ECAI 2025
I. Lynce et al. (Eds.)

doi:10.3233/FAIA251113

Early Detection of Skills Demand with Deep Learning-Based Timeline Cover on Temporal Knowledge Graph

Giorgio Lazzarinetti[a,b,*], **Christian Scquizzato**[b], **Alessandro Mangone**[b] **and Jacopo Marcolini**[b]

[a]Università degli Studi Milano-Bicocca, Milano, Italy
[b]BID Company S.r.l., Milano, Italy, https://www.bidcompany.it/
ORCID (Giorgio Lazzarinetti): https://orcid.org/0000-0003-0326-8742, ORCID (Christian Scquizzato): https://orcid.org/0009-0005-8961-5427, ORCID (Alessandro Mangone): https://orcid.org/0000-0002-6406-536X, ORCID (Jacopo Marcolini): https://orcid.org/0000-0001-7312-7123

Abstract. Early identification of emerging skills is critical to align workforce development with labor market trends, however current skills forecasting and detection methods fall short since they rely heavily on historical data or depend on predefined taxonomies. In this paper, first, we propose to frame skill emergence as a combinatorial optimization problem over temporal networks. This problem, known as the Minimum Timeline Cover (MinTCover) problem, has the goal of identifying minimal activity intervals for entities, offering interpretable summaries of when they become relevant. Second, we design a system that combines real-time skills extraction from job posting and temporal knowledge graph construction using large language models, and we introduce a loss-enhanced version of a deep learning-based MinTCover solver to model skill emergence. Experiments show that on the one hand, our loss-based MinTCover solver improves state-of-the-art approaches achieving an average 1.17% reduction in total timeline span, and on the other hand, the overall pipeline achieves high performance in identifying emerging skills, outperforming other network-based clustering approaches, with better temporal alignment and fewer premature detections. Our method is the first to model early skills detection as a combinatorial problem and to operate in a fully unsupervised setting collecting data in real time, demonstrating its potential for dynamic labor market monitoring and proactive identification of emerging skills.

1 Introduction

In recent years, the rapid transformation of the labor market, driven by digitalization and automation, has made it necessary to fundamentally rethink the skills required of workers. According to the Future of Jobs Report 2025 by the World Economic Forum [27], by the end of 2025 nearly half of the global workforce will need reskilling, while many traditional roles will either disappear or be radically reshaped. In this context, it becomes crucial anticipating shifts in skill demands both for companies and for educational institutions. Timely identification of emerging competencies enables businesses to stay aligned with industry trends and helps bridge the gap between education and employment. This need is recognized, in fact, also by large-scale initiatives such as the European's project on skills anticipation and matching [3, 4], which emphasize evidence-based labor forecasting.

Existing methods for skill demand analysis fall into two main categories: forecasting established trends and detecting emerging skills. Traditional forecasting approaches typically rely on historical data, often using statistical models or deep learning (DL) techniques. For example, recurrent neural networks have been used to model temporal dynamics in job ads [5], while labor data and expert input have supported long-term planning strategies [12]. However, these methods are constrained by their dependence on extensive historical datasets and often fail to adapt to fast-paced technological changes. To cope with these drawbacks, network-based methods such as temporal skills clustering [22] have been proposed. This approach works directly on job postings and requires less historical data, identifying emerging trends by grouping skill mentions into temporal co-occurrence clusters and tracking their evolution over time. If, on the one hand, this method is effective in adapting to fast changes, on the other it relies on a set of predefined skills, not being suitable for automatic early skills detection. Approaches for detecting emerging skills automatically aim to capture nascent trends but often suffer from fragmented and inconsistent data sources. Some strategies rely on predicting emerging skills defined as previously low-demand skills experiencing a surge in hiring demand [28], others integrate natural language processing and skill taxonomies to link curricula with market trends by combining data from job advertisements, course descriptions, and resumes to enable a fine-grained analysis of skill gaps [17], or use machine learning (ML) for career guidance based on skill profiles [23]. Despite their promise, these methods struggle with scalability and robustness across domains.

To address these limitations, we propose to model the early skills detection problem as an instance of the *Minimum Timeline Cover* (MinTCover) problem [20, 10, 7, 6, 8, 15, 16], a recently introduced combinatorial optimization task on temporal graphs. MinTCover aims to extract compact and interpretable timelines by identifying the minimal activity intervals necessary to cover all interactions in a temporal network. Although the problem is NP-hard, efficient sub-optimal solutions have been proposed, including a DL-based model that offers promising results despite relying on heuristic

* Corresponding Author. Email: giorgio.lazzarinetti@bidcompny.it.

post-processing that does not ensure true minimality [16]. To frame early skills detection as a MinTCover task, we leverage knowledge graphs (KGs), which offer a flexible framework for modeling dynamic relationships and have already shown potential in domains like resume parsing and skill matching [11, 24], as well as dynamic network analysis [18, 2]. Thus, we put forward a system that extracts skills and temporal relationships from job posting with large language models (LLMs), resulting in a dynamic data-driven graph structure where each node represents a skill and edges represent co-occurrences in job postings over time, providing a principled foundation for detecting emerging skills with high temporal resolution and without relying on predefined taxonomies. Contrary to clustering-based or taxonomy-dependent methods, our method requires neither predefined skill sets nor historical frequency thresholds, enabling detection of previously unseen competencies in real time.

In this setting, our contributions are twofold. First, we apply MinTCover to early skills detection - an area where existing methods struggle with both scalability and effectiveness - by mapping emerging skills to activity intervals in temporal KGs. This enables the identification of in-demand competencies ahead of traditional forecasting models. Notably, our system successfully identifies novel and valid emerging skills not yet included in formal taxonomies, as confirmed by expert validation. Second, we propose an algorithmic improvement over existing MinTCover solvers by incorporating a loss-based refinement step, ensuring true minimality and improving both predictive accuracy and summarization efficiency with respect to state-of-the-art approaches over a well established test dataset.

The remainder of the paper is structured as follows: Section 2 introduces the MinTCover theoretical framework; Section 3 details the KG construction approach and loss-based optimization; Section 4 presents our experiments and evaluation metrics; and Section 5 concludes with final remarks and future directions.

2 Problem Formulation

2.1 Minimum Timeline Cover Problem

Let $G = (V, E)$ be an unweighted undirected *temporal graph*, where V is the set of *vertices* and E the set of *temporal edges* (u, v, t), with $u, v \in V$ vertices and $t \in T$ the *timestamp* indicating the time that an interaction between vertices u and v takes place. We denote as as $deg(u) = \sum_{t=1}^{T} degree((u, t))$ the *global degree* of a vertex u, i.e. the number of temporal edges incident in u in the overall time domain and as *density* the ratio of the number of edges $|E|$ with respect to the maximum possible edges: $d = \frac{2|E|}{|V|(|V|-1)|T|}$.

Given two numbers s_u, e_u, with $s_u \leq e_u$ we define $I_u = [s_u, e_u]$ as the *activity interval* of vertex u and $\mathcal{T} = \{I_u\}_{u \in V}$ as an *activity timeline* of G. Given an interval $I_u = [s_u, e_u]$, $\delta(I_u) = e_u - s_u$ is the *span* of interval I_u.

Definition 1 (Timeline Cover)**.** *Given a temporal graph $G = (V, E)$ and an activity timeline $\mathcal{T} = \{I_u\}_{u \in V}$, we say that $\mathcal{T}$ **covers G** if $\forall (u, v, t) \in E,\ t \in I_u$ or $t \in I_v$.*

The goal is to find a timeline that has the most compact intervals possible according to the *sum-span* of a timeline $\mathcal{T}$, $S(\mathcal{T}) = \sum_{u \in V} \delta(I_u)$.

Problem 1 (MinTCover)**.** *Given a temporal graph $G = (V, E)$, find a timeline $\mathcal{T} = \{I_u\}_{u \in V}$ that covers G and minimizes the sum-span $S(\mathcal{T})$.*

This problem is NP-Hard [10] and, when considering more than one time interval, not even approximable within any constant factor (deciding whether there exists a solution of span 0 is indeed an NP-complete problem) [20, 10, 7, 6]. Although no scalable exact solutions are available in the literature, some approximate [20, 8], heuristic [15] and DL-based [16] solutions have been proposed. Among these, the most effective approach is proposed in [16], where the authors suggests to train a DL model based on graph neural network (GNN), transformers and pointer network (PN) to improve solution quality with respect to other approximations and heuristics based solutions, against a little increase in terms of execution time.

2.2 Skills Demand Early Detection

The MinTCover problem provides a natural framework for detecting emerging skills in temporal KGs derived from job postings. To understand this connection, the fundamental idea is to represent each skill as a vertex of a graph where temporal edges denote co-occurrences within job descriptions over time. The goal is to assign each skill an activity interval $I_u = [s_u, e_u]$, such that for every edge (u, v, t), the timestamp t lies within I_u or I_v. By minimizing the total length of these intervals, MinTCover reveals when a skill becomes essential and when it fades into irrelevance.

On the one hand, this approach distinguishes true skill emergence from sporadic appearances: a skill becomes active only if its presence is required to cover interactions that no other skill can. Rare mentions are ignored unless indispensable or until they become relevant, allowing the framework to identify meaningful transitions in labor demand. From the other, it captures both emergence and decline: a skill is activated when it becomes necessary to cover new co-occurrences and deactivated when others suffice to cover its former role. Thus, it implicitly models demand frequency - frequently required skills will have longer intervals, while marginal ones may never be activated or will quickly fade.

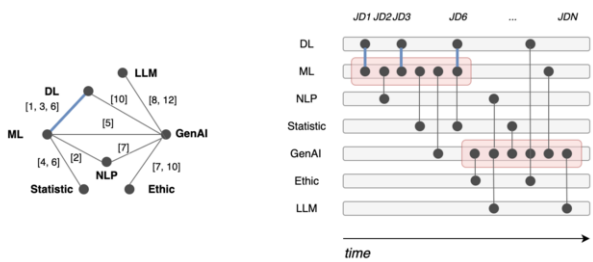

Figure 1. A static co-occurrence graph (left) representing fixed relationships among entities and a temporal framework, emphasizing how activity timelines capture dynamic interactions over time. Edges in the left static KG correspond to co-occurrences at specific timestamps: consider, as an example, the edges highlighted in blue in job descriptions (JD) 1, 3 and 6, which correspond to the edge between vertices ML and DL with the corresponding vector of labels. On the right, these edges are activated across time, and the MinTCover algorithm selects minimal intervals during which each skill is indispensable. Red timelines illustrate the MinTCover concept: identifying minimal intervals summarizing key temporal patterns.

Figure 1 illustrates this process: on the left, a co-occurrence graph depicts skills as vertices, with edges representing their co-appearance in job postings over time: every time a new job posting is processed new skills and co-appearances are added as new vertices and new edges to the co-occurrence graph. Each edge is labeled with the array of timestamps of their appearance in the job posting: thus, when parsing a job posting if an edge is already present the array of times-

tamps used as label is enriched with the new timestamp. On the right, a temporal network model visualizes these temporal interactions, showing at each timestamp the vertices and edges extracted from the corresponding job posting. This highlights how timelines of entity and time-interval pairs can provide rich insights into significant events. For instance, the launch of ChatGPT in November 2022 rapidly transformed the workflows of data scientists in ML projects, prompting companies to seek professionals with expertise in Generative AI. In Figure 1 these timelines highlighted in red capture these pivotal transitions, i.e. the initial focus on traditional ML skills and the subsequent demand for Generative AI expertise, underscoring the central role of skills in this context. This approach to mapping event-driven timelines is fundamental to addressing the MinTCover problem.

Overall, MinTCover is a robust and formally grounded approach for detecting new skills early enough to warrant strategic response. Indeed, it offers a principled approach to identifying skills that genuinely impact the labor market, enabling timely interventions such as curriculum updates, upskilling, or strategic planning.

3 Methodology

To leverage the MinTCover problem for early skill detection, we first construct a temporal KG from job postings: each job posting is parsed into a local KG annotated with its timestamp. These KGs are aggregated over time by stacking them into a unified temporal structure, where edges retain the timestamp of their first appearance.

Building KGs from unstructured text is a challenging task, often tackled using various DL methods [30]. Recent advances in LLMs like GPT-4o [29, 9], combined with prompt engineering [21], allow effective KG extraction without fine-tuning. We follow the methodology proposed in [14], adapting LLM prompts to extract skills and relationships following the classical ML-based text-to-KG pipeline[1].

3.1 Knowledge Graph Construction

To build the KG, we adopt a prompt-engineering framework that combines both standard practices and different prompting strategies, like zero-shot [25], few-shot [1] and chain-of-thought [26]. The framework breaks down the process into four essential elements: *Role*, *Instructions*, *Steps*, and *Expectations*.

The LLM is assigned the task of converting job descriptions into KGs, where nodes represent normalized technical skills, edges connect skills co-mentioned within the same sentence or paragraph and date records the posting's publication date. These aspects are specified, together with some examples, in the Role and Instructions sections. In building the KG we only focus on hard technical skills. The Steps section breaks down the task into manageable steps in a chain-of-thought fashion, ensuring a logical progression. Here we ask the LLM to follow the classical steps of the text-to-KG pipeline, first extracting all the skills from the job posting normalizing them considering synonyms and linguistic variations, then creating a list of all the nodes and linking nodes with edges for skills that appear in the same sentence, and finally refining the graph checking for consistency. We link two nodes only if they appear in the same sentence. Indeed, an analysis of real job description showed that, in general, skills that are related appear together in the same sentence. Over time, as new technologies or skills emerge, they will initially appear alongside older ones in job descriptions. Eventually, the older skills will fade away, leaving only the new ones and other related competencies. By linking nodes this way, we create a KG that effectively captures these transitions, making it well-suited for early skill detection within the MinTCover framework. In the final Expectations section we define the ultimate objective of the prompt, guiding the LLM's focus toward a specific outcome structure (a dictionary-like structure with list of nodes, list of edges and timestamps) also providing an example to further instruct the LLM on the end goal.

[1] This includes entity classification, relationship extraction, entity disambiguation and linking, knowledge integration and knowledge refinement.

Knowledge Graph Construction Prompt

```
ROLE: You are an algorithm that
must convert job descriptions into a
knowledge graph.
INSTRUCTIONS: Create a knowledge graph
in which:
- Nodes: Each node represents a
unique skill extracted from the job
description. Focus on skills tied to
specific activities or technologies
(e.g., map "experience with Docker" to
"Docker").
- Edges: An edge exists if the
corresponding skills are mentioned
together in the same sentence or
paragraph.
Focus only on hard technical skills.
STEPS: To build the knowledge graph
follow these steps:
1) Extract the skills: Analyze the job
description and extract all technical
skills, normalizing for synonyms and
variations.
2) Create the nodes: Generate a list of
unique skills resolving ambiguities.
3) Create the edges: Link every pair of
skills mentioned in the same sentence or
paragraph.
4) Create the knowledge graph: Integrate
the extracted skills and edges into
a cohesive knowledge graph structure
eventually refining it by checking for
consistency.
EXPECTATIONS: Provide the resulting
knowledge graph in a dictionary-like
structure including:
- List of nodes: unique identifiers for
each skill.
- List of edges: array of skill pairs.
- Date: publication date (DD-MM-YYYY).
Example:
Input: "A pharmaceutical company is
looking for a Data Scientist with
experience in Python and Machine
Learning. Published on: 2023-10-15".
Output: { "nodes": [ "Data Science",
"Python", "Machine Learning"],
"edges": [ ["Data Science", "Python"],
["Data Science", "Machine Learning"],
["Python", "Machine Learning"]], "date":
"15-10-2023" }.
```

This prompt structure ensures the generation of KGs with consistent formatting and temporal metadata. Once converted, job postings are chronologically ordered and aligned by node sets to create a complete temporal KG for each specific role, enabling the application of MinTCover for fine-grained skill trend analysis.

3.2 Heuristic Optimization

To compute skill timelines, we start from DLMINTC+, the DL-based approach proposed in [16], which leverage DL frameworks to properly represent node sequences and select activity intervals. This solver operates in five stages. First, each node is characterized via temporal degree-based features to capture its local structure at each timestamp. Then, the temporal graph is embedded, encoding node features over time with GNN. These sequences are aggregated to model the full temporal evolution of each node and processed with a Transformer. A pointer mechanism then selects activity intervals for each node based on the learned temporal representations. Finally, to ensure complete coverage, since the selected intervals may not form a valid timeline cover, an iterative post-processing step ensures full edge coverage by refining the output intervals accordingly, thus covering uncovered edges while minimizing the overall sum-span $S(\mathcal{T}) = \sum_{u \in V}(e_u - s_u)$. Given a temporal graph $G = (V, E)$ and intervals $\mathcal{T}$, the algorithm proceeds as follows:

1. Initialize: Identify uncovered edges $U = \{(u, v, t) \in E \mid t \notin I_u \wedge t \notin I_v\}$.
2. Expand Intervals: For each $(u, v, t) \in U$, extend either I_u or I_v with minimal span increase to include t.
3. Iterate: Repeat until all edges are covered.

This yields a refined set $\mathcal{T}'$ that guarantees coverage of all temporal edges and local minimality - each expansion step is minimal at the time of application. However, global minimality is not guaranteed. To improve this, we introduce an enhanced version in Algorithm 1, incorporating a loss-based reduction step, similar to the one proposed in [15]. This phase identifies and removes timestamps that do not contribute to edge coverage, producing more compact and efficient timelines.

The algorithm consists of 2 phases and begins with the initial intervals produced by the Minimal Timeline Completion algorithm proposed in [16] for each node. In a first phase we compute a loss value (which corresponds to the number of covered edges that would become uncovered after reducing the interval of a node) for each node–timestamp pair which are at the boundary of the timeline (thus, only initial and final node-timestamp pair for each timeline), reflecting whether that exact pair is solely responsible for covering any edge. Then, in the reduction phase, all timestamps with zero loss (i.e., not uniquely covering any edge) are removed. After each reduction the loss is recomputed for all nodes.

3.2.1 Minimality Considerations

Algorithm 1 yields a final set of intervals that both covers all edges and is *minimal*, i.e. no further shrinking is possible without losing coverage, as proved in Theorem 1.

Theorem 1. *Let $\mathcal{T}$ be the output of Algorithm 1 on the temporal graph $G = (V, E)$, with initial timeline $\mathcal{T}'$ computed by the Minimal Timeline Completion Algorithm. Then $\mathcal{T}$ is a minimal timeline cover of G according to Problem 1; that is, no further contraction of any interval can be performed without uncovering at least one edge.*

Algorithm 1: Minimal Timeline Completion with Loss-Based Reduction

Input: $G = (V, E)$, $\mathcal{T}' = \{I_u = [s_u, e_u]\}_{u \in V}$ (from Minimum Timeline Completion)
Output: Adjusted intervals $\mathcal{T}$ covering E and reduced to minimal

```
Phase 1: Loss Computation
    Initialize loss((v,t)) ← 0 for all (v,t) where
     t ∈ [s_v, e_v].
    foreach (u,v,t) ∈ E do
        // Check coverage at t by u and v
        covers_u ← I(t ∈ [s_u, e_u]);
         covers_v ← I(t ∈ [s_v, e_v]);
        // if exactly one endpoint covers t
        if covers_u + covers_v = 1 then
            if covers_u = 1 then
                loss((u,t)) ← loss((u,t)) + 1;
            end
            else
                loss((v,t)) ← loss((v,t)) + 1;
            end
        end
    end
Phase 2: Final Reduction (Shrinking)
    foreach v ∈ V do
        for t = s_v to e_v do
            if loss((v,t)) = 0 then
                // Removing t does not uncover
                   any edge exclusively covered
                   by (v,t)
                Remove t from [s_v, e_v] and recompute loss
                 for other nodes;
            end
        end
    end
    return T;
```

To provide a demonstration to Theorem 1 we split the proof into three parts. Since Algorithm 1 starts with the initial completion for which it has been shown the output is a valid coverage, we firstly show that coverage is preserved after reduction, then that loss value of any vertex in the timeline $\mathcal{T}$ does not decrease and finally that the timeline $\mathcal{T}$ returned after the shrinking phase is minimal.

Proof. (1) Let us denote by $\mathcal{T}^i$ the timeline at the i-th iteration of the shrinking phase and by $\mathcal{T}^0 = \mathcal{T}'$ the input timeline. By construction, $\mathcal{T}^0$ is a valid cover. We show by induction that if $\mathcal{T}^i$ is a timeline cover, then $\mathcal{T}^{i+1}$ is also a cover. Since $\mathcal{T}^0$ covers E (by the end of the extending phase), it follows that every subsequent $\mathcal{T}^i$ remains a cover. Indeed, if in the i-th iteration no vertex–timestamp pair is removed, then $\mathcal{T}^{i+1} = \mathcal{T}^i$, so coverage clearly remains intact. Otherwise, if in the i-th iteration a pair (v, t) with $loss((v, t)) = 0$ is removed, it means (v, t) was not uniquely covering any edge. Formally, for every edge $(u, v, t) \in E$, there exists another vertex–timestamp pair (u, t) such that $t \in I_u$ (or equivalently $t \in I_v$ but with positive loss). Hence its removal does not uncover any edge, ensuring that $\mathcal{T}^{i+1}$ also remains a valid cover. Therefore, by induction, $\mathcal{T}^i$ is a cover for all i, in particular $\mathcal{T}^*$ at the final iteration.

(2) Next, we show that the $loss$ value for each remaining vertex–timestamp pair (v, t) never decreases during the shrinking phase.

- If $loss((v,t)) > 0$ at iteration i, then (v,t) is not removed at the i-th iteration. As a result, its $loss$ value cannot decrease; it either remains the same or may increase if other pairs covering the same edges are removed.
- If $loss((v,t)) = 0$ at iteration i, then (v,t) is removed in the i-th iteration. Once removed, we do not track its loss value any further.
- Furthermore, when a different pair (u,t') is removed, any edge that (u,t') was covering might now rely solely on (v,t) (if it has the same timestamp t), which could increase $loss((v,t))$ if (v,t) remains in $\mathcal{T}$.

Hence, no existing pair (v,t) has its $loss$ value decreased over the course of the shrinking phase.

(3) Finally, suppose for contradiction that $\mathcal{T}^*$ is not minimal, i.e., there exists a pair $(v,t) \in \mathcal{T}^*$ such that removing it still leaves $\mathcal{T}^*$ a valid cover. By definition, this implies $loss((v,t)) = 0$, since (v,t) does not uniquely cover any edge. However, from part (2), once a pair (v,t) has positive loss, it stays positive; conversely, if it has $loss((v,t)) = 0$ at iteration i, it is removed at that same iteration. Thus, no vertex–timestamp pair with zero loss can remain in $\mathcal{T}$ at the end. Therefore, (v,t) cannot exist in $\mathcal{T}^*$ with $loss((v,t)) = 0$ and we reach a contradiction. This contradiction shows that every pair (v,t) in $\mathcal{T}^*$ must be essential for coverage, confirming that $\mathcal{T}^*$ is minimal.

□

3.2.2 Complexity Considerations

Let $n = |V|$ and $m = |E|$. Phase 1 performs $loss$ computation which depends on the number of vertices $(v,t) \in \mathcal{T}$. Since we add, for each vertex $v \in V$ exactly two timestamps (the start and end of the activity interval, namely, s_v, e_v), the dimension of the computed activity timeline is exactly $|\mathcal{T}| = 2n$. It follows that the complexity for the initialization of the loss is $O(n)$. Phase 2 is the shrinking phase, whose complexity depends on the number of updates performed over the $loss$ values. Since each vertex is updated at most once for each edge incident in it, the total number of possible updates is bounded to the sum of the global degrees of each vertex, i.e., $\sum_{v \in V} deg(v) = 2m$, thus the complexity of the shrinking phase is $O(m)$. Therefore, the overall complexity of the algorithm is $O(m+n)$.

In summary, Algorithm 1 not only ensures coverage of all temporal edges but also prunes away superfluous timestamps through loss, guaranteeing that the final solution is minimal with respect to interval contraction.

4 Experimental Evaluation

4.1 Performance Metrics

We evaluate our approach through two analyses. First, we assess the impact of the loss-based refinement in the Minimal Timeline Completion algorithm, comparing the total sum-span $S(\mathcal{T})$ produced by the baseline DLMINTC+ approach [16] and our enhanced version with loss-based reduction, using the DIMACS benchmark graphs [19], which encompasses diverse network structures. Both methods share identical architectures, hyperparameters, and training setups to ensure a fair comparison based on sum-span minimization. Second, to evaluate early skill detection capabilities, we crafted a dataset of job postings and related skills which has been published for reproducibility purpose [13]. In this dataset we compare predicted activity intervals with ground truth annotations over a dataset of job postings spanning 2000–2024. This dataset includes 250 postings (on average 10 per year across five sectors: Banking, Insurance, Consultancy, Pharma, Retail), focused on the Data Science role [2]. We use this dataset both to evaluate the quality of skill extraction, matching the extracted skills against those detected by expert analyzing the set of job postings, and to compare the results of our MinTCover-based approach for early skills detection with the network-based method that uses temporal skill clustering [22]. We compare the results with this approach since in the literature is the only one that completely relies on job posting as source of skills detection and do not need a lot of historical data to predict the trends, even if it can perform forecasting only for a predefined set of skills, thus not being completely unsupervised. For the latter case, we adopt standard precision (P) and recall (R) metrics as in Equations 1 and 2,

$$\mathrm{P} = \frac{|\text{Extracted} \cap \text{Ground Truth}|}{|\text{Extracted}|} \tag{1}$$

$$\mathrm{R} = \frac{|\text{Extracted} \cap \text{Ground Truth}|}{|\text{Ground Truth}|} \tag{2}$$

while for the former we use Intersection over Union (IoU) as in Equation 3, which measures the overlap between predicted and true active intervals for each skill u, capturing both correct detection and false alarms,

$$\mathrm{IoU}(u) \;=\; \frac{\left| I_{\mathrm{real}}(u) \;\cap\; I_{\mathrm{pred}}(u) \right|}{\left| I_{\mathrm{real}}(u) \;\cup\; I_{\mathrm{pred}}(u) \right|} \tag{3}$$

and Detection Delay (Δ) as in Equation 4, which captures how early or late the algorithm detects a skill relative to its actual onset,

$$\Delta(u) \;=\; s_{\mathrm{pred}}(u) \;-\; s_{\mathrm{real}}(u). \tag{4}$$

These metrics jointly assess detection accuracy, timeliness, and the model's ability to avoid premature or delayed activation of skills.

4.2 Experimental Results

Firstly, we evaluate the efficiency of our method by measuring the reduction in total sum-span $S(\mathcal{T})$ achieved by our loss-optimized refinement over the baseline DLMINTC+ solver. Table 1 summarizes the results across a diverse collection of temporal graphs, reporting for each graph its density D and the absolute reduction ΔS in sum-span. The full dataset includes graphs with a number of nodes ranging from 24 to 89,269 (mean: 6,800), edges from 554 to over 18 million (mean: 1.2 million), and temporal spans from 1 to over 736,000 timestamps (mean: 40,000). In general, higher-density graphs tend to exhibit fewer timestamps and represent highly interactive environments over short durations (e.g., animal behavior or sensor contact), whereas lower-density graphs typically span longer temporal windows with sparser interactions (e.g., online or communication networks). Our method consistently improves over the baseline, with identical performance observed in only two cases. On average, it reduces the total sum-span by 1.17%, with the largest absolute gains found in high-volume datasets such as *fb-wosn-friends* and *ia-digg-reply*, where even small percentage improvements translate into substantial temporal compression.

[2] Ground truth intervals were annotated by three domain experts, marking when each skill was genuinely in demand. We only focus on hard technical skills and we computed inter-annotator agreement for all the skills extracted observing high consistency across annotators. When disagreement occurred, a consensus process was used to define the final reference intervals.

Graph	d	ΔS
aves-sparrow-social	3.89×10^{-1}	0
aves-wildbird-network	1.17×10^{-1}	1.80×10^{1}
copresence-InVS13	4.39×10^{-3}	2.00×10^{4}
copresence-InVS15	2.50×10^{-3}	1.00×10^{4}
copresence-LH10	4.53×10^{-3}	7.00×10^{3}
copresence-LyonSchool	7.24×10^{-2}	3.00×10^{3}
copresence-SFHH	5.56×10^{-3}	5.00×10^{3}
copresence-Thiers13	3.88×10^{-2}	2.00×10^{4}
email-dnc	1.13×10^{-6}	1.00×10^{3}
fb-wosn-friends	8.49×10^{-10}	2.00×10^{6}
ia-contacts-dublin	8.98×10^{-8}	9.00×10^{3}
ia-contacts-hypertext2009	6.27×10^{-4}	5.00×10^{3}
ia-digg-reply	2.26×10^{-9}	4.00×10^{5}
ia-hospital-ward-proximity	1.24×10^{-3}	6.00×10^{3}
ia-primary-school-proximity	1.39×10^{-3}	3.00×10^{3}
ia-prosper-loans	6.77×10^{-7}	4.00×10^{3}
ia-retweet-pol	5.93×10^{-9}	3.00×10^{4}
insecta-ant-colony1	4.41×10^{-1}	4.00×10^{1}
insecta-ant-colony3	4.74×10^{-1}	5.00×10^{1}
insecta-ant-colony5	4.23×10^{-1}	1.10×10^{2}
mammalia-raccoon-proximity	1.42×10^{-1}	7.00×10^{0}
rec-amz-Baby	1.06×10^{-9}	5.00×10^{4}
reptilia-tortoise-network-bsv	2.01×10^{-2}	1.20×10^{0}
reptilia-tortoise-network-fi	6.92×10^{-4}	0
SFHH-conf-sensor	2.47×10^{-4}	9.00×10^{3}

Table 1. Comparison of temporal coverage reduction (ΔS) across various temporal graphs. ΔS represents the difference in total sum-span between the baseline DLMINTC+ solution and our loss-optimized refinement. Results are presented across graphs with varying density d, computed as the ratio of observed to possible temporal edges. Densely connected graphs tend to span shorter time windows with fewer timestamps (e.g., sensor or proximity networks), while sparse graphs often cover broader temporal ranges (e.g., online interaction or communication networks). Overall, our refinement achieves consistent reductions in total sum-span across both dense and sparse graphs, with the most substantial absolute improvements observed in large, low-density networks—where timeline compression has the highest impact.

Secondly, we measure our LLM-based approach's skills extraction ability measuring Precision and Recall comparing the set of extracted skills against the ground truth generated by domain experts [3]. Each skill in the ground truth was also categorized as either *established* or *emerging*, based on its historical presence in curricula, policy documents, and prior job postings. A skill was considered matched if it appeared in both the extracted and ground-truth sets. As shown in Table 2, our method achieves high performance on both categories. For established skills, we observe a precision of 0.97 and recall of 0.93, indicating strong consistency with well-known competencies. For emerging skills, despite their inherently more variable representation, the model achieves a recall of 0.96, confirming its suitability for early trend detection. The lower precision (0.64) in this category reflects the presence of valid but not yet annotated skills, further highlighting the discovery potential of our approach. A manual inspection confirmed that the majority of these, indeed, correspond to valid and emerging competencies not previously annotated such as *GenAIOps*, *Quantistic AI*, and *Causal AI*, which were absent from historical taxonomies, suggesting that the measured precision underestimates the system's true accuracy. This highlights the ability of the extraction component not only to preserve existing knowledge, but also to adapt to rapidly evolving labor market trends.

[3] Consider that on a standard machine (4 CPU, 16 GB RAM), the full pipeline processes 100 job postings in under 8 minutes, with 70% of the time spent on LLM inference. This step is highly parallelizable and can be cached for efficiency.

SkillType	GroundTruth	Extracted	Matched	P	R
Established	60	58	56	0.97	0.93
Emerging	24	36	23	0.64	0.96

Table 2. Precision and recall of the extracted skills, broken down by type. A skill is considered matched if it appears in both the extracted set and the ground truth. Manual inspection confirmed that several unmatched skills—such as GenAIOps and Quantistic AI—were indeed valid emerging skills not previously annotated, suggesting that the measured precision underestimates the system's true accuracy.

Finally, we assess early skill detection performance using the IoU score and detection delay Δ. Figure 2 shows the distribution of IoU scores between the predicted skill intervals and the expert-annotated ground truth. For the network-based clustering approach, since it requires to define skills in advanced, we perform the analysis for the set of skills extracted by the LLM pipeline, thus being able to exactly compare the performance on each skill. Our method achieves a higher mean IoU of 0.697, compared to 0.622 for the baseline. While the baseline distribution is flatter and skewed toward mid-range values, our approach displays a pronounced concentration in the upper range indicating that it is not only more precise in identifying when skills are active, but also significantly more consistent across different skill categories.

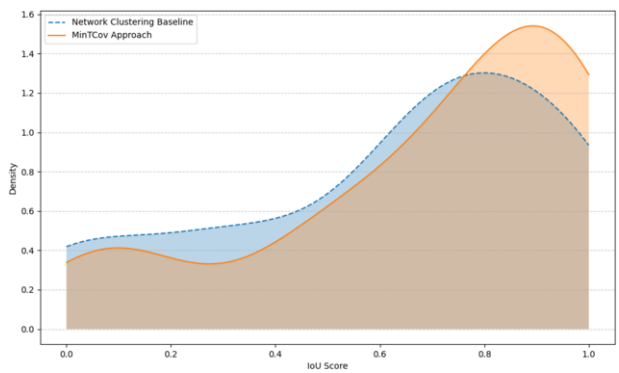

Figure 2. Comparison of IoU distributions for predicted vs. ground truth skill intervals for our MinTCover-based approach and the baseline clustering-based approach.

Figure 3 reports the Δ distribution in months. While the baseline achieves a slightly lower mean delay of 1.0 months, our method records a comparable average of 1.41 months. However, a closer look at the distribution reveals key differences: our approach exhibits a sharper concentration around zero, indicating more accurate timing overall, whereas the baseline shows a flatter spread with more frequent deviations. In particular, our method achieves a better balance between early and late detections, avoiding both premature activations and excessive lags. This behavior reflects the design of our temporal model, which favors minimal, context-aware activation windows over generalized, cluster-driven trends. As a result, the proposed method not only predicts when a skill becomes relevant with greater precision, but also provides a temporally faithful representation of its emergence - crucial for actionable forecasting and timely policy intervention.

As seen, even if effective, one of the main drawbacks of the network-based clustering baseline is that it relies on a predefined list of skills and, thus, merely observes the evolution of known competencies. Our method, instead, operates in a fully unsupervised setting. Skill ex-

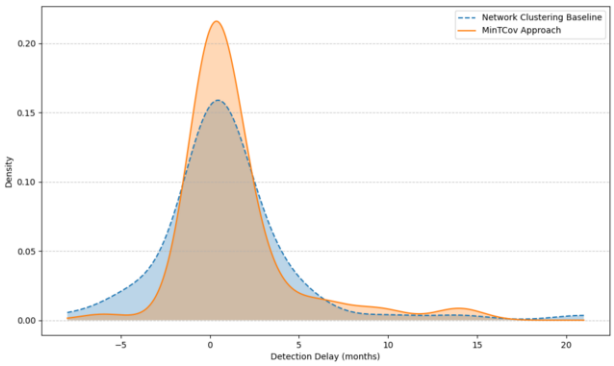

Figure 3. Comparison of detection delay Δ in months for our MinTCover-based approach and the baseline clustering-based approach.

traction is performed dynamically using LLMs, allowing the system to identify and track emerging skills in real time, even those not previously annotated or catalogued. Nonetheless, our skills extraction pipeline is modular and could be applied on top of the network-based clustering framework to enhance its responsiveness, making it actually suitable for early skills detection.

5 Conclusion and Future Works

In this work, we pursued a dual objective: advancing the state of the art in solving the MinTCover problem, and leveraging this improvement for the critical task of early skill detection in the labor market.

To this end, we proposed a novel end-to-end framework that combines dynamic skill extraction through LLMs, temporal KG construction, and an enhanced MinTCover solver. This architecture allows for fully unsupervised and real-time tracking of emerging skills, overcoming limitations of existing approaches that either rely on predefined taxonomies or require extensive historical datasets. Our method is among the first to enable precise, scalable early detection of in-demand competencies directly from job postings.

Experimental results demonstrate the strength of our solution on both fronts. On the optimization side, the proposed loss-based refinement step leads to a 1.17% average reduction in total sum-span over diverse benchmark graphs - without compromising scalability - confirming the effectiveness of the proposed minimality guarantee. On the application side, our approach outperforms a network-based clustering baseline in early skills detection, achieving a higher mean IoU (0.697 vs. 0.622) and a more temporally centered detection delay, with fewer extreme deviations.

Moreover, our skill extraction pipeline shows high precision and recall even for emerging skills, where variability and ambiguity are inherently higher. These results validate the potential of combining temporal reasoning and real-time LLM-based extraction for next-generation labor market intelligence systems. Although our experiments focused on the Data Science domain, the framework is domain-agnostic and can be adapted to other sectors, provided sufficient job postings are available.

Future work will focus on benchmarking different KG construction strategies and prompt design choices to assess their impact on downstream forecasting performance. Additionally, we plan to extend the framework to incorporate alternative data sources (e.g., patents, course platforms, and social forums) and move toward a probabilistic formulation of skill relevance to better model gradual adoption and decay over time. Moreover, while the current implementation assumes undirected and unlabeled co-occurrence graphs, future work will explore extending the approach to support edge directionality and semantic types to leverage the full expressiveness of KGs.

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[2] L. Cai, X. Mao, Y. Zhou, Z. Long, C. Wu, and M. Lan. A survey on temporal knowledge graph: Representation learning and applications. *CoRR*, abs/2403.04782, 2024. doi: 10.48550/ARXIV.2403.04782. URL https://doi.org/10.48550/arXiv.2403.04782.

[3] Cedefop. Cedefop - skills anticipation and matching project, 2017. URL https://www.cedefop.europa.eu/en/projects/assisting-eu-countries-skills-matching. Accessed: March 17, 2025.

[4] Cedefop. Skills anticipation: looking to the future. Briefing Note 9124 EN, European Centre for the Development of Vocational Training (Cedefop), 2017. URL https://www.cedefop.europa.eu/en/publications/9124#group-downloads.

[5] M. M. G. de Macedo, W. Clarke, E. Lucherini, T. Baldwin, D. Q. Neto, R. A. de Paula, and S. Das. Practical skills demand forecasting via representation learning of temporal dynamics. In V. Conitzer, J. Tasioulas, M. Scheutz, R. Calo, M. Mara, and A. Zimmermann, editors, *AIES '22: AAAI/ACM Conference on AI, Ethics, and Society, Oxford, United Kingdom, May 19 - 21, 2021*, pages 285–294. ACM, 2022. doi: 10.1145/3514094.3534183. URL https://doi.org/10.1145/3514094.3534183.

[6] R. Dondi. Untangling temporal graphs of bounded degree. *Theor. Comput. Sci.*, 969:114040, 2023. doi: 10.1016/J.TCS.2023.114040. URL https://doi.org/10.1016/j.tcs.2023.114040.

[7] R. Dondi and M. Lafond. An FPT algorithm for temporal graph untangling. In N. Misra and M. Wahlström, editors, *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, September 6-8, 2023, Amsterdam, The Netherlands*, volume 285 of *LIPIcs*, pages 12:1–12:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi: 10.4230/LIPICS.IPEC.2023.12. URL https://doi.org/10.4230/LIPIcs.IPEC.2023.12.

[8] R. Dondi and A. Popa. Exact and approximation algorithms for covering timeline in temporal graphs. *Annals of Operations Research*, 04 2024. doi: 10.1007/s10479-024-05993-8.

[9] L. Fan, L. Li, Z. Ma, S. Lee, H. Yu, and L. Hemphill. A bibliometric review of large language models research from 2017 to 2023. *ACM Trans. Intell. Syst. Technol.*, 15(5):91:1–91:25, 2024. doi: 10.1145/3664930. URL https://doi.org/10.1145/3664930.

[10] V. Froese, P. Kunz, and P. Zschoche. Disentangling the computational complexity of network untangling. *Theory Comput. Syst.*, 68(1):103–121, 2024. doi: 10.1007/S00224-023-10150-Y. URL https://doi.org/10.1007/s00224-023-10150-y.

[11] N. Goyal, J. S. Kalra, C. Sharma, R. Mutharaju, N. Sachdeva, and P. Kumaraguru. Jobxmlc: Extreme multi-label classification of job skills with graph neural networks. In A. Vlachos and I. Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 2136–2146. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EACL.163. URL https://doi.org/10.18653/v1/2023.findings-eacl.163.

[12] International Labour Organization and Organisation for Economic Cooperation and Development. Approaches to anticipating skills for the future of work. Technical report, International Labour Organization, 2018. URL https://www.ilo.org/publications/approaches-anticipating-skills-future-work.

[13] G. Lazzarinetti. Emerging skills temporal knowledge graphs from job postings (2000–2024), 2025. URL https://doi.org/10.5281/zenodo.15230061. Data set.

[14] G. Lazzarinetti, S. Manzoni, and I. Zoppis. Automating resume analysis: Knowledge graphs via prompt engineering. In A. Artale, G. Cortellessa, and M. Montali, editors, *AIxIA 2024 - Advances in Artificial Intelligence - XXIIIrd International Conference of the Italian Association*

for Artificial Intelligence, AIxIA 2024, Bolzano, Italy, November 25-28, 2024, Proceedings, volume 15450 of *Lecture Notes in Computer Science*, pages 214–227. Springer, 2024. doi: 10.1007/978-3-031-80607-0_17. URL https://doi.org/10.1007/978-3-031-80607-0_17.

[15] G. Lazzarinetti, S. Manzoni, I. Zoppis, and R. Dondi. Fastmintc+: A fast and effective heuristic for minimum timeline cover on temporal networks. In P. Sala, M. Sioutis, and F. Wang, editors, *31st International Symposium on Temporal Representation and Reasoning, TIME 2024, October 28-30, 2024, Montpellier, France*, volume 318 of *LIPIcs*, pages 20:1–20:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi: 10.4230/LIPICS.TIME.2024.20. URL https://doi.org/10.4230/LIPIcs.TIME.2024.20.

[16] G. Lazzarinetti, R. Dondi, S. Manzoni, and I. Zoppis. Dlmintc+: A deep learning based algorithm for minimum timeline cover on temporal graphs. *Algorithms*, 18(2):113, 2025. doi: 10.3390/a18020113. URL https://doi.org/10.3390/a18020113.

[17] C. Mason, H. Chen, D. Evans, and G. Walker. Illustrating the application of a skills taxonomy, machine learning and online data to inform career and training decisions. *The International Journal of Information and Learning Technology*, 40, 06 2023. doi: 10.1108/IJILT-05-2022-0106.

[18] O. Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Math.*, 12(4):239–280, 2016. doi: 10.1080/15427951.2016.1177801. URL https://doi.org/10.1080/15427951.2016.1177801.

[19] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In B. Bonet and S. Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 4292–4293. AAAI Press, 2015. doi: 10.1609/AAAI.V29I1.9277. URL https://doi.org/10.1609/aaai.v29i1.9277.

[20] P. Rozenshtein, N. Tatti, and A. Gionis. The network-untangling problem: from interactions to activity timelines. *Data Min. Knowl. Discov.*, 35(1):213–247, 2021. doi: 10.1007/S10618-020-00717-5. URL https://doi.org/10.1007/s10618-020-00717-5.

[21] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P. S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. Pham, G. Kroiz, F. Li, H. Tao, A. Srivastava, H. D. Costa, S. Gupta, M. L. Rogers, I. Goncearenco, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. Hoyle, and P. Resnik. The prompt report: A systematic survey of prompt engineering techniques. *CoRR*, abs/2406.06608, 2025. doi: 10.48550/ARXIV.2406.06608. URL https://doi.org/10.48550/arXiv.2406.06608.

[22] E. M. Sibarani and S. Scerri. Generating an evolving skills network from job adverts for high-demand skillset discovery. In R. Cheng, N. Mamoulis, Y. Sun, and X. Huang, editors, *Web Information Systems Engineering - WISE 2019 - 20th International Conference, Hong Kong, China, November 26-30, 2019, Proceedings*, volume 11881 of *Lecture Notes in Computer Science*, pages 441–457. Springer, 2019. doi: 10.1007/978-3-030-34223-4_28. URL https://doi.org/10.1007/978-3-030-34223-4_28.

[23] T. Subha, R. Ranjana, B. Aarthi, S. Pavithra, and M. S. Srinidhi. Skill analysis and scouting platform using machine learning. In *2022 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, pages 1–6, 2022. doi: 10.1109/IC3IOT53935.2022.9767872.

[24] Y. Wang, Y. Allouache, and C. Joubert. Analysing CV corpus for finding suitable candidates using knowledge graph and BERT. In *Proceedings of the Thirteenth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA 2021)*, pages 26–31, Valencia, Spain, 2021. International Academy, Research, and Industry Association (IARIA). doi: 10.1109/DBKDA.2021.00012.

[25] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

[26] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

[27] World Economic Forum. The future of jobs report 2025, 2024. URL https://www.weforum.org/reports/the-future-of-jobs-report-2025. Accessed January 2025.

[28] R. Yazdanian, R. L. Davis, X. Guo, F. Lim, P. Dillenbourg, and M. Kan. On the radar: Predicting near-future surges in skills' hiring demand to provide early warning to educators. *Comput. Educ. Artif. Intell.*, 3:100043, 2022. doi: 10.1016/J.CAEAI.2021.100043. URL https://doi.org/10.1016/j.caeai.2021.100043.

[29] G. Yenduri, R. M, C. S. G., S. Y, G. Srivastava, P. K. R. Maddikunta, D. R. G, R. H. Jhaveri, B. Prabadevi, W. Wang, A. V. Vasilakos, and T. R. Gadekallu. GPT (generative pre-trained transformer) - A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *IEEE Access*, 12:54608–54649, 2024. doi: 10.1109/ACCESS.2024.3389497. URL https://doi.org/10.1109/ACCESS.2024.3389497.

[30] L. Zhong, J. Wu, Q. Li, H. Peng, and X. Wu. A comprehensive survey on automatic knowledge graph construction. *ACM Comput. Surv.*, 56(4):94:1–94:62, 2024. doi: 10.1145/3618295. URL https://doi.org/10.1145/3618295.