SPHINX: A System for Metapath-based Entity Exploration in Heterogeneous Information Networks

Serafeim Chatzopoulos
Univ. of the Peloponnese &
"Athena" RC
Greece
schatzop@uop.gr

Kostas Patroumpas
"Athena" RC
Greece
kpatro@athenarc.gr

Alexandros Zeakis
"Athena" RC
Greece
azeakis@athenarc.gr

Thanasis Vergoulis
"Athena" RC
Greece
vergoulis@athenarc.gr

Dimitrios Skoutas
"Athena" RC
Greece
dskoutas@athenarc.gr

ABSTRACT

We present SPHINX, a system for metapath-based entity exploration in Heterogeneous Information Networks (HINs). SPHINX allows users to define different views over a HIN based on both automatically selected and user-defined metapaths. Then, entity ranking and similarity search can be performed over these views to find and explore entities of interest, taking also into account any spatial or temporal properties of entities. A Web-based user interface is provided to facilitate users in performing the various functionalities supported by the system, including metapath-based view definition, index construction, search parameters specification, and visual comparison of the results.

PVLDB Reference Format:
Serafeim Chatzopoulos, Kostas Patroumpas, Alexandros Zeakis, Thanasis Vergoulis, Dimitrios Skoutas. SPHINX: A System for Metapath-based Entity Exploration in Heterogeneous Information Networks. *PVLDB*, 13(12): 2913-2916, 2020.
DOI: https://doi.org/10.14778/3415478.3415507

1. INTRODUCTION

Heterogeneous Information Networks (HINs) are graphs comprising different types of nodes (entities) and edges (relationships) [4]. HINs offer an intuitive and generic model for representing complex information in various domains. A core concept for analyzing HINs is that of *metapath*, which is a path defined on the schema of the HIN [5]. Metapaths represent relationships of different semantics between entities of the same or different type, providing a mechanism for exploring and analyzing a HIN from multiple perspectives. Thus, they are fundamental for several types of analyses in HINs, ranging from similarity joins to HIN embeddings and recommendations [5, 1, 3].

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415507

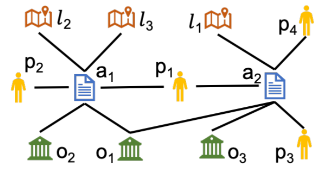


Figure 1: Illustrative example of a HIN containing articles, persons, organizations and locations.

Figure 1 presents an illustrative example of a HIN containing *articles* (a_1, a_2), *persons* (p_1, p_2, p_3, p_4), *organizations* (o_1, o_2, o_3) and *locations* (ℓ_1, ℓ_2, ℓ_3). Consider metapaths that connect two persons if they are mentioned in the same article (`PAP`) or in two articles mentioning the same person (`PAPAP`), organization (`PAOAP`) or location (`PALAP`). Then, p_2 and p_3 are connected according to `PAPAP` and `PAOAP`, but not according to `PAP` and `PALAP`.

Hence, both the neighborhood and the centrality of a node in a HIN become *relative* to the metapaths under consideration. This raises the need for tools that can facilitate users in defining and computing different *views* of a HIN based on different (combinations of) metapaths, which can then be used to compute and compare answers to questions such as which entities are the most important (central) in the network or having the highest similarity (common neighbors) to a query entity. The task becomes even more complex in the presence of entity types that are additionally associated with *spatial* or *temporal* properties (e.g., geospatial coordinates or timestamps). Spatial and temporal proximity are important factors in several analyses. Yet, since spatial and temporal relationships are typically not represented explicitly in the network structure, they cannot be captured by metapaths. Thus, analysis methods that are purely metapath-based will inevitably overlook these aspects, resulting in significant loss of information that is present in the data.

Motivated by the above, we have developed SPHINX[1], a system for metapath-based entity exploration in HINs, including support for spatial and temporal entities. SPHINX includes a *workflow manager* for executing preprocessing and offline tasks involving metapath-based view materialization, index construction and random walk computations, as

[1] `http://sphinx.magellan.imsi.athenarc.gr/`

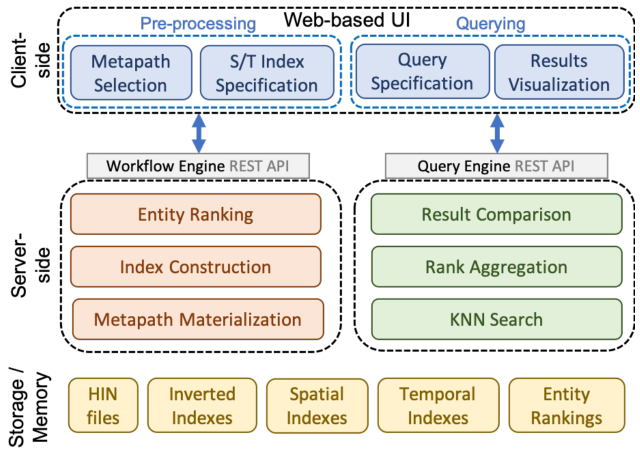


Figure 2: SPHINX architecture.

well as a *query engine* for executing top-k similarity search queries. A Web-based *user interface* is provided for users to preprocess a HIN, submit queries, and visualize the results. The main functionalities of SPHINX are:

- Compute and index different views of a HIN based on automatically selected or user-defined metapaths.
- Index spatial and temporal attributes of entities.
- Execute random walks on selected HIN views to generate entity rankings.
- Find the top-k most similar entities to a query with respect to different weighted combinations of metapaths and/or spatio-temporal attributes.
- Visualize and compare the results obtained from different views or weight parameters.

2. SYSTEM OVERVIEW

The main components of SPHINX are illustrated in Figure 2 and can be distinguished in the following core parts:

- *Workflow Engine.* The workflow engine handles tasks that are executed offline, as they typically have longer execution times, especially in large HINs. This includes metapath-based view materialization, index construction, and entity ranking via random walk computations. We use Apache Airflow[2], an open-source and scalable platform that allows to programmatically author, schedule and monitor workflows.
- *Query Engine.* The query engine executes top-k similarity search queries containing preferences over different combinations of metapaths and/or spatial or temporal attributes. For each criterion, a k-nearest neighbor (KNN) search is triggered, and the individual ranked lists of results are aggregated to produce the final top-k answers. Appropriate indices are built to speed up these KNN queries, including inverted indices, R-trees and B+ trees.
- *User Interface.* A Web-based user interface provides views for selecting metapaths, specifying indices to be constructed, defining query conditions and weight parameters, and visualizing the results. Client-side and server-side components communicate via REST APIs.

[2]`https://airflow.apache.org/`

3. MAIN FUNCTIONALITIES

3.1 Workflow Engine

The workflow engine is responsible for long-running tasks that are executed offline. It provides a REST API that allows triggering these tasks and monitoring their execution. Implementations for the following main tasks are included.

Metapath-based View Materialization. This is a core component for any metapath-based method for HIN exploration and analysis. It materializes a view of the HIN according to one or more specified metapaths. Such a view is essentially a set of edges, where an edge between two entities in the HIN is created if these entities are connected with one or more paths of the specified type. Consequently, this allows to define, for each entity, a set of *neighbors* that is *relative* to the given metapath. In particular, if the metapath under consideration is *cyclic*, i.e., the source and target nodes are of the same type, then the resulting view is a homogeneous network; otherwise, it is a bipartite graph, connecting entities of the source type to entities of the target type.

Index Construction. This component allows the creation of three types of indices. For each metapath-based materialized view, an inverted index is constructed on the set of relative neighbors of each entity for that metapath. Moreover, for spatial and temporal attributes, an R-tree and B+ tree are constructed, respectively. Creating these indices is necessary for performing similarity search over the respective metapaths or spatial and temporal attributes.

Entity Ranking. Once a homogeneous network has been constructed by materializing a view of the HIN based on a cyclic metapath, this component executes a random walk process to assign ranking scores to the nodes of this network. Specifically, we use the PageRank algorithm for this purpose. The result is, for each entity, a *relative* ranking score with respect to the metapath under consideration.

3.2 Query Engine

The query engine executes KNN queries and rank aggregation. Top-k queries are more intuitive in this setting, since it is not straightforward for a user to specify similarity thresholds, especially when the query involves multiple attributes of different types. Specifically, given a query entity, SPHINX can identify the top-k most similar entities in the HIN, according to a given set of criteria. These criteria can involve one or more metapaths, having as source type the type of the query entity, as well as spatial or temporal attributes, if applicable to the query entity. Moreover, different weights can be assigned to different criteria.

Formally, a query is a tuple $Q = \langle \mathcal{T}, \mathcal{C}, k \rangle$, where $\mathcal{T}$ is the type of entity to search for, $\mathcal{C} = \mathcal{C}_m \cup \mathcal{C}_s \cup \mathcal{C}_t$ are conditions over metapaths, spatial and temporal attributes, respectively, and k is the number of results to return. Each condition $C \in \mathcal{C}$ is a tuple of the form $C = \langle c, v, w \rangle$, where c denotes an attribute name, v specifies the desired value for that attribute, and $w \in (0, 1]$ is a weight parameter. For conditions on metapaths, c specifies a metapath starting from $\mathcal{T}$ and ending at another entity type $\mathcal{T}'$, while v is a set of entity ids of type $\mathcal{T}'$. For conditions on spatial or temporal attributes, c denotes the name of the attribute, and v is a spatial point or a timestamp, respectively.

Processing a query involves two components, namely KNN Search and Rank Aggregation, whose operations are interleaved, as described next.

KNN Search. Given a condition, this component executes a KNN query returning a ranked list of results. For conditions on metapaths, we execute top-k set similarity join [6], exploiting the respective inverted index that has been constructed earlier. This retrieves the nodes having the most similar sets of neighbors with respect to the given metapath. For spatial and temporal attributes, we execute KNN queries using the respective R-tree and B+ tree indices. A challenge arises from the fact that different similarity measures or distance functions are used for different types of conditions. Next, we explain how to obtain unified ranking scores.

For each condition C, we use a distance function d to measure the distance between the query value and the entity value with respect to that condition. Recall that, if $C \in \mathcal{C}_m$, these values are sets; hence, in that case, we choose d to be the Jaccard distance. If $C \in \mathcal{C}_s \cup \mathcal{C}_t$, we use as d the L^2 norm, which, for temporal attributes, corresponds to the absolute value of the difference between two timestamps, whereas, for spatial attributes, corresponds to the Euclidean distance of two points.

We then need a way to scale these distances in each condition, so that the results become comparable. To this end, given a query value v_q and the respective value v_e of an entity e, we define a *scaled* (i.e., relative) distance δ as:

$$\delta(v_q, v_e) = \frac{d(v_q, v_e)}{d_k} \tag{1}$$

where d_k is the distance of the k-nearest neighbor in that condition. Given these relative distances, we can now define a ranking score for each condition as:

$$score(v_q, v_e) = e^{-\lambda\delta(v_q, v_e)} \tag{2}$$

where λ is an exponential decay parameter.

Rank Aggregation. The KNN queries for each query condition can be executed in parallel, each one producing a ranked list of results. Each result is a tuple of the form $R = \langle id, score\rangle$, where id is the entity id and $score$ is calculated according to Equation 2. Notice that rank aggregation is also applied in the same manner for aggregating individual entity rankings that have been computed offline.

The goal of rank aggregation is to return a global top-k list of results, where each entity e is ranked according to an *aggregate score* over all conditions in the query Q as:

$$\gamma(Q, e) = \sum_{C \in \mathcal{C}} w_c \times score_c \tag{3}$$

where w_c is the weight assigned to condition C. This is performed using the Threshold Algorithm [2]. This algorithm scans each ranked list in parallel, performing two actions: (a) it computes the aggregate score of each *seen* item, and (b) it maintains a non-increasing upper bound on the score of *unseen* items. The search terminates once k items have been found with scores higher than the current unseen upper bound. These items constitute the global top-k results.

For entity ranking, the Threshold Algorithm can be applied directly on the individual ranked lists that have been computed already. However, for similarity search it requires two adaptations, since the individual ranked lists are not fully available but are instead constructed on the fly as a result of KNN queries. First, the rank aggregation component includes a mechanism for looking up the information about an entity using its id. This is required for computing the aggregate score of entities that have been seen only in a subset of the ranked lists. Second, it is possible that the contents of the ranked lists are exhausted before the search has produced k results. In that case, new KNN queries with larger values for k need to be issued.

Result Comparison. The top-k similar entities to a query depend on the specified conditions as well as the respective weight parameters. To facilitate and guide the user in selecting different query conditions and weight parameters, this component computes the differences between two result sets. Given two top-k lists L_i and L_j, we compute: (a) the Spearman's rank correlation coefficient, which provides an indicator of the overall agreement between the two rankings, and (b) for each individual result, a score $\Delta = rank_i - rank_j$, indicating the difference between its rankings in the two lists, or `null` if it does not appear in the other list.

4. USER INTERFACE

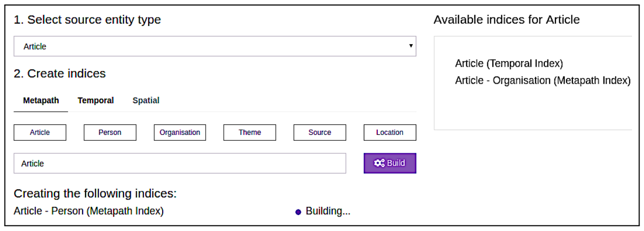
SPHINX provides a Web-based user interface that allows users to perform the functionalities described above and visualize the results. The UI includes the following pages.

Indexing. In this view (Fig. 3a), the user can select a type of entity, and then choose one or more metapaths starting from this entity type. For each selected metapath, a button allows to trigger the respective workflow for materializing the view and creating the inverted index. To bootstrap the metapath selection process, SPHINX automatically suggests certain metapaths. Specifically, given the schema of the HIN, it suggests, for each pair of entity types $\mathcal{T}$ and $\mathcal{T}'$: (a) the shortest path from $\mathcal{T}$ to $\mathcal{T}'$ and (b) the shortest cycle from $\mathcal{T}$ to $\mathcal{T}$ via $\mathcal{T}'$. The intuition for relying on shortest paths to suggest default metapaths lies on the observation that longer metapaths are typically less meaningful [5]. Finally, if the entity type under consideration also has spatial or temporal properties, the respective buttons are enabled for building spatial or temporal indices.

Similarity Search. In this view (Fig. 3b), the user can issue similarity search queries and view the results. For the selected entity type, the indexed metapaths, as well as any spatial or temporal properties, if applicable, are listed. The user can select which conditions to include as criteria in the search, and specify a value and a weight for each condition. The number k of results can also be determined. The retrieved results are displayed in a list, ordered by their score, and showing the attribute values in each one, which allows the user to inspect why each result matches the search.

Ranking. This view is very similar to that for search; the difference is that it does not include spatial/temporal conditions and query values. The user can select one or more metapaths, specify their weights, and retrieve a ranked list of entities according to these conditions.

Result Comparison. For each executed similarity search or ranking operation, an entry is added in the user's history for this session with a timestamp and the respective results. Then, the user can select two entries from the history and use the result comparison page (Fig. 3c) to compare the results. First, the Spearman's rank correlation coefficient between the two ranked lists is displayed. Then, a pie chart is presented, showing how many results appear in both lists, and, among those, for how many the ranking has increased,


1. Select source entity type

Article

2. Create indices

Metapath | Temporal | Spatial

Article | Person | Organisation | Theme | Source | Location

Article

Build

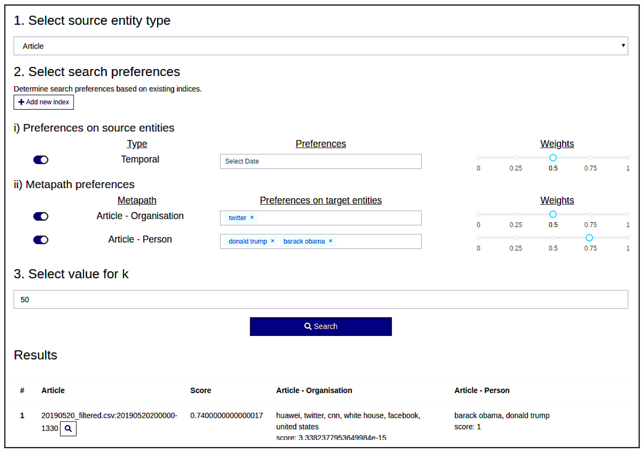
Creating the following indices:

Article - Person (Metapath Index) ● Building...

Available indices for Article

Article (Temporal Index)
Article - Organisation (Metapath Index)

(a) Index creation.


1. Select source entity type

Article

2. Select search preferences

Determine search preferences based on existing indices.

+ Add new index

i) Preferences on source entities

Type	Preferences	Weights
Temporal	Select Date	0.5

ii) Metapath preferences

Metapath	Preferences on target entities	Weights
Article - Organisation	twitter ×	0.5
Article - Person	donald trump × barack obama ×	0.75

3. Select value for k

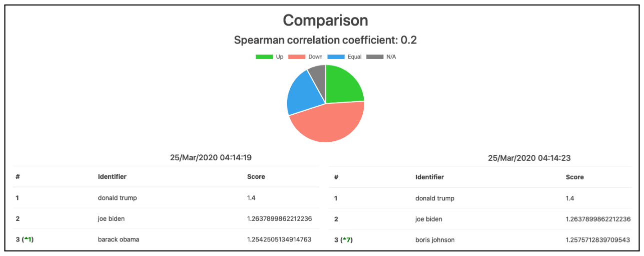
50

Search

Results

#	Article	Score	Article - Organisation	Article - Person
1	20190520_filtered.csv:20190520200000-1330	0.7400000000000017	huawei, twitter, cnn, white house, facebook, united states score: 3.3382377953649984e-15	barack obama, donald trump score: 1

(b) Similarity search.


Comparison

Spearman correlation coefficient: 0.2

25/Mar/2020 04:14:19

#	Identifier	Score
1	donald trump	1.4
2	joe biden	1.2637899862212236
3 (^1)	barack obama	1.2542505134914763

25/Mar/2020 04:14:23

#	Identifier	Score
1	donald trump	1.4
2	joe biden	1.2637899862212236
3 (^7)	boris johnson	1.2575712839709543

(c) Result comparison.

Figure 3: User Interface.

decreased or remained the same. Finally, for each entry in each list, a number is included indicating the difference for this result with respect to the other list. Thus, this view assists users in exploring how different conditions or weight parameters affect the results, and which entities are more stable or more sensitive to these factors.

5. DEMONSTRATION SCENARIO

To demonstrate SPHINX, we have prepared a scenario based on news articles and associated entities collected from the GDELT project[3]. Specifically, we use all articles from CNN and BBC during 2019. The resulting HIN contains the following entity types: `Articles` (71,422), `Persons` (105,261), `Organizations` (43,214), `Locations` (16,823) and `Themes` (9,230). Each article is associated with a timestamp and each location with geospatial coordinates.

First, we will present to the users the schema of this HIN and introduce them to the concept of metapaths, explaining how different metapaths express relationships with different semantics between the entities. We will guide them through the indexing page of SPHINX, showing how to select metapaths suggested by the system or specify custom ones in order to create and index different views of the HIN. Then, we will demonstrate the following default scenarios:

- *Ranking of persons.* We have preprocessed, indexed and ranked *persons* according to the metapaths `PAP`, `PAOAP` and `PALAP`. The users will be able to retrieve top-k persons according to any of these metapaths, or combinations of them with varying weights, and visually compare the results. For instance, top-3 persons for `PAOAP` include Donald Trump, Joe Biden and Nancy Pelosi, while for `PALAP`, Nancy Pelosi is replaced by Boris Johnson. This indicates that the centrality of Boris Johnson in the network increases, when persons are connected through articles mentioning the same location rather than the same organization.
- *Ranking of organizations.* We use the following metapaths: `OAO`, `OAPAO` and `OALAO`. As with persons, the users will be able to view and compare the ranked results for any combinations of these criteria. We can observe that the top organizations tend to be less sensitive to different metapaths or weights.
- *Article search.* We use the following metapaths: `AP`, `AO` and `AL`. The users will be able to specify their preferences on any of these, as well as on the *publication date*, and then search for top-k articles with these criteria. We can observe how the nearest neighbors of an article change with varying preferences.
- *Location search.* We use the following metapaths for *locations*: `LAP`, `LAO` and `LAT`. The users will be able to specify their preferences on any of these, as well as on *geocoordinates*. Then, they can search for top-k locations with these criteria, and observe how the nearest neighbors of a location change accordingly.

6. ACKNOWLEDGMENTS

This work was supported by the EU H2020 project SmartDataLake (825041).

7. REFERENCES

[1] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*, pages 135–144, 2017.

[2] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4):11:1–11:58, 2008.

[3] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu. Heterogeneous information network embedding for recommendation. *IEEE Trans. Knowl. Data Eng.*, 31(2):357–370, 2019.

[4] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu. A survey of heterogeneous information network analysis. *IEEE Trans. Knowl. Data Eng.*, 29(1):17–37, 2017.

[5] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11):992–1003, 2011.

[6] C. Xiao, W. Wang, X. Lin, and H. Shang. Top-k set similarity joins. In *ICDE*, pages 916–927, 2009.

[3]`https://www.gdeltproject.org/`