Node Embeddings

Here we will have a look at node embeddings and how to further reduce their dimensionality to be able to visualize them in a 2D plot.

Note about data dependencies

PageRank centrality and Leiden community are also fetched from the Graph and need to be calculated first. This makes it easier to see in the visualization if the embeddings approximate the structural information of the graph. If these properties are missing you will only see black dots all of the same size without community coloring. In future it might make sense to also run a community detection algorithm co-located in here to not depend on the order of execution.

References

- jqassistant
- Neo4j Python Driver
- Tutorial: Applied Graph Embeddings
- Visualizing the embeddings in 2D
- Fast Random Projection
- scikit-learn TSNE
- AttributeError: 'list' object has no attribute 'shape'

```
The scikit-learn version is 1.3.0.
The pandas version is 1.5.1.
```

Preparation

Create Graph Projection

Create an in-memory undirected graph projection containing Package nodes (vertices) and their dependencies (edges).

Generate Node Embeddings using Fast Random Projection (Fast RP)

Fast Random Projection calculates an array of floats (length = embedding dimension) for every node in the graph. These numbers approximate the relationship and similarity information of each node and are called node embeddings. Random Projections is used to reduce the dimensionality of the node feature space while preserving pairwise distances.

The result can be used in machine learning as features approximating the graph structure. It can also be used to further reduce the dimensionality to visualize the graph in a 2D plot, as we will be doing here.

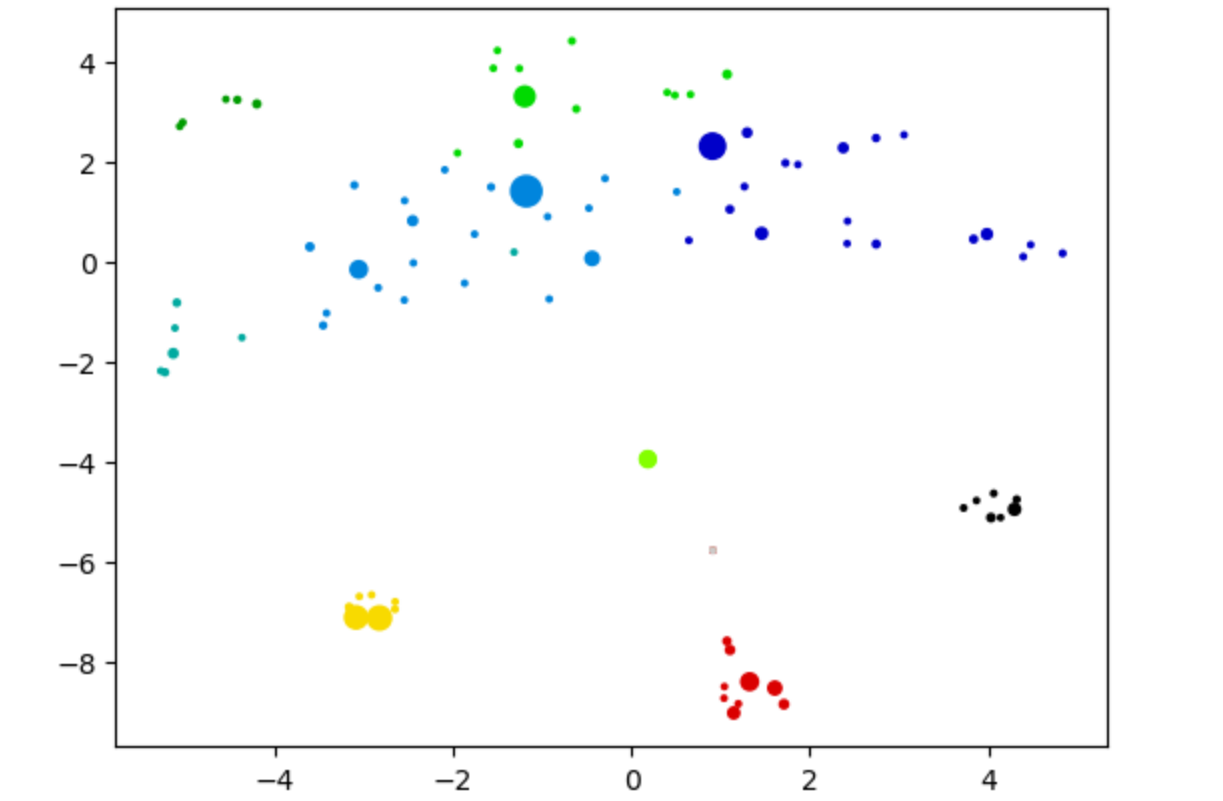
Dimensionality reduction with t-distributed stochastic neighbor embedding (t-SNE)

This step takes the original node embeddings with a higher dimensionality (e.g. list of 32 floats) and reduces them to a 2 dimensional array for visualization.

> It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data.

(see https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html#sklearn.manifold.TSNE)

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 93 samples in 0.000s...
[t-SNE] Computed neighbors for 93 samples in 0.228s...
[t-SNE] Computed conditional probabilities for sample 93 / 93
[t-SNE] Mean sigma: 0.617398
[t-SNE] KL divergence after 250 iterations with early exaggeration: 51.613049
[t-SNE] KL divergence after 1000 iterations: 0.067252
```


Package nodes positioned by their dependency relationships using t-SNE
4
2
0
−2
−4
−6
−8
−4
−2
0
2
4