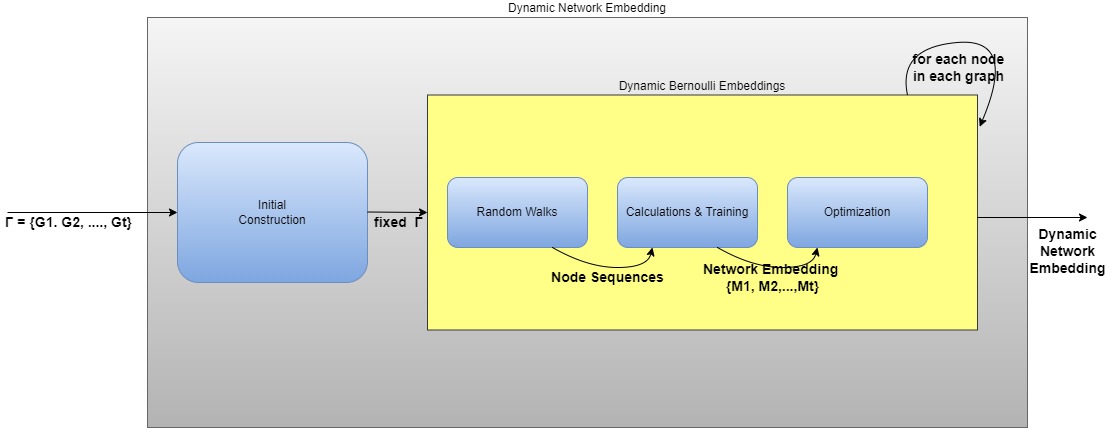
**Dynamic Network Embeddings for Network Evolution Analysis**

***Summary***



**Initial Construction**

Dynamic networks are represented as a series of graphs, where each graph corresponds to a specific period. There are two primary methods for embedding these graphs:

Constant Time Interval: Each graph represents events occurring within a fixed time window between t and t+1.

Constant Event Count: Each graph represents a fixed number of events, ensuring uniformity in events count across graphs, regardless of time intervals.

To ensure smooth transitions and maintain continuity across the dynamic network, an overlap is introduced between adjacent graphs. This overlap creates a connection between consecutive graphs, preserving the flow of information and minimizing disruptions in the network's temporal dynamics.

**Random Walk**

The random walk is a process that starts from a given source node and explores the graph. For each node in every graph, the Random Walk model performs multiple random walks (defined by a parameter L), each of a specified length (another parameter R).

To optimize runtime, these random walks are executed in parallel across all graphs.

The output of this process is a node sequence matrix W generated for each graph.

**Calculations and Training**

The goal of this stage is to compute embeddings (numerical representations) for each node in the dynamic network across multiple timestamps. These embeddings capture:

* Proximity: Similar nodes are placed close in the embedding space.
* Temporal Continuity: The embeddings of stable nodes change minimally across timestamps.
* Efficiency: The embeddings are low-dimensional, reducing computational complexity.

**Key Components of the Calculation Process**

1. Embedding Vectors:
   * **Node embedding**  A low-dimensional vector that represents node g at time t. It captures the node’s relationships within the network at that timestamp.
   * **Context vector** A shared vector for each node, representing its role or interactions across all timestamps. It acts as a reference point to determine how nodes relate to their context.

Think of as the position of a node in the network at time t, and as the general "type" of the node based on its connections.

1. Node Indicator Vector
   * This is a binary vector where =1 if node g is part of the context for the i-th position in a random walk at time t and otherwise =0.
   * It ensures that the model correctly identifies which nodes are part of a given context.

This vector acts as a "flag" to indicate the presence of a node in a specific context, simplifying the computation of relationships.

1. Context Relationship Score :
   * This score measures how well a node g fits its context.

*: Embedding of node g at time t.*

*: Context vector of node g.*

*: Size of the context window.*

* + Calculated as:

This is like calculating the similarity between a node’s embedding and the aggregated embeddings of its neighbours.

1. Likelihood Functions:

To ensure embeddings preserve proximity and temporal continuity, the model defines two likelihood functions:

* **Positive Likelihood** (:

This measures how well the model predicts the true relationships between nodes in the network.

* **Negative Likelihood** (:

Here, represents a sampling distribution for "negative" nodes (nodes not part of the true context). Negative sampling reduces computational complexity.

**Negative Sampling**: Instead of calculating for all possible negative nodes (which would be computationally expensive), a small subset is sampled at random. This speeds up training while retaining accuracy.

1. Regulariztion:

To ensure temporal consistency and prevent overfitting, two regularization terms are added:

* + For the context vector:

This ensures that the context vectors remain stable and bounded.

* + For temporal continuity:

This penalizes large changes in embeddings between consecutive timestamps.

Regularization ensures the embeddings are smooth and consistent over time.

**Optimization Process**

1. **Objective Function**: The combined objective function is:

The aim is to maximize the positive likelihood and minimize the negative likelihood, while maintaining stability via regularization.

1. **Stochastic Gradient Descent (SGD)**:
   * The model uses SGD to iteratively update and based on their gradients.
   * Gradients are computed with respect to the objective function.
   * A carefully chosen learning rate ensures convergence.

Think of SGD as the model’s way of "learning" by repeatedly adjusting the embeddings to better reflect the network structure.

**Practical Considerations**

1. **Pretraining**: The model can use static embeddings (e.g., from methods like Node2Vec) as a starting point for training. This accelerates convergence and improves results.
2. **Alignment Across Time**: By sharing context vectors across all timestamps, the model aligns embeddings in the same vector space, ensuring temporal consistency.

**Optimization**

To optimize the model, the positive and negative contexts are calculated separately. Since most dynamic networks contain an extremely large number of negative context nodes, the model computes only a sampled distribution of negative context nodes. This approach significantly reduces computational runtime.

Moreover, the model accounts for the difference between each vector in the matrices Mt and Mt-1​. This difference is calculated based on the context of the vector in the matrix for the next timestamp.

**Why Does This Work?**

* **Captures Node Relationships**: By focusing on the contexts generated by random walks, the embeddings reflect the local and global structure of the network.
* **Maintains Temporal Stability**: Regularization prevents embeddings from changing too much across timestamps unless there are significant structural changes.
* **Efficient Computation**: Techniques like negative sampling and SGD make the process scalable to large networks.

**Citation Dynamic Network Embeddings**

***Strategies for Incorporating Timestamps into Dynamic Network Embeddings***

* + 1. Weighting Edges:

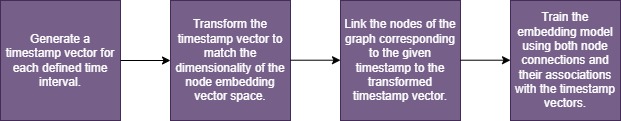
For every citation between articles, the edge weight will be determined by the time elapsed between their publication dates. This elapsed time will be converted into a numeric value and assigned as the weight of the temporal edge. Incorporating this strategy into the random walk model allows the algorithm to account for temporal relationships, enhancing the analysis of citation patterns and network dynamics.

A diagram of a process

Description automatically generated with medium confidence

* + 1. Temporal Embedding Vector:

Each timestamp will be represented by a unique vector. The graph corresponding to a specific timestamp will connect its nodes to this timestamp vector. This approach allows the training process to incorporate temporal changes directly, enabling the model to draw conclusions about how relationships evolve over time.



* + 1. Publication Embedding:

To ensure effective analysis of graph proximity between articles, the publication date of each article can be integrated into the embedding process. Incorporating the publication date as a feature of the nodes enhances the ability to calculate proximity by accounting for temporal relationships, thereby improving the embedding's relevance and accuracy.

***Strategies for Graphs Modelling Considering Timestamps***

* The graphs are constructed to contain a fixed number of events, ensuring consistent graph sizes over time. This strategy maintains uniformity in graph structure, which facilitates better comparisons and analysis across different time intervals.
* The overlap between graphs at consecutive timestamps does not need to be extensive, as edges in the graphs are only added over time and not removed. This minimizes redundancy while still preserving temporal continuity.