IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 January 2024, accepted 21 February 2024, date of publication 23 February 2024, date of current version 1 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3369409

RESEARCH ARTICLE

A Blackboard Model for Flexible and Parallel Text Annotation

MARC GALLOFRÉ OCAÑA AND ANDREAS L. OPDAHL
Department of Information Science and Media Studies, University of Bergen, Bergen, 5004 Vestland, Norway
Corresponding author: Marc Gallofré Ocaña (marc.gallofre@uib.no)

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

ABSTRACT Creating rich semantic text annotations is a complex process that involves combining multiple natural-language annotation approaches. This annotation process is often approached sequentially and includes pre-processing steps and techniques that build on the outputs of others. However, combining them is not trivial, because some annotation approaches comprise chains of steps or build on other already pre-existing annotations, some pre-processing steps may be common to several techniques, and many newer techniques are even end-to-end which have diluted the need for specific pre-processing steps. Yet it can be beneficial to combine the different approaches because they solve different annotation problems and, even when they solve the same problem, they may have complementary strengths. Whereas existing works often approach the annotation process sequentially, we argue that it can instead be implemented as a partly sequential, partly parallel and concurrent collaboration between independent components. The Blackboard Model is a long-established problem-solving paradigm that deals with complex problems where multiple knowledge sources contribute independently towards the solution. In this work, we study the feasibility of the Blackboard Model for creating rich semantic annotations from text as part of a larger big-data-ready AI system for supporting journalists and newsrooms.

INDEX TERMS Blackboard model, knowledge graph, big data, deep learning, semantic technologies, natural language processing, information extraction.

I. INTRODUCTION

Annotating natural-language texts semantically is a complex information extraction process, which is often approached as a sequential process that involves multiple natural-language processing (NLP) steps [1]. We call this an *annotation pipeline*. Pipelines are often used in research and industrial projects that aim to extract information contained in documents, e.g., news, web pages, reports, court records, and medical diagnoses. The purpose is to create more structured representations of the information that can later be exploited for different purposes such as, but not limited to, building data lakes and knowledge graphs, conducting data and network analysis, and supporting information exchange, validation, retrieval, augmentation, and inference [1]. An annotation pipeline can involve different tasks as shown in Figure 1. It typically starts with tasks for cleaning and preparing the data, like tokenisation and part-of-speech tagging, which may be common to several further tasks. The extracted information may not be required in the following tasks that solve different annotation problems but some tasks may build on it. Yet, it is usually propagated through all tasks, including the outputs of the subsequent tasks. This can create dependencies between the tasks that can propagate errors and introduce bottlenecks and data duplication.

Unfortunately, annotation pipelines lack flexibility and are not easy to modify. Adding or removing components is challenging due to the dependencies between tasks. If one would deploy an annotation pipeline as depicted in Figure 1, this would rather contain dependencies between the steps similar to the ones in Figure 2. For example, as shown in Figure 2, named-entity recognition (NER) and named-entity linking (NEL) may depend on the results of coreference resolution models to improve the results [3] and the resolution of dark entities may need to compare the results of both NER and NEL to identify the missed

The associate editor coordinating the review of this manuscript and approving it for publication was Chang Choi.

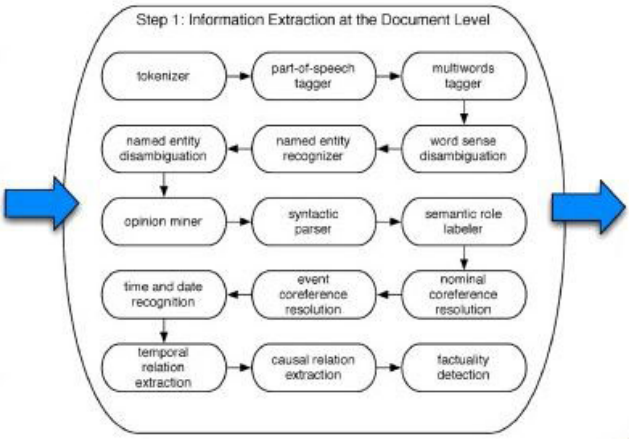

FIGURE 1. Sequential annotation process for NLP (excerpt from [2]).

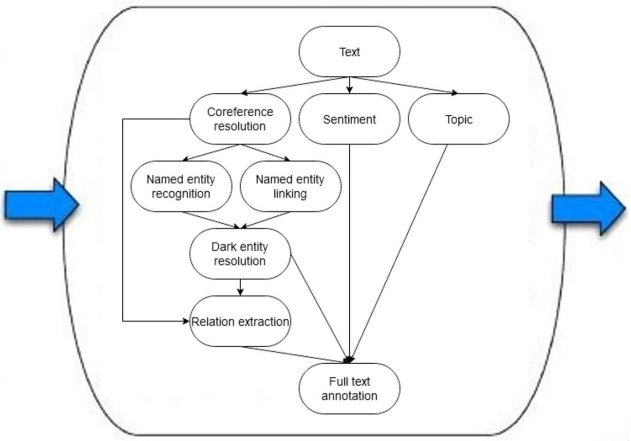

FIGURE 2. Example of dependencies in the annotation process for our use case.

links [4]. Similarly, extracting the relations between the entities in the text may benefit from resolving coreferences and entities. However, it is difficult to adapt to new data and annotation types as the annotation pipeline may need to be modified substantially to accommodate these changes. In addition, different components may require or benefit from different types of specialised hardware like TPUs and GPUs and specific RAM sizes or access to external services and APIs. As the components of the annotation pipeline are not fully decoupled, it can be cumbersome to assign suitable computing resources, parallelise and scale them. Recently, end-to-end NLP, foundation models and large language models (LLMs) [5], [6], [7] have achieved state-of-the-art results on many annotation tasks without depending on pre-processing steps or other models. But there will still be a need to combine LLM annotations with annotations provided by simpler models and by other LLMs to fully annotate texts. We expect that future systems will need to interact with more than one LLM.

Hence, a more flexible alternative to the annotation pipeline is needed. However, this is not a trivial problem, because the annotation tasks may involve steps that solve the same annotation problem in different and perhaps complementary ways, and other tasks may build on the outputs of the previous ones. Yet, it can be beneficial to combine them because they solve different annotation problems, and even when they solve the same problems, they may have complementary strengths. This means that the annotation process is not fully sequential, but it must support sequentially dependent and independent combinations, where some steps rely on the outputs of others and some on the same input to produce the same output. Therefore, a solution is needed that is flexible enough to integrate new techniques, such as the end-to-end deep learning models, transformers [8], foundation models and LLMs [5], as they become available to solve existing and new annotations problems.

We show that the Blackboard Model [9] can be used as an alternative to the current annotation pipeline approach. This tried-and-tested model is a problem-solving paradigm that deals with complex problems [10]. It is formalised through a blackboard where a problem is presented, and knowledge sources which are expert agents capable of solving parts of the problem each. The knowledge sources act independently of each other in a parallel way. When the different parts of the problem are solved, they are merged to create the solution. In our case, a knowledge source is a software component that performs text annotation tasks, whether pre-processing, like tokenisation, or semantic annotation tasks, like named entity linking. The blackboard is a temporary storage of more or less annotated versions of each new text, which are finally merged to become fully annotated. Therefore, the annotations must be represented in a format that can be easily unified in order to be merged. This approach provides a flexible, extensible, maximally parallel, and thus scalable solution for NLP annotation, reducing the dependencies and data duplication, and facilitating integration. Therefore, in this work, we ask: ''How can the Blackboard Model be adapted for rich semantic annotation of text?'' For the rest of this paper, we refer to the Blackboard Model as the annotation blackboard to distinguish it from the annotation pipeline.

We explore the following use case: annotating a real-time stream of news to create knowledge graphs that represent the news stories. However, the present paper only discusses the first part of this use case: the rich semantic annotation of text. Our use case is tackled in the context of a larger project and big-data-ready AI system [11], [12], [13] that explores semantic technologies and knowledge graphs to provide journalists with newsworthy information from social media and news sources [14], [15]. We built the *annotation blackboard* on top of big-data-ready technologies and used semantic vocabularies and technologies to annotate and represent news. These technologies have proven to provide scalability, distribution and flexibility to Big Data by design. Some other works have already explored the usage of big-data technologies to parallelise the annotation pipeline with promising results [16]. In comparison, our approach differs because it utilises an alternative paradigm to the annotation pipeline that can easily accommodate

new annotation techniques and reduce dependencies between components.

The article is organised as follows: Section II extends the definition of the Blackboard Model. Section III details our implementation of the Blackboard Model for NLP annotation. Section IV presents the conducted experiment and its results. Section V discusses our contribution, and Section VI states our conclusions and suggests further work.

II. BACKGROUND

A. BLACKBOARD MODEL

The Blackboard Model was introduced in the 1970s as a problem-solving paradigm for complex problems [10]. It offers a parallel and concurrent approach as an alternative to sequential problem-solving models. The Blackboard Model [9] consists of two types of components: the blackboard and the knowledge sources. The blackboard is a data structure, such as a database or common repository, where the problem is presented to the knowledge sources. The knowledge sources are agents that have the required knowledge or access to it and are specialised in solving one part of the problem. Each knowledge source contributes to solving the problem by providing a partial solution. The different contributions can be sequentially dependent on one another or completely independent. The partial solutions are shared on the blackboard so that other knowledge sources can use them. When all knowledge sources have contributed, the partial solutions are combined to form the final solution.

This approach reduces bottlenecks as the whole process can be parallelised easily (but with sequential output-input dependencies accounted for), instead of a sequential process where each task waits for the previous one. It also reduces data duplication as data and contributions are shared in the blackboard and do not need to be propagated from one task to the other. As the knowledge sources are independent, they can be easily scaled, replicated and replaced. However, this approach requires a coordination model to activate the knowledge sources [17].

III. NLP ANNOTATION AS A BLACKBOARD APPROACH

The information extraction process in an annotation pipeline consists of various steps that range from specific filtering and pre-processing to annotation tasks. Pre-processing tasks typically include downcasing, lemmatisation, removal of stopwords and/or punctuation, vectorisation, etc. Common annotation tasks are sentiment analysis, topic labelling, named entity recognition and linking, relation extraction, etc. Each of these tasks can be implemented as a software component that has a clear purpose and functional boundary. Several interrelated tasks can also be implemented in the same software component. In either case, we can consider these components as knowledge sources.

In an annotation pipeline, the information is often forwarded as input to the next task that only uses the information it needs to expand the annotations and add them to the output. Instead, in our annotation blackboard, this information is stored in a common repository, from which each component retrieves the inputs it needs and to which it generates its outputs. Hence, we can instantiate this common repository as a blackboard. To fully move from an annotation pipeline to an annotation blackboard, we must not only instantiate the software components and common repository as knowledge sources and blackboard but also follow a new processing and coordination model.

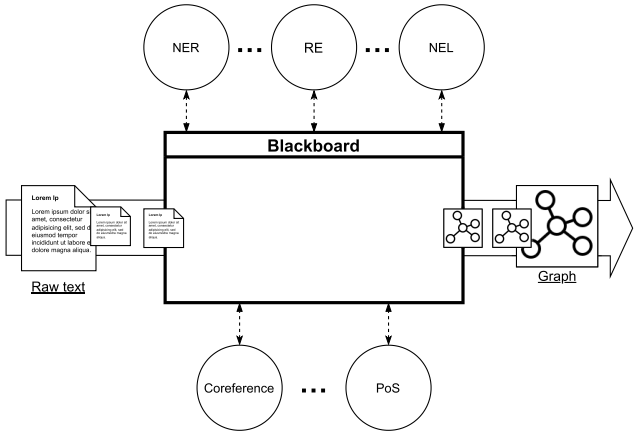

FIGURE 3. Annotation blackboard representation.

In our use case, a stream of news (raw text) is semantically annotated with the concepts and relations in the text [3], [18], Figure 3 illustrates the instantiation of the annotation blackboard. In it, the different annotation tasks such as named-entity recognition (NER), relation extraction (RE), named-entity linking (NEL), coreference resolution, part-of-speech tagging (PoS) and others are types of knowledge sources. Each annotation task can be realised by one or many knowledge sources, these can be any type of model from rule-based and machine-learning models to foundation models fine-tuned to specific tasks or calls to external API services. In our system, we used small transformer-based and other models for their simplicity, availability and low resource consumption as they can run in single CPU instances (i.e., an implementation of RoBERTa [19] from SpaCy called en_core_web_trf for NER, PoS and coreference, a fine-tuned BERT model from the OpenNRE [20] for REL, a fine-tuned BERT model for sentiment classification, a fine-tuned DeBERTa model for natural language inference for topic classification [21], and the DBpedia Spotlight model for NEL [22]). Using LLMs, especially those that are fine-tuned for instruction-following tasks, may require implementing extra steps in the blackboard prototype to guarantee and validate that the outputs are always consistent with the desired format.

In the model illustrated in Figure 3 the outputs of pre-processing tasks like coreference resolution and PoS tagging do not need to be used by all of the other knowledge sources as some of them may be implemented as end-to-end models. Instead, only those knowledge sources that need these outputs will access them.

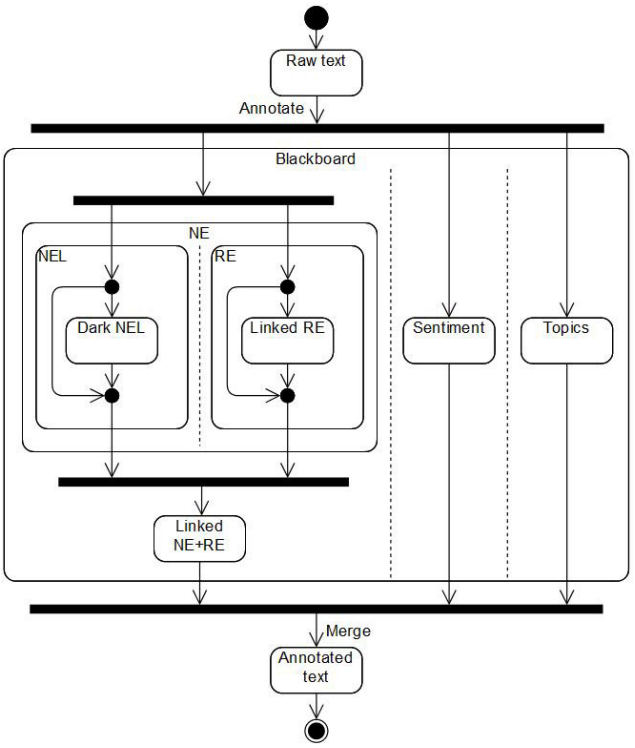

FIGURE 4. UML state machine diagram[1] of the annotation blackboard.

Figure 4 shows a state diagram of a part of the annotation blackboard process. It illustrates how each raw input text is sent to the blackboard. Inside the blackboard, the text can be annotated by several independent knowledge sources. Annotation with named entities and relations can, in turn, involve new independent and concurrent transformations as well as sub-states. As soon as named entities have been annotated, the entities can also be linked. Identifying unlinkable (dark) entities [4] is a possible transformation. In parallel with NER and NEL, relations between the named entities can be identified and possibly linked. Finally, the different annotations are merged to produce an annotated output text. Figures 3 and 4 only show a small number of annotation techniques. They do not cover all possible pre-processing steps, nor do they account for other existing or future annotation techniques such as stance detection, event detection, frame analysis, etc.

Different annotation components can support single transformations or groups of related transformations. For example, one component can perform only NER on raw texts, another can perform NEL on texts with NE annotations, and a third component can perform both NER and NEL on raw texts. Similarly, one component can perform RE on texts with named entity annotations, and another can perform combined NER, RE and RE linking on raw texts. All these different possibilities for combining annotations, of which some are proper extensions of others, can be accommodated by a hierarchy of annotations that range from fine-grained to

[1] http://www.omg.org/spec/UML/2.5/

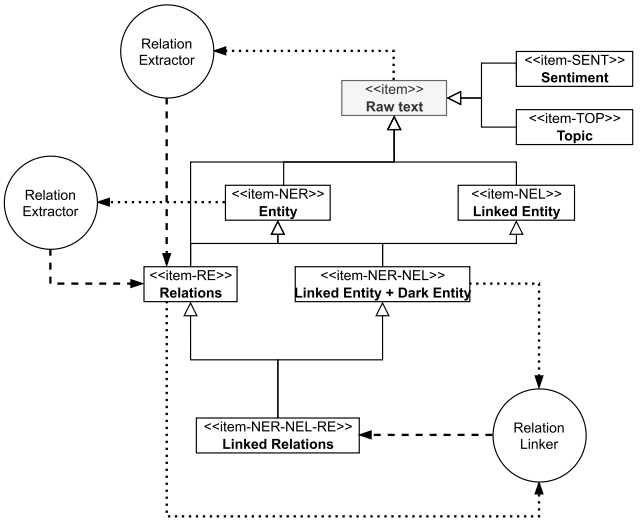

FIGURE 5. Hierarchy of annotations.

more complex annotations as illustrated in Figure 5. This figure also shows the possibilities that our adaptation of the Blackboard Model provides for using the information in the hierarchy. For example, tasks like relation extraction can be implemented by two different knowledge sources, have as input different levels of annotations, and the same knowledge source can have more than one input from different annotations.

This model facilitates extensibility because, at any time, we can add more types of knowledge sources without interfering with the others. As the information is always shared through the blackboard, new knowledge sources of the same type can just read from and write to the same parts of the blackboard independently. This is crucial, as new annotation components continue to become available. If a new type of knowledge source needs a new type of input or generates a new type of output, this new information can be added and used as part of the hierarchy of annotations, without affecting the other knowledge sources. This also applies to pre-processing tasks.

Our adaptation of the Blackboard Model implies that the blackboard must differentiate and be able to represent the different types in the annotation hierarchy, the representation format must always allow the addition of types of annotations without interfering with the current ones, and the different annotations must be mergeable. Therefore, the blackboard must be implemented using technologies that facilitate information access, merging and differentiation, and the knowledge sources must employ flexible formats for representing and structuring information. We decided to use semantic technologies and vocabularies to represent annotated texts and Apache Kafka[2] and ksqlDB[3] to implement the blackboard.

[2] kafka.apache.org
[3] ksqldb.io

TABLE 1. Vocabularies used in this work.

Prefix	Vocabulary IRI
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
owl	http://www.w3.org/2002/07/owl#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#
itsrdf	http://www.w3.org/2005/11/its/rdf#
nif	http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#

```
<http://example.org/doc1>
  a nif:Context ,
  nif:OffsetBasedString ;
  nif:beginIndex "0"^^xsd:nonNegativeInteger
      ;
  nif:endIndex "83"^^xsd:nonNegativeInteger
      ;
  nif:isString "Pau Casals, the legendary
      Catalan cellist, received the United
      Nations Peace Medal." .
```

LISTING 1. Raw text represented in NIF.

Semantic technologies and vocabularies provide language-agnostic and machine-understandable representations that allow us to homogenise the inputs and outputs of the knowledge sources. Hence, by using the same representations, any new knowledge source can easily be adapted to use and share the information on the blackboard. There are different semantic vocabularies for representing NLP annotations [23] such as NAF [24] and NIF [25]. Among them, we chose NIF because it provides a vocabulary and an ontology that let us identify each annotation uniquely; annotate spans, sentences and annotations with quality measures like confidence and precision; and easily introduce new types of annotations. By being able to represent and differentiate annotations at any specific level, we can also merge annotations that are included in or overlap with other annotations. In addition, NIF is a widely used vocabulary on the semantic web. Table 1 lists the vocabularies we use to represent the annotations in this work.

Apache Kafka is a highly scalable event streaming platform that follows a publish (write)/subscribe (read) protocol for communication and data storage. In Apache Kafka, the information is constantly published into independent streams of topics that we can use to represent the different types in the annotation hierarchy (i.e., each type is represented as a topic).[4] These topics, in turn, are distributed and can be subscribed to by multiple knowledge sources simultaneously and their outputs can be published to other topics. Each message (or text) in a topic has a unique key, which can be used to identify versions of the same original text that have been differently annotated and thus stored in different topics.

KsqlDB is a database for stream processing built on top of Apache Kafka that provides a query language similar to SQL and facilitates merge-on-read of topics. To merge annotations

[4]An example of the Kafka topics can be seen in Figure 5 where the name of the topic is placed between << >> above the name of each type of annotation.

```
<http://example.org/doc1#offset_0_9>
  a nif:Phrase, nif:OffsetBasedString;
  nif:beginIndex "0"^^xsd:nonNegativeInteger
      ;
  nif:endIndex "10"^^xsd:nonNegativeInteger
      ;
  nif:anchorOf "Pau Casals" ;
  itsrdf:taClassRef <http://dbpedia.org/
      ontology/Person> ;
  itsrdf:taConfidence 0.97 ;
  itsrdf:taIdentRef <http://dbpedia.org/
      resource/Pau_Casals> .

<http://example.org/doc1#offset_57_83>
  a nif:Phrase, nif:OffsetBasedString;
  nif:beginIndex "57"^^xsd:
      nonNegativeInteger ;
  nif:endIndex "83"^^xsd:nonNegativeInteger
      ;
  nif:anchorOf "United Nations Peace Medal"
      ;
  nif:taClassRef <http://dbpedia.org/
      ontology/Award> ;
  itsrdf:taConfidence 0.92 ;
  itsrdf:taIdentRef <http://dbpedia.org/
      resource/United_Nations_Peace_Medal> .
```

LISTING 2. Named entity linking annotations represented in NIF.

```
<http://example.org/doc1#offset_0_83>
  a nif:Relation, nif:OffsetBasedString ;
  nif:beginIndex "0"^^xsd:nonNegativeInteger
      ;
  nif:endIndex "83"^^xsd:nonNegativeInteger
      ;
  nif:head <http://example.org/doc1#
      offset_0_9> ;
  nif:tail <http://example.org/doc1#
      offset_57_83> ;
  nif:taRelationRef <https://dbpedia.org/
      ontology/award>;
  itsrdf:taConfidence 0.9  .
```

LISTING 3. Relation extraction annotations represented in NIF. We used a extended version of NIF as explained in Section IV-A.

from different knowledge sources, either as an input to a knowledge source or to generate the final output, we run a join query using the same key on the different topics with ksqlDB. The current implantation uses a simple time windowing to collect the results from the knowledge sources. By combining Apache Kafka and ksqlDB, we can differentiate and represent the different types of the annotation hierarchy and add new types of annotations (topics) without affecting the current ones. This flexibility is achieved in part by using NIF, as it makes the outputs of the knowledge source comparable, reusable and mergeable.

To further illustrate the process, we can take as an example the following sentence as input:

Pau Casals, the legendary Catalan cellist, received the United Nations Peace Medal.

This sentence will be represented in NIF as shown in Listing 1 and published in the topic `item` with a unique key. The current implementation uses an MD5 hash of

```
CREATE STREAM "item-NER-NEL"
WITH (
    KAFKA_TOPIC = 'item-NER-NEL',
    PARTITIONS = 10,
    REPLICAS = 3,
    VALUE_FORMAT='JSON',
    KEY_FORMAT='KAFKA'
)
AS SELECT
    ner.key AS KEY,
    ner.value AS "ner_value",
    nel.value AS "nel_value"
FROM "item-NER" ner
INNER JOIN "item-NEL" nel
    WITHIN 2 MINUTES
    GRACE PERIOD 0 SECONDS
    ON ner.key = nel.key;
```

LISTING 4. Stream join query

the sentence to create the key. The sentence can be then further processed by the knowledge sources to, for example, identify the named entities linked to DBpedia as shown in Listing 2 and extract the relations as shown in Listing 3. Each knowledge source will publish its outputs in the corresponding topic (e.g., `item-NEL` and `item-RE`).

A join query like the one in Listing 4 will merge the different topics into one within a time window. The current implementation of the join query on ksqlDB runs continuously and streams the results as soon as there is a match within the specified time windows. Then, a final knowledge source will combine the different outputs to resolve and build the final annotation.

IV. EVALUATION

We conducted an empirical evaluation to compare the performance of the annotation blackboard to a sequential annotation pipeline for transforming text into rich semantic annotations. Our evaluation only compares the annotation blackboard with the sequential annotation pipeline, because they use two distinct approaches and we have not identified other approaches for text annotation. Today, most of the solutions use sequential annotation pipelines where some parts are executed in parallel using big-data-ready technologies to speed and parallelise the process. However, the underlying paradigm is the same.

The experiment includes knowledge sources that perform topic annotation, sentiment annotation, coreference resolution, NER, NEL, dark entity resolution and RE linking. The NER and NEL use either raw text or the coreference resolution as input, the dark entity resolution uses the outputs of NER and NEL, while the RE linking can use either the outputs of the NER or REL. The other knowledge sources only use raw text. A final knowledge source uses the outputs of the previous knowledge sources to represent their annotations as graphs following an event/item annotation ontology [18].

To conduct the experiment, we used feeds of news articles written in English. The articles were gathered from a wide

```
nif:sentiment
  a owl:ObjectProperty ;
  rdfs:label "sentiment"@en ;
  rdfs:comment """The sentiment of a string"
      ""@en ;
  rdfs:domain nif:PropertyBasedAnnotation ;
  rdfs:range rdfs:Resource .
```

LISTING 5. Extension of NIF to annotate sentiments using external resources.

variety of sources provided by the NewsAPI,[5] a service that provides news feeds from more than 80.000 news sources and blogs worldwide. The news articles varied in length, ranging from 18 to 34561 characters with a median text length of 2249 characters per article. The shortest articles consisted only of the title as we were unable to download its content due to paywalls or other limitations. The generated graphs varied in length accordingly, ranging from 9 to 5101 triples, with a median of 722 per article.

We run the experiment as part of a larger prototype of a big-data-ready AI system described in [13] to simulate a scenario closer to a real-life solution. The experiment runs on a cluster of 22 cloud instances with a total of 73 vCPU and 228GB RAM.[6] We implemented the annotation blackboard according to the descriptions in the previous section. For the sequential pipeline, we used the same knowledge sources as in the annotation blackboard but executed them sequentially. To annotate the news items we used the NIF vocabulary. However, we had to extend it to include the sentiment, coreference and RE linking annotations.

A. NIF EXTENSION

The NIF 2.0 specification [25] does not support annotating sentiment, coreferences and semantic relations between entities in the text. Therefore, we had to extend NIF 2.0 to include additional types of NLP annotations. To extend the NIF ontology, we aimed to reuse and adhere to the existing concepts and definitions.

NIF already includes the `nif:sentimentValue` and the `nif:topic` properties to annotate sentiments and topics respectively at either document (`nif:Context`) and sentence or word (`nif:String`) level. However, `nif:sentimentValue` is intended to annotate decimal values from -1 to 1. In our case, the `nif:sentimentValue` is not suitable as we annotate sentiment concepts using resources from external repositories such as DBpedia (e.g., http://dbpedia.org/resource/Joy). The way we annotate sentiment is similar to `nif:topic` definition. Therefore, we decided to extend NIF by adding a new type of property, the `nif:sentiment`, which is

[5] newsapi.org

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

```
nif:Coreferential
  a owl:Class ;
  rdfs:label "Coreferential"@en ;
  rdfs:comment """Text annotation denoting
      the start and end of two coindexed
      expresions. It starts at the start of
      the first expression and ends a the end
       of the second expression. The referent
       is represented by nif:head and the
      coreferential by nif:tail ."""@en ;
  rdfs:subClassOf nif:Structure ;
  owl:hasKey (nif:head nif:tail) .
```

LISTING 6. Extension of NIF to annotate coreferentials.

```
nif:CoreferenceResolution
  a owl:Class ;
  rdfs:label "CoreferenceResolution"@en ;
  rdfs:comment "Coreference resolution of
      Context string denoted by nif:
      wasConvertedFrom ."@en ;
  rdfs:subClassOf nif:Context .
```

LISTING 7. Extension of NIF to annotate coreference resolutions.

based on and follows the same definition as the `nif:topic`. Listing 5 shows the `nif:sentiment` definition.

As for coreference, we annotate expressions that are coreferential and the resolved text where all the coreferentials are substituted by their respective referents. The current NIF does not support these types of linguistic annotations. However, it does offer `nif:head` and `nif:tail` properties, which we used to annotate the co-indexed expressions (`nif:String`). The `nif:head` annotates the referent, and the `nif:tail` annotates the coreferential. To distinguish coreferentials from other annotations, we extended NIF by adding a new subclass of `nif:Structure` called `nif:Coreferential`, as shown in Listing 6. To distinguish the resolved text that contains the coreference resolutions from other annotations, we created a new subclass of `nif:Context` called `nif:CoreferenceResolution` as shown in Listing 7. We reused the property `nif:wasConvertedFrom` to indicate the text (`nif:Context`) from which the resolved text is derived.

We annotate semantic relations between entities present in the text using properties from DBpedia (e.g., http://dbpedia.org/property/locatedIn and http://dbpedia.org/ontology/occupation). However, NIF does not provide an explicit vocabulary to annotate semantic relations between expressions (`nif:String`). While it does provide properties to annotate linguistic dependency relations between words, these are intended to follow the dependency grammar theory. To address this limitation, we extended NIF by adding a new subclass of the `nif:Structure` called `nif:Relation` as shown in Listing 8. We also created two new properties to represent the `nif:Relation` with a resource from an external source: the `nif:taMsRelationRef` and `nif:taRelationRef`. The `nif:taMsRelationRef`

```
nif:Relation
  a owl:Class ;
  rdfs:label "Relation"@en ;
  rdfs:comment """Text annotation denoting
      the start and end of two related
      expresions. It starts at the start of
      the first expression and ends a the end
       of the second expression. The relation
       is a directed property from the nif:
      head to the nif:tail ."""@en ;
  rdfs:subClassOf nif:Structure ;
  owl:hasKey (nif:head nif:tail nif:
      taMsRelationRef) .
```

LISTING 8. Extension of NIF to annotate coreferentials.

represents the most significant semantic relation, while the `nif:taRelationRef` represents more than one relevant relation.

In addition to extending the NIF ontology, we also needed to extend the hash-based identifiers proposed in the NIF specification [26]. The hash-based identifiers provide an alternative to the offset-based identifiers used in the listings of section III. The main difference between these two types of identifiers is that the hash-based identifiers use the MD5 algorithm to hash the surrounding text of the annotations as part of the identifier, rather than using a sequential number for the text plus the offsets of the annotation. While hashing the text provides a universal identification, the hash-based identifiers can produce the same identifier when the same word is annotated in different positions with the same surrounding text. This can be addressed by increasing the amount of surrounding text used for creating the identifier. However, this solution may not always work, particularly for small tweets or texts. Therefore, to create unique and universal identifiers, we added the offsets to the hash-based identifiers. This solution ensures that the same annotation in different positions with the same surrounding text will always have different identifiers.

B. RESULTS

In Table 2 we report the process time and wall time for each approach and the improvement of the annotation blackboard over the sequential pipeline. Process time refers to the amount of time it takes for the CPU to complete a task, while wall time refers to the amount of time elapsed from the start to the end of a task, including any time spent waiting for resources or performing other tasks. As the underlying NLP models in both approaches are the same, the resulting annotations are identical. Therefore, we did not use information retrieval (IR) metrics like accuracy to evaluate the results. The Appendix A shows two examples of the resulting annotations, including annotations from each NLP process. Further work is needed to evaluate in detail how the proposed approach improves flexibility and maintainability compared to traditional tightly-coupled pipelines.

The results of the experiment show that the annotation blackboard outperformed the sequential pipeline in terms of

TABLE 2. Median time [range] in seconds for the annotation blackboard and the sequential pipeline and the time improvement of the annotation blackboard compared to the sequential pipeline.

	Process time	Wall time
Blackboard	0.286 [0.029–2.23]	29.999 [3.152–88.55]
Sequential	0.335 [0.038–2.289]	41.612 [3.505–110.337]
Improvement	12.584% (1.143x)	22.52% (1.29x)

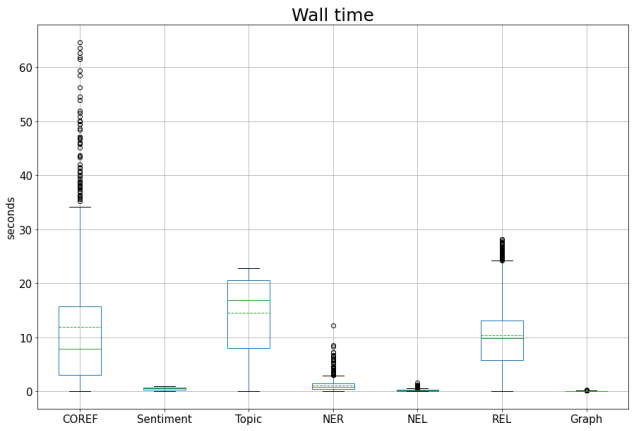

FIGURE 6. Wall times for the different NLP processes, including process sleep and network times.

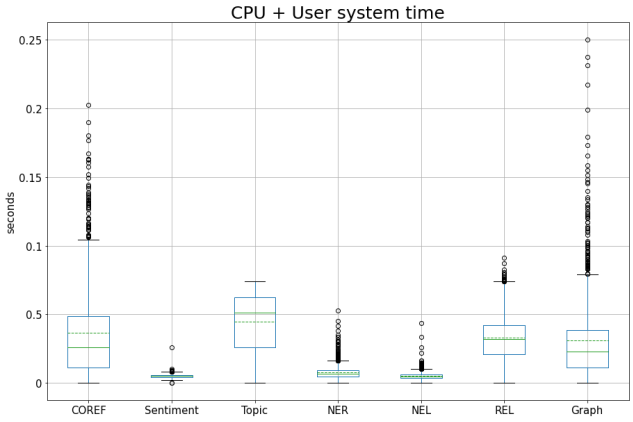

FIGURE 7. CPU and user system process times for the different NLP processes.

both process time and wall time per article. The boxplots in figures 6 and 7 show the distribution of time taken by each knowledge as wall time and process times respectively. This can help identify which knowledge sources take longer to run and potentially be used to optimise their performance.

V. DISCUSSION

Our experiment demonstrates that using the Blackboard Model to generate rich semantic annotations from raw text is feasible. The Blackboard Model reduces the complexity and dependencies of the annotation pipeline using decoupled knowledge sources. This allows the system to be easily extended with new knowledge sources, that can add additional capabilities or fulfil other use cases without the need to redesign the system or modify it extensively. It also facilitates parallelization and scalability of the system.

As the knowledge sources are independent units that employ common standards, they can be deployed as microservices and scaled effortlessly. In our experiment, each knowledge source has been deployed as a dockerised application. We have also separated the NLP models from the knowledge sources, achieving better resource allocation, fast updates, replication and reuse. This separation is reflected in the difference between the wall time and process time figures 6 and 7. The process time does not include the inference and processing time of the model. One possible alternative to reduce the time of the models is running these models on GPUs rather than on CPUs. Overall, this makes the Blackboard Model an attractive paradigm for building flexible, scalable and modular systems.

However, in our experiment, we observed that current semantic vocabularies for text annotation do not cover all the different types of annotations in NLP. For instance, we needed to extend NIF because it does not have explicit support for annotating relations between entities, sentiments and coreference. This can pose challenges for integrating the outputs of knowledge sources and requires adapting these vocabularies to specific cases. As a result, the annotation system may vary across different systems.

A. INTEGRATION WITH FOUNDATION MODELS

Foundation models [5] are currently leading the state-of-the-art in various artificial intelligence areas, including NLP. Examples of foundation models are InstructGPT [6] and GPT4 [7]. These models have opened many new possibilities as they can provide remarkable results for tasks that have not even been trained for. The annotation blackboard can be implemented by integrating one or more foundation models. In this case, each knowledge source will be responsible for a different task and interact with the foundation model through specific prompts tailored to the task. Consequently, a single foundation model may be accessed by multiple knowledge sources at the same time.

B. JOURNALISTIC USE CASES

Our proposed annotation blackboard can be extended with additional knowledge sources to support different journalistic use cases in newsrooms [27], [28]:

1) PERSONALISED NEWS CREATION

Knowledge sources tailored to journalists' interests or news angles can provide journalists with more personalised news feeds and help newsrooms to better adapt to their audience or focus [29], [30]. For example, a knowledge source can be added to connect dispersed and hidden pieces of information to a particular story of interest or frame the story with a new angle targeting a particular audience.

2) AUTOMATED NEWS ARTICLE SUMMARISATION

As soon as partial annotations containing the key concepts and relevant information are generated, other knowledge

sources can use them to summarise the text [31]. This can save journalists time in sifting through large amounts of text and allow them to quickly identify the most important information.

3) FACT CHECKING

Knowledge sources can connect to knowledge graphs and external repositories such as DBpedia and Wikidata. These repositories can be used to verify claims and statements made by politicians, public figures, and other sources [32].

C. OTHER USE CASES

The Blackboard Model can be employed in other domains with multimodal data, such as audio, time series and images.

1) STOCK MARKET PREDICTION

An annotation blackboard can provide a live view of the market by combining multiple knowledge sources that leverage different types of data such as historical and current price data, market trends, financial reports, news articles and social media [33]. This facilitates other knowledge sources to generate more accurate predictions and analyses of the stock market.

2) SMART ENVIRONMENTS

In smart environments such as smart cities, factories and buildings with large amounts of diverse IoT sensors, each knowledge source would be responsible for processing data from a specific type of sensor [34]. The annotation blackboard would integrate the partial solutions that can be then used for decision making like optimising energy usage and routing, detecting and responding to security threats and managing traffic flow.

VI. CONCLUSION

We have shown the applicability of the Blackboard Model in the context of NLP for transforming news into rich semantic annotations. Our experiment shows that the annotation blackboard outperforms the sequential approach in terms of process time and wall time. This is due to the parallel and concurrent nature of the Blackboard Model. Besides, the Blackboard Model offers additional advantages as it can easily accommodate new knowledge sources, allowing for easy scaling, adaptation to different purposes and integration of diverse techniques.

One of the main challenges of the Blackboard Model is the coordination model, which we have addressed by employing Apache Kafka and ksqlDB. Apache Kafka provided off-the-shelf coordination, hence alleviating the need for developing a specific component designated to coordinate the different knowledge sources. KsqlDB brings a flexible mechanism for merging the outputs of the knowledge source. In addition, we employed the NIF vocabulary and semantic representation to represent the outputs of the knowledge sources, easing the integration and understanding of the knowledge sources. The combination of these technologies makes the Blackboard Model a viable alternative for integrating and scaling AI pipelines to Big Data solutions. One limitation of using Apache Kafka ksqlDB to coordinate and merge the outputs of the knowledge sources is handling large messages that exceed the memory limitations. This can be addressed by temporarily storing the data of the messages in databases and only passing metadata. This can considerably reduce the size of the messages, as only the needed identifiers to retrieve the data are passed.

Alternatives to Apache Kafka can be considered to implement the blackboard such as Apache ActiveMQ[7] and RabbitMQ.[8] We decided to use Apache Kafka as this provides a native distributed architecture and is designed for high message volumes and real-time processing, providing a better fit to our requirements and throughput than Apache ActiveMQ and RabbitMQ. Albeit, as the proposed annotation blackboard is not bound to a particular technology, the alternatives can be adopted since they provide similar message distribution protocols as the publish/subscribe protocol. Ultimately, the technological decision relies on the requirements of the deployed solution.

We believe that our findings provide valuable insights for researchers and practitioners working on text annotation. Further research can explore the potential of the Blackboard Model in other NLP applications, as well as investigate the use of different knowledge sources to solve the same annotation task and the integration of different types of representations such as vector embeddings. Further investigation is also needed on semantic vocabularies to cover different types of annotations. To further scale the annotation blackboard, in future work, we are interested in the possibility of building the knowledge sources as Spark jobs and combining Apache Kafka with Apache Spark.[9]

Future big-data AI systems will need similar solutions as the one proposed in this work to scale their annotation or similar pipelines and easily integrate the new advances in AI. Hence, employing the Blackboard Model as proposed in this work may be relevant and applicable to other domains with similar needs.

APPENDIX A
ANNOTATION EXAMPLES

Listings 9 and 10 provide a simplified example of the result of the annotation blackboard. To facilitate the identification of the knowledge source that generated each annotation type, we indicated in the listings the knowledge source that produced the following annotations with a comment that start with #. These examples also represent the situation where not all the knowledge sources contributed directly to the final result, a situation that happens when the output of some knowledge source is not required to annotate the text due to

[7] activemq.apache.org
[8] rabbitmq.com
[9] spark.apache.org

```
<http://example.org/doc1>
 a nif:Context ,
 nif:OffsetBasedString ;
 nif:beginIndex "0"^^xsd:nonNegativeInteger;
 nif:endIndex "83"^^xsd:nonNegativeInteger ;
 nif:isString "Pau Casals, the legendary
     Catalan cellist, received the United
     Nations Peace Medal." ;

#Sentiment
 nif:sentiment http://dbpedia.org/resource/
     Joy ;

#Topic
 nif:topic http://dbpedia.org/resource/
     History .

#NER
<http://example.org/doc1#offset_0_9>
 a nif:Phrase, nif:OffsetBasedString;
 nif:beginIndex "0"^^xsd:nonNegativeInteger;
 nif:endIndex "10"^^xsd:nonNegativeInteger ;
 nif:anchorOf "Pau Casals" ;

#NEL
 itsrdf:taClassRef <http://dbpedia.org/
     ontology/Person> ;
 itsrdf:taConfidence 0.97 ;
 itsrdf:taIdentRef <http://dbpedia.org/
     resource/Pau_Casals> .

#NER
<http://example.org/doc1#offset_57_83>
 a nif:Phrase, nif:OffsetBasedString;
 nif:beginIndex "57"^^xsd:nonNegativeInteger
      ;
 nif:endIndex "83"^^xsd:nonNegativeInteger ;
 nif:anchorOf "United Nations Peace Medal" ;

#NEL
 nif:taClassRef <http://dbpedia.org/ontology
     /Award> ;
 itsrdf:taConfidence 0.92 ;
 itsrdf:taIdentRef <http://dbpedia.org/
     resource/United_Nations_Peace_Medal> .

#REL
<http://example.org/doc1#offset_0_83>
 a nif:Relation, nif:OffsetBasedString ;
 nif:beginIndex "0"^^xsd:nonNegativeInteger;
 nif:endIndex "83"^^xsd:nonNegativeInteger ;
 nif:head <http://example.org/doc1#
     offset_0_9> ;
 nif:tail <http://example.org/doc1#
     offset_57_83> ;
 nif:taRelationRef <https://dbpedia.org/
     ontology/award>;
 itsrdf:taConfidence 0.9  .
```

LISTING 9. Example of the result of the annotation process. The process that generated each of the annotation is marked above the annotation as a comment (i.e., #Sentiment, #Topic, #NER, #NEL and #REL).

the lack of information or the absence of relevant content to be annotated. This implies that the knowledge source cannot perform any annotation because there is nothing in the text that falls within its domain of expertise.

```
<http://example.org/doc2>
 a nif:Context ,
 nif:OffsetBasedString ;
 nif:beginIndex "0"^^xsd:nonNegativeInteger;
 nif:endIndex "41"^^xsd:nonNegativeInteger ;
 nif:isString "The cat ate the fish. It is
     sleeping now." ;

#Topic
 nif:topic http://dbpedia.org/resource/
     Animal .

#NER
<http://example.org/doc2#offset_0_6>
 a nif:Phrase, nif:OffsetBasedString;
 nif:beginIndex "0"^^xsd:nonNegativeInteger;
 nif:endIndex "7"^^xsd:nonNegativeInteger ;
 nif:anchorOf "The cat" .

#NER
<http://example.org/doc2#offset_22_23>
 a nif:Phrase, nif:OffsetBasedString;
 nif:beginIndex "22"^^xsd:nonNegativeInteger
      ;
 nif:endIndex "24"^^xsd:nonNegativeInteger;
 nif:anchorOf "It" ;

#COREF
<http://example.org/doc2#offset_0_23>
 a nif:Coreferential, nif:OffsetBasedString
     ;
 nif:beginIndex "0"^^xsd:nonNegativeInteger;
 nif:endIndex "24"^^xsd:nonNegativeInteger ;
 nif:head <http://example.org/doc1#
     offset_0_6> ;
 nif:tail <http://example.org/doc1#
     offset_22_23> .

<http://example.org/doc2_1>
 a nif:CoreferenceResolution ,
 nif:OffsetBasedString ;
 nif:beginIndex "0"^^xsd:nonNegativeInteger;
 nif:endIndex "46"^^xsd:nonNegativeInteger ;
 nif:isString "The cat ate the fish. The cat
      is sleeping now." ;
 nif:wasConvertedFrom <http://example.org/
     doc2> .
```

LISTING 10. Simplified example of the result of the annotation process for a coreference resolution.

ACKNOWLEDGMENT

The computations were performed on the Norwegian Research and Education Cloud (NREC), using resources provided by the University of Bergen and the University of Oslo (https://www.nrec.no).

REFERENCES

[1] R. Grishman, "Twenty-five years of information extraction," *Natural Lang. Eng.*, vol. 25, no. 6, pp. 677–692, Nov. 2019.

[2] 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, and R. Segers, "NewsReader: Using knowledge resources in a cross-lingual reading machine to generate more knowledge from massive streams of news," *Knowl.-Based Syst.*, vol. 110, pp. 60–85, Oct. 2016.

[3] T. Al-Moslmi, M. Gallofré Ocaña, A. L. Opdahl, and C. Veres, "Named entity extraction for knowledge graphs: A literature overview," *IEEE Access*, vol. 8, pp. 32862–32881, 2020.

[4] M. Van Erp, F. Ilievski, M. Rospocher, and P. Vossen, "Missing Mr. Brown and buying an Abraham Lincoln—Dark entities and DBpedia," in *Proc. NLP-DBPEDIA@ ISWC*, H. Paulheim, M. van Erp, A. Filipowska, P. N. Mendes, and M. Brümmer, Eds., 2015, pp. 81–86.

[5] R. Bommasani et al., "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.

[6] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 27730–27744.

[7] OpenAI et al., "GPT-4 technical report," 2023, *arXiv:2303.08774*.

[8] T. Wolf et al., "TransFormers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstration*, Q. Liu and D. Schlangen, Eds., Oct. 2020, pp. 38–45.

[9] H. P. Nii, "The blackboard model of problem solving and the evolution of blackboard architectures," *AI Mag.*, vol. 7, p. 38, Jun. 1986.

[10] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy, "The hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty," *ACM Comput. Surv.*, vol. 12, no. 2, pp. 213–253, Jun. 1980.

[11] M. Gallofré Ocaña, L. Nyre, A. L. Opdahl, B. Tessem, C. Trattner, and C. Veres, "Towards a big data platform for news angles," in *Proc. Norwegian Big Data Symp.*, 2018, pp. 1–14.

[12] M. Gallofré Ocaña and A. L. Opdahl, "Developing a software reference architecture for journalistic knowledge platforms," in *Proc. ECSA Companion*, R. Heinrich, R. Mirandola, and D. Weyns, Eds., 2021, pp. 1–8.

[13] M. Gallofré Ocaña and A. L. Opdahl, "A software reference architecture for journalistic knowledge platforms," *Knowl.-Based Syst.*, vol. 276, Sep. 2023, Art. no. 110750.

[14] T. A. A. Al-Moslmi, M. Gallofré Ocaña, A. L. Opdahl, and B. Tessem, "Detecting newsworthy events in a journalistic platform," in *Proc. 3rd Eur. Data Comput. Journalism Conf.*, 2019, pp. 3–5.

[15] A. L. Opdahl, T. Al-Moslmi, D.-T. Dang-Nguyen, M. Gallofré Ocaña, B. Tessem, and C. Veres, "Semantic knowledge graphs for the news: A review," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1–38, Jul. 2023.

[16] M. Kattenberg, Z. Beloki, A. Soroa, X. Artola, A. Fokkens, P. Huygen, and K. Verstoep, "Two architectures for parallel processing of huge amounts of text," in *Proc. 10th Int. Conf. Lang. Resour. Eval.*, May 2016, pp. 4513–4519.

[17] H. P. Nii, "Blackboard application systems, blackboard systems and a knowledge engineering perspective," *AI Mag.*, vol. 7, p. 82, Jul. 1986.

[18] A. L. Opdahl and B. Tessem, "Ontologies for finding journalistic angles," *Softw. Syst. Model.*, vol. 20, no. 1, pp. 71–87, Feb. 2021.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[20] X. Han, T. Gao, Y. Yao, D. Ye, Z. Liu, and M. Sun, "OpenNRE: An open and extensible toolkit for neural relation extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP), Syst. Demonstrations*, 2019, pp. 169–174.

[21] P. He, J. Gao, and W. Chen, "DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing," 2021, *arXiv:2111.09543*.

[22] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes, "Improving efficiency and accuracy in multilingual entity extraction," in *Proc. 9th Int. Conf. Semantic Syst.*, Sep. 2013, pp. 121–124.

[23] P. Cimiano, C. Chiarcos, J. P. McCrae, and J. Gracia, *Representing Annotated Texts As RDF*. Cham, Switzerland: Springer, 2020, pp. 61–87.

[24] A. Fokkens, A. Soroa, Z. Beloki, N. Ockeloen, G. Rigau, W. R. Van Hage, and P. Vossen, "NAF and GAF: Linking linguistic annotations," in *Proc. 10th Joint ISO-ACL SIGSEM Workshop Interoperable Semantic Annotation*, H. Bunt, Ed., 2014, pp. 9–16.

[25] S. Hellmann, J. Lehmann, S. Auer, and M. Brümmer, "Integrating NLP using linked data," in *Proc. The Semantic Web—ISWC 2013*. Berlin, Germany: Springer, 2013, pp. 98–113.

[26] S. Hellmann, J. Lehmann, and S. Auer, "Linked-data aware URI schemes for referencing text fragments," in *Knowledge Engineering and Knowledge Managemen*. Berlin, Germany: Springer, 2012, pp. 175–184.

[27] M. Gallofré Ocaña and A. L. Opdahl, "Challenges and opportunities for journalistic knowledge platforms," in *Proc. CIKM Workshops*, 2020, pp. 1–9.

[28] M. Gallofré Ocaña and A. Opdahl, "Supporting newsrooms with journalistic knowledge graph platforms: Current state and future directions," *Technologies*, vol. 10, no. 3, p. 68, May 2022.

[29] B. Tessem, M. Gallofré Ocaña, and A. L. Opdahl, "Construction of a relevance knowledge graph with application to the local news angle," in *Proc. 5th Symp. Norwegian AI Soc.*, 2023, pp. 1–12.

[30] M. G. Ocaña, T. Al-Moslmi, and A. L. Opdahl, "Knowledge graph semantic annotation and population with real-time events data from GDELT," in *Proc. IEEE 24th Conf. Bus. Informat. (CBI)*, vol. 2, Jun. 2022, pp. 65–72.

[31] L. Huang, L. Wu, and L. Wang, "Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 5094–5107.

[32] E. Maliaroudakis, K. Boland, S. Dietze, K. Todorov, Y. Tzitzikas, and P. Fafalios, "ClaimLinker: Linking text to a knowledge graph of fact-checked claims," in *Proc. Companion Web Conf.*, Mali, Apr. 2021, pp. 669–672.

[33] S. Elhammadi, L. V. S. Lakshmanan, R. Ng, M. Simpson, B. Huai, Z. Wang, and L. Wang, "A high precision pipeline for financial knowledge graph construction," in *Proc. 28th Int. Conf. Comput. Linguistics*, Barcelona, Spain, 2020, pp. 967–977.

[34] S. Chun, J. Jung, X. Jin, S. Seo, and K.-H. Lee, "Designing an integrated knowledge graph for smart energy services," *J. Supercomput.*, vol. 76, no. 10, pp. 8058–8085, Oct. 2020.

MARC GALLOFRÉ OCAÑA received the B.Sc. and M.Sc. degrees in informatics engineering from the Polytechnic University of Catalonia-BarcelonaTech, Barcelona, in 2015 and 2017, respectively, and the Ph.D. degree from the University of Bergen, Norway, in 2023. His research interests include big-data systems, semantic knowledge graphs, AI, and software architectures. As part of his research career, he has been involved in the organization of multiple workshops and international conferences and workshops for the industry.

ANDREAS L. OPDAHL received the Ph.D. degree from the Norwegian University of Science and Technology, in 1992. He is currently a Professor in information systems development with the University of Bergen, Norway. His research interests include ontologies and knowledge graphs, enterprise and IS modelling, and safety and security requirements. He is the author, coauthor, or co-editor of more than a 100 peer-reviewed research articles that have been cited more than 3700 times. He is a member of IFIP WG5.8 on Enterprise Interoperability and WG8.1 on Design and Evaluation of Information Systems. He serves regularly as a reviewer for premier international journals and on the program committees and as an organizer of the most renowned international conferences and workshops in his fields of interest.