Knowledge-Based Systems 276 (2023) 110750

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A Software Reference Architecture for Journalistic Knowledge Platforms[☆]

Marc Gallofré Ocaña [*], Andreas L. Opdahl

University of Bergen, Department of Information Science and Media Studies, Bergen, 5020, Norway

A R T I C L E I N F O

Article history:
Received 18 October 2022
Received in revised form 5 June 2023
Accepted 24 June 2023
Available online 30 June 2023

Keywords:
Software reference architecture
Newsrooms
Knowledge graphs
Journalism
Artificial intelligence
Big data

A B S T R A C T

Newsrooms and journalists today rely on many different artificial-intelligence, big-data and knowledge-based systems to support efficient and high-quality journalism. However, making the different systems work together remains a challenge, calling for new unified journalistic knowledge platforms. A software reference architecture for journalistic knowledge platforms could help news organisations by capturing tried-and-tested best practices and providing a generic blueprint for how their IT infrastructure should evolve. To the best of our knowledge, no suitable architecture has been proposed in the literature. Therefore, this article proposes a software reference architecture for integrating artificial intelligence and knowledge bases to support journalists and newsrooms. The design of the proposed architecture is grounded on the research literature and on our experiences with developing a series of prototypes in collaboration with industry. Our aim is to make it easier for news organisations to evolve their existing independent systems for news production towards integrated knowledge platforms and to direct further research. Because journalists and newsrooms are early adopters of integrated knowledge platforms, our proposal can hopefully also inform architectures in other domains with similar needs.

1. Introduction

News organisations today are forced to constantly adapt their business models to digital media innovations to increase information quality, competitiveness and growth [1,2]. Potentially news-relevant information can come from almost any type of source and in any data format. The daily global production of news exceeds 100.000 articles [3], while social media generate similar volumes within a second. Consequently, news organisations can benefit from using big data and artificial intelligence (AI) solutions to manage information, extract knowledge and create value for more and more journalistic purposes [4] including: identifying and contextualising newsworthy events to find connections along millions of articles in investigative journalism; facilitating data visualisation with the support of storytelling techniques in digital journalism; automating news writing utilising structured data to automatically create and publish reports about markets, sports and weather (a.k.a. robot journalism, algorithmic journalism or automated journalism); and, providing real-time fact-checking tools to identify fake claims using external knowledge bases in political journalism.

Unsurprisingly, both research and industry agree on the relevance and challenges associated with future AI systems across domains [5]. Particularly, future AI systems must be semantically sound and explainable, as well as foster trustworthy AI. To achieve this, these systems must be able to integrate sub-symbolic deep learning, symbolic knowledge representation and logical reasoning [6]. Knowledge graphs are a topical choice for knowledge representation and reasoning [7] alongside neural networks for implementing sub-symbolic AI. As the world is constantly changing, these systems must incorporate continuous-learning techniques to keep deep learning (DL) and machine learning (ML) models up-to-date.

Journalistic knowledge platforms. An emerging type of information system that integrates AI, big data and knowledge bases to support high-quality journalism [8]. In this article, we refer to these systems as *Journalistic Knowledge Platform* (JKPs). JKPs harvest and analyse news and social media information over the net in real time [3] and leverage encyclopedic sources [9,10]. News-relevant information is semantically annotated and represented in knowledge bases using linked open data (LOD) [11] and AI techniques like natural language processing (NLP) [12]. The resulting knowledge bases are exploited with data analysis, reasoning and information retrieval techniques to provide journalists with meaningful background knowledge and newsworthy information [13,14], as well as to help journalists and readers dive more deeply into information, events and story lines [15–17].

☆ This work was supported by the Norwegian Research Council IKTPLUSS project 275872.

* Corresponding author.
E-mail addresses: marc.gallofre@uib.no (M. Gallofré Ocaña), andreas.opdahl@uib.no (A.L. Opdahl).

https://doi.org/10.1016/j.knosys.2023.110750

JKPs typically implement different mechanisms for interacting with the system, for example, to provide live feeds and alerts and to search for information. Because JKPs combine and represent personal data from different sources, they must also consider the privacy policies [18]. To combat the dissemination of fake news and misinformation, JKPs must also manage the provenance of news and its sources, facilitating its identification. All these aspects make JKPs a particularly complex kind of big-data knowledge-centric intelligent system.

Work on JKPs has so far been driven by the research community, albeit often in collaboration with industry. We envision that the field will continue to gain industrial traction in the near term. Because journalists and newsrooms are early adopters of integrated knowledge platforms in general, we also envision that our work in the journalistic domain can also inform other knowledge-intensive domains that rely critically on exploiting high-volume, high-velocity and high-variety information sources.

Software reference architecture. Today, most news organisations rely on many current, independent and task-specific production systems. However, depending on multiple systems entails a higher resource footprint compared to integrated systems that reduce code and data duplication. The utilisation of multiple systems also increases the cost of coordinating developer teams and providers, as well as the cost of maintaining and updating the systems. Organisations may lose control over their data and knowledge because their systems do not share common data repositories nor representations or are provided as Software as a Service (SaaS) by third parties. As a consequence, organisations may miss out on opportunities for exploiting potentially news-relevant information [8]. These concerns can be addressed by having a clear system design and a system architecture that allows organisations to integrate and expand their solutions in a coherent manner over time. Therefore, in this article, we propose a software reference architecture (SRA) for JKPs. The proposed architecture can also serve as an example of a more general high-level architecture for future big-data AI systems that combine deep learning and knowledge graphs and that support evolving knowledge. We are focusing in particular on JKPs that employ semantic knowledge graphs [7] for knowledge representation.

An SRA for JKPs should provide news organisations with a blueprint and associated advice for how to evolve its many current systems towards a cohesive, comprehensive, and integrated JKP [8]. On the organisational level, central challenges are that JKPs (a) are complex systems that must balance many concerns [3] and are thus challenging to adopt without architectural guidance; (b) must interoperate with a wide variety of in-house legacy systems and external services [13]; and, (c) are long-term investments that must be able to evolve to incorporate future best-of-breed components that replace or come in addition to existing ones [3]. On the technical level, JKPs need to support (a) the ingestion of big data from diverse sources, (b) the semantic annotation, representation and enrichment of news-relevant items [15]; (c) the inclusion of diverse mechanisms for serving potential newsworthy events and information [10,16]; (d) the addition of processes for continuously evolving and adapting machine learning models, ontologies and schemas [9]; (e) the integration of explainable sub-symbolic and symbolic AI approaches [8]; and, (f) the control of data privacy and provenance [18]. The research literature on JKPs has focused on the application side and addressed different challenges for news production. However, the authors are not aware of other lines of work that have studied the architecture of JKPs specifically.

SRA for JKP. Researchers have proposed several software architectures that deal with big data in general [19–24], but few of them deal with the central challenges that JKPs face and, to the best of our knowledge, none of them deals with them all. For example, the current big-data architectures are typically designed for data analysis and immutable data, whereas the focus of JKPs is on exploring and understanding knowledge that evolves over time. In addition, few current big-data architectures consider the integration of knowledge bases and AI in detail. Therefore, In this article, we address the question: "What would be a good software reference architecture for journalistic knowledge platforms?" We propose a software reference architecture that addresses the central challenges of journalistic knowledge platforms and integrates artificial intelligence and knowledge bases to support journalists and newsrooms. We also introduce two novel types of components: one for continuously improving and updating AI models and the other for curating knowledge representations. The first component includes services to monitor data and schema changes and update the models respectively. The second component scans knowledge representations to enrich the content and rectify inconsistencies or missing information. This architecture is the first of its kind proposed for the systems described in this work. Unlike existing big-data architectures focused on immutable data and data analysis, the proposed architecture focuses on evolving knowledge and analysing knowledge representations. The design of the architecture is primarily grounded in the research literature but also relies on our practical experience with developing a series of JKP prototypes in collaboration with the industry. We believe the proposed architecture can be adaptable to other domains with characteristics similar to news production.

In a previous publication [25], the authors have outlined a preliminary version of an SRA for JKPs. The present article extends the earlier outline in several ways: it presents the high-level qualities that an SRA for JKPs must satisfy; it explains the architectural principles that guided the design; it describes a generic architecture for big-data knowledge-based AI systems; it further elaborates the description and argumentation of the SRA specific to JKPs; and it compares the coverage of the proposed SRA for JKPs with the research literature.

The remainder of the article is organised as follows: Section 2 defines our terminology and introduces SRAs, knowledge graphs, embeddings and vector databases. Section 3 describes our research method. Section 4 analyses the related literature. Section 5 outlines the high-level required qualities for an SRA. Section 6 presents the SRA for JKPs. Section 7 evaluates the proposed SRA. Finally, Section 8 states our conclusions and plans for further work.

2. Background

2.1. Central terms

By *big-data technology* we mean the recent generation of middleware that accommodate web-scale data processing and storage. For example, the big-data technologies we use in our work include Apache Kafka and Cassandra. By *knowledge bases* we mean data repositories that maintain strong semantic definitions of and links between the data. Our work focuses on knowledge graphs, using techniques such as RDF, OWL and SPARQL, and Blazegraph for storage. By *AI* we mean symbolic and sub-symbolic techniques, including machine and deep learning for tasks like natural-language processing. Examples of AI techniques we use in our work are named entity linking, relation extraction and inference rules. We also refer the reader to our previous work on the usage of knowledge graphs for news [26] and our review on JKPs [8] for further details on big data, knowledge bases and AI techniques. We proceed to discuss a few other central terms in more detail.

2.2. Software reference architecture

A software reference architecture (SRA) "is a generic architecture for a class of systems that is used as a foundation for the design of concrete architectures from this class" [27]. It defines the basic software elements and data flows and captures the best practices for designing and implementing complex systems and their functionalities.

Two types of SRAs can be distinguished: practice-driven and research-driven [28]. Practice-driven SRAs are based on practical experience developing concrete architectures in a domain. They describe the "best practices" and address legacy problems. Research-driven SRAs address areas that are expected to become important in the future but where there are few or no development experiences yet. They are based on theoretical reflections grounded in the research literature.

2.3. Knowledge graphs

Knowledge graphs provide symbolic representations through concepts, relations and logic rules. According to [7], knowledge graphs capture and abstract knowledge using graph-based data models wherein entities of interest are represented as nodes and the relations between them as edges of the graph. Ontologies and rules are employed to define the semantics and terms of the graph, reason about it, and ease data integration. Knowledge graphs are particularly relevant for systems that integrate and extract value from heterogeneous and dynamic data. They are exact symbolic representations that do not require large amounts of data to become meaningful. Their workings are easy to explain to humans, but managing large graphs efficiently can be hard for computers. Compared to relational and NoSQL models, knowledge graphs facilitate semantic integration, flexible data and schema evolution, along with graph query languages for exploring complex relations through arbitrary-length paths.

2.4. Embeddings and vector databases

Embedding techniques are used in machine and deep learning to represent concepts as vectors in a latent space. Concepts can be extracted from a wide variety of data from text, images, and audio to sequences like DNA and molecular structures. Vectors are generated through mathematical models and provide sub-symbolic representations, positioning concepts in the embedding space according to similarity or other relations. These vectors, as sub-symbolic representations, have a stochastic component and require large amounts of data to be meaningful. They are hard to explain to humans, but even large collections of vectors can be efficiently managed by computers.

Well-known techniques for word and text embedding are word2vec [29] and transformers [30] like BERT [31] and, most recently, GPT-3 [32]. These techniques are particularly relevant for systems that exploit the semantic and contextual similarity between data and used in many AI applications like natural language processing, chatbots, image recognition, and recommendation.

Storing large collections of vectors requires specialised databases with optimised storage, access and search. Vector databases are an emerging technology for storing and indexing vectors efficiently and provide functionalities for retrieving vectors using similarity search algorithms like HSNW [33] and FAISS [34]. Examples of vector databases are Milvus[1], Weaviate[2] and Vald[3].

[1] milvus.io
[2] weaviate.io
[3] vald.vdaas.org

3. Method

To design and validate the SRA, we follow an established method for designing empirically-grounded reference architectures [35]. The method comprises six steps, where the initial five steps provide the "empirical foundation" and the sixth provides the "empirical validity" as illustrated in Fig. 1:

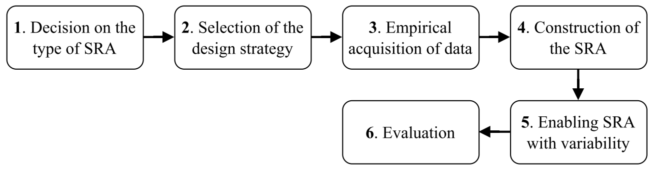
1. Decision on the type of SRA
2. Selection of the design strategy
3. Empirical acquisition of data
4. Construction of the SRA
5. Enabling SRA with variability
6. Evaluation

Fig. 1. Construction process of the SRA for JKPs.

Step 1 – Decision on type of SRA: We chose the preliminary and facilitation type of SRA described in [27]. This is a type of research-driven of SRA that aims to facilitate guidelines for designing systems and concrete architectures that are likely to become important in the future and will be utilised by multiple organisations. The guidelines are designed by researchers in collaboration with interested software organisations and grounded on existing research literature and practical experience.

Step 2 – Selection of design strategy: We chose a research-driven strategy because we expect JKPs to become increasingly important in the future and we have not identified a substantial number of industrial implementations and experiences on JKPs.

Step 3 – Empirical acquisition of data: We carefully selected 13 research projects that matched our definition of JKPs (see Section 4.1). A detailed, qualitative meta-analysis review was presented in [8] to derive the challenges, opportunities, main stakeholders, information, functionalities, techniques, components and concerns in JKPs. We also drew on our practical experiences developing a series of JKP prototypes with Wolftech[4], a software developer for the international newsroom market [36,37], and collaborating with a large cluster of news media industry partners in the MediaFutures research centre[5] [38].

Step 4 – Construction of SRA: Firstly, we identified the related literature on JKPs and software architectures for big data and semantic technologies (see Section 4). Secondly, since we could not find a suitable architecture for JKPs, we decided to propose a new SRA grounded on the specific research literature, our practical experiences and the general literature on architectural principles. From this, we derived the main required qualities for JKPs (see Section 5) and selected suitable architecture principles (see Section 6.1). Thirdly, the required qualities were mapped into architectural elements, such as components, functionalities, data stores and data flows. These identified elements were then further mapped into a high-level architecture view of the SRA (see Section 6), guided by the architecture principles. Finally, we instantiated the SRA as a proof-of-concept prototype of a JKP (see Section 7), which we used to iteratively improve the SRA.

[4] wolftech.no
[5] mediafutures.no

Step 5 – Enabling SRA with variability: By iterating over the SRA design and continuously developing and testing the JKP prototype, we improved and refined the preliminary design decisions reported in [25,37] concerning the architecture, components, principles and semantics. To facilitate the adaptation of the SRA in other domains, we drew insights from generic big-data architectures to identify common characteristics shared by our SRA for JKPs and big-data AI architectures. As a result, we propose a high-level view of our architecture for JKPs that can potentially be adapted to big data and AI systems in other domains. We also show how the high-level view is refined into our specific SRA for JKPs and further instantiated into our prototype JKP. As JKPs are an emerging field and there are not enough research results or experience available to empirically back-up architecture variants, a more thorough investigation of variability must be left for future work.

Step 6 – Evaluation of the SRA: Following the established method for empirically-grounded SRA development [35], we have evaluated our SRA proposal in two ways:

Mapping-based evaluations: To ensure fulfilment of the required functional qualities for an SRA for JKP (Section 5.2), we systematically mapped these qualities to the architecture components (Table 3). Similarly, we mapped the required non-functional qualities (Section 5.3) to both architecture principles (Table 2) and architecture component (Table 4). Finally, to ensure that our SRA proposal accounts for the components, functionalities and goals of JKPs reported in the existing research literature (Section 4.1), we systematically mapped them into architecture components (Table 5).

Evaluations by prototype development and testing: We have validated the viability of the high-level architecture view (Section 6.2) by refining it into a concrete SRA for JKPs (Section 6.3). Furthermore, we have validated the feasibility of this concrete SRA for JKPs by instantiating it into a running JKP prototype that has been iteratively developed and tested (Section 7.2).

To construct and evaluate our prototype in Steps 4–6, we have followed a design science approach [39], which "supports a pragmatic research paradigm that calls for the creation of innovative artefacts to solve real-world problems" [40]. Within the field of information systems, design-science researchers often adopt an iterative process comprising three different cycles [41], which involve understanding the application context or environment, studying and improving the theoretical framework, and evaluating the artefacts [42]. Accordingly, we have iteratively designed and validated our SRA for JKPs by developing and refining artefacts informed by the relevant literature, while considering the contextual environment of both developers and users.

4. Related literature

4.1. Journalistic knowledge platforms

Several JKPs have been proposed in the research literature. We summarise the projects we have identified, along with their industry partners, in Table 1. We can broadly categorise them into two groups: the earlier JKPs (until around 2010), which primarily focused on implementing the Semantic Web idea [43] within newsrooms, and the more recent JKPs (after 2010), which combined semantic technologies [44] with machine- and deep-learning approaches.

The earlier JKPs employed semantic technologies and ontologies to automate the metadata annotation process, combine different knowledge bases, and formalise media standards. They used ontologies in NLP pipelines, together with LOD, to automatically annotate news archives and feeds with metadata about topics, keywords, categories and other relevant information (e.g., persons, places, organisations, sentiments and relations). For example, *PlanetOnto* [45] focused on providing a knowledge management system to provide personalised semantic retrieval and search in news archives. *Neptuno* [47] developed tools for creating, maintaining and exploring news archives. *AnnoTerra* [48] proposed a prototype for integrating earth science data sources to enhance news feeds from NASA's Earth Observatory using knowledge bases. *SemNews* [49] focused on automating metadata annotation for semantic search and monitoring of RSS feeds. *Hermes* [59] proposed a framework for searching and classifying news to support decision-makers. The *BBC* used knowledge graphs and LOD to link information across news articles, enrich their Content Management System (CMS) and recommend news [9,51]. *NEWS* [13] automatised the metadata annotation of news and images and provided news intelligent information retrieval services using semantic technologies.

More recent JKPs focused on identifying and analysing events and advancing machine and deep learning for supporting journalism. A common thread among them, and some of the earlier examples, was that they deal with big data. *EventRegistry* [15] developed a tool for collecting news articles from around 75.000 multilingual sources, identifying and extracting information about the events, and summarising and visualising events from close to 200.000 articles daily. *NewsReader* [16] presented a platform for machine reading of multilingual streams of news and extracting information about what, who, where and when for representing events temporally using knowledge graphs, for example allowing users to find networks of actors and their implication over time. The platform was tested on nearly 2.5 million news articles and extracted over 1.1 billion triples from these articles. *Reuters* [55,56] developed a real-time platform to analyse around 12 million tweets per day from Twitter to identify and verify newsworthy events before they are reported by other news agencies and automate news production processes. *SUMMA* [3] developed a multilingual and multimedia platform employing NLP techniques for monitoring internal and external live media, including TV and radio broadcasts, and providing services for data journalists. *INJECT* [58] developed a tool to support journalists by providing creative angles on news stories. *ASRAEL* [10] presented a system for aggregating news articles and utilising the Wikidata knowledge base for describing and clustering events in news from a corpus of over 2 million articles.

Typical problems faced in these projects are: huge volumes of heterogeneous data, some of them arriving in real time [3,15,16]; complex processing pipelines that combine NLP, machine learning and knowledge representation [10,15,16]; and integration of legacy and external systems [9,13]. These are problems that typically call for architectural guidance.

4.2. Software architectures for big data and semantic technologies

According to existing big-data architecture reviews [19–24], only four architectures for big data [19,23,60,61] have considered semantic technologies. *LMS* [60] was designed for providing a middleware for sensor data and the Internet of Things (IoT). *SOLID* [61] adapted the principles of the *Lambda* processing architecture [62] to RDF for gathering, storing and serving big data in real time. *Bolster* [19] extended the Lambda architecture by adding a new semantic layer to represent machine-readable metadata, contrary to JKPs that represent the data semantically.

Table 1
Selected projects. N = news media partner and T = technology partner.

Project	Industry partners	References
PlanetOnto	–	[45,46]
Neptuno	Diari SEGRE[N] and iSOCO[T]	[47]
AnnoTerra	NASA's Earth Observatory[N]	[48]
SemNews	–	[49]
Hermes	–	[50]
BBC CMS	BBC[N]	[9,51]
NEWS	Agencia EFE[N], Agencia ANSA[N] and Ontology Ldt.[T]	[13,52]
Event Registry	–	[15]
NewsReader	LexisNexis[T], The Sensible Code Company (before ScraperWiki)[T] and Synerscope[T]	[16,53]
Reuters Tracer	Reuters[N]	[54–56]
SUMMA	LETA[N], BBC Monitoring[N], Deutsche Welle[N] and Priberam Labs[T]	[3,57]
INJECT	Adresseavisen[N], AFP[N], The Globe and Mail[N], Stibo[T]	[58]
ASRAEL	AFP[N]	[10]
News Angler[a]	Wolftech[T]	[25,36]

[a]News Angler is the research project in which the authors are involved.

SmartLAK [23] focused on supporting learning analytic services and defines components for validation and inference based on ontologies. However, none of them covers mechanisms for semantic data enrichment, continuously pushing live data streams, or continually (re-)training machine learning models. Four proposed architectures [63–66] have considered maintaining and updating ML models and defined specific components for storing and training them, but none of them considered semantic technologies like knowledge graphs and ontologies. Furthermore, none of the existing architectures considers the curation of knowledge representations. In conclusion, none of them is a suitable starting point for an SRA for JKPs.

5. Required qualities for the SRA

5.1. Approach

To drive the design and evaluation of our SRA, we systematically derived the required high-level qualities from earlier JKP projects reported in the literature and our previous studies [8,18]. We used the most recent JKPs to derive the qualities, while the earlier JKPs provided supplementary insights to support and augment these qualities. We divided the JKP qualities into functional (i.e., specific behaviours that the system must implement) and non-functional (i.e., general properties of the system). In addition, we identified the required general qualities for any SRA from the literature [27,67], i.e., being feasible, representative, essential, easy to grasp, long-lasting and technology independent. The derived qualities are also corroborated by the current literature on big data architectures [21,22,24] and AI systems [5], as well as they align with quality attributes of ISO/IEC 25010 [20].

5.2. Required functional qualities

`Annotating` To better manage, analyse and derive knowledge to support journalists in creating high-quality stories, JKPs must annotate content with relevant information such as people, organisations, places, relations, categories, themes and other metadata [3,15,46–50,55,58]. JKPs use semantic annotations to facilitate the representation of the meaning of concepts and relations, standardise annotations using well-defined schemas and ontologies, and improve relation and concept mining [9,10,13,16].

`Knowledge-representation` JKPs are knowledge-centric systems that provide knowledge representations of news, events and background information and the relations between them [3,9,10,15,55,58]. Most of the JKPs are focused on exploiting the relations and connections between information. Hence, JKPs must employ systems that facilitate working with relations and updating the knowledge representations [13,16,46–50].

`Enriching` JKPs must implement mechanisms to update and expand the extracted information. Because the information can change over time (e.g., a newly elected head of a government) and some other information may not be completed (e.g., an article referencing a country instead of the city where an event took place), journalists need to constantly have access to up-to-date and fine-grained information to produce high-quality journalism [3,9,10,13,15,16,46–50].

`Schema-evolution` As novel developments and themes may appear, JKPs must facilitate schema evolution [13,16,46–50]. To do so, the technologies used to represent and store schemas must provide flexible and easy mechanisms to update them.

`Model-updating` AI models must be constantly updated to follow new information and news development [13,15,46,55]. JKPs must implement mechanisms for continuously evolving ML models to provide state-of-the-art results and adapt them to new events and users' needs.

`Storage` JKPs deal with a variety of data and representation formats [3,9,10,13,15,16,46–50,55,58]. They need to access information in different ways, for example, obtaining most recent feeds in real-time, reading data in bulk, retrieving historical data and finding similar texts. Therefore, JKPs must employ different databases for specific purposes to optimise storage and access.

`Push` JKPs must continuously push potentially newsworthy events to journalists [3,13,46–49,55]. This is achieved by, for example, sending feeds or alerts to journalists according to their preferences or current work.

`Pull` JKPs must provide services for pulling information from the knowledge base [3,9,10,13,15,16,46–50,55,58]. These services typically require direct interaction with the user to search information.

`Data-ownership` Stories generated with the support of JKPs are disseminated to a broad or even worldwide audience and the information resources need to be protected as they may be subject to ownership and usage policies [13,48]. JKPs must keep track of these policies and their implications, especially when information is merged or derived from multiple sources.

`Privacy` Merging and connecting information from different sources and social media can lead to data privacy challenges [18]. Hence, JKPs must implement mechanisms to monitor potential data privacy violations.

`Provenance` Information about the source from where the information has been derived helps journalists to assess the quality of news and find its origins. In addition, metadata about the process and version that gathered, modified or updated the information facilitates the traceability of the process and detection of errors [13,16,48]. Thus, JKPs must facilitate keeping track of the metadata associated with the information sources and processes.

5.3. Required non-functional qualities

`Interoperability` JKPs interoperate with heterogeneous in-house legacy systems, external services and other JKPs [3,9,10,13,16,46–49]. To do so, they need to provide clear meaning and data representations, as well as use standard formats, interfaces and exchange protocols.

`Modularity` JKPs must be able to incorporate future components that replace existing ones [9,13,16,48]. This guarantees the addition of new components to adapt the JKPs to particular users' needs and update them with future best-of-breath solutions. Hence, the implemented components need to be independent, modular and abstract.

`Scalability` To deal with large volumes of news-relevant information and sources, JKPs must employ tools and storage systems designed to increase to, efficiently support and uniformly process big data volumes [3,9,10,13,15,16,47,55].

`Velocity` JKPs support news production where time is a critical factor and delays can lower the value of information. News-relevant information is rapidly and continuously produced and broadcast worldwide [3,16,47,55]. JKPs must obtain this information, process it, analyse it, and make it available as soon as possible to maximise its value.

`Variety` News-relevant information is produced and broadcast as unstructured and structured data [3,9,13,16,47,48]. It comprise diverse modalities of data like audio, video and images, structuration principles like tables and graphs, time cycles like live and historic, and formats like plain text, RDF and JPEG. Therefore, JKPs must be able to ingest, process and store varied data consistently.

`Knowledge-evolution` As the world is constantly evolving, current events and developments become past and are preceded by new ones [13,15,50]. Therefore, JKPs must implement components that can adapt their behaviour in response to emerging entities, events and relations along with new terms and their meaning.

`Sub-/symbolic-AI` JKPs integrate sub-symbolic and symbolic techniques. This integration benefits both approaches: sub-symbolic techniques may be enhanced with logic and reasoning from symbolic AI, and symbolic AI may be sped up with sub-symbolic techniques [6,68]. To support this integration, JKPs must facilitate both symbolic representations like knowledge graphs and ontologies and sub-symbolic data like models and training materials.

`Trustworthy-AI` As JKPs support journalists in creating stories that may effect society, journalists need to trust the system [9]. Hence, following the European guidelines on AI [69], JKPs must allow journalists to take informed decisions, ensure data privacy and integrity and provide transparent, traceable and explained solutions.

5.4. Qualities addressed by the JKP projects

In Appendix we trace from which projects each quality has been derived (see Tables A.6 and A.7). The analysed projects addressed qualities primarily related to the technical and research challenges such as `Annotating`, `Knowledge-representation`, `Enriching`, `Schema-evolution`, `Storage`, `Push`, `Pull`, `Interoperability`, `Scalability` and `Variety`. Only a limited number of projects addressed qualities that support the validation of the newsroom production such as `Data-ownership` and `Provenance`. Despite early examples of `Model-updating` in the earliest projects, advances in machine learning increased its relevance in the newer projects, while `Sub-/symbolic-AI` remains unexplored. The rise of web-scale volumes made `Velocity` relevant. Earlier projects explicitly addressed `Modularity`, which newer projects achieve implicitly through technological decisions. Few projects considered the `Knowledge-evolution` as opposed to static knowledge. `Privacy` has not been addressed by any project, despite its importance for complying with regulations such as the GDPR. `Trustworthy-AI`, relevant for providing safe systems and understanding their outcomes, was only addressed by one project. We particularly emphasise on the underrepresented yet relevant qualities for future systems.

6. Software reference architecture for JKPs

6.1. Architectural principles

We propose a set of architectural principles for the SRA for JKPs. These principles are composed of different architectural patterns and technologies that we consider the most appropriate to fulfil the elicited high-level non-functional qualities as illustrated in Table 2.

Microservice architecture pattern. Microservices is an architectural pattern for applications where every functionality is deployed as its own service and often independent from the others [70]. Components in a microservice system are self-contained, loosely coupled, technology neutral, reusable and specialised. They typically communicate via clear APIs. These characteristics facilitate components replacement, integration, scaling and distribution. This pattern is an ideal candidate for an SRA for JKPs as it provides `Interoperability` and `Modularity` by design. Components designed following the microservice architecture principles can (a) be easily deployed, integrated and updated because they have clear boundaries and minimal technological dependencies on other components; (b) be dynamically replicated to meet specific processing loads; and, (c) be utilised independently or in collaboration with other components to fulfil business functionalities. Solutions like *Docker*[6] containers can be used to improve the availability, `Scalability`, replaceability and deployment of microservices.

Liquid architecture pattern. Liquid architecture [71] is an architecture pattern for integrating nearline and offline big-data processing with two distinguished layers: the processing and the messaging layer. The processing layer executes ETL-like jobs for different back-end systems. The messaging layer follows a topic-based publish/subscribe communication model where streams of incoming messages are identified by topics. Jobs can read from selected topics and output to new ones. Messages can contain metadata annotation such as timestamps that are used to provide stateful and incremental processing. Each job is an isolated resource that may perform several tasks and communicate with other jobs, creating a dataflow processing graph. Compared to

[6] www.docker.com

Table 2
Connection between non-functional qualities and architecture principles.

	Microservices	Liquid	Blackboard	Semantic technologies
`Interoperability`	✓			✓
`Modularity`	✓			✓
`Scalability`	✓	✓		
`Velocity`		✓		
`Variety`				✓
`Knowledge-evolution`			✓	✓
`Sub-/symbolic-AI`			✓	✓
`Trustworthy-AI`				✓

the well-known Lambda [62] and Kappa [72] architectures, Liquid does not duplicate the code, as opposed to Lambda; and, it does not need to reprocess the current data view to run batch jobs, as opposed to the Kappa. Unlike other architecture patterns like Phi [73] that offer similar benefits, Liquid provides resource isolation and incremental data processing, as well as it removes the need for duplicating the data for downstream processing. The Liquid architecture pattern is an excellent candidate for an SRA for JKPs, because it is designed for meeting the `Scalability` and `Velocity` requirements of big data, and reduces the development, maintenance efforts and hardware demands [71]. Event-streaming solutions like *Apache Kafka*[7] can be employed to implement the message layer.

Blackboard model. The blackboard model is a problem-solving approach to solve complex problems where different kinds of domain knowledge and expertise are needed [74]. In a blackboard-based system, independent components cooperate to solve problems using a shared knowledge base (viz., the blackboard) [75]. These components activate when there is a change in the knowledge base or an event that meet a certain condition. They may also modify the knowledge base to contribute towards the solution. The behaviour of these components depends on the current state of the knowledge base and adapts as the knowledge evolves. This increases the response of the components of JKPs to `Knowledge-evolution`. As components cooperate to solve a problem by sharing resources or preliminary solutions, components can use the output of a sub-symbolic method to build on a symbolic one or vice versa. Hence, the blackboard model facilitates the integration of `Sub-/symbolic-AI`.

Semantic technologies. Semantic technologies encompass technologies and standards in the context of the Semantic Web [43] that deal with the meaning of the data rather than its structure [44]. They are designed to represent entities, their relationships and attributes using well-defined ontologies, and optionally, logic rules. Semantic technologies are commonly used to construct knowledge graphs and integrate LOD [7]. Software architectures of JKPs implementing semantic technologies may benefit from (a) language neutrality and clear representations of data and meaning to improve `Interoperability` and `Modularity` between systems and `Variety` of sources and data formats; (b) LOD sources that constantly update their knowledge bases like DBpedia and Wikidata to improve `Knowledge-evolution`; (c) using knowledge graphs to facilitate `Trustworthy-AI` by improving explanations, for example, by exposing connections and relations, providing context and showing semantic similarities [76]; and, (d) the symbolic representations that these technologies provide to enable `Sub-/symbolic-AI` integration.

6.2. High-level view

Fig. 2 proposes a high-level SRA for journalistic knowledge platforms, which implements the architectural principles. The architecture is centred around the knowledge base and includes four types of components. The *Knowledge Base* manages all the information needed to operate the system, including data stores, knowledge representations, schemas, ontologies, metadata, and even AI models which are provided and deployed as a service to facilitate their access and integration in the architecture. The knowledge base utilises unique identifiers, such as IRIs, to ensure consistent representation across services. This eases the integration of knowledge representations and vectors from AI models, as they share the same identifiers. To integrate the knowledge base with the rest of the system, the system is designed following the blackboard model, where the knowledge base is placed in the centre of the system and communicates bi-directionally with the other components. The *Input* components collect and analyse relevant information from outside the platform and store it in the knowledge base. The *Output* components provide services to push relevant information to users and let users pull information from the knowledge base on demand. The *Learner* components employ continuous-learning techniques to keep the ML/DL models, ontologies and schemas up to date. The *Curator* components maintain and improve knowledge base contents. The principal difference between the Learner and the Curator is that the Learner is focused on improvement at the type or meta level, whereas the Curator is focused on the instance or production level. For example, the Learner may include services for automatic re-training of ML/DL models or semi-manual update of the schema and ontology. The Curator services deal with knowledge fusion and enrichment tasks such as addressing knowledge contradictions, updating knowledge representations based on external knowledge bases like DBpedia, merging similar knowledge representations, and controlling potential privacy violations as the ones described in [18].

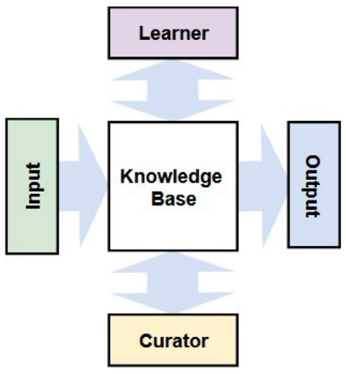

Fig. 2. High-level view of the architecture.

7 kafka.apache.org

To facilitate machine-readable and understandable data, all components utilise semantic descriptions to represent and describe the content, thereby reducing ambiguities and facilitating integration and communication. Furthermore, to guarantee to ensure traceability of every piece of information back to the generating process, every service of the SRA maintains `Provenance` information. Our proposed architecture goes beyond the existing big-data architectures in the literature, as it explicitly incorporates components like the Curator and Learner, providing clear pathways for their inclusion.

6.3. SRA for JKPs

Fig. 3 illustrates the SRA for JKPs in more detail. While the high-level view in Fig. 2 may apply to other knowledge-based domains, this architecture is specific to news work. To align with existing literature on JKPs and enhance comprehension, we have renamed the components of the SRA accordingly, providing a nomenclature that reflects their most common intended purposes. We also decided to split the Output component into a Feeder and a Retriever to differentiate between the `Pull` and `Push` types of interaction in JKPs. As a result, the SRA for JKPs is composed of six groups of components, namely, the *Ingestor*, *Knowledge Base*, *Curator*, *Learner*, *Feeder* and *Retriever*, each consisting of several microservices. A less refined version of the architecture was presented in [25], which did not consider, among other things, explicit components for training and updating AI models and schemas nor the Current Window and Vector Store of the Knowledge Base.

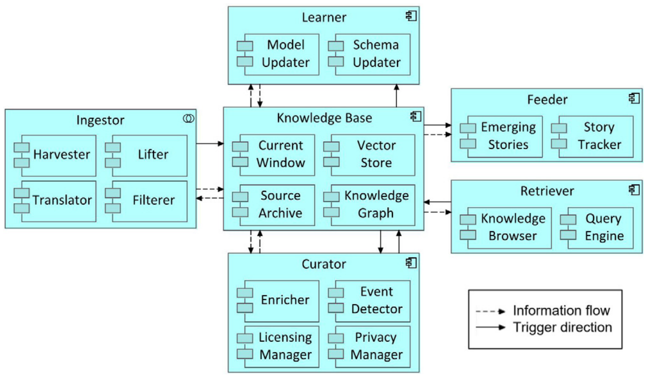

Fig. 3. The SRA for JKPs (represented with ArchiMate 3.1 notation).

6.3.1. Ingestor

The Ingestor collects and `Annotating` potentially news-relevant information items such as news articles and social media messages, multimedia files and structured data from online sources. The most relevant components are the *Harvester* and the *Lifter*. The Harvester continuously downloads and ingests scheduled and real-time news-relevant items from sources like RSS, APIs and websites. To handle data `Variety`, multiple harvesters can be deployed, each of them targeting specific sources or data formats. The Lifter annotates and transforms these news-relevant items into `Knowledge-representations` using semantic technologies and AI techniques before they are uploaded to the Knowledge Base. The resulting knowledge representations can be using RDF and predefined ontologies, such as the Event Description Ontology [77]. Ontologies must be designed general enough to facilitate `Schema-evolution` and `Interoperability` between services. Lifters are composed of different AI modules specialised in different tasks (e.g., named entity recognition and face recognition), which are designed to be replaced or extended (`Modularity`) to follow the state-of-the-art [12]. To combine the results from the different AI modules and improve data `Interoperability`, these can use vocabularies for representing annotations like NLP Interchange Format (NIF) [78] or NLP Annotation Format (NAF) [79]. Each annotation must provide information about its quality (e.g., accuracy and support values), the `Provenance` to trace back to the source and process that generated it, the `Data-ownership` and the terms of use.

Additional services like the *Translator* and *Filterer* can be added to pre-process and clean the collected news-relevant items. For example, the Translator service can be utilised for translating the text into a canonical language, while the Filter can handle tasks like normalising data types, standardising formats, and filtering out advertisements. In some cases, it may be necessary to employ other services that can group micro-texts, such as Twitter messages, into chunks of similar messages, enabling them to be processed collectively. These micro-texts may not be relevant enough on their own, but they may turn relevant when analysed together or when many similar texts occur at the same time or within the same location.

As a result, the Ingestor performs the real-time transformations once and near the source before they are stored in the Knowledge Base, and processed further by the Feeder and Curator. As this provides both the raw data and knowledge representations from the beginning, it facilitates the deployment and integration of `Sub-/symbolic-AI` approaches. By processing the data near the sources, we also avoid software and data duplication, reducing the computational resources needed to deploy the platform. Hence, this can have an impact on the overall power and resource utilisation.

6.3.2. Knowledge base

The Knowledge Base provides persistent `Storage` and is composed of multiple specialised databases for different data, including raw files, metadata, `Knowledge-representation`, schemas, ontologies, vectors and ML/DL models. The use of IRIs facilitates data identification across databases and provides `Provenance`. The usage of specialised databases is encouraged to optimise data storage as they can efficiently manage specific types of data and perform better on certain types of queries and workloads. For example, by only storing the knowledge representations in a knowledge graph and storing the raw data text in a different database, we can reduce resource utilisation associated with knowledge representations and improve performance when exploring relationships between concepts.

The *Source Archive* can be composed of multiple databases for managing a `Variety` of raw files such as text and multimedia files. The *Knowledge Graph* stores the knowledge and schema representations. It represents news-related knowledge and provides a historical data view that can be updated to capture `Knowledge-evolution` and `Schema-evolution`. Semantic technologies like RDF and triple-store databases facilitate both the `Knowledge-evolution` and `Schema-evolution` because they provide flexible data representations and natural integration with linked data. The Knowledge Graph is used as a hub to provide `Interoperability` with the different repositories and integrate legacy archives as it can be used to manage data lakes [80]. The *Current Window* provides a live and dynamic view of the most recent data and knowledge representations coming from the Ingestor, Curator and Retriever. It must provide real-time responses and support streaming operations. Many machine and deep learning solutions are often based on embedding techniques for representing the meaning of, for example, text, graphs and images. The *Vector Store* stores these embeddings in vector databases. Vector databases facilitate the availability and search of vectors, reduce the need for re-computing them and provide similarity functions that can be used to optimise information retrieval. Some vector databases offer the possibility of storing the metadata associated with the vector (e.g., if multiple news-relevant items have been used to generate the vectors, we can add their IRIs as metadata). This can enhance the level `Trustworthy-AI` on JKPs, as it allows for more explainable AI by providing `Provenance` to the vector representations and their resulting outcomes. At the same time, interlinking all databases and vector representations with IRIs simplifies data collection and generation for solutions that integrate `Sub-/symbolic-AI` and update ML/DL models.

These storage services must handle large volumes of data, intensive write and read operations in real-time (`Velocity`), and horizontally scale (`Scalability`). For example, distributed databases like *Apache HBase*[8] and *Cassandra*[9] can store large data volumes. Although many of the open-source graph databases and triple stores with support for RDF and SPARQL do not provide support for scaling horizontally, some of them can hold more than one billion (10^9) triples [81] (e.g., *Blazegraph*[10] and *Jena TDB*[11]). Strategies like partitioning the graph databases according to resource types/predicates, temporal aspects, themes and geolocations, or a combination of these can be employed for distributing graph databases.

6.3.3. Curator

The main purpose of the Curator is to make the Knowledge Base as useful as possible for journalistic purposes. The *Enricher* enhances `Knowledge-representation` using external information from the LOD (e.g., Wikidata). It enriches the Knowledge Graph by adding linked data retrieved from the LOD cloud to expand the represented news-relevant information and events (`Enriching`) and updates or corrects them following the latest advances (`Knowledge-evolution`). The *Privacy Manager* monitors incoming data to identify and propagate prohibitions, permissions, obligations and violations [18] and outputs alerts that need to be rectified by the user. The *Licensing manager* controls the data copy-rights and licensing in the Knowledge Base. For example, when different permissions are merged, the licensing manager maintains data usage obligations and restrictions, identifies the `Data-ownership` conflicts and adds the corresponding missing information.

Additional services can be added to analyse news-relevant information and events and produce newsworthy information for journalists. For example, the *Event Detector* detects newsworthy events from social media and other sources. An aggregator service can incrementally relate and cluster news-relevant items into more comprehensive and reliable event representations or storylines. A network analyser service can identify and analyse different types of connections between actors and their relations with the events. An angle detector service can derive the news angles that fit an event [17,82]. An analogy service can find analogies between different news-relevant items [83]. The Curator can also contain services to provide explanations of the AI results to users (`Trustworthy-AI`).

6.3.4. Learner

The *Learner* provides services to keep the AI models and schemas up-to-date. The *Model Updater* uses continuous-learning techniques such as incremental or online training approaches to improve those models that depend on the frequency and context of words and entities or user preferences. The process of `Model-updating` can be triggered when a significant frequency of an unknown word/entity is detected to incorporate it into the model (e.g., the first mentions of "COVID-19"), use the information from the last week to evolve the model (e.g., after unveiling a corruption case some politicians should be placed closer to the corruption theme), or adapt the recommendations following the current work of a journalist (e.g., when the journalist starts working on a new story). To generate training materials, the Model Updater can access the stored data in the Knowledge Base and the latest version of external repositories such as Wikidata and Dbpedia or the most recent and current events in the Current Window. The resulting updates can be stored in the Vector Store or change the models used by other services, creating a new version of the model. The *Schema updater* evolves current schemas or mappings like themes or categories to include newer elements or remove obsolete ones based on the incoming or on-demand data (`Schema-evolution`).

6.3.5. Feeder

The Feeder monitors streams of linked data coming from the Current Window to `Push` live information to journalists. It continuously pushes newsworthy feeds and alerts based on users' preferences. For example, the *Emerging Stories* service can be implemented to identify live stories that are gaining attention from various publishers or social media platforms, and a *Story Tracker* to push stories that are related to the current journalists' work. As the Feeder is intended to push information to journalists as soon as it is captured or generated, it improves the `Velocity` of information transmission and discovery and reduces delays. The Feeder implement end-points where other services can connect to get live feeds and alerts (`Interoperability`).

6.3.6. Retriever

Retriever allows users to `Pull` information from the JKP on demand. For example, the *Query Engine* facilitates querying and analysis of data from the knowledge base, generating data visualisations, access to taxonomies, and retrieval from news and multimedia archives. It can provide an end-point with pre-packaged queries for particular purposes, like finding news stories related to a particular person and retrieving relevant information for a given event. The *Knowledge Explorer* provides access to background and related information from external sources. Additional services can provide tools such as currency and time converters, related story retrieval, and suggestions to enhance news stories, including news angles. These services are also exposed as API to allow external users and systems to pull information from the JKP (`Interoperability`).

7. Validation

As explained in the Method section, we have validated our proposed SRA in two ways: (1) by mapping between each of the required qualities from Section 5 and our SRA; and (2) by iteratively developing and testing a prototype implementation of the SRA.

7.1. Mapping

We established mappings to verify that the proposed SRA fulfils all the required qualities. To do so, we examine which components contribute towards each quality and how.

8 hbase.apache.org
9 cassandra.apache.org
10 www.blazegraph.com
11 jena.apache.org

Table 3
Mapping between functional required qualities and components.

Functional quality	Ingestor	Knowledge Base	Curator	Learner	Retriever	Feeder
`Annotating`	News items annotation					
`Push`						Feeds and alerts
`Pull`					Query on demand	
`Model-updating`				Learning techniques		
`Enriching`			LOD addition			
`Knowledge-representation`	News items to graphs		LOD addition			
`Storage`		Persistent databases				
`Schema-evolution`				Schema updates		
`Data-ownership`	Terms-of-use metadata		Terms-of-use monitoring			
`Privacy`			Data privacy monitoring			
`Provenance`	Tracing metadata	IRI	Tracing metadata	Tracing metadata	Tracing metadata	Tracing metadata

Table 4
Mapping between non-functional required qualities and components.

Non-functional quality	Component	Principle
`Interoperability`		Microservices, Semantic Tech.
`Modularity`		Microservices
`Scalability`		Liquid Architecture, Microservices, Blackboard Model
`Velocity`	Feeder	Liquid Architecture
`Variety`	Ingestor, Knowledge Base	Blackboard Model, Semantic Tech.
`Knowledge-evolution`	Curator, Learner	
`Sub-/symbolic-AI`		Blackboard Model, Semantic Tech.
`Trustworthy-AI`	Curator	

Functional qualities:. To validate that all required functional qualities (Section 5.2) are covered by the SRA, we mapped them to the architecture components. Table 3 shows the components responsible for providing or realising each functional quality and highlights the key aspects that support it. Certain qualities may be associated with multiple components.

As shown in Table 3, the SRA defines Ingestor components for `Annotating` news items and transforming them into `Knowledge-representations`, which are enriched in the Curator with LOD. It also defines components such as the Feeder to `Push` live feeds and alerts and the Retriever to `Pull` information on demand. To keep ML/DL models and schemas up-to-date, the SRA defines the Learner that employs different learning techniques to evolve them. The news-relevant information is persisted in specialised databases in the Knowledge Base. To keep track of `Data-ownership`, the Ingestor adds terms-of-use information to each news item and the Curator monitors them, as well as, potential `Privacy` violations. All components add tracing metadata to provide `Provenance` and utilise IRIs to facilitate identification across services.

Non-functional qualities:. The non-functional qualities (Section 5.3) depend on the architectural principles, the development decisions and specific components, as shown in Table 4. To achieve `Interoperability`, the SRA is based on semantic technologies and vocabularies, schemas, linked data and open standards. These technologies provide language neutrality, formal data representations, open definitions and clear `Knowledge-representation`. This brings data understanding at a conceptual level and facilitate data integration and fusion.

To achieve `Modularity`, the SRA is based on microservice principles. The different components must have clear functional boundaries, so they can be deployed independently and interact between them effortlessly. This facilitates the replacement and addition of new components without affecting the current ones or modifying them. Well-defined boundaries also facilitate communication with external users.

The data ingestion part of the SRA is inspired by the Liquid architecture and microservice principles to handle big data `Scalability` and `Velocity` needs. The Liquid architecture combined with microservices offers a development pattern for designing scalable systems. The annotation components are built upon the blackboard model which is designed for parallel processing and ease scalability. Data `Variety` can be handled by adding new processes to the Lifter for the different types of data and formats. To combine different types of data and annotations, the SRA for JKPs uses semantic technologies to represent them. In addition, the SRA is designed with a knowledge base that can integrate specialised databases for the different types of data. In addition, the SRA enhances the response time with components like the Feeder.

We designed the SRA with the Curator and the Learner components to manage the `Knowledge-evolution`. The Curator contains services for maintaining and improving knowledge representations using external and internal knowledge. While the Learner contains services for updating the ML/DL models to adapt knowledge extraction and representation to current developments.

The integration of `Sub-/symbolic-AI` is accomplished by using the blackboard model and semantic technologies. The SRA makes the source data and its symbolic and sub-symbolic representations available in the knowledge base, while it keeps the involved concepts aligned using IRIs. This allows the design of solutions that can exploit both types simultaneously.

Table 5
Mapping between architecture components and projects.

Project	Ingestor	Knowledge Base	Curator	Learner	Retriever	Feeder
PlanetOnto	✓	✓		✓	✓	✓
Neptuno	✓	✓		✓	✓	
Annoterra	✓	✓	✓	✓	✓	
SemNews	✓	✓		✓	✓	
Hermes	✓	✓	✓	✓	✓	
BBC CMS	✓	✓	✓		✓	
NEWS	✓	✓	✓	✓	✓	✓
Even Registry	✓	✓	✓	✓	✓	
NewsReader	✓	✓	✓	✓	✓	
Reuters Tracer	✓	✓	✓	✓	✓	✓
SUMMA	✓	✓	✓		✓	✓
INJECT	✓	✓	✓		✓	
ASRAEL	✓	✓	✓		✓	

We designed the SRA to favour `Trustworthy-AI` through services that control `Privacy` and `Data-ownership`, provide `Provenance` and enhance explainability through the usage of semantic technologies and symbolic representation.

Comparison with existing JKPs. Finally, to validate that our proposed SRA is able to account for all the elements of the various JKPs reported in the literature (Section 4.1), we have reviewed them carefully and mapped their elements into the corresponding parts of our SRA, as shown in Table 5. Because many of the analysed projects built pipeline-based systems with little architectural description, the elements we mapped were sometimes suggested solutions and processing steps based on their goals and functionalities. We managed to map all the related JKPs into our SRA which indicates that our proposed SRA for JKPs is able to account for existing JKPs reported in the research literature.

7.2. Prototype

In order to evaluate the feasibility of our proposal, we developed a prototype platform that instantiates the SRA for JKPs (Fig. 4).

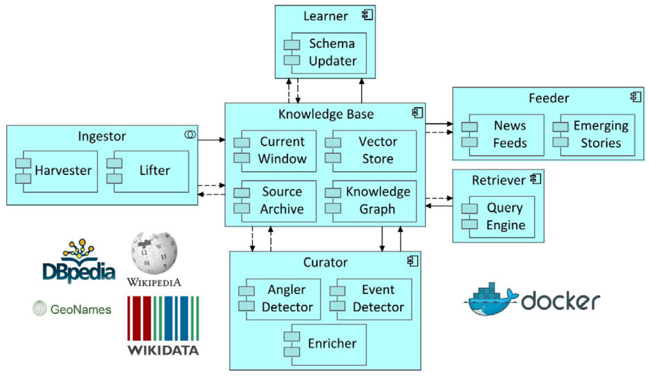

Fig. 4. The instantiated architecture for the JKP prototype.

Ingestor. We implemented the Ingestor with services for Harvesting and Lifting. Our Harvester crawls news-related websites and harvests RSS feeds, Twitter accounts, NewsAPI[12] and GDELT[13]. The Twitter API provides real-time tweet streams from specific accounts, geographical areas or topics. NewsAPI aggregates and provides streams of news articles from over 80000 news sources and blogs. GDELT provides semi-structured information about conflict events, collected from news all around the world and automatically translated into English from 65 different languages. We observed that RSS support is declining among news organisations. This makes aggregation services like NewsAPI and GDELT a solid alternative to consider, as they provide access to a larger number of news sources. We also observed that data from news organisations' and journalists' Twitter accounts cannot be used straightforwardly as many messages only provide links to news articles or little information. Hence, these messages need to be aggregated in chunks of information before they are processed and the information from the links must be downloaded. Harvesting news from different sources also creates duplicates that need to be filtered out. Many of them can be easily identified using URLs. However, it is not always trivial to filter similar news, because the same URL can provide updated content or a different URL can report the same news with a few newsworthy modifications in the content or even contradictions.

Our Lifter [12] transforms the news and event streams into semantic knowledge representations in real time according to an event-description ontology [77]. It combines out-of-the-box NLP systems such as DBpedia Spotlight[14] and SpaCy[15] and different end-to-end deep learning models for semantically annotating potentially news-relevant textual items with named entity linking, relation extraction, sentiment and topic annotations, and links to Wikidata and DBpedia. To integrate these annotations, we use the NLP Interchange Format (NIF) [78]. As these components have been designed as microservices with clear functional boundaries to facilitate `Interoperability` and `Modularity`, variants of the same components can be used to transform unstructured data from RSS feeds and structured data from GDELT to knowledge graphs. Furthermore, to scale the prototype in order to handle the large amount of data produced by GDELT, we replicated the GDELT Lifter components to avoid bottlenecks.

Knowledge base. The Knowledge Base includes a Source Archive service implemented with Apache Cassandra, a Knowledge Graph implemented with Blazegraph, a Current Windows built with Apache Kafka and ksqlDB as a stream store. We are also in the process of incorporating a vector store implemented with Vald. Cassandra is used to store the textual information together with the IRIs of the news-relevant items represented in the knowledge graph. This decision allows us to reduce the data stored in the knowledge graph; provide provenance by tracking news representations back to their source; and facilitate new training material for ML models based on the current state of our system. The Knowledge Graph is distributed over four instances of Blazegraph: one dedicated to storing news-relevant items from news articles and Twitter messages, and three to storing the events from GDELT. These four instances ingest around 11M ($11 \cdot 10^6$) triples daily from news and tweets, and another 11M ($11 \cdot 10^6$)

12 newsapi.org
13 www.gdeltproject.org
14 www.dbpedia-spotlight.org
15 spacy.io

from GDELT events. In a period of 6 months, it can ingest more than 4B ($6 \cdot 10^9$) triples in total. We have observed that these large amounts of triples cannot be held in a single triple-store instance without affecting its performance.

Curator. The Curator implements an Enricher, an Event detector and an Angle detector. Our Enricher extends the annotated items with location-related background information extracted from DBpedia, Wikidata and other LOD sources. Our Event Detector provides journalists with aggregated and real-time events detected from GDELT streams. The Angle Detector analyses the representations of the news items to identify location angles for a set of selected locations of interest [77]. As the Angler Detector analyses the news-relevant item representations from all sources, we had to replicate it to meet the velocity and volume demands. The Learner implements a Schema updater that monitors incoming GDELT events to identify new themes and update our themes hierarchy and mappings accordingly.

Feeder and Retriever. The Feeder provides an API that exposes a feed of annotated news-relevant items from the Knowledge Base and allows external users to interact with our system. To explore co-development with external contributors, we ran a research challenge[16] where external developers were invited to submit solutions using live feeds directly from the knowledge base [84]. In addition, we collaborated in another research challenge[17] where participants had access to news and images from our system to explore the connection between text and images. Our Retriever exposes several APIs to access the Knowledge Base and other services of the system. On top of the Retriever API, we developed an editing interface for journalists to recommend relevant information for the story the journalist is working on and provide background information for the entities present in the text. We have observed that extracting background information from Wikidata and DBpedia presents many challenges as entities from the same categories are not in general represented following the same structure and using the same properties.

Infrastructure. Our prototype runs on 28 cloud instances (with a total of 94 vCPU, 312 GB RAM and 20 TB disk)[18]. We used Ansible and Terraform to automatically set up the instances and Docker Swarm for orchestrating a total of 114 services as containerised applications (i.e., 70 services related to the JKP and the rest for monitoring these services and the cloud instances). These services run as containerised applications and are exposed through APIs that facilitate their replacement with newer versions without affecting the performance of the platform. We also decided to decouple the ML/DL models from the applications by exposing them through APIs, allowing us to switch the model at any time. By following these principles, we developed a system that allowed us to experiment with and meet the `Scalability` and `Velocity` requirements. Our prototype downloads news items and transforms them into graphs within an average of 21.112 seconds per news item, with a standard deviation of 9.906 seconds, including sleep and network waiting times. If we only take the system and user CPU time, our prototype takes an average of 0.246 seconds per item, with a standard deviation of 0.083 seconds. The text of the news items varies in length, with an average length of 3305.18 characters per item and a standard deviation of 3742.22. This reflects on the extracted graphs that have an average of 712.38 triples per graph, with a standard deviation of 550.355. To further evaluate these qualities, we plan to conduct stress-testing experiments with our prototype by, for example, processing all tweets produced by Twitter in a single day or re-processing in a single day the equivalent of the harvested news in a month.

To support communication between services, we serialised the messages using *JSON-LD*[19] and semantic vocabularies, as well as employed Apache Kafka as a message broker. JSON-LD is a popular extension of JSON for serialising linked data. Apache Kafka is a framework where independent services communicate through subscription to topics and production and consumption of messages with associated metadata (e.g., topic, key and timestamp). To add or duplicate a service, we only needed to assign it to the desired message stream. This also allowed us to duplicate services to meet specific workloads or add new ones effortlessly.

Earlier JKP prototypes, that ran on a simpler infrastructure without an equally carefully planned architecture [36], have already implemented additional functionalities, which we plan to adapt into the current prototype. In the development of the prototype, different people contributed to adapting the old components and creating new ones while the JKP was running. This was in part possible by meeting the modularity requirement.

7.3. Threats to validity

Completeness. Completeness deals with the selection of earlier projects that our work as based on. Our selection of JKP projects has been systematically carried out in the context of an extensive literature review of research on knowledge graphs for the news [26]. We have not included non-JKP projects in our work because JKPs exhibit a unique combination of characteristics that we have not seen in other domains, such as real time, web-scale data volumes, social media, text, multimedia, reference information from other sources and evolving concepts and stories. A limitation is that our work only covers the English-language literature, and we have not identified any relevant JKPs developed in geographical regions outside Europe, Canada and USA. We have also only covered the research literature. Although there are many commercial tools available for journalists and newsrooms, they tend to be focused on single tasks and not on the platform and architecture levels we address in this work.

Project access. Project access deals with the availability and reliability of information about earlier projects that our work is grounded in. We have selected primary accounts of JKP projects that are published in and available through reputed and peer-reviewed international journals and conferences. However, as we did not have access to the code of the related JKPs for re-implementing them following our SRA, nor access to their input and output data for comparing them with our proposed solution, we could not run more detailed comparative evaluations.

Internal validity. Internal validity deals with the clarity of the connections between evidence and conclusions. The qualitative validation is based on our own reading and interpretation of the requirements as presented in the primary studies. We traced each requirement (both functional and non-functional) backwards to its source, both in Section 5 and in the Appendix, and mapped each requirement forward to a specific design decision in Section 7.1. In addition to these mappings, the prototype is a direct instantiation of the SRA for JKPs, and we have shown that it adheres to all of the architecture principles outlined in Section 6.1. Although the authors of this work are involved in the development of the News Hunter platform, we have sought to reduce bias by limiting the contribution of News Hunter to supporting and extending the analysis of the literature and design of the SRA.

16 https://multimediaeval.github.io/editions/2021/tasks/emergingnews

17 https://multimediaeval.github.io/editions/2022/tasks/newsimages

18 The cloud instances run on different models of CPU (Intel Xeon CPU E5-2680 v3 @ 2.50 GHz, Intel Xeon CPU E5-2680 v4 @ 2.40 GHz, Intel Xeon Gold 6226 CPU @ 2.70 GHz, Intel Xeon Gold 5317 CPU @ 3.00 GHz, AMD EPYC 7452 @ 2.35 GHz) and memory speeds (2133 MT/s, 2400 MT/s, 2933 MT/s, 3200 MT/s) respectively.

19 www.w3.org/TR/json-ld11

External validity. External validity deals with the usefulness and validity of our results in other JKP contexts and other big-data and AI domains beyond journalism. To ensure usefulness and validity in a broad range of JKP contexts, we have taken all the relevant research projects we have found into account and we have collaborated with industrial users of more focussed journalistic tools. To facilitate usefulness and validity in other domains with similar required qualities, we have presented a high-level view of the SRA that can potentially be adapted to other application areas. However, our SRA has only been validated for JKPs, and it would need further validation to be used for other purposes. Language and region aspects should not pose a problem in terms of generalisation, but multimedia analysis remains an area for further research.

8. Conclusion

Grounded in the existing literature and supported by our practical experience, we have proposed an empirically-grounded SRA for JKPs. The purpose was to make it easier for news organisations to evolve their existing independent systems for news production towards integrated journalistic knowledge platforms and to direct further research. Although the SRA has been driven by the needs of journalism and news organisations, we have also presented a more high-level view of the SRA that can potentially serve as a proposal for a generic architecture for big-data and knowledge-based AI systems in other domains.

Our architectural decisions are based on reported experiences with existing platforms, supported by our own experience developing a JKP in collaboration with industry partners. The SRA is based on proven architecture concepts and is designed to be technology independent, open-ended and long-lasting, with components and services that can be replaced and integrated with other systems. It covers those components and functionalities that are essential for JKPs and introduces Learner and Curator components that are not considered in the previous literature. It provides a vocabulary to compare and understand different realisations of JKPs.

To demonstrate the feasibility of the proposed SRA, we have implemented a proof-of-concept prototype of JKP that instantiates it. We have developed the prototype iteratively and incrementally in order to continuously evaluate our SRA design. Validating the SRA in a newsroom production environment and assessing its real and perceived value for practitioners is left for further work.

In further work, we also want to explore the combination of knowledge graphs and vector databases and its implication for our architectural decisions. One possible benefit is improved explainability. We want to investigate how the results of machine learning techniques that employ vectors can be explained by analysing the knowledge representations related to those vectors. Moreover, we want to explore the benefits of expanding the Learner to learn from whole system, to learn not only from the AI models and knowledge representations but also from the usage and performance of each component. This will allow the Learner to adapt and personalise the components to the user's needs and the domain of the AI system.

CRediT authorship contribution statement

Marc Gallofré Ocaña: Conceptualization, Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Andreas L. Opdahl:** Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Appendix. Required qualities and their information sources

Table A.6
Functional required qualities and the projects that dealt with them.

	Annotating	Knowledge-representation	Enriching	Schema-evolution	Model-updating
PlanetOnto	✓	✓	✓	✓	✓
Neptuno	✓	✓	✓	✓	
Annoterra	✓	✓	✓	✓	
SemNews	✓	✓	✓	✓	
Hermes	✓	✓	✓	✓	
BBC CMS	✓	✓	✓		
NEWS	✓	✓	✓	✓	✓
Event Registry	✓	✓	✓		✓
NewsReader	✓	✓	✓	✓	
Reuters Tracer	✓	✓			✓
SUMMA	✓	✓	✓		
INJECT	✓	✓			
ASRAEL	✓	✓	✓		

	Storage	Push	Pull	Data-ownership	Privacy	Provenance
PlanetOnto	✓	✓	✓			
Neptuno	✓	✓	✓			
Annoterra	✓	✓	✓	✓		✓
SemNews	✓	✓	✓			
Hermes	✓		✓			
BBC CMS	✓		✓			
NEWS	✓	✓	✓	✓		✓
Event Registry	✓		✓			
NewsReader	✓		✓			✓
Reuters Tracer	✓	✓	✓			
SUMMA	✓	✓	✓			
INJECT	✓		✓			
ASRAEL	✓		✓			

Table A.7
Non-functional required qualities and the projects that dealt with them.

	Interoperability	Modularity	Scalability	Velocity
PlanetOnto	✓			
Neptuno	✓		✓	✓
Annoterra	✓	✓		
SemNews	✓			
Hermes				
BBC CMS	✓	✓	✓	
NEWS	✓	✓	✓	
Event Registry			✓	
NewsReader	✓	✓	✓	✓
Reuters Tracer			✓	✓
SUMMA	✓		✓	✓
INJECT				
ASRAEL	✓		✓	

	Variety	Knowledge-evolution	Sub-/symbolic-AI	Trustworthy-AI
PlanetOnto				
Neptuno	✓			
Annoterra	✓			
SemNews				
Hermes		✓		
BBC CMS	✓			✓
NEWS	✓	✓		
Event Registry		✓		
NewsReader	✓			
Reuters Tracer				
SUMMA	✓			
INJECT				
ASRAEL				

References

[1] C. Beckett, New Powers, New Responsibilities: a Global Survey of Journalism and Artificial Intelligence, Tech. Rep., Polis, London School of Economics and Political Science, 2019, URL https://blogs.lse.ac.uk/polis/2019/11/18/new-powers-new-responsibilities.

[2] J. Vázquez Herrero, S. Direito-Rebollal, A.S. Rodrí guez, X. García, Journalistic Metamorphosis: media Transformation in the Digital Age, Springer International publishing, 2020, http://dx.doi.org/10.1007/978-3-030-36315-4.

[3] U. Germann, R. Liepins, G. Barzdins, D. Gosko, S. Miranda, D. Nogueira, The SUMMA platform: A scalable infrastructure for multi-lingual multi-media monitoring, in: Proceedings of ACL 2018, System Demonstrations, 2018, http://dx.doi.org/10.18653/v1/P18-4017.

[4] S.C. Lewis, O. Westlund, Big data and journalism, Digit. Journal. 3 (3) (2015) http://dx.doi.org/10.1080/21670811.2014.976418.

[5] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A.M. Vollmer, S. Wagner, Software engineering for AI-based systems: A survey, ACM Trans. Softw. Eng. Methodol. 31 (2) (2022) http://dx.doi.org/10.1145/3487043.

[6] A. d'Avila Garcez, L.C. Lamb, Neurosymbolic AI: The 3rd wave, 2020, arXiv:2012.05876.

[7] Aidan Hogan, et al., Knowledge graphs, ACM Comput. Surv. 54 (4) (2021) http://dx.doi.org/10.1145/3447772.

[8] M. Gallofré Ocaña, A.L. Opdahl, Supporting newsrooms with journalistic knowledge graph platforms: Current state and future directions, Technologies 10 (3) (2022) http://dx.doi.org/10.3390/technologies10030068.

[9] Y. Raimond, T. Scott, S. Oliver, P. Sinclair, M. Smethurst, Use of semantic web technologies on the BBC web sites, in: Linking Enterprise Data, 2010, http://dx.doi.org/10.1007/978-1-4419-7665-9_13.

[10] C. Rudnik, T. Ehrhart, O. Ferret, D. Teyssou, R. Troncy, X. Tannier, Searching news articles using an event knowledge graph leveraged by wikidata, in: Companion Proceedings of the 2019 World Wide Web Conference, 2019, http://dx.doi.org/10.1145/3308560.3316761.

[11] C. Bizer, T. Heath, T. Berners-Lee, Linked data: The story so far, in: Semantic Services, Interoperability and Web Applications: Emerging Concepts, IGI global, 2011, pp. 205–227.

[12] T. Al-Moslmi, M. Gallofré Ocaña, Lifting news into a Journalistic Knowledge Platform, in: Proceedings of the CIKM 2020 Workshops, 2020, URL http://ceur-ws.org/Vol-2699/paper42.pdf.

[13] N. Fernández, D. Fuentes, L. Sánchez, J.A. Fisteus, The NEWS ontology: Design and applications, Expert Syst. Appl. 37 (12) (2010) http://dx.doi.org/10.1016/j.eswa.2010.06.055.

[14] T.A.A. Al-Moslmi, M. Gallofré Ocaña, A.L. Opdahl, B. Tessem, Detecting newsworthy events in a journalistic platform, in: The 3rd European Data and Computational Journalism Conference, 2019.

[15] G. Leban, B. Fortuna, J. Brank, M. Grobelnik, Event registry: Learning about world events from news, in: Proceedings of the 23rd International Conference on World Wide Web, 2014, http://dx.doi.org/10.1145/2567948.2577024.

[16] P. Vossen, R. Agerri, I. Aldabe, A. Cybulska, M. van Erp, A. Fokkens, E. Laparra, A.-L. Minard, A.P. Aprosio, G. Rigau, M. Rospocher, R. Segers, NewsReader: Using knowledge resources in a cross-lingual reading machine to generate more knowledge from massive streams of news, in: Special Issue Knowledge-Based Systems, Vol. 110, Elsevier, 2016, http://dx.doi.org/10.1016/j.knosys.2016.07.013.

[17] E. Motta, E. Daga, A.L. Opdahl, B. Tessem, Analysis and design of computational news angles, IEEE Access (2020).

[18] M. Gallofré Ocaña, T. Al-Moslmi, A.L. Opdahl, Data privacy in Journalistic Knowledge Platforms, in: Proceedings of the CIKM 2020 Workshops, 2020, URL http://ceur-ws.org/Vol-2699/paper44.pdf.

[19] S. Nadal, V. Herrero, O. Romero, A. Abelló, X. Franch, S. Vansummeren, D. Valerio, A software reference architecture for semantic-aware big data systems, Inf. Softw. Technol. 90 (2017) http://dx.doi.org/10.1016/j.infsof.2017.06.001.

[20] B. Sena, A.P. Allian, E.Y. Nakagawa, Characterizing big data software architectures: A systematic mapping study, in: Proceedings of the 11th Brazilian Symposium on Software Components, Architectures, and Reuse, SBCARS '17, Association for Computing Machinery, New York, NY, USA, 2017, http://dx.doi.org/10.1145/3132498.3132510.

[21] B. Sena, L. Garcés, A.P. Allian, E.Y. Nakagawa, Investigating the applicability of architectural patterns in big data systems, in: Proceedings of the 25th Conference on Pattern Languages of Programs, PLoP '18, The Hillside Group, USA, 2018.

[22] C. Avci, B. Tekinerdogan, I.N. Athanasiadis, Software architectures for big data: A systematic literature review, Big Data Anal. 5 (1) (2020) 1–53.

[23] P. Ataei, A.T. Litchfield, Big data reference architectures, a systematic literature review, in: ACIS 2020 Proceedings, (30) 2020, URL https://aisel.aisnet.org/acis2020/30.

[24] T.V.R. da Costa, E. Cavalcante, T. Batista, Big data software architectures: An updated review, in: O. Gervasi, B. Murgante, E.M.T. Hendrix, D. Taniar, B.O. Apduhan (Eds.), Computational Science and Its Applications – ICCSA 2022, Springer International Publishing, Cham, 2022, pp. 477–493.

[25] M. Gallofré Ocaña, A.L. Opdahl, Developing a software reference architecture for journalistic knowledge platforms, in: ECSA2021 Companion Volume, 2021.

[26] A.L. Opdahl, T. Al-Moslmi, D.-T. Dang-Nguyen, M. Gallofré Ocaña, B. Tessem, C. Veres, Semantic knowledge graphs for the news: A review, ACM Comput. Surv. (2022) http://dx.doi.org/10.1145/3543508.

[27] S. Angelov, P. Grefen, D. Greefhorst, A framework for analysis and design of software reference architectures, Inf. Softw. Technol. 54 (4) (2012) http://dx.doi.org/10.1016/j.infsof.2011.11.009.

[28] S. Angelov, J.J. Trienekens, P. Grefen, Towards a method for the evaluation of reference architectures: Experiences from a case, in: Software Architecture. ECSA 2008, 2008, http://dx.doi.org/10.1007/978-3-540-88030-1_17.

[29] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, 10.48550/ARXIV.1301.3781.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), in: Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017, URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[31] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186, http://dx.doi.org/10.18653/v1/N19-1423, (Long and Short Papers).

[32] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Adv. Neural Inf. Process. Syst. 33 (2020) 1877–1901.

[33] Y.A. Malkov, D.A. Yashunin, Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, IEEE Trans. Pattern Anal. Mach. Intell. 42 (4) (2020) 824–836, http://dx.doi.org/10.1109/TPAMI.2018.2889473.

[34] J. Johnson, M. Douze, H. Jégou, Billion-scale similarity search with GPUs, IEEE Trans. Big Data 7 (3) (2019) 535–547.

[35] M. Galster, P. Avgeriou, Empirically-grounded reference architectures: A proposal, in: Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS, Association for Computing Machinery, 2011, http://dx.doi.org/10.1145/2000259.2000285.

[36] A. Berven, O.A. Christensen, S. Moldeklev, A.L. Opdahl, K.J. Villanger, A knowledge-graph platform for newsrooms, Comput. Ind. 123 (2020) http://dx.doi.org/10.1016/j.compind.2020.103321.

[37] M. Gallofré Ocaña, L. Nyre, A.L. Opdahl, B. Tessem, C. Trattner, C. Veres, Towards a big data platform for news angles, in: 4th Norwegian Big Data Symposium, NOBIDS 2018, 2018, URL http://ceur-ws.org/Vol-2316/paper1.pdf.

[38] A. Tverberg, I. Agasøster, M. Grønbæck, R.S. Marius Monsen, K. Eikeland, E. Trondsen, L. Westvang, T.B. Knudsen, E. Fiskerud, R. Skår, S. Stoppel, A. Berven, G.S. Pedersen, P. Macklin, K. Cuomo, L. Vredenberg, K. Tolonen, A.L. Opdahl, B. Tessem, C. Veres, D.-T. Dang-Nguyen, E. Motta, V.J. Setty, WP3 2021 M3.1 Report the Industrial Expectations to, Needs from and Wishes for the Work Package, Tech. Rep., University of Bergen, MediaFutures, 2021.

[39] H.A. Simon, The Sciences of the Artificial, MIT Press, 1996.

[40] A. Hevner, S. Chatterjee, Design science research in information systems, 2010, http://dx.doi.org/10.1007/978-1-4419-5653-8_2.

[41] A.R. Hevner, A three cycle view of design science research, Scand. J. Inf. Syst. 19 (2) (2007) URL https://aisel.aisnet.org/sjis/vol19/iss2/4.

[42] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS Q. 28 (1) (2004) URL http://www.jstor.org/stable/25148625.

[43] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Sci. Am. 284 (5) (2001) 34–43, URL http://www.jstor.org/stable/26059207.

[44] N. Shadbolt, T. Berners-Lee, W. Hall, The semantic web revisited, IEEE Intell. Syst. 21 (3) (2006) 96–101, http://dx.doi.org/10.1109/MIS.2006.62.

[45] J. Domingue, E. Motta, PlanetOnto: From news publishing to integrated knowledge management support, IEEE Intell. Syst. Appl. 15 (3) (2000) 26–32, http://dx.doi.org/10.1109/5254.846282.

[46] Y. Kalfoglou, J. Domingue, E. Motta, M. Vargas-Vera, S. Buckingham Shum, Myplanet: An ontology driven web based personalised news service, in: Proceedings of International Joint Conference on Artificial Intelligence, Vol. 2001, International Joint Conferences on Artificial Intelligence, 2001, pp. 44–52, URL http://ceur-ws.org/Vol-47/kalfoglou.pdf.

[47] P. Castells, F. Perdrix, E. Pulido, M. Rico, R. Benjamins, J. Contreras, J. Lorés, Neptuno: Semantic web technologies for a digital newspaper archive, in: The Semantic Web: Research and Applications. ESWS 2004, 2004, http://dx.doi.org/10.1007/978-3-540-25956-5_31.
[48] D.B. Ramagem, B. Margerin, J. Kendall, AnnoTerra: Building an integrated earth science resource using semantic web technologies, IEEE Intell. Syst. 19 (3) (2004) http://dx.doi.org/10.1109/MIS.2004.3.
[49] A. Java, T. Finin, S. Nirenburg, SemNews: A semantic news framework, in: The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, 2006, URL https://www.aaai.org/Papers/AAAI/2006/AAAI06-316.pdf.
[50] K. Schouten, P. Ruijgrok, J. Borsje, F. Frasincar, L. Levering, F. Hogenboom, A semantic web-based approach for personalizing news, in: Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10, ACM Press, Sierre, Switzerland, 2010, p. 854, http://dx.doi.org/10.1145/1774088.1774264.
[51] G. Kobilarov, T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, R. Lee, Media meets semantic web – how the BBC uses DBpedia and linked data to make connections, in: The Semantic Web: Research and Applications, Vol. 5554, 2009, http://dx.doi.org/10.1007/978-3-642-02121-3_53.
[52] N. Fernández, J.M. Blázquez, J.A. Fisteus, L. Sánchez, M. Sintek, A. Bernardi, M. Fuentes, A. Marrara, Z. Ben-Asher, NEWS: Bringing semantic web technologies into news agencies, in: The Semantic Web - ISWC 2006, 2006, pp. 778–791, http://dx.doi.org/10.1007/11926078_56.
[53] M. Rospocher, M. van Erp, P. Vossen, A. Fokkens, I. Aldabe, G. Rigau, A. Soroa, T. Ploeger, T. Bogaard, Building event-centric knowledge graphs from news, J. Web Semant. 37–38 (2016) 132–151, http://dx.doi.org/10.1016/j.websem.2015.12.004.
[54] Q. Li, S. Shah, X. Liu, A. Nourbakhsh, R. Fang, TweetSift: Tweet topic classification based on entity knowledge base and topic enhanced word embedding, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 2429–2432, http://dx.doi.org/10.1145/2983323.2983325.
[55] X. Liu, Q. Li, A. Nourbakhsh, R. Fang, M. Thomas, K. Anderson, R. Kociuba, M. Vedder, S. Pomerville, R. Wudali, R. Martin, J. Duprey, A. Vachher, W. Keenan, S. Shah, Reuters tracer: A large scale system of detecting & verifying real-time news events from Twitter, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 207–216, http://dx.doi.org/10.1145/2983323.2983363.
[56] X. Liu, A. Nourbakhsh, Q. Li, S. Shah, R. Martin, J. Duprey, Reuters tracer: Toward automated news production using large scale social media data, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 1483–1493.
[57] S. Miranda, D. Nogueira, A. Mendes, A. Vlachos, A. Secker, R. Garrett, J. Mitchel, Z. Marinho, Automated fact checking in the news room, in: The World Wide Web Conference, WWW '19, Association for Computing Machinery, 2019, pp. 3579—3583, http://dx.doi.org/10.1145/3308558.3314135.
[58] N. Maiden, K. Zachos, A. Brown, G. Brock, L. Nyre, A. Nygård Tonheim, D. Apsotolou, J. Evans, Making the news: Digital creativity support for journalists, in: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1–11, http://dx.doi.org/10.1145/3173574.3174049.
[59] F. Frasincar, J. Borsje, L. Levering, A semantic web-based approach for building personalized news services, Int. J. E-Business Res. (IJEBR) 5 (3) (2009) 35–53.
[60] D. Le-Phuoc, H.Q. Nguyen-Mau, J.X. Parreira, M. Hauswirth, A middleware framework for scalable management of linked streams, J. Web Semant. 16 (2012) http://dx.doi.org/10.1016/j.websem.2012.06.003, The Semantic Web Challenge 2011.
[61] M.A. Martínez-Prieto, C.E. Cuesta, M. Arias, J.D. Fernández, The SOLID architecture for real-time management of big semantic data, Future Gener. Comput. Syst. 47 (2015) http://dx.doi.org/10.1016/j.future.2014.10.016, Special Section: Advanced Architectures for the Future Generation of Software-Intensive Systems.
[62] N. Marz, How to beat the CAP theorem, 2011, URL http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html.
[63] P. P, Reference architecture and classification of technologies, products and services for big data systems, Big Data Res. 2 (4) (2015) http://dx.doi.org/10.1016/j.bdr.2015.01.001.
[64] D. Xu, D. Wu, X. Xu, L. Zhu, L. Bass, Making real time data analytics available as a service, in: Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures, QoSA '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 73–82, http://dx.doi.org/10.1145/2737182.2737186.
[65] G.M. Sang, L. Xu, P. de Vrieze, A reference architecture for big data systems, in: 2016 10th International Conference on Software, Knowledge, Information Management & Applications, SKIMA, 2016, pp. 370–375, http://dx.doi.org/10.1109/SKIMA.2016.7916249.
[66] L. Heilig, S. Voß, Managing cloud-based big data platforms: A reference architecture and cost perspective, in: Big Data Management, Springer International Publishing, Cham, 2017, pp. 29–45, http://dx.doi.org/10.1007/978-3-319-45498-6_2.
[67] S. Martínez-Fernández, C.P. Ayala, X. Franch, H.M. Marques, Benefits and drawbacks of software reference architectures: A case study, Inf. Softw. Technol. 88 (2017) http://dx.doi.org/10.1016/j.infsof.2017.03.011.
[68] M.K. Sarker, L. Zhou, A. Eberhart, P. Hitzler, Neuro-symbolic artificial intelligence: Current trends, 2021, arXiv preprint arXiv:2105.05330.
[69] High-Level Expert Group on Artificial Intelligence, Ethics Guidelines for Trustworthy AI, Tech. Rep., European Commission, 2019, URL https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai.
[70] N. Dragoni, S. Giallorenzo, A.L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, L. Safina, Microservices: Yesterday, today, and tomorrow, in: Present and Ulterior Software Engineering, Springer International publishing, 2017, http://dx.doi.org/10.1007/978-3-319-67425-4_12.
[71] R.C. Fernandez, P.R. Pietzuch, J. Kreps, N. Narkhede, J. Rao, J. Koshy, D. Lin, C. Riccomini, G. Wang, Liquid: Unifying nearline and offline big data integration, in: Proceedings of the 7th Biennial Conference on Innovative Data Systems Research, CIDR, 2015.
[72] J. Kreps, Questioning the lambda architecture, 2014, URL https://www.oreilly.com/radar/questioning-the-lambda-architecture.
[73] F. Cerezo, C.E. Cuesta, J.C. Moreno-Herranz, B. Vela, Deconstructing the Lambda architecture: An experience report, in: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), 2019, http://dx.doi.org/10.1109/ICSA-C.2019.00042.
[74] H.P. Nii, The blackboard model of problem solving and the evolution of blackboard architectures, AI Mag. 7 (2) (1986) 38.
[75] I.D. Craig, Blackboard systems, Artif. Intell. Rev. 2 (2) (1988) 103–118.
[76] F. Lecue, On the role of knowledge graphs in explainable AI, Semantic Web 11 (1) (2020) 41–51.
[77] A.L. Opdahl, B. Tessem, Ontologies for finding journalistic angles, Softw. Syst. Model. (2020) http://dx.doi.org/10.1007/s10270-020-00801-w.
[78] S. Hellmann, J. Lehmann, S. Auer, M. Brümmer, Integrating NLP using linked data, in: The Semantic Web – ISWC 2013, 2013, http://dx.doi.org/10.1007/978-3-642-41338-4_7.
[79] A. Fokkens, A. Soroa, Z. Beloki, N. Ockeloen, G. Rigau, W.R. Van Hage, P. Vossen, NAF and GAF: Linking linguistic annotations, in: Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation, 2014.
[80] H. Dibowski, S. Schmid, Using knowledge graphs to manage a data lake, in: INFORMATIK 2020, Gesellschaft für Informatik, Bonn, 2021, pp. 41–50, http://dx.doi.org/10.18420/inf2020_02.
[81] W3C, Largetriplestores, 2020, URL https://www.w3.org/wiki/LargeTripleStores.
[82] B. Tessem, M. Gallofré Ocaña, A.L. Opdahl, Construction of a relevance knowledge graph with application to the LOCAL news angle, in: Nordic Artificial Intelligence Research and Development, 2023.
[83] B. Tessem, Analogical news angles from text similarity, in: Artificial Intelligence XXXVI, 2019, http://dx.doi.org/10.1007/978-3-030-34885-4_35.
[84] M. Gallofré Ocaña, A.L. Opdahl, D.-T. Dang-Nguyen, Emerging News task: Detecting emerging events from social media and news feeds, MediaEval, 2021.