have been evaluated and the results have been documented in D2.3. For the evaluation, datasets from the literature were utilized, but to increase the robustness of the hate speech detection service, both ATC and DW also developed two new datasets (documented in D2.3). For Personal Companion, the architecture with the best performance was employed. Specifically, a two-step approach has been adopted, with a binary classifier first (hate-non-hate) and then a multiclass classifier -utilizing llama-3.3-70b-versatile.

Fallacy Detection. Logical fallacies are forms of reasoning that may appear valid but are, in fact, flawed. They arise when an error in argumentation is made—either unintentionally or as a rhetorical strategy to persuade an audience. In everyday language, the term “fallacy” often refers to any widely accepted belief that is actually false, without necessarily implying a flaw in logical reasoning. However, in argumentation research, the term “logical fallacy” is often shortened to “fallacy,” recognizing that not all reasoning errors strictly violate formal logic.

Identifying fallacies in everyday discourse is crucial not only for limiting the spread of non-credible content but also for fostering critical thinking skills. This, in turn, helps address flawed argumentation at its root and supports healthy, democratic debate. Despite its significance, fallacy detection remains a challenging task in natural language processing (NLP). Key difficulties include accounting for variability in human annotations, the mistaken assumption that only one fallacy type can occur per text, and the common practice of labeling at the document or post level—overlooking the fact that different segments of a single text may contain multiple, distinct fallacies (Habernal, Wachsmuth, Gurevych, & Stein, 2024). For this purpose, within T2.2, we addressed the above issues by proposing a novel fallacy detection dataset composed of 3,425 short news articles in English, comprising fallacies of 12 different types, to check if LLMs can identify fallacies in realistic and complex textual environments. Then, the performance of zero-shot and fine-tuned Llama-2 was tested. Fine-tuned Llama-2 returned promising results and, hence, for the prototype version of Personal Companion, we have employed this solution. However, within the T7.3, we plan to investigate the performance of more LLMs, taking into consideration their environmental impact.

Finally, it is worth noting that, to the best of our knowledge, research relevant to hate speech detection and fallacy detection mainly focuses on the use of LLMs for small texts (such as media posts), e.g., Hussain & Aslam (2024). However, hate speech and logical fallacies are also prevalent in larger contexts (Hu, Long, & Wang, 2024), such as media articles, in which case the problem can become highly challenging due to potential implied information between the paragraphs (hence, the problem cannot be simply reduced to text segmentation). Our evaluation results, documented in D2.2 and D2.3, show that both services cope well with

longer texts.

4. System Description

4.1.1 Prototype Architecture

Personal Companion is a web-based service that aims to assist journalists and fact-checkers in assessing the truthfulness and reasoning involved in a media article that they produce/consume. Personal Companion is implemented as a web-based application providing end-users with a single point of entry to all service modules.

All functionalities are implemented as independent APIs, which are called by the Personal Companion's web interface. However, the APIs' independent deployment allows for all functionalities to be made accessible to third parties, allowing for the sharing of resources developed by AI-CODE among the project's partners. This modular architecture and implementation also allow for a more flexible uptake and exploitation strategy, since all modules can also be easily deployed separately for exploitation cases where this may be of interest.

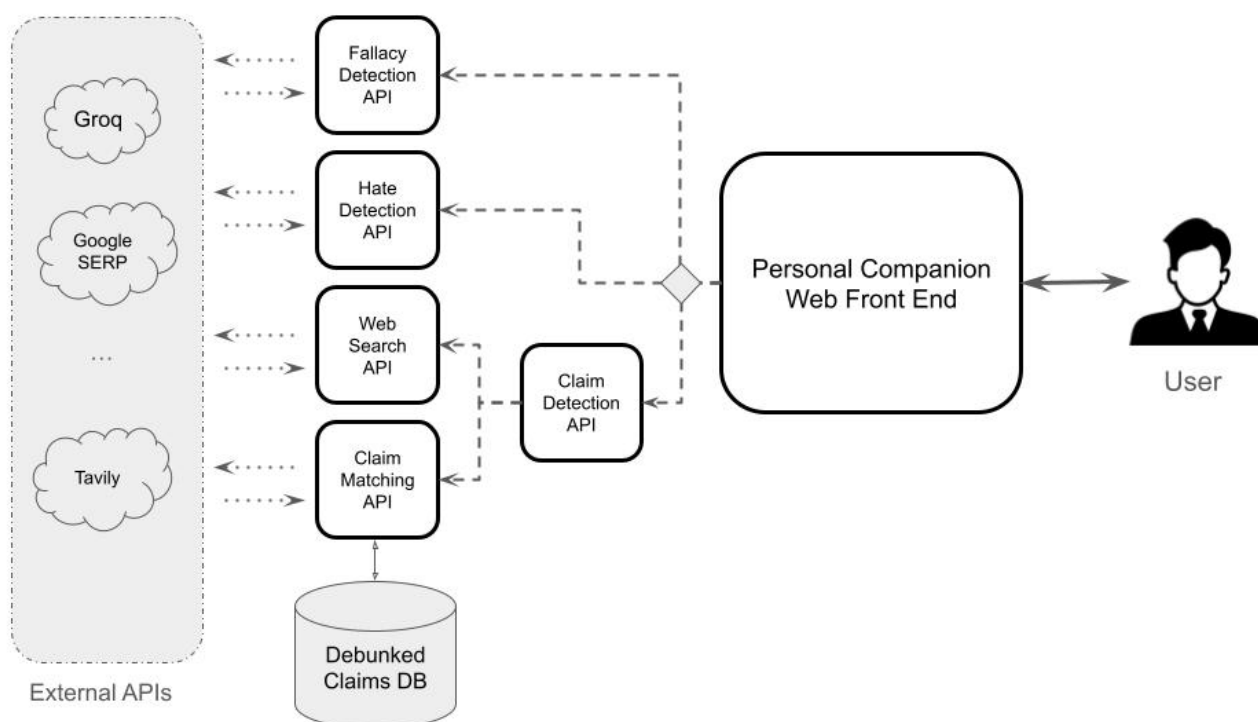


Figure 1 Personal Companion Architecture

The architecture diagram illustrates a web-based personal companion tool designed to help users assess the credibility and quality of online content. At the center of the system is the Personal Companion Web Front End, which serves as the main interface through which the user interacts with the platform. When a user submits content—such as text or links—the front end communicates with a set of internal APIs to analyze the input from multiple perspectives. These include the Fallacy Detection API, which identifies logical fallacies; the Hate Detection API, which scans for hate speech; and the Claim detection API, which identifies factual claims in the content.

The Claim Detection API is not provided as a separate option to the user. Instead, it is invoked as a prerequisite to the claim matching and web search APIs, which require a claim (and not the complete text) as input. Once claims are detected, the Claim Matching API compares them against entries in the Debunked

Claims Database (which contains the debunking reports, i.e., reports that aim to expose and correct false, misleading, or deceptive claims) to verify their accuracy. To support this verification, the Web Search API interfaces with external APIs such as Groq, Google SERP, and Tavily to retrieve relevant information from the web. These external data sources supplement the fact-checking and analysis process, enhancing the system’s reliability and depth. The Claim detection API also works in tandem with the Web Search API to enrich the claim detection process. Overall, this system integrates automated content analysis, real-time fact-checking, and user interaction to support more informed and critical media consumption.

4.2 Technical Description

4.2.1 Functional description

This first release of Personal Companion consists of four main functionalities: 1) matching to debunked claims, 2) matching claims to other articles, 3) logical fallacy detection, and 4) hate speech detection. A user can assess an article by selecting one or multiple functionalities at the same time.

Next, we provide the functional description of these four functionalities, and then a detailed technical description of Personal Companion’s technical infrastructure follows in Section 4.2.

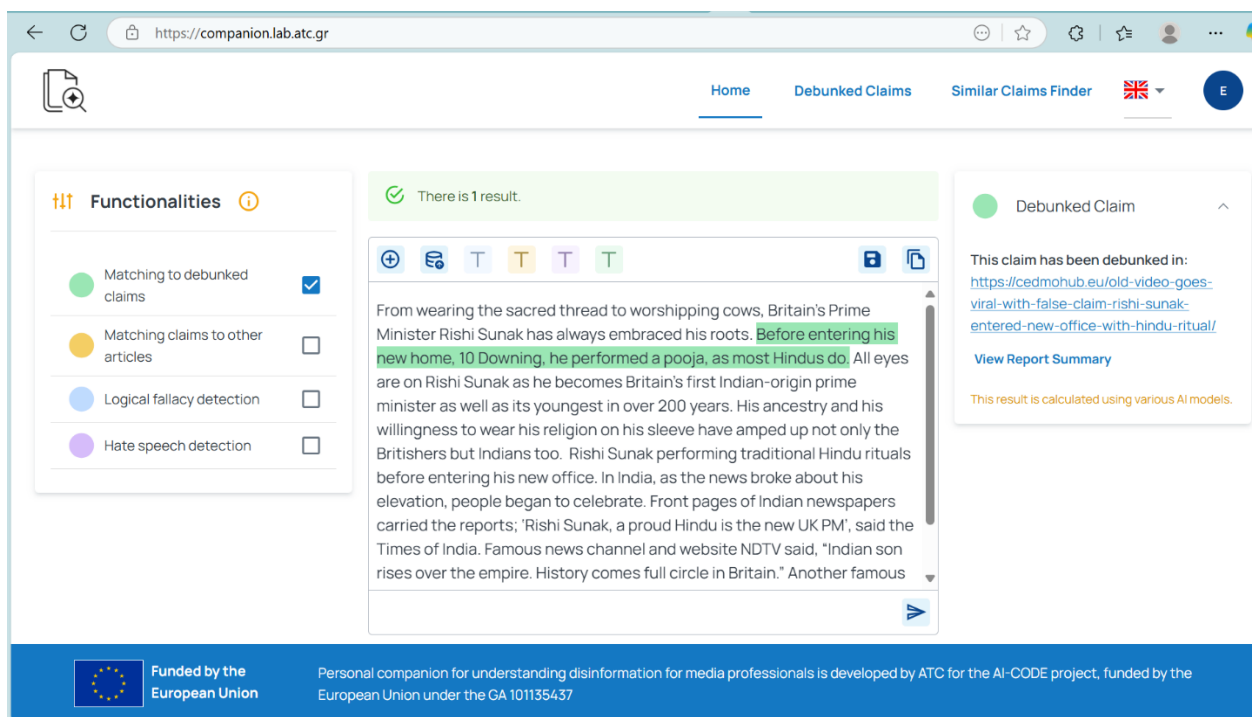


Figure 2 Claim Matching to Debunked Claims.

Matching to debunked claims: The Personal Companion service contains the “Debunked Claims” database that contains debunking reports along with a set of metadata, including the debunked claim, the URL of the report, etc. The “Matching to debunked claims” component checks if the *main* claim of a provided article is *semantically similar* to any of the debunked claims stored in the database. If this is the case, the main claim of the article is highlighted and an accordion appears, providing a summary (by clicking on the “View Report”) and the URL of the debunking report (Figure 2).

Matching claims to other articles: In contrast to the claim matching component, which is based on a “closed-world” assumption, this functionality performs an “open-world search”, i.e., it may return any relevant information found on the internet at the time of search. It takes as input an article, detects its *main* claim, and then searches the web for articles that have semantically similar claims. It highlights the main claim in

the text and returns the URLs of the relevant articles. Additionally, it automatically searches Bluesky⁸ and Google Fact Checks.

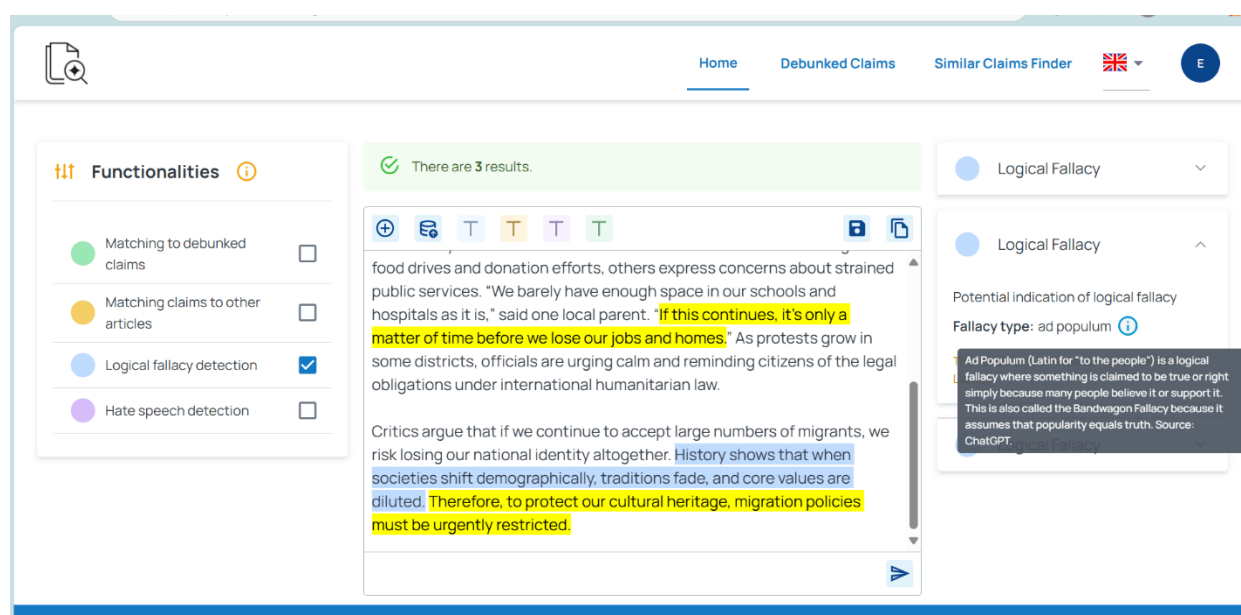


Figure 3 The Logical Fallacy Detection Result.

Logical Fallacy detection: Takes as input an article and checks if it contains certain types of logical fallacies. It highlights the fallacious sentence and returns the logical fallacy type and, upon request, its meaning (Figure 3). Personal Companion detects the following 12 types of fallacies:

1. **Ad hominem:** an excessive attack on an individual or a group instead of addressing someone's argument or opinion.
2. **Ad populum:** claiming a truth or affirming something is good or correct because many people think so.
3. **Circular reasoning:** a premise explicitly or implicitly assumes that its conclusion is true rather than providing any supporting statements.
4. **Deductive fallacy:** the text has an error in the formal logical reasoning.
5. **Equivocation:** a word or phrase is used in one sense in one premise and in another sense in some other needed premise or in the conclusion.
6. **Fallacy of credibility:** an appeal is made to some form of ethics, authority, or credibility.
7. **Fallacy of extension:** an argument that attacks an exaggerated or caricatured version of your opponent's position.
8. **False dilemma:** presenting only two options or sides when there are many options or sides.
9. **False causality:** assuming that two correlated events must also have a causal relation.
10. **Faulty generalization:** drawing a broad conclusion based on a limited sample of population.
11. **Intentional fallacy:** an argument has some element that shows the intent of a speaker to win an argument without actual supporting evidence.
12. **Red herring (fallacy of relevance):** the argument supporting the claim diverts the attention to issues which are irrelevant to the claim at hand.

⁸ While our initial implementation uses the publicly available SearchPosts API, after the original implementation the endpoint was found to no longer be accessible. However, as Bluesky offers most endpoints in two versions: user authenticated and public, our future approach will focus on the authenticated version.

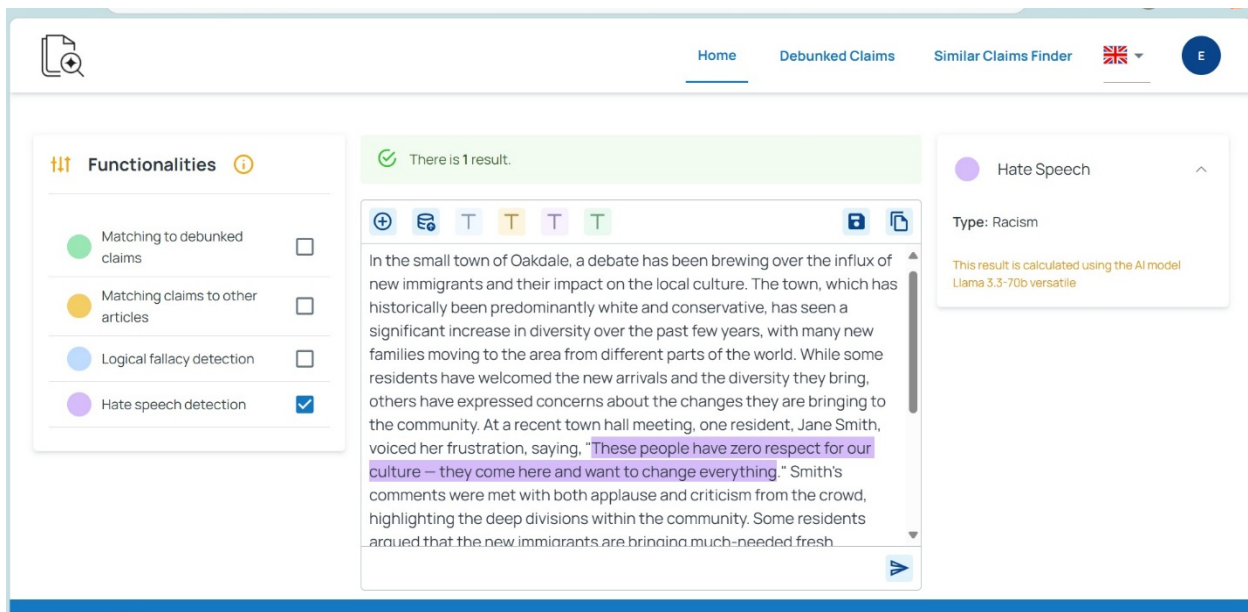


Figure 4 Hate Speech Detection Result.

- **Hate speech detection:** Takes as input an article and checks if it contains hate speech. It highlights the sentence with hate speech and returns the hate speech type (Figure 4). For Personal Companion, in line with the literature described in Section 2, hate speech is defined as: *any form of communication—spoken, written, symbolic, or digital—that incites hatred, discrimination, or violence against a specific group or individual based on protected characteristics. These characteristics include race, ethnicity, nationality, disability, religion or belief system, gender, gender identity, or sexual orientation.*

Hence, Personal Companion detects and classifies four types of hate speech:

- Racism, i.e., related to the race of an individual or a group.
- Sexism, related to expressions of misogyny / misandry.
- Sexual orientation, connected to sexual orientation and gender identity.
- Religious chauvinism, referring to the religion of an individual or a group.

Next, the technical implementation of the above mentioned functionalities is described in detail.

4.2.2 Technical Implementation

4.2.2.1 Claim detection

The claim detection implementation relies on LLMs to extract the claims presented in an input text and match each claim to the text snippet that represents it. The implementation uses the llama3-8b-8192 model, provided by Groq.⁹ This model is a relatively small, open-source LLM, with a relatively low cost.

A crude solution to the problem would be to simply provide the LLM with the text under examination, along with instructions based on the guidelines 1-6 detailed in Section 3. However, we have opted to follow a more detailed approach, aiming to break down the claim detection task into distinct smaller tasks. This is a common prompting technique, which is known to improve LLM reliability and performance¹⁰. To this end, we built a pipeline of transformations (prompt chains) on the input text using LLMs, to finally derive the text snippets containing the claims. The complete pipeline is illustrated in Figure 5.

⁹ <https://groq.com/>

¹⁰ https://www.promptingguide.ai/techniques/prompt_chaining

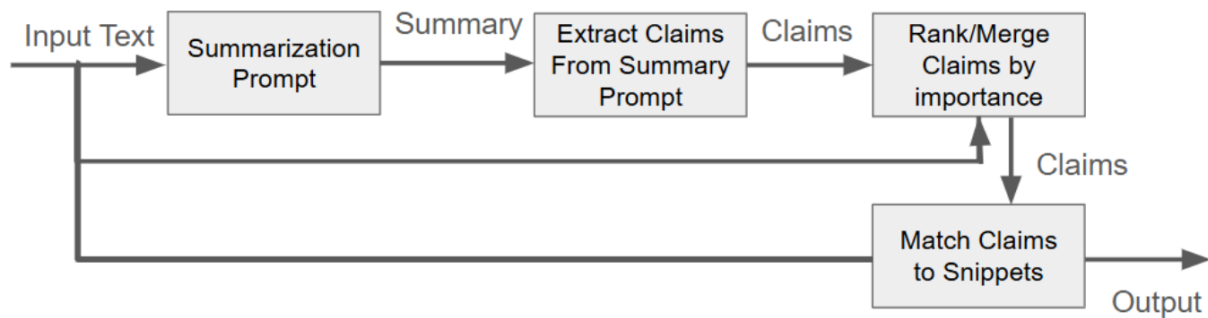


Figure 5. LLM / prompting pipeline for claim detection.

The pipeline consists of the following steps:

1. **Summarization:** initially the LLM is instructed to summarize the input text. To produce this summary the LLM is prompted to focus on verifiable facts, figures and data, analyze the sender's main intention, and focus on keyword indicators carrying domain-specific information (in line with the guidelines presented in Section 3). The summarization step is useful in the case of larger texts, reducing the context window for subsequent tasks, and enabling the focus on the most important details included in the text.
2. **Claim detection from the Summary:** the LLM is provided the previously generated input text summary and is instructed to write the claims included in it, along with the same guidelines provided for the summary generation.
3. **Merge/Rank Claims by Importance:** in this step the LLM is presented with the original input text along with the previously extracted claims. The instruction given aims to enhance the claims with any missing pieces of information, to make them valid standalone sentences (decontextualization), and to merge semantically overlapping ones. Additionally, the LLM is instructed to write the resulting claims in order of importance.
4. **Match Claims to Snippets:** in this final step the LLM is presented with the claims produced in step 3, along with the original input text. The instruction is to provide for each claim the passage in the input text, verbatim, where the claim is expressed.

API¹¹. The claim detection functionality is deployed as an API endpoint, written in Python's Flask framework. The API endpoint expects a JSON object as its input, with the following format:

```
{
  "message": <input_text>
}
```

The output produced is another JSON object of the following structure:

```
{
  "claims": [
    { "claim": <text_snippet_a> },
    { "claim": <text_snippet_b> },
    ...
  ]
}
```

Important note: In this first prototype of Personal Companion we use the *most important* claim returned by the Merge/Rank Claims by Importance step. However, in line with the user needs outlined in Sexrion 2.2, this functionality is designed to allow the retrieval of multiple claims. Hence, in T7.3 we plan to retrieve multiple claims and automatically assess their veracity using the Claim Matching and Web Search technologies

¹¹ The swagger URL is (as of 6/4/2025): https://claim-detection-vc.ilabhub.atc.gr/apidocs/#/Claim%20Detection%20API/post_extract_claims_after_summarizing | route: /extract_claims_after_summarizing

presented next.

4.2.2.2 Claim Matching

The claim matching component allows a user to input a claim (i.e., a sentence expressed as a factual statement), which is evaluated in terms of its truth, based on the resources in the Debunked Claims database. The Debunked Claims database currently contains debunking reports from Politifact¹² and Elections24Check¹³. Specifically, the component examines the claim, compares it to resources in the database and returns a decision about whether the claim has been debunked. It is important to note, that this decision is made under a “closed-world” assumption, i.e., should the system output that the claim has not been debunked, this does not mean that no debunking report exists against the claim (e.g., on the internet), but that the curated resources (the database) do not record such information. Naturally, a media organization will benefit more from this service if it maintains curated the Debunked Claims database (e.g., considering particular sources as trustworthy and excluding other fact-checks from non-trustworthy outlets).

The claim matching service consists of a vector DB (which is the transformation of the Debunked Claims database in an embedding space), an API, and an AI Agent developed using LangGraph and LangChain. At the high level, the system can be described as shown in Figure 6.

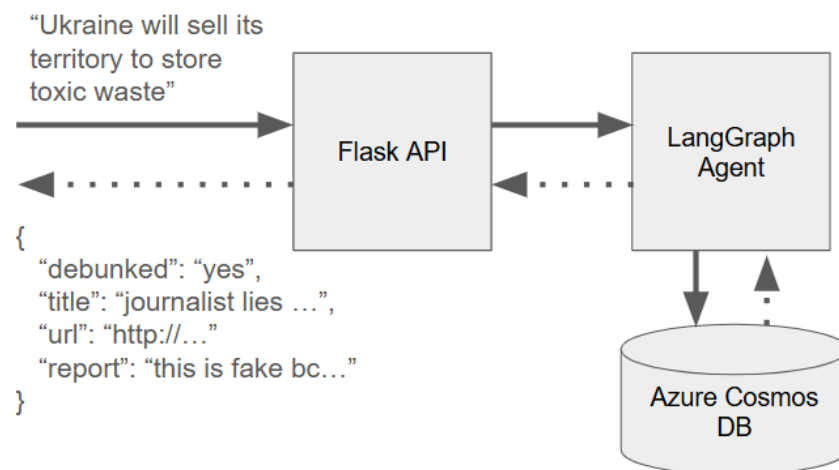


Figure 6 High-level system architecture of the Claim Matching component

Figure 6 shows the flow of information, and the components involved in producing a judgement about the claim. A user sends a claim to the API. The API spawns the AI agent. The agent first communicates with the vector DB to retrieve a set of top-k most relevant documents. Note, there is no similarity threshold cutoff, i.e., the most similar documents to the claim are retrieved, regardless of whether they actually are related to the claim. Following this, the LangGraph Agent decides on the relevance of the retrieved documents and produces a judgement about whether the examined claim is debunked based on the documents found.

Vector DB. The vector DB employed by the service is a NoSQL Cosmos DB set up on Azure. The database can also be interacted with via a Swagger UI.¹ Text-embedding-3-large has been used by the database to embed debunking reports and user queries are embedded in the same embedding space. These embeddings were selected on the basis that they support multiple languages. The fields recorded in the database include:

- Claim: the text corresponding to the claim being debunked
- Title: the fact-check title
- Article: the complete text of the fact-check

¹² <https://www.kaggle.com/datasets/rmisra/politifact-fact-check-dataset>

¹³ <https://elections24.efcsn.com/>

- url: a link to the fact-check
- article_date: the fact-check's publication date

API.¹⁴ The API is currently exposed using Python's Flask framework. It consists of a single endpoint `"/claim_match"`, which takes as input a JSON with a single key, `"message"`, its value being the claim to be matched against the records in the database. A request to the API endpoint triggers the execution of the agentic workflow (see next section), which produces the final response of the API:

The input is structured as follows:

```
{
  "message": <input_text>
}
```

The output produced is another JSON object of the following structure:

```
{
  "debunked": <takes the value yes/no>
  "title": < title of the fact check based on which the claim was found to be
debunked (empty if debunked = no)>
  "url": <url of the fact check based on which the claim was found to be debunked
(empty if debunked = no)>
  "report": <short text explaining why the claim in questions is false>
}
```

Agentic Workflow. The agentic workflow implemented by the claim matching component is graphically depicted in Figure 7. In short, the workflow proceeds as follows: initially the Agent receives a claim. The first step in the workflow is to retrieve from the vector DB the documents that are closer in the embedding space (**retrieve** node). Following this, the documents and claim are presented to an LLM. The LLM's role is to produce a judgement, based on the relevance of the retrieved documents to the claim (**grade_documents** node). Particularly, the LLM is used as a classifier, which outputs one of the labels 'CONFIRM', 'NEGATE', and 'NO DECISION', based on whether the examined claim is confirmed, debunked, or no relevant decision can be made, respectively, when considering only the information from the article retrieved (currently its title). If no document is considered as relevant (i.e., all labels were 'NO DECISION'), the workflow ends, and the claim examined is considered as not debunked. If documents relevant to the claim are found (i.e., at least one labelled as 'CONFIRM', or 'NEGATE'), an answer is produced about the claim's status as debunked/not debunked³ (**generate_answer** node). After an answer is produced, the Agent is called to self-correct, i.e., evaluate whether the generated answer is grounded in the document provided (**evaluate_generated_answer** node). If so, it proceeds with generating the final judgement (**debunked_verdict** node), otherwise the Agent is called to self-correct (**correct_ungrounded_response** node). Finally, the Agent parses the JSON object in the format required to be returned by the API (**create_final_json** node).

¹⁴ The swagger URL is (as of 6/4/2025): <https://claim-matching.lab.atc.gr/apidocs/> | route: `/claim_match`

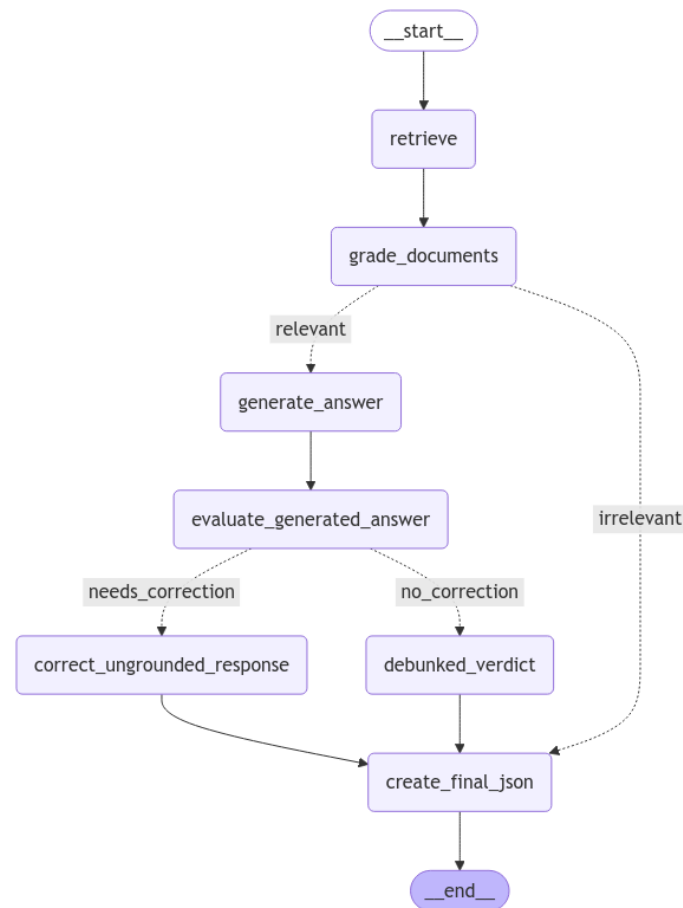


Figure 7 Agentic AI workflow.

Model Selection. Currently, the Agentic workflow makes use of a small Llama model from Groq, namely Llama3-8b-8192. This choice was made due to several advantages it offers: first, given this model is relatively small, it allows for potential disengagement from an external API and moving towards a local deployment in the future, should there be need. Further, the agentic workflow was designed with more individual steps, each with a small scope, to allow using a small model on all nodes, for reasons of cost effectiveness and environmental impact. Finally, while Groq as a provider claims to be energy efficient, given that it uses LPUs instead of GPUs for LLM inference,¹⁵ selecting smaller models should generally ensure reduced energy consumption.¹⁶

4.2.2.3 Web Search

The web search functionality takes a claim as its input, searches the web based on keywords extracted from this claim, and returns a set of documents/URLs grouped into sets of resources depending on their content's relation to the input claim (e.g., whether the sources found support the claim, debunk it, or relate to it in other ways). In contrast to the claim matching component, which is based on a "closed-world" assumption, the web search functionality performs an "open-world search", i.e., it may return any relevant information found on the internet at the time of search.

This endpoint implements a Q&A functionality based on agentic RAG (Singh et al., 2025). The implemented workflow expects a claim (i.e., a sentence) as its input, then performs web search on various search engines and web resources, based on this claim. After performing web search based on search keywords produced by generative AI, it categorizes the collected results into different classes. The categorization leverages LLMs

¹⁵ https://groq.com/wp-content/uploads/2024/07/GroqThoughts_PowerPaper_2024.pdf

¹⁶ arxiv.org/html/2411.11892v1

as classifiers, an approach that has been popular in the literature recently, and has been shown to sometimes outperform specialized classifiers (Wang, Pang, & Lin, 2024). In particular, the web resources collected are categorized into the following classes: ‘debunking’ (web resources that debunk the claim), ‘supporting’ (web resources which support the claim), and ‘related’ (web resources may only partially address the claim, or relate to the claim in other ways, covering other similar topics, etc.). During runtime, an additional category, ‘unrelated’ is produced, but documents in this category are removed before a response to the user is produced. Finally, if sources (i.e., websites) were found which debunk the claim, the endpoint returns a short debunking report, based on the information from these sources.

API.¹⁷ The endpoint was built with efficiency in mind, and employ —where possible— asynchronous functions. To this end, it was built using Python’s FastAPI framework, a framework specifically built for developing APIs. The API developed consists of a single endpoint named ‘/web_search/’. The endpoint is invoked via an HTTP POST request, with the following input JSON schema as its payload:

```
{
  "claim": <input_claim>
}
```

The endpoint’s responses, in turn, conform to the following schema:

```
{
  "claim": "string",
  "debunking": [
    {
      "context": "string",
      "metadata": {}
    }
  ],
  "supporting": [
    {
      "context": "string",
      "metadata": {}
    }
  ],
  "related": [
    {
      "context": "string",
      "metadata": {}
    }
  ],
  "debunk_report": "string"
}
```

The response contains a list of resources classified under the categories ‘debunking’ and ‘supporting’ the claim and being otherwise ‘related’ to the claim. Each resource includes a context, which is typically the text snippet found in a Search Engine Result Page (**SERP**). Additionally, different types of metadata may be included per resource, depending on how it was found (i.e., which type of web search). The minimal metadata set for each document which is returned, includes the ‘origin’ (the domain name of the URL), ‘source’ (the complete URL of the resource) and ‘found_via’ (corresponding to the resource(s) where the URL was found).

Process Workflow. The web search workflow is relatively linear. It performs several web search engine calls, which execute in parallel. At invocation time, an LLM first produces sets of keyword search queries, based on

¹⁷ The swagger URL is (as of 6/4/2025): <https://aic-web-search.lab.atc.gr/apidocs/>

the input claim. These are used as input for several web search processes, which collect the returned web resources (URLs) along with any related text found and any additional metadata. Following the web searches, all collected documents are merged, filtered based on their embedding similarity to the claim, and afterward labelled based on their relation to the claim (e.g., ‘debunk’, ‘support’). Finally, if debunking documents are found, the workflow uses them in a RAG setting, to produce a debunking report included in the final response. For the prototype version of the workflow, the workflow is set to fail if either search keywords, or document classification labels cannot be produced. In future versions, the agentic workflow can be set to retry the failed steps for a predetermined number of times before failing.

The complete workflow is written using Python’s LangGraph framework. All LLM calls are based on Python’s LangChain framework. Three web search engines and their related Python libraries are used: google SERP¹, Tavily² (a search engine specifically designed for question answering with LLMs), and Duckduckgo.³ Additionally, the workflow performs a search on Bluesky (covering a next-generation social media-tailored search) and Google Fact Checks (to focus specifically on fact-checking of claims). The complete workflow is illustrated in Figure 8.

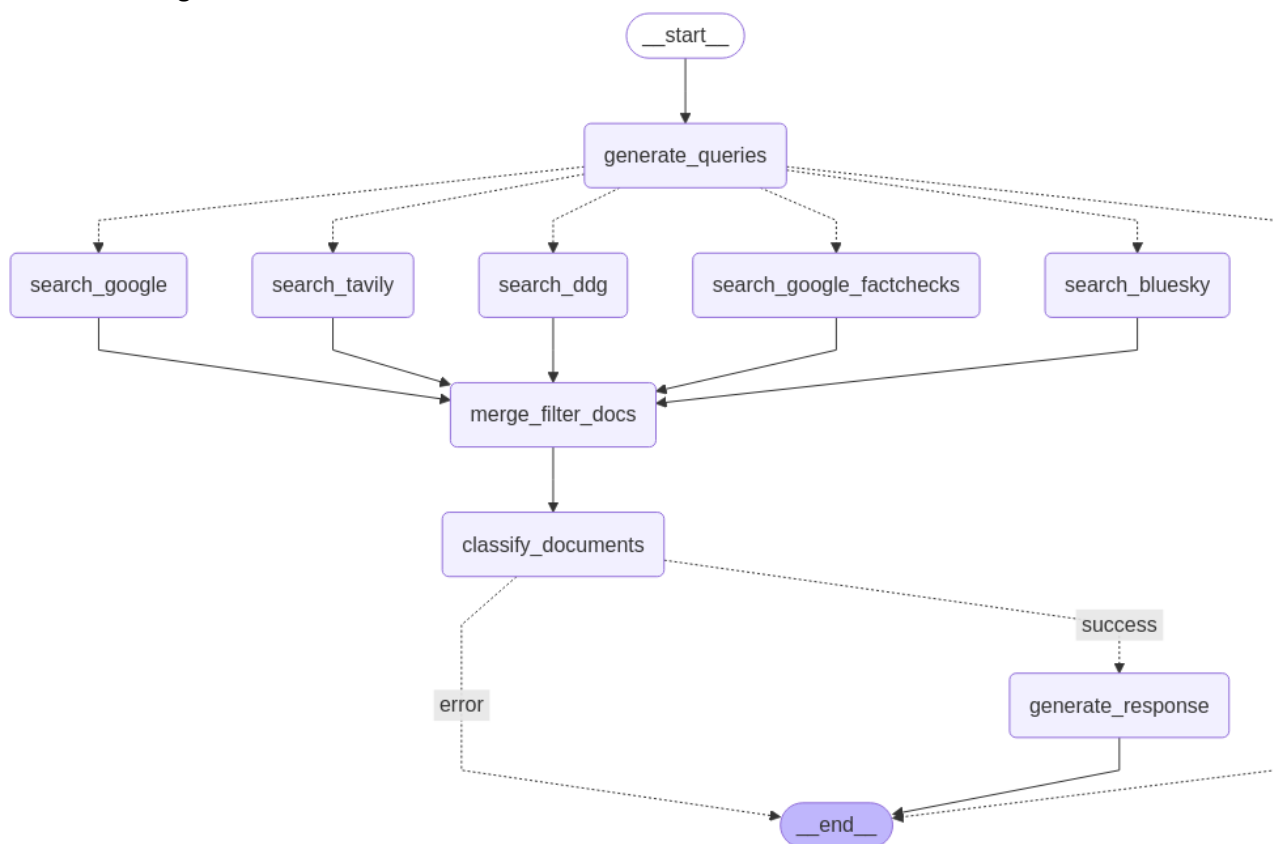


Figure 8 Web Search Workflow

The workflow is built with the initial aim of providing a comprehensive web search result set in the prototype application. The goal is to run multiple variations of the same query on various search engines for a relatively exhaustive search. In practice, this is currently limited, given that most web search services impose rate limits (see node descriptions below).

The nodes involved in the workflow illustrated in Figure 8 address the following functionalities:

- **generate_queries:** this node exploits the role-playing prompting technique, a technique which has been shown to improve LLM effectiveness in certain scenarios (Kong, et al., 2024). In particular, it casts the LLM to the role of a fact-checker who is experienced at performing web search to verify information. The LLM is instructed to produce two different sets of keyword queries, one short- and one long-format. The short format in particular is suitable for search on microblogging social media,

like Bluesky (we found that too many keywords in Bluesky's API tended to return no results, likely due to required exact matches of all keywords). The rationale behind producing different keyword queries is to mimic human behaviour when using search engines: a human using search engines like google may try searching for a particular topic using different phrasing and/or combinations of search terms, in order to explore a wider result set of web pages, compared to when issuing a single query. Our aim is to produce a comprehensive result set and broaden the information collected on the search.

- **search_bluesky**¹⁸: this node performs concurrent queries on Bluesky, using all sets of generated short queries. The node uses functionalities provided by the [Bluesky API](#).⁴
- **search_ddg**: this node performs web search using the Duckduckgo search engine. We note that Duckduckgo is subject to rate limits, in the absence of a subscription. In the current prototype version, this node performs a single query per execution, using the input claim, verbatim, as the search query.
- **search_google**: this node performs a query on Google's official SERP API. We note that Google's SERP API is a paid service, and its free tier is limited to 100 requests per month. For prototyping this is considered sufficient, however, for real usage this node will either require replacement, or a subscription to the service.
- **search_google_factchecks**: this node performs concurrent queries (using the sets of short and long keyword searches generated) on the Google Factchecks⁵ API. In addition to gathering the textual context of the results, this service also provides information about the claimant, the rating of the claim etc., information which is also collected by the workflow.
- **search_tavily**: this node performs a search against the Tavily search engine. We note that Tavily is a paid service. The free tier allocates 1000 credits per month to a user, which in the setting of the web search functionality amounts to about 500 queries. Given these limits, under heavy usage, this node will require replacement, or a subscription.
- **merge_filter_docs**: this node merges all documents collected in the previous web search stages based on their URL (a website may be found, for example, both through tavily and google, with different text contexts from each search engine). The node further calculates for each merged document its text embedding and removes all documents with very low embedding similarities compared to the examined claim.
- **classify_documents**: this node examines the textual context of the retrieved URLs/documents and classifies them into those supporting/debunking the claim, or otherwise being related to it (i.e., not fully answering, being on a different but related topic), using an LLM as a classifier. The node removes any documents that were labelled as UNRELATED to the examined claim.
- **generate_response**: this node is responsible for producing the final output format, containing all data collected in the previous steps. Further, should debunking documents have been collected through web search, it generates a debunking report, which is also included in the final response.

Model Selection. Three nodes of the web search workflow require the use of generative AI: generate_queries, validate_documents, and generate_response. The generative AI functionalities are provided through LLM calling via the Groq API. With production deployment in mind, where possible, we have used small LLM models, to reduce costs, exploit the less restrictive rate limits¹⁹ imposed on these models by the provider, and decrease energy demands.

In the current prototype setup, the following models are used:

- **generate_queries**: llama-3.1-8b-instant. This model is among the cheapest offerings by Groq, and its small size ensures reduced energy demand. Given the simplicity of the required task, no larger

¹⁸ As previously noted, while our initial implementation uses the publicly available SearchPosts API, after the original implementation the endpoint was found to no longer be accessible. However, as Bluesky offers most endpoints in two versions: user authenticated and public, our future approaches will focus on the authenticated version

¹⁹ <https://console.groq.com/docs/rate-limits>

models were examined.

- **validate_documents:** llama-4-maverick-17b-128e-instruct. This model was chosen after benchmarking different models on the document relevance to the claim classification task (Section 5).
- **generate_response:** meta-llama/llama-4-scout-17b-16e-instruct. This is the smaller offering of the newest class of llama models available in Groq, providing limited costs, compared to both the alternative llama4 variant, as well as larger llama3 versions.

All the above are among the cheapest offerings in Groq, while being relatively small models in terms of parameters, which should in turn reduce energy consumption.

4.2.2.4 Hate Speech Detection

The research conducted for the hate speech detection tool is fully described in D2.3, where the exploration of various architectures and models is described in detail. Specifically, after an extensive comparison of SLMs and LLMs against datasets available in the literature and a dataset annotated manually for this project by partner DW, it is found that the best performing model with accuracy >91% is Llama-3.3-70b-versatile and the best performing architecture is a two-step classification. With this architecture, the task is split into two sequential prompts for improved accuracy:

- **Hate Speech Presence Detection Prompt:** A binary classification few-shot example prompt instructs the model to determine whether the input contained hate speech ("Hate" vs "No Hate").
- **Hate Speech Classification Prompt:** If the first LLM call detects hate speech, a second few-shot example prompt is issued to classify the hate speech into the appropriate category (Racism, Sexism, Religious or Sexual Orientation). Multiple hate categories can be detected within a single input; each linked to one or more specific text snippets. The prompt also instructs the model to highlight the specific phrases or sentences that triggered the detection. This approach raises the number of LLM calls to two per input if hate speech is detected in the first call. Otherwise, if no hate speech is detected the calls per input are limited to one.

This two-step design aims to reduce false positives during category classification by first confidently filtering non-hate content. It also allows for more focused instructions in the Hate Speech Classification Prompt, tailored specifically to category identification, leading to improved overall classification accuracy.

API.²⁰ The hate speech detection functionality is deployed as an API endpoint, written in Python's Flask framework. The API endpoint expects a JSON object as its input, with the following format:

```
{
  "message": <input_text>
}
```

Supposing that the text contains two hateful sentences: sentence_1, sentence_2, where sentence_1 is categorized with two types of hate (e.g., racism, sexual orientation) and sentence_2 with one type, the output produced is another JSON object of the following structure:

```
{
  Output: {message: "sentence_1",
           responseText: "{category_1}"},
          {message: "sentence_1",
           responseText: "{category_2}"},
          {message: "sentence_2",
           responseText: "{category_1}"}
}
```

²⁰ The swagger URL is (as of 6/4/2025): <https://hs.ilabhub.atc.gr/docs> | route: /hate_speech_two_step

4.2.2.5 Fallacy Detection

The research conducted for the fallacy detection tool is fully described in D2.2. Specifically, a dataset composed of 3,425 short news articles in English was created using LLMs and the dataset (Sourati et al., 2023), comprising fallacies of 12 different types, to check if LLMs can identify fallacies in realistic and complex textual environments. For the prototype version of Personal Companion, we have employed a Llama-2 model fine-tuned in our generated dataset.

API.²¹ The fallacy detection functionality is deployed as an API endpoint, written in Python's Flask framework. The API endpoint expects a JSON object as its input, with the following format:

```
{
  "text": "Your text to analyze for fallacies"
}
```

The endpoint's response, in turn, conforms to the following schema:

```
{
  "signals": [
    { "name": "FALLACY",
      "spans": [
        { "start": 0,
          "end": 45
        }
      ]
    },
  ],
  "fallacies": [
    { "fallacy_type": "string",
      "fallacious_text": "string",
      "confidence_score": decimal }
  ]
}
```

4.2.2.6 Technical Infrastructure

The Personal Companion is built using the microservices architecture, consisting of independently deployed components. This design enables precise scalability adjustments for each component individually. Figure 13 below depicts the hybrid deployment mode that is adopted, where some resources are deployed as Kubernetes services in an on premises multi-node cluster, while some as container services hosted on cloud resources (Azure).

²¹ The swagger URL is (as of 6/4/2025): <http://74.234.136.116:8080/docs> [gpu-based/old] | route: /api/v1/disinformation_signals

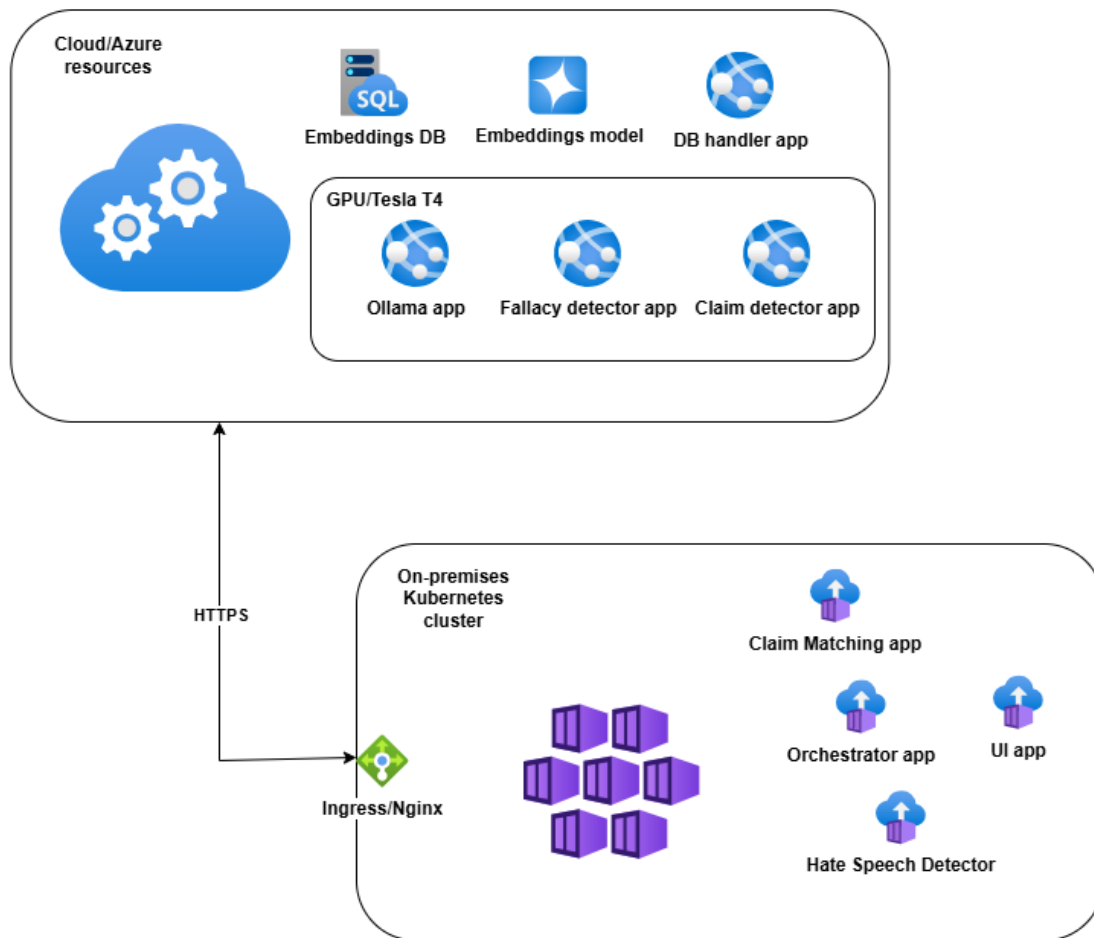


Figure 9 Deployment Architecture of Personal Companion

The cloud services consist of different deployment types:

- **Embeddings model:** an Azure OpenAI Service-based model, text-embedding-3-large, as a managed service.
- **Embeddings DB:** a managed service based on the Azure Cosmos DB, a fully managed NoSQL, relational, and vector database. Serverless configuration is applied to the database containers.
- **DB handler application:** exposes a layer of functions over the NoSQL and vector database (similarity search) and runs in a similar mode (serverless as a container app with auto scaling enabled).
- **Ollama app:** the Ollama language model framework runs as a docker container on a GPU enabled daemon. The docker runtime is hosted on an instance powered by Nvidia Tesla T4 GPU (1 vCPU - 16Gb).
- **Fallacy detector application:** runs in containerized mode on the GPU enabled docker runtime.
- **Claim detector application:** runs in containerized mode on the GPU enabled docker runtime.

The on-premises deployment mode utilizes a multi-node Kubernetes cluster, with all services exposed via a NGINX-based ingress controller serving as a load balancer. Each component supports parameterization through environment variables, and Kubernetes deployment scripts are integrated into the CI/CD pipeline. Resource limitations and scaling configurations differ between the presentation layer services (UI app) and the backend services (claim matching, hate speech detector and orchestrator app). The orchestrator app is deployed in high availability mode using replica sets since it is the component that maintains the real-time communication with the UI web application. All services are exposed over HTTPS.

5. System Validation

5.1 Data-centric evaluation results

The data-centric evaluation results for multi-class hate speech detection—achieving 92% accuracy on unseen, real-world data manually annotated by media professionals for AI-CODE—are documented in Deliverable D2.3. The results for fallacy detection, which achieved 86% accuracy on a custom, manually annotated dataset created for AI-CODE, are presented in Deliverable D2.2. Next, the data-centric evaluation for the components: i) claim matching to debunked claims, and ii) matching to other articles, are presented in detail. The data-centric evaluation for the claim detection functionality will be performed in T7.2.

5.1.1 Evaluation of Claim Matching to Debunked Claims Component

Due to the lack of relevant datasets in the literature, we conducted a sanity check to gain insights into the end-to-end accuracy of the current prototype. To this end, we applied the following evaluation procedure:

- We randomly selected 100 claims currently stored in the system's Vector DB (all coming from Elections24). These claims were provided to openAI's GPT-4o-mini model with the request for each of them to be paraphrased. The reasoning was that a claim is not expected to be input by users with the exact same wording as it is recorded in the database. The paraphrased claims collected this way make up the intended true positives of our data (i.e., claims that the system *should* identify as debunked in the database).
- We randomly selected 100 claims currently recorded by Politifact.⁷ The claims found this way are intended to be the true negatives, i.e., claims which the claim matching service *should not* find in the database (note that this is not necessarily true for all claims, since the two databases may have overlaps in the debunked claims they record – we expect this set to contain a significant number of true negatives, nonetheless).

For all claims collected (200 in total), we ran the claim matching API and recorded the verdicts produced for each claim. The total accuracy (true positives and true negatives) was calculated at 88% (i.e., 24 out of 200 claims were incorrectly matched). Table 3 presents the results in detail per dataset.

Table 3 Accuracy Results for Claim Matching.

Dataset	Elections24Check	Politifact
% of correctly identified as debunked	83%	0%
% of incorrectly identified as debunked	0%	7%
% of correctly identified as not debunked	0%	93%
% of incorrectly identified as not debunked	17%	0%

We note that in the case of Politifact, 7 claims out of the 100 retrieved documents from the database, which were erroneously judged by the LLM as being relevant to the claim. These cases led to the workflow producing the judgment that the examined claim was debunked. However, we note that currently, the LLM producing document relevance judgements is only provided article titles, which lack substantial information; hence, adding an article summary to the text examined should alleviate these erroneous judgements.

For Elections24check, a case can be made of the claims currently recorded in the database: some records include insufficient information and/or badly formed textual claims. An example is “Footage displays Palestinian inmates in an Israeli correctional facility”, which is reminiscent of a newspaper article title, but the exact claim that it presents is not clearly written. Another example is the claim in the database “The current level, despite contributions from the industry, is the lowest it has ever been”. This could refer to many different things, e.g., carbon dioxide, sea levels, etc. From its current form, significant context is missing to allow for a correct match against the database. Hence, we expect a manually curated database without

such ambiguities to lead to improved accuracy results.

5.1.2 Evaluation of Matching to Other Articles Component

A prototype of the workflow detailed in Section 4.2.2.3 was initially developed without particular focus on the prompts and LLM used for document labelling (as debunking, supporting, or being otherwise relevant to the input claim). Initially, the smallest Llama offerings from Groq were used for quick prototyping. After manual testing of a complete working version of the end-to-end workflow, we noticed obvious cases which presented errors in the document labelling, at least in terms of how a human annotator might judge the relevance of a SERP context to a given claim. Following this observation we selected four sample claims which we identified as debunked (“Volodymyr Zelensky bought a winery from singer Sting in Italy, in the Tuscany region, for 75 million euros”, “Volodymyr Zelensky sold Ukraine territory for use as a toxic waste dump”, “Wind turbines emit infrasound that is harmful to human health”, “Rishi Sunak performed traditional Hindu rituals before entering his new office”) in different fact-check organizations. With these four claims we invoked the workflow, and we collected all documents returned by the related web searches. This provided a set of 122 documents.

After examining the labels provided by the workflow for each document, we initially identified the need to better define the categories a document can belong to concerning the claim. In particular, the original categories used were ‘debunk’, ‘support’, ‘related’ (partially addressing the claim), and ‘similar’ (results not directly related to the claim, but on related topics and/or events). From this process, we identified the need for the additional category ‘unrelated’, as especially in the case of Bluesky search, many retrieved posts were completely irrelevant to the claim.

Given the 122 collected texts, we examined the associated claim and context provided by the SERPs. Given the textual context and the information gathered by the URL and other metadata, we manually annotated the document’s relation to the claim²² and then used a simple prompt that required the LLMs to produce one of the labels we had assigned manually. We then evaluated different LLMs in terms of their accuracy, given our annotations. We tested the following models:

- llama-3.1-8b-instant,
- llama-3.3-70b-versatile,
- meta-llama/llama-4-maverick-17b-128e-instruct,
- meta-llama/llama-4-scout-17b-16e-instruct,
- qwen-qwq-32b.

These models were chosen due to having the highest rate limits in Groq, hence would be most suitable in a production setting. However, the accuracy of all models ranged between 0.59 and 0.65.

Following this, we also experimented with a Chain of Thought prompt (Wei et al, 2022), asking the LLMs to detail their “thought process” based on which they arrived at the final labels. However, (apart from affecting some LLMs negatively in terms of accuracy), the observed accuracy was still low, ranging from 0.51 to 0.67. We concluded that chain-of-thought reasoning for this task does not provide significant gains, while at the same time increasing the tokens generated, hence leading to more energy consumption and higher cost.

Following these experiments, we studied the confusion matrices for each model and observed many errors among the ‘similar’ and ‘related’ classes. This led to our redefinition of classes, based on the following

²² Note, that examination of the complete content in a web page might change the assigned label. However, at this point we depend only on the provided SERP context and do not perform additional scraping of web pages (which may be subject to legal issues). The intuition is for the web search agent to perform a quick search and arrive at a decision through a quick readthrough of search results, mimicking the behavior of a human quickly looking up information on the internet.

reasoning: the focal point of the web search functionality should be to find: a) documents debunking a claim and b) documents supporting a claim. The remainder should be classified as either irrelevant or relevant/similar in a broader sense. Hence, we chose the labels ‘debunk’ (a document debunks the claim, the dataset contained 42 such documents), ‘support’ (the claim is supported by the document, the dataset contained 8 such documents), ‘other’ (the document otherwise relates to the claim without completely debunking/supporting it, the dataset contained 49 such documents) and ‘unrelated’ (the document is irrelevant to the claim, the dataset contained 23 such documents). The random baseline for this dataset is 0.319. We used a simple prompt instructing the LLMs to assign a label based on these classes. Table 4 presents the accuracy of the examined models.

Table 4 Accuracy results of the examined models

Model	Accuracy	F1-score for Debunked Class	F1-score for Supporting Class
llama-3.1-8b-instant	0.520	0.533	0.520
llama-3.3-70b-versatile	0.569	0.598	0.667
llama-4-maverick-17b-128e-instruct	0.699	0.731	0.500
llama-4-scout-17b-16e-instruct	0.602	0.659	0.588
qwen-qwq-32b	0.715	0.773	0.636

Based on the above, while qwen-qwq-32b has a slightly better accuracy overall, we selected llama-4-maverick-17b-128e-instruct. We made this choice, given that the difference in accuracy is relatively low (a difference of 0.016), and qwen-qwq-32b is a “reasoning” model, i.e., it always spawns a chain of thought response. This affects the overall workflow a) in terms of generated tokens, incurring higher costs and energy consumption, and b) in terms of response time. Especially the latter must be considered, given that a workflow may require tens of SERP contexts to be classified in relation to the claim. Hence the tokens generated should be kept to a minimum in a production setting.

To conclude, we note that the task of correctly assigning a class label on search engine results is not trivial, given that LLMs are presented with text snippets which may lack coherence or all details required for a sound conclusion to be drawn.

5.2 Ethics by Design

The development of Personal Companion has been guided by the principles of Trustworthy AI, as outlined in the Ethics Guidelines for Trustworthy AI developed by the High-Level Expert Group on AI (AI-HLEG). This effort has been carried out in close collaboration with WP9, particularly Task 9.3, which is dedicated to designing and implementing an ethical assessment framework for all AI-CODE services. Several ethical requirements were initially addressed in the user requirements analysis presented in Deliverable D3.1, and this preliminary analysis has since been revisited and refined in light of ongoing system development. Table 5 below summarizes the key ethical and human oversight requirements identified during this process and outlines how the Personal Companion addresses each through specific design and implementation decisions. The requirements are organized according to the seven categories of Trustworthy Artificial Intelligence (AI), as defined in the Ethics Guidelines for Trustworthy AI:

1. Human Agency and Oversight (HAO)
2. Technical Robustness and Safety (TSA)
3. Privacy and Data Governance (PDG)
4. Transparency (TRA)
5. Diversity, Non-discrimination and Fairness (FAI)
6. Societal and Environmental Well-being
7. Accountability

Table 5 Ethical design considerations based on the requirements for Trustworthy AI.

#	Ethical requirements	Design and Implementation
HAO-01	It shall be apparent to the user interacting with the application that they are interacting with an AI system	For each outcome presented to the user there is an explicit mention that is generated by AI model(s) (e.g., Figure 4). This approach improves transparency and ensures that it is clear to users that AI systems are embedded in the service. Furthermore, this information is added in the “Disclaimer” in “Terms and Conditions” which the user can see during the sign up. This will further be improved with the use of models cards, instead of static descriptions for the AI models.
HAO-02	The capabilities of the application shall be communicated clearly to the user.	This is facilitated through the UI, specifically the user can click on the “i” button on “Functionality” to obtain more descriptive information.
HAO-05	The level of human oversight (HITL, HOTL, HIC) is clearly defined and appropriate for the use case	The human has full oversight as each output is presented in detail (by highlighting the detected sentences and providing the relevant assessments). Also, the service is limited to detecting disinformation/misinformation, hate speech, fallacies. It does not proceed to automatically change anything in the assessed article.
TSA-02a	The accuracy, reliability, and reproducibility of the application shall be monitored, documented, and communicated to the user over time.	This will be facilitated through the model cards developed in T6.2 and T7.2. Steps are being undertaken to integrate them.
TSA-02b	There are processes in place to ensure that the level of accuracy of the AI system to be expected by end-users and/or subjects is properly communicated.	This will be facilitated through the model cards developed in T6.2 and T7.2. Steps are being undertaken to integrate them.
TSA-03	The data used to build the application shall be up-to-date, of high quality, complete and representative of the deployment environment.	Personal Companion’s modular architecture allows for easy integration of up-to-date models trained on recent high-quality datasets (following recommendation 5.2.5). This ensures the system remains relevant to real-world conditions and helps prevent performance degradation caused by data distribution shifts over time.
PDG-01	The application shall minimize the amount of data collected on users.	The collected data are limited to email address, password (encrypted), organization name in the sign-up. Additionally, based on FAI-03 requirement the service will also collect the feedback provided by each user (but not tight to their profile). The user is notified about the stored data and the processing methods in “Terms & Conditions”, that can be seen during the sign up process.
TRA-02	The decisions of the application shall be understandable by the user (XAI) as far as reasonably achievable.	All functionalities are explained to the user in the informative link on “Functionalities. Also, the user will be enabled to cross check by providing the output in an analytic way, i.e., by highlighting the claim and providing the type of fallacy/hate speech/ the matching claim. Also, when the user the definition of the detected type of hate speech/fallacy will provided further understanding of the results.

TRA-03	The application shall inform users about the purpose, criteria, and limitations of the decisions generated by the AI system.	A description card will be added to describe all mentioned requirements
FAI-03	The application shall have a mechanism that allows for flagging issues related to bias, discrimination or poor performance.	Feedback mechanism is under development and will be incorporated (by M24) collecting input text, output of service and feedback from user.

In addition, the ethics advisory board of the project provided the following feedback on the service, which was reviewed with the service leader and CEPS. Table 6 below summarizes the feedback in the left column and the measures that have been adopted or will be adopted and implemented in response within T7.3 to the feedback in the right column.

Table 6 Ethics Advisory Board Recommendations and Implementation Measures.

EAB recommendation	Implementation measure
Re-word ‘assist <i>young</i> professionals’ to just ‘professionals’ because seniority level does not prevent mistakes e.g. logical fallacies	While this wording couldn’t be changed for the past deliverable, it has been adjusted and will be considered in communication materials for this service.
Promising that <i>all</i> logical fallacies will be detected is likely to create unrealistic expectations. The wording should be more precise.	The wording will be changed to reflect that this service captures 12 different types of logical fallacies. Transparency on these 12 different types will be offered through the model cards or the UI. Currently, the user sees which type of fallacy is detected, not the full list that can be detected. It is acknowledged (currently in Terms & Conditions in the sign up page) that the service does not have a 100% accuracy rate. This will be reflected in the wording in description for deliverables and in user-facing communication.
Improve transparency on the reliability of this service.	This will be facilitated through the integration of model cards (T6.2/T7.2).
There should be more clarity for the user how the system operates and its limitations, e.g. how is ‘claim’ defined, what type of logical fallacies are incorporated, and how are conditional sentences processed.	Model cards will be implemented. Additional textual information on the different functionalities are incorporated in the UI, e.g. through the info button in “Functionalities”.
It should be acknowledged that there is a difference between claim and fact.	The exact definition of the claim, as defined for the purposes of this application (provided in Section 2) is provided through the info button in “Functionalities”.

6. Conclusion and Future Outlook

This deliverable has presented the prototype version of the Personal Companion for media professionals—a practical, AI-powered tool designed to assist media practitioners in identifying and assessing disinformation within news content. The tool integrates multiple advanced components—including claim matching to debunked claims, web-based evidence retrieval, logical fallacy detection, and hate speech identification—into a user-friendly interface that supports both automated analysis and human oversight. The development process was driven by real-world needs identified through co-creation workshops and stakeholder engagement, ensuring that the prototype aligns with the workflows and expectations of journalists, editors, and fact-checkers. Technical implementation choices were made with transparency, scalability, and sustainability in mind, leveraging LLMs in an efficient, modular architecture. Initial evaluation results show a

strong performance across components, demonstrating the system’s potential in real-world editorial settings.

As part of the immediate next steps (by M24), to be implemented under task T7.3 and in preparation for the validation phase in task T8.3, several improvements are planned—particularly in the automated identification of factual inaccuracies. Currently, only the main claim in an input text is detected and assessed; upcoming iterations will expand this to detect multiple claims per article, allowing users to select which claims to search on the database with the debunked claims, or on the Web. Additionally, the Agent’s decision in the claim matching component is currently based solely on the title of the debunking report. Future updates will evaluate the impact of providing additional context—such as a short summary or the full article text—to various nodes in the agentic workflow. In parallel, the fallacy detection module will be further refined through exploration of alternative architectures and models, aiming to achieve an optimal balance between performance and environmental impact—consistent with the project’s commitment to sustainable and trustworthy AI.

Further planned enhancements include the automatic generation of reports consisting of a detected claim and a corresponding set of matching sources. Additionally, the user interface will be redesigned to support updated functionalities and to accommodate both novice and expert users. Based on user profiles defined in D3.2, different access levels will be implemented—for instance, learners will have read-only access to the debunked claims database, while professional fact-checkers will be granted editing capabilities under specific constraints. To ensure increased usability, the revised interface will be tested and iteratively improved through user trials conducted under task T7.3, with finalization expected by M26 to allow sufficient time for broader validation during task T8.3.

To address the ethical requirements for the Personal Companion prototype, we plan to focus on enhancing transparency, oversight, and fairness through both technical improvements and user interface design. Planned activities include the integration of model cards (T6.2/T7.3) to communicate AI model capabilities, limitations, and reliability, thereby improving user understanding and trust. A feedback mechanism will be implemented by the start of T8.3 to enable users to report potential issues related to bias, discrimination, or model performance, supporting human-in-the-loop oversight. Finally, continuous evaluation of detection methods using up-to-date, representative data from the literature and real use case scenarios within T8.3 will ensure robustness, while ongoing collaboration with WP9 will guide ethical assessment in line with the Trustworthy AI framework.

7. References

- Aggarwal, P., & Zesch, T. (2022). Analyzing the Real Vulnerability of Hate Speech Detection Systems against Targeted Intentional Noise. *Proceedings of the Eighth Workshop on Noisy User-generated Text (W-NUT 2022)* (p. 230–242). Gyeongju, Republic of Korea: Association for Computational Linguistics.
- Amazeen, L. G., & Michelle, A. (2019, 02). Fact-Checking as Idea and Practice in Journalism. *Oxford University Press*.
- Arya, G., Hasan, M. K., Bagwari, A., Safie, N., Islam, S., & Ahmed, F. R. (2024). Multimodal hate speech detection in memes using contrastive language-image pre-training. *IEEE Access*, 22359–22375.
- Balalau, O., Bertaud-Velten, P., El Fraihi, Y., Gaur, G., Goga, O., Guimaraes, S., . . . Saadi, B. (2024). FactCheckBureau: Build your own fact-check analysis pipeline. *33rd ACM International Conference on Information and Knowledge Management* (p. 5185–5189). Association for Computing Machinery.
- Bertani, A., Mazzeo, V., & Gallotti, R. (2024). Decoding the News Media Diet of Disinformation Spreaders. *Entropy*, 26(270).
- Choi, E. C., & Ferrara, E. (2024). Automated Claim Matching with Large Language Models: Empowering Fact-Checkers in the Fight Against Misinformation. *WWW '24: Companion Proceedings of the ACM Web Conference 2024*. Singapore.
- Davidson, T., Warmesley, D., Macy, M., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. *Americal Association for Artificial Intelligence* (p. 512–515). Proceedings of the international AAAI conference on web and social media.
- De Gibert, O., Perez, N., García-Pablos, A., & Cuadros, M. (2018). Hate speech dataset from a white supremacy forum. *arXiv:1809.04444*.
- Deng, Z., Schlichtkrull, M., & Vlachos, A. (2024). Document-level claim extraction and decontextualisation for fact-checking. *62nd Annual Meeting of the Association for Computational Linguistics* (p. 11943–11954). Association for Computational Linguistics.
- Fortuna, P., & Nunes, S. (2018). A survey on automatic detection of hate speech in text. *ACM Computing Surveys*, 1-30.
- Habernal, I., Wachsmuth, H., Gurevych, I., & Stein, B. (2024). Fine-grained Fallacy Detection with Human Label Variation. *arXiv:2502.13853*.
- Hu, Q., Long, Q., & Wang, W. (2024). Decomposition dilemmas: Does claim decomposition boost or burden fact-checking performance? *arXiv:2411.02400*.
- Hu, Q., Long, Q., & Wang, W. (2024). Decomposition dilemmas: Does claim decomposition boost or burden fact-checking performance? *arXiv:2411.02400*.
- Hussain, A., & Aslam, A. (2024). Hate speech against women and immigrants: A comparative analysis of machine learning and text embedding techniques. *Journal of Applied Research and Technology*, 548-559.
- Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R., . . . Dong, X. (2024). Better Zero-Shot Reasoning with Role-Play Prompting. *arXiv:2308.07702v2*.
- Mavroudis, V. (2024). *LangChain v0.3*. Preprint.
- Metropolitansky, D., & Larson, J. (2025). Towards Effective Extraction and Evaluation of Factual Claims. *arXiv:2502.10855*.
- Min, S., Krishna, K., Lyu, X., L. M., Yih, W., Koh, P., . . . Hajishirzi, H. (2023). FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. *Conference on Empirical Methods in Natural Language Processing* (p. 12076–12100). Association for Computational Linguistics.
- Myers, D., Mohawesh, R., Chellaboina, V. I., Sathvik, A. L., Venkatesh, P., Ho, Y.-H., . . . Jararweh, Y. (2024). Foundation and large language models: Fundamentals, challenges,. *Cluster Comput*, 1-124.
- Nakov, P., Corney, D., Hasanain, M., Alam, F., Elsayed, T., Barrón-Cedeño, A., . . . Da San Martino, G. (2021).

- Automated Fact-Checking for Assisting Human Fact-Checkers. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. New York.
- Ni, J., Shi, M., Stambach, D., Sachan, M., Ash, E., & Leippold, M. (2024). AFaCTA: Assisting the annotation of factual claim detection with reliable LLM annotators. *62nd Annual Meeting of the Association for Computational Linguistics* (p. 1890–1912). Association for Computational Linguistics.
- Nockleby, J., Levy, L., Karst, K., & Mahoney, D. (2000). Encyclopedia of the american constitution. Hate speech., (p. 1277–1279).
- Revanth G., R., Chinthakindi, S. C., Wang, Z., Fung, Y., Conger, K., Elsayed, A., . . . Ji, K. (2022). NewsClaims: A New Benchmark for Claim Detection from News with Attribute Knowledge. *Conference on Empirical Methods in Natural Language Processing* (p. 6002–6018). Association for Computational Linguistics.
- Schmidhuber, M., & Kruschwitz, U. (2024). LLM-based synthetic datasets: Applications and limitations in toxicity detection. *Proc. LRECCOLING*, (p. 37).
- Scirè, A., Bejgu, A. S., Tedeschi, S., Ghonim, K., Martelli, F., & Navigli, R. (2024). Truth or Mirage? Towards End-to-End Factuality Evaluation with LLM-Oasis. *CoRR abs/2411.19655*.
- Shahid, W., Jamshidi, B., Hakak, S., Isah, H., Khan, W. Z., Khan, M. K., & Choo, K.-K. R. (2024). Detecting and Mitigating the Dissemination of Fake News: Challenges and Future Research Opportunities. *IEEE Transactions on Computational Social Systems*, 11(4).
- Singh, A., & al., e. (2025). Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. *arXiv:2501.09136*.
- Srba, I., Razuvayevskaya, O., Augusto, J., Móro, R., Schlicht, I. B., Tonelli, S., . . . Bieliková, M. (2024). A Survey on Automatic Credibility Assessment of Textual Credibility. *CoRR, abs/2410.21360*.
- Stambach, D., Webersinke, N., Binger, J., Kraus, M., & Leippold, M. (2023). Environmental claim detection. *61st Annual Meeting of the Association for Computational Linguistics* (p. 1051–1066). Association for Computational Linguistics.
- Standards, C. (s.d.). Available from: <https://transparency.meta.com/en-gb/policies/>.
- Wang, Z., Pang, Y., & Lin, Y. (2024). Smart Expert System: Large Language Models as Text Classifiers. *arXiv-2405*.
- Wei, J., & al", ". (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 24824-24837.
- Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., . . . Hu, X. (2024). Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond. *ACM Trans. Knowl. Discovery Data*, 1-32.
- Zanartu, F., Cook, J., Wagner, M., & García, J. (2024). A technocognitive approach to detecting fallacies in climate misinformation. *Scientific Reports, Nature*.