Gelly:Bipartite Graph

Attention Bipartite Graph currently only supported in Gelly Java API.

Bipartite Graph

A bipartite graph (also called a two-mode graph) is a type of graph where vertices are separated into two disjoint sets. These sets are usually called top and bottom vertices. An edge in this graph can only connect vertices from opposite sets (i.e. bottom vertex to top vertex) and cannot connect two vertices in the same set.

These graphs have wide application in practice and can be a more natural choice for particular domains. For example to represent authorship of scientific papers top vertices can represent scientific papers while bottom nodes will represent authors. Naturally an edge between a top and a bottom nodes would represent an authorship of a particular scientific paper. Another common example for applications of bipartite graphs is relationships between actors and movies. In this case an edge represents that a particular actor played in a movie.

Bipartite graphs are used instead of regular graphs (one-mode) for the following practical reasons: * They preserve more information about a connection between vertices. For example instead of a single link between two researchers in a graph that represents that they authored a paper together a bipartite graph preserves the information about what papers they authored * Bipartite graphs can encode the same information more compactly than one-mode graphs

Graph Representation

A `BipartiteGraph` is represented by: * A `DataSet` of top nodes * A `DataSet` of bottom nodes * A `DataSet` of edges between top and bottom nodes

As in the `Graph` class nodes are represented by the `Vertex` type and the same rules apply to its types and values.

The graph edges are represented by the `BipartiteEdge` type. A `BipartiteEdge` is defined by a top ID (the ID of the top `Vertex`), a bottom ID (the ID of the bottom `Vertex`) and an optional value. The main difference between the `Edge` and `BipartiteEdge` is that IDs of nodes it links can be of different types. Edges with no value have a `NullValue` value type.

```java
BipartiteEdge<Long, String, Double> e = new BipartiteEdge<Long, String, Double>(1L, "id1",
0.5);

Double weight = e.getValue(); // weight = 0.5
```

Graph Creation

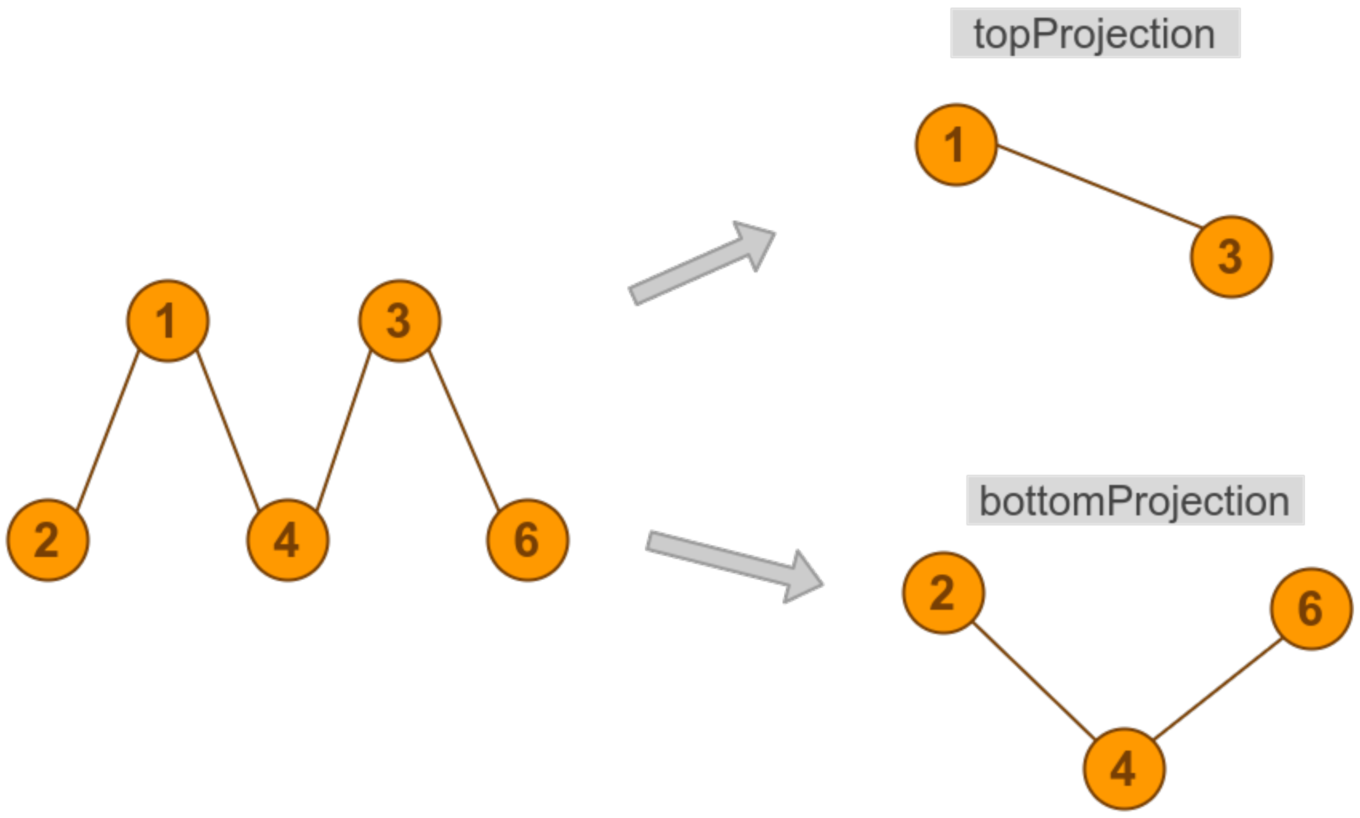
You can create a `BipartiteGraph` in the following ways:

- from a `DataSet` of top vertices, a `DataSet` of bottom vertices and a `DataSet` of edges:

```
ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();

DataSet<Vertex<String, Long>> topVertices = ...

DataSet<Vertex<String, Long>> bottomVertices = ...

DataSet<Edge<String, String, Double>> edges = ...

Graph<String, String, Long, Long, Double> graph = BipartiteGraph.fromDataSet(topVertices,
bottomVertices, edges, env);
```

Graph Transformations

- **Projection**: Projection is a common operation for bipartite graphs that converts a bipartite graph into a regular graph. There are two types of projections: top and bottom projections. Top projection preserves only top nodes in the result graph and creates a link between them in a new graph only if there is an intermediate bottom node both top nodes connect to in the original graph. Bottom projection is the opposite to top projection, i.e. only preserves bottom nodes and connects a pair of nodes if they are connected in the original graph.



Gelly supports two sub-types of projections: simple projections and full projections. The only difference between them is what data is associated with edges in the result graph.

In the case of a simple projection each node in the result graph contains a pair of values of bipartite edges that connect nodes in the original graph:

```java
ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
// Vertices (1, "top1")
DataSet<Vertex<Long, String>> topVertices = ...

// Vertices (2, "bottom2"); (4, "bottom4")
DataSet<Vertex<Long, String>> bottomVertices = ...

// Edge that connect vertex 2 to vertex 1 and vertex 4 to vertex 1:
// (1, 2, "1-2-edge"); (1, 4, "1-4-edge")
DataSet<Edge<Long, Long, String>> edges = ...

BipartiteGraph<Long, Long, String, String, String> graph =
BipartiteGraph.fromDataSet(topVertices, bottomVertices, edges, env);

// Result graph with two vertices:
// (2, "bottom2"); (4, "bottom4")
//
// and one edge that contains ids of bottom edges and a tuple with
// values of intermediate edges in the original bipartite graph:
// (2, 4, ("1-2-edge", "1-4-edge"))
Graph<Long, String, Tuple2<String, String>> graph bipartiteGraph.projectionBottomSimple();
```

Full projection preserves all the information about the connection between two vertices and stores it in `Projection` instances. This includes value and id of an intermediate vertex, source and target vertex values and source and target edge values:

```java
ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
```

```java
// Vertices (1, "top1")
DataSet<Vertex<Long, String>> topVertices = ...

// Vertices (2, "bottom2"); (4, "bottom4")
DataSet<Vertex<Long, String>> bottomVertices = ...

// Edge that connect vertex 2 to vertex 1 and vertex 4 to vertex 1:
// (1, 2, "1-2-edge"); (1, 4, "1-4-edge")
DataSet<Edge<Long, Long, String>> edges = ...

BipartiteGraph<Long, Long, String, String, String> graph =
BipartiteGraph.fromDataSet(topVertices, bottomVertices, edges, env);

// Result graph with two vertices:
// (2, "bottom2"); (4, "bottom4")
// and one edge that contains ids of bottom edges and a tuple that
// contains id and value of the intermediate edge, values of connected vertices
// and values of intermediate edges in the original bipartite graph:
// (2, 4, (1, "top1", "bottom2", "bottom4", "1-2-edge", "1-4-edge"))
Graph<String, String, Projection<Long, String, String, String>> graph
bipartiteGraph.projectionBottomFull();
```