Node Embeddings

This notebook demonstrates different methods for node embeddings and how to further reduce their dimensionality to be able to visualize them in a 2D plot.

Node embeddings are essentially an array of floating point numbers (length = embedding dimension) that can be used as "features" in machine learning. These numbers approximate the relationship and similarity information of each node and can also be seen as a way to encode the topology of the graph.

Considerations

Due to dimensionality reduction some information gets lost, especially when visualizing node embeddings in two dimensions. Nevertheless, it helps to get an intuition on what node embeddings are and how much of the similarity and neighborhood information is retained. The latter can be observed by how well nodes of the same color and therefore same community are placed together and how much bigger nodes with a high centrality score influence them.

If the visualization doesn't show a somehow clear separation between the communities (colors) here are some ideas for tuning:

- Clean the data, e.g. filter out very few nodes with extremely high degree that aren't actually that important
- Try directed vs. undirected projections
- Tune the embedding algorithm, e.g. use a higher dimensionality
- Tune t-SNE that is used to reduce the node embeddings dimension to two dimensions for visualization.

It could also be the case that the node embeddings are good enough and well suited the way they are despite their visualization for the down stream task like node classification or link prediction. In that case it makes sense to see how the whole pipeline performs before tuning the node embeddings in detail.

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 if the embeddings approximate the structural information of the graph in the plot. If these properties are missing you will only see black dots all of the same size.

References

- jqassistant

- Neo4j Python Driver
- Tutorial: Applied Graph Embeddings
- Visualizing the embeddings in 2D
- scikit-learn TSNE
- AttributeError: 'list' object has no attribute 'shape'
- Fast Random Projection (neo4j)
- HashGNN (neo4j)
- node2vec (neo4j) computes a vector representation of a node based on second order random walks in the graph.
- Complete guide to understanding Node2Vec algorithm

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

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

The following function takes the original node embeddings with a higher dimensionality, e.g. 64 floating point numbers, and reduces them into a two 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)

1. Java Packages

1.1 Generate Node Embeddings using Fast Random Projection (Fast RP) for Java Packages

Fast Random Projection is used to reduce the dimensionality of the node feature space while preserving most of the distance information. Nodes with similar neighborhood result in node embedding with similar vectors.

```
The results have been provided by the query filename: ../cypher/Node_Embeddings/Node_
Embeddings_0a_Query_Calculated.cypher
```

	codeUnitName	shortCodeUnitName	projectName	communityId	centrality	embedding
0	org.axonframework.modelling.command	command	axon-modelling-4.9.3	0	0.155609	[-0.2997018098831177, -0.1952381432056427, 0.2...
1	org.axonframework.modelling.command.inspection	inspection	axon-modelling-4.9.3	0	0.152740	[-0.34666895866394043, -0.20513851940631866, 0...
2	org.axonframework.modelling.command.legacyjpa	legacyjpa	axon-modelling-4.9.3	0	0.016234	[-0.3526989817619324, -0.24959245324134827, 0....
3	org.axonframework.modelling.saga	saga	axon-modelling-4.9.3	1	0.349429	[-0.11967099457979202, 0.2634722888469696, -0....
4	org.axonframework.modelling.saga.metamodel	metamodel	axon-modelling-4.9.3	1	0.315690	[-0.12892881035804749, 0.3700981140136719, -0....

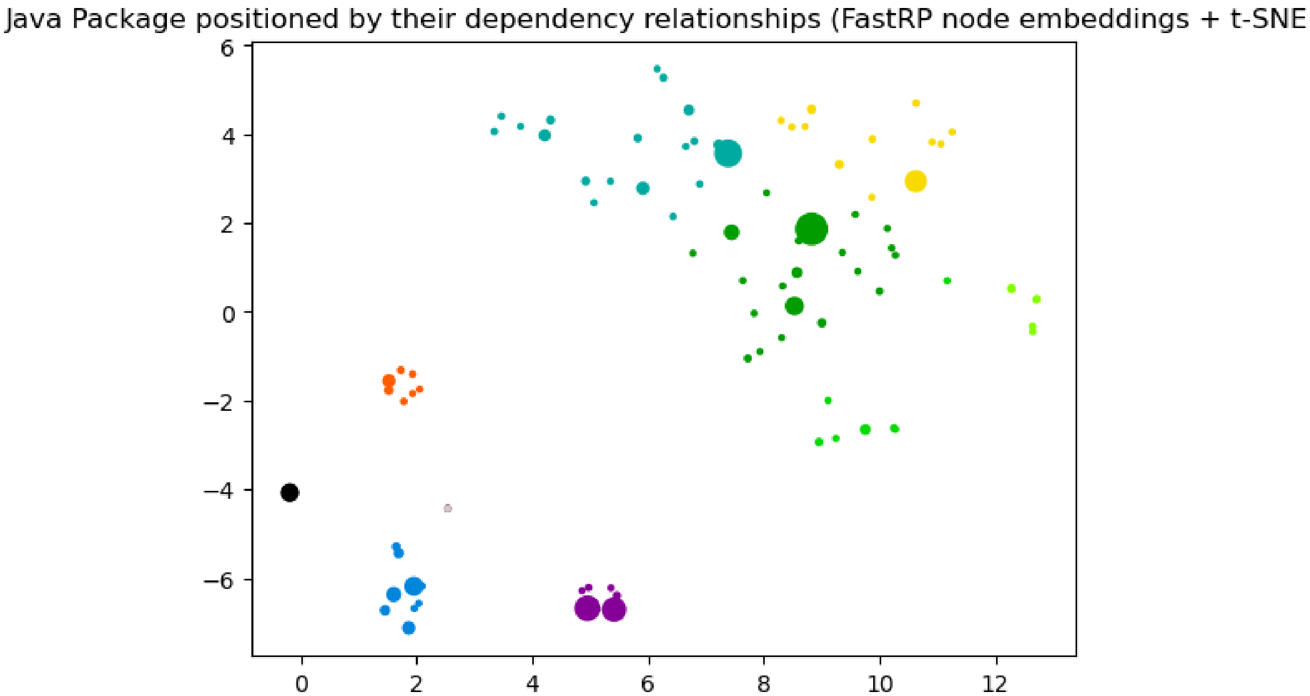
1.2 Dimensionality reduction with t-distributed stochastic neighbor embedding (t-SNE)

This step takes the original node embeddings with a higher dimensionality, e.g. 64 floating point numbers, and reduces them into a two dimensional array for visualization. For more details look up the function declaration for "prepare_node_embeddings_for_2d_visualization".

```
t-SNE: Sample size (Number of nodes)=93
t-SNE: perplexity=30.0
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 93 samples in 0.000s...
[t-SNE] Computed neighbors for 93 samples in 0.082s...
[t-SNE] Computed conditional probabilities for sample 93 / 93
[t-SNE] Mean sigma: 0.655771
[t-SNE] KL divergence after 250 iterations with early exaggeration: 48.126137
[t-SNE] KL divergence after 800 iterations: 0.050790
(93, 2)
```

	codeUnit	artifact	communityId	centrality	x	y
0	org.axonframework.modelling.command	axon-modelling-4.9.3	0	0.155609	-0.188775	-4.077829
1	org.axonframework.modelling.command.inspection	axon-modelling-4.9.3	0	0.152740	-0.187209	-4.069431
2	org.axonframework.modelling.command.legacyjpa	axon-modelling-4.9.3	0	0.016234	-0.216587	-4.061156
3	org.axonframework.modelling.saga	axon-modelling-4.9.3	1	0.349429	4.956002	-6.678513
4	org.axonframework.modelling.saga.metamodel	axon-modelling-4.9.3	1	0.315690	5.414717	-6.707891

1.3 Visualization of the node embeddings reduced to two dimensions



1.4 Node Embeddings for Java Packages using HashGNN

HashGNN resembles Graph Neural Networks (GNN) but does not include a model or require training. It combines ideas of GNNs and fast randomized algorithms. For more details see HashGNN. Here, the latter 3 steps are combined into one for HashGNN.

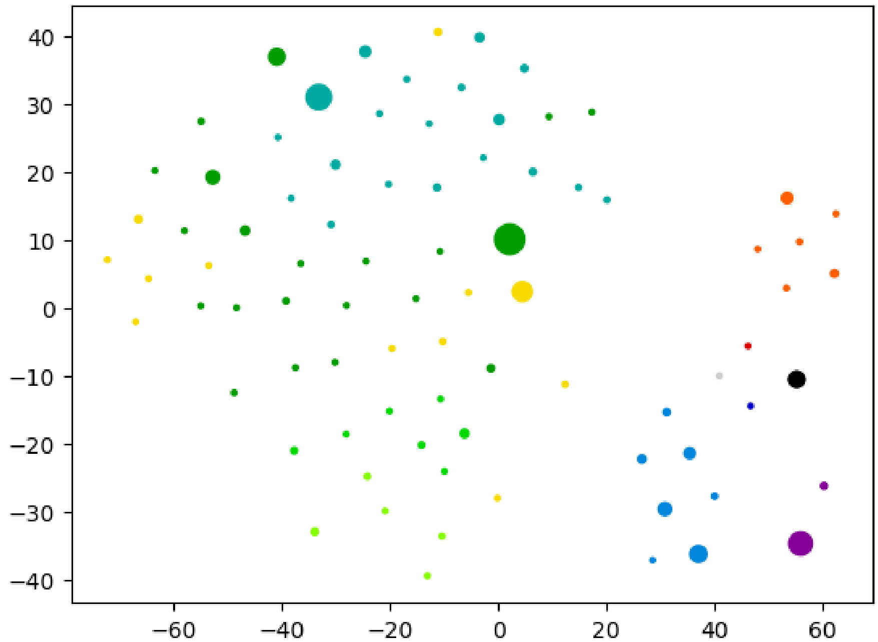
```
The results have been provided by the query filename: ../cypher/Node_Embeddings/Node_
Embeddings_0a_Query_Calculated.cypher
```

	codeUnitName	shortCodeUnitName	projectName	communityId	centrality	embedding
0	org.axonframework.modelling.command	command	axon-modelling-4.9.3	0	0.155609	[-0.8660253882408142, 0.21650634706020355, -0....
1	org.axonframework.modelling.command.inspection	inspection	axon-modelling-4.9.3	0	0.152740	[-0.8660253882408142, 0.21650634706020355, -0....
2	org.axonframework.modelling.command.legacyjpa	legacyjpa	axon-modelling-4.9.3	0	0.016234	[-0.8660253882408142, 0.21650634706020355, -0....
3	org.axonframework.modelling.saga	saga	axon-modelling-4.9.3	1	0.349429	[0.8660253882408142, -0.4330126941204071, -0.6...
4	org.axonframework.modelling.saga.metamodel	metamodel	axon-modelling-4.9.3	1	0.315690	[0.8660253882408142, -0.4330126941204071, -0.6...

```
t-SNE: Sample size (Number of nodes)=93
t-SNE: perplexity=30.0
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 93 samples in 0.000s...
[t-SNE] Computed neighbors for 93 samples in 0.002s...
[t-SNE] Computed conditional probabilities for sample 93 / 93
[t-SNE] Mean sigma: 2.844971
[t-SNE] KL divergence after 250 iterations with early exaggeration: 47.890877
[t-SNE] KL divergence after 1000 iterations: 0.095945
(93, 2)
```

	codeUnit	artifact	communityId	centrality	x	y
0	org.axonframework.modelling.command	axon-modelling-4.9.3	0	0.155609	55.115955	-10.507041
1	org.axonframework.modelling.command.inspection	axon-modelling-4.9.3	0	0.152740	55.115955	-10.507041
2	org.axonframework.modelling.command.legacyjpa	axon-modelling-4.9.3	0	0.016234	55.115955	-10.507041
3	org.axonframework.modelling.saga	axon-modelling-4.9.3	1	0.349429	55.845608	-34.690697
4	org.axonframework.modelling.saga.metamodel	axon-modelling-4.9.3	1	0.315690	55.845608	-34.690697

Java Package positioned by their dependency relationships (HashGNN node embeddings + t-SNE)



2.5 Node Embeddings for Java Packages using node2vec

```
The results have been provided by the query filename: ../cypher/Node_Embeddings/Node_
Embeddings_0a_Query_Calculated.cypher
```

	codeUnitName	shortCodeUnitName	projectName	communityId	centrality	embedding
0	org.axonframework.modelling.command	command	axon-modelling-4.9.3	0	0.155609	[0.09866541624069214, -2.0777676105499268, 1.5...
1	org.axonframework.modelling.command.inspection	inspection	axon-modelling-4.9.3	0	0.152740	[0.13471074402332306, -2.0186145305633545, 1.6...
2	org.axonframework.modelling.command.legacyjpa	legacyjpa	axon-modelling-4.9.3	0	0.016234	[0.077057845890522, -1.932939052581787, 1.5184...
3	org.axonframework.modelling.saga	saga	axon-modelling-4.9.3	1	0.349429	[0.186208114027977, -0.8858957886695862, 0.633...
4	org.axonframework.modelling.saga.metamodel	metamodel	axon-modelling-4.9.3	1	0.315690	[0.05101150646805763, -0.9541137218475342, 0.4...

```
t-SNE: Sample size (Number of nodes)=93
t-SNE: perplexity=30.0
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 93 samples in 0.000s...
[t-SNE] Computed neighbors for 93 samples in 0.001s...
[t-SNE] Computed conditional probabilities for sample 93 / 93
[t-SNE] Mean sigma: 0.647886
[t-SNE] KL divergence after 250 iterations with early exaggeration: 50.072601
[t-SNE] KL divergence after 900 iterations: 0.131736
(93, 2)
```

	codeUnit	artifact	communityId	centrality	x	y
0	org.axonframework.modelling.command	axon-modelling-4.9.3	0	0.155609	1.841828	3.176171
1	org.axonframework.modelling.command.inspection	axon-modelling-4.9.3	0	0.152740	1.830896	3.161993
2	org.axonframework.modelling.command.legacyjpa	axon-modelling-4.9.3	0	0.016234	1.834886	3.152582
3	org.axonframework.modelling.saga	axon-modelling-4.9.3	1	0.349429	-2.549339	2.266320
4	org.axonframework.modelling.saga.metamodel	axon-modelling-4.9.3	1	0.315690	-2.399351	2.300525

Java Package positioned by their dependency relationships (node2vec node embeddings + t-SNE)