

Semantically-Enabled Data Spaces for Intelligent Data Asset Discovery^{*}

Harry Nakos, Theodore Dalamagas, Stelios Sartzetakis, Nikos Kapetanas, Danae Pla Karidi, and Yannis Stavrakas[†]

Information Management Systems Institute, Athena Research Center, Athens, Greece

Abstract

We demonstrate KnowDS, a system that extends Data Spaces with semantic search capabilities. We show how KnowDS can leverage the minimal descriptive metadata of the data assets in a Data Space to construct a relevance graph, and how it uses Steiner trees to find relevant items that do not contain any of the query terms. We describe the KnowDS system in detail, and present a running example that showcases its functionality.

Keywords

Data Spaces, semantic search, Steiner tree

1. Introduction and Motivation

Data Spaces (DS) represent a paradigm shift in data sharing, enabling secure, sovereign, and interoperable exchange of data across organizations. Unlike traditional centralized platforms, data remains with the data owner, ensuring control over its usage. This decentralized model is facilitated by standardized technical frameworks and common governance rules.

Key components in the Data Space architecture [1] include trusted *connectors*, identity management services, metadata catalogs, and policy enforcement engines. These components ensure data findability, accessibility, and adherence to pre-defined usage agreements. Data Spaces foster collaboration, allowing participants to create new data-driven services and business models. Initiatives like the International Data Spaces (IDS) [1] and Gaia-X [2] are driving the development and adoption of these interoperable ecosystems, that are crucial for unlocking the full potential of the data economy.

Discovering data assets (namely datasets, together with usage rules) in a Data Space is an essential first step for *data consumers*. It relies heavily on structured metadata catalogs, often leveraging standards like DCAT [3]. Users can search and filter these catalogs using descriptive metadata to identify relevant data offerings. The underlying *connectors* within the Data Space ensure secure access and facilitate subsequent data transactions for the acquisition of the selected data assets.

Such discovery of data assets through filtering is marginally adequate, and leaves a lot of room for improvement. The capability for semantic search would significantly enhance data discovery in Data Spaces. By leveraging knowledge graphs and conceptual understanding, semantic search would bridge the gap between human language and machine processing, leading to an easier and more productive user experience. Towards this end, we envision semantically-enabled Data Spaces for efficient discovery of data assets. Such a Data Space would:

- use not only the descriptive metadata of the data assets, but also external semantic sources, to construct and maintain knowledge graphs,
- guide the data consumer to formulate a search query based on natural language,
- leverage the knowledge graphs to identify the data assets that best match the query criteria,
- prompt the data consumer to examine the results and initiate the Data Space process for acquiring the desirable data asset(s).

In this demonstration, we present *KnowDS*, a system that implements an important first step towards semantically-enabled Data Spaces. KnowDS is implemented as an extension of the *soivity* Data Space

^{*} SEMANTiCS 25: 21st International Conference on Semantic Systems, September 03–05, 2025, Vienna, Austria

[†] xnakos@athenarc.gr, dalamag@athenarc.gr, stelios@athenarc.gr, nkapetanas@athenarc.gr, danae@athenarc.gr, ys@athenarc.gr

(Community Edition) [4], currently one of the leaders in the field, but could be easily adapted to other Data Space products. KnowDS demonstrates how semantic search capabilities can be introduced into Data Spaces.

Challenges. With KnowDS we address the following challenges: (a) how to retrieve more – but still relevant – results than those returned by a simple filtering, (b) how to rank the results according to their semantic relevance, (c) how to employ a simple method of querying that does not require a model of the domain knowledge of the Data Space, (d) how to take advantage of the simple metadata of the assets that are already present in the Data Space, and (e) how to remain domain-agnostic and readily adapt to different DS and different types of data assets.

Contribution. KnowDS is a prototype that integrates with the *soivity* Data Space, and adds semantic search capabilities on Data Spaces, without modifying the DS source code or configuration. Two key points on DS integration are: (a) to keep the KnowDS internal database in sync with the data assets of the Data Space (see section 2.1), and (b) once a data asset is selected, to provide the user with a pointer to acquire the data asset through the Data Space procedures. Moreover: (c) KnowDS shows how to create a semantic graph from minimal DS asset metadata (see section 2.2), and (d) uses a Steiner tree [5] to identify additional relevant data assets, which would not have been included in the results by a simple keyword search (see section 2.3).

An evaluation of the KnowDS approach is planned for the immediate future, which will cover testing against real datasets, assessment of the improvement of asset discoverability, efficiency issues, and adaptability to different DS technologies.

2. System Overview

A typical data asset retrieval in a Data Space involves two major steps:

- As a first step, the data consumer examines the available data assets through the web interface of the *soivity Dataspace Portal*. The Dataspace Portal (formerly called Authority Portal) is a centralized catalogue managing authorized users and available data assets. Although the data remains with its owner (namely the data provider) in a distributed fashion, the Dataspace Portal maintains a list of asset metadata enabling the discovery of data assets. This list is kept in a PostgreSQL [6] database.
- Then, as a second step, having found the desired data asset in the list, the data consumer is directed to the provider of the data asset in order to acquire it. This is done through a process in which the consumer and the provider Connector modules communicate and carry out the transaction.

KnowDS enhances the discovery of the data asset in the first step. The second step (data acquisition) is not affected by KnowDS. In what follows, we present some key functions of KnowDS.

2.1. Data Synchronization

KnowDS needs the asset metadata stored in the DS in order to implement its search method. *soivity* DS keeps those metadata in a PostgreSQL database, while KnowDS uses Apache Solr [7] as its data repository. The challenge is twofold. First, to keep KnowDS in sync with the Data Space metadata. KnowDS should be notified about any change to DS asset metadata, and update its own data repository and internal structures to reflect this change. The second challenge is to do this without interfering with the internal workings of the DS. This will allow KnowDS to easily adapt to other DS platforms and products.

To this end, KnowDS uses Debezium [8], an open-source platform for capturing changes in data. Debezium monitors the log files of a target database (in this case the *soivity* PostgreSQL log files) and notifies KnowDS of any changes that occur. Debezium supports a number of popular data repositories, therefore it should be straightforward to adapt KnowDS to other DS products. Essentially, the parameterization of Debezium is the only step needed for adding KnowDS to some existing DS.

2.2. Semantic Graph Creation

Each data asset in a Data Space is associated with descriptive metadata that is available throughout the DS. This metadata consists of simple fields, usually based on DCAT. KnowDS uses only a few of those fields, namely the *title*, the *tags*, and the *description*. The *title* and *description* are self-explanatory. The *tags* field is a set of keywords related to the data asset. Moreover, a couple of identifiers are used: the *connector endpoint* identifies the connector of the data provider, and the *asset ID* identifies the data asset itself. Those two identifiers allow the data consumer to start a negotiation for the specific data asset with the data provider, through their respective connectors.

KnowDS uses the *tags* fields to build a *relevance graph* connecting data assets.

- Each node in the relevance graph represents a data asset. An edge between two nodes represents the tags that the nodes have in common, and the label of the edge is the common tags separated by commas (see Figure 1 for an example, the different node colors become meaningful in section 3).
- Each edge has a number attached, indicating the number of common tags that the edge represents. This number is called relevance strength and denotes the relevance between the nodes connected by the edge. The more common keywords between two assets, the higher the relevance strength between them. Obviously, if two nodes do not share any tag, they are not connected by an edge.

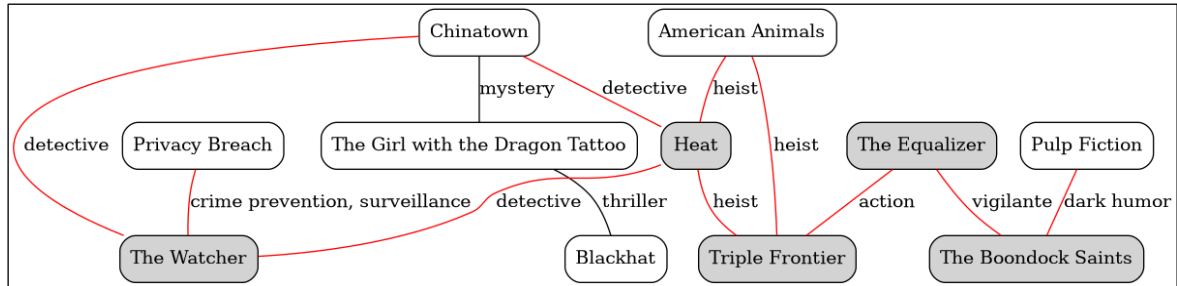


Figure 1: The Steiner tree of the relevance graph from the movies running example.

We assume that the tags have a semantic orientation and refer to the concepts that characterize the data asset. It will also be beneficial, although not necessary, if the tags are selected from a curated, controlled vocabulary, which is a reasonable assumption.

2.3. Data Asset Matching

To find a data asset, the data consumer provides KnowDS with a set of query terms. KnowDS aims to find the relevant data assets in the DS, and rank them according to their semantic proximity to the terms, using the relevance graph. The task proceeds in three steps:

- First, we find the set of nodes in the relevance graph that contain one or more of the provided terms in their tags, title, or description. We call this set the set of direct results.
- Then, we construct the Steiner tree [5] that contains all the nodes in the direct results set. The Steiner tree is the least-cost connected subgraph of the relevance graph that contains these nodes (see Figure 1). We set the cost between two nodes connected by an edge with relevance strength N to be $1 / N$. Intuitively, a node that connects two or more of the nodes in the direct result set will be part of the Steiner tree, even if this node is not part of the direct result set. The result of the query will consist of all the nodes of the Steiner tree, and will include the direct result set plus the nodes that act as in-betweens. This way, the results may contain nodes that are indeed relevant, but do not contain any of the initial terms. When constructing the Steiner tree, we also consider the case that the relevance graph is a disconnected graph.
- Finally, we rank the results. The nodes of the Steiner tree are ranked by combining two score components. The first score is the text relevance score, which reflects how well the textual

content of each node matches the query terms. This score captures the fact that nodes containing more search terms are more relevant. The second score is the connectivity score that indicates the importance of each node within the Steiner tree. This score is calculated as the sum of the relevance strength of all edges of the node. Higher connectivity score means that the node exhibits stronger connections with its neighbors, constituting a more central node that bridges many related nodes. The final ranking score for each node, called combined relevance score, is then computed by multiplying the (normalized) text relevance score by the (normalized) connectivity score. Therefore, a node must have both a high textual relevance score, and a high connectivity score to be ranked high.

The result of the query is the nodes of the Steiner tree, ranked according to the combined relevance score explained above.

3. User Interface and Demonstration

We used a dataset about movies for our running example. We set up a simple DS consisting of two connectors, one for the data provider and another one for the data consumer. The data assets in our example are movie trailers (short video files). The DS descriptive metadata includes some basic information about the respective movies, namely the movie type, the movie title, and a short textual description. Each data asset has an *asset ID*, identifying the asset, and a *connector endpoint*, pointing to the connector in charge of the asset (in our trivial case, this is always the data provider connector).

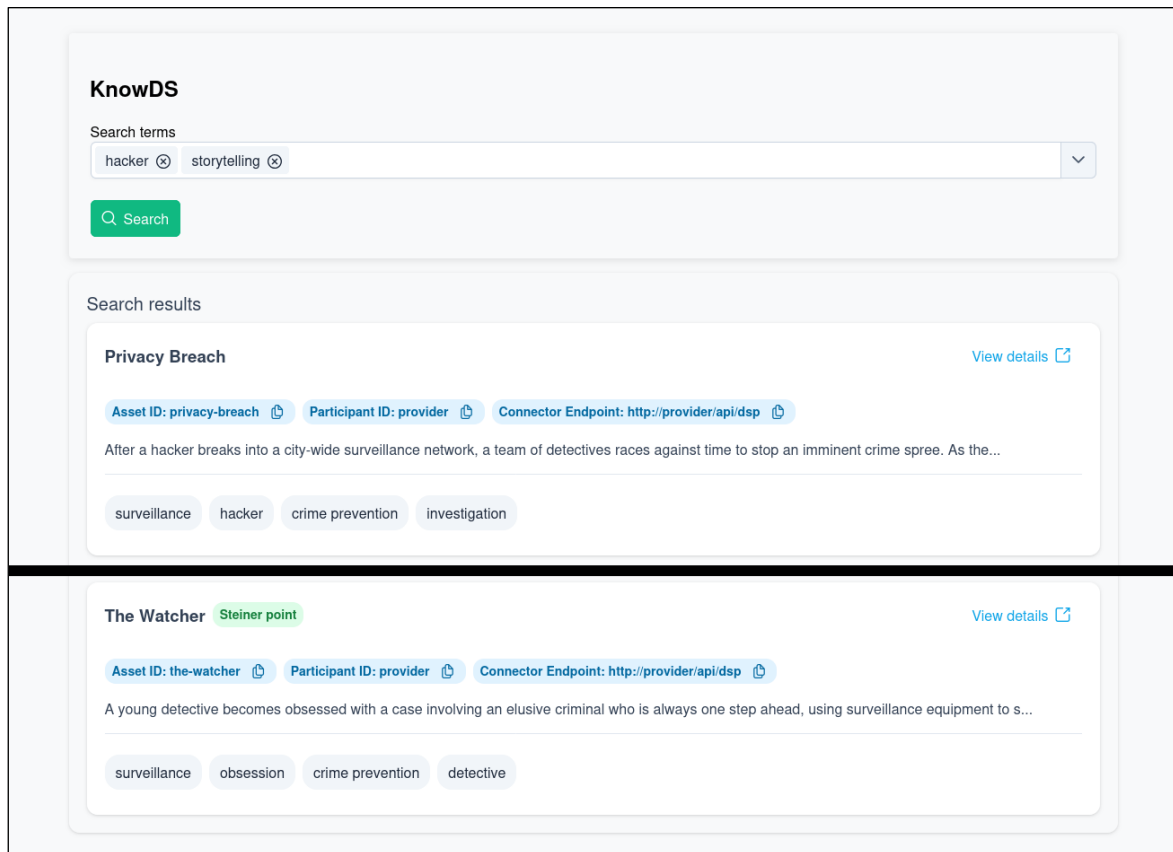


Figure 2: The KnowDS User Interface demonstrating the running example.

The KnowDS UI with the running example is depicted in Figure 2. The user (data consumer) enters the search terms “hacker” and “storytelling”, with the OR operator assumed. Note that the user may select a query term from a drop-down list of the existing tags (i.e., “hacker”), or insert a different term (i.e., “storytelling”). In future versions of KnowDS we plan to add more operators and expand the set of query terms by adding synonyms: for each term we will consult external sources such as WordNet [9] and DBpedia [10] to get a list of relevant words, that would maximize the possibility to find relevant

content in the DS by searching the *title* and *description* metadata fields. The results of this query correspond to the nodes of the Steiner tree in Figure 1. The white nodes in Figure 1 form the set of direct results, while the gray nodes are in-between nodes that connect the white nodes. The results are presented ranked, as already discussed. For brevity, Figure 2 depicts only the first and the last items of the result list, with a thick black line representing the rest of the results. The last result item corresponds to a gray node in the Steiner tree, meaning that it does not include any of the query terms, and is denoted by the flag “Steiner point” on the UI.

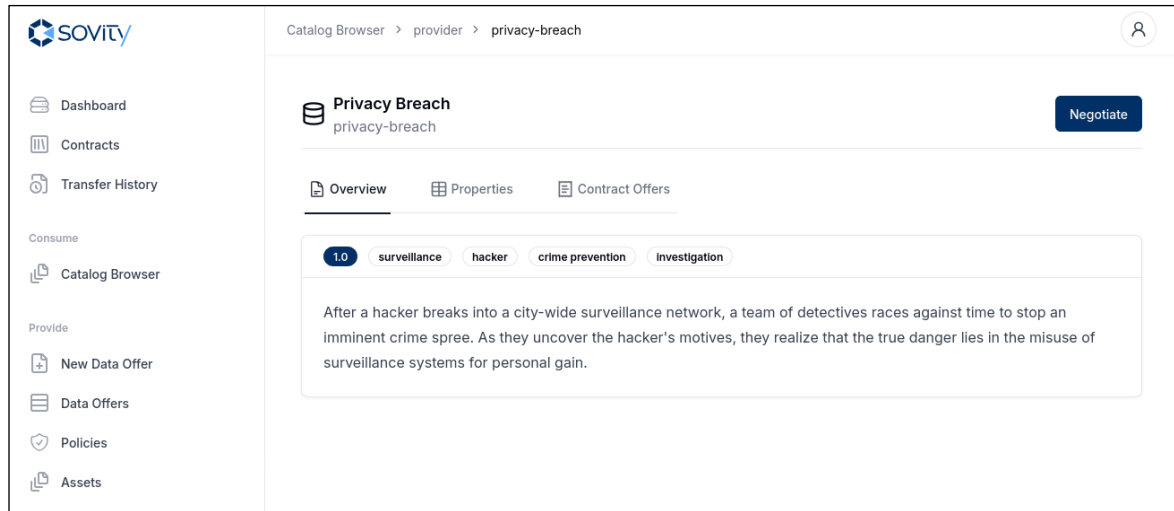


Figure 3: Data consumer ready to negotiate a data asset with the data provider connector.

Assuming the user is interested in the first item, titled “Privacy Breach”, the matching *asset ID* and *connector endpoint* can be used by an authorized data consumer to negotiate the corresponding data asset, through the soivity DS connectors as depicted in Figure 3.

The KnowDS installation with the running example presented in this paper is available at: <https://knowds.imsi.athenarc.gr/>

Acknowledgements

This work is partially funded by the *UNDERPIN* project (Digital Europe Program, Grant Agreement no. 101123179), and the *DataBri-X* project (Horizon Europe Research and Innovation Program, Grant Agreement no. 101070069).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

- [1] International Data Spaces, <https://internationaldataspaces.org/>, June 2025.
- [2] Gaia-X, <https://gaia-x.eu/>, June 2025.
- [3] Data Catalogue Vocabulary (DCAT), <https://www.w3.org/TR/vocab-dcat-3/>, June 2025.
- [4] soivity, <https://soivity.de/>, June 2025.
- [5] Dreyfus, S. E., & Wagner, R. A. (1971). The Steiner problem in graphs. *Networks*, 1(3), 195-207.
- [6] PostgreSQL, <https://www.postgresql.org/>, June 2025.
- [7] Apache Solr, <https://solr.apache.org/>, June 2025.
- [8] Debezium, <https://debezium.io/>, June 2025.
- [9] WordNet, <https://wordnet.princeton.edu/>, June 2025.
- [10] DBpedia, <https://www.dbpedia.org/>, June 2025.