Lecture 8: Learning with Graphs

Michael Cochez

Deep Learning 2020

VRIJE UNIVERSITEIT AMSTERDAM

dlvu.github.io

THE PLAN

part 1: Introduction - Why graphs? What are embeddings?

part 2: Graph Embedding Techniques

part 3: Graph Neural Networks

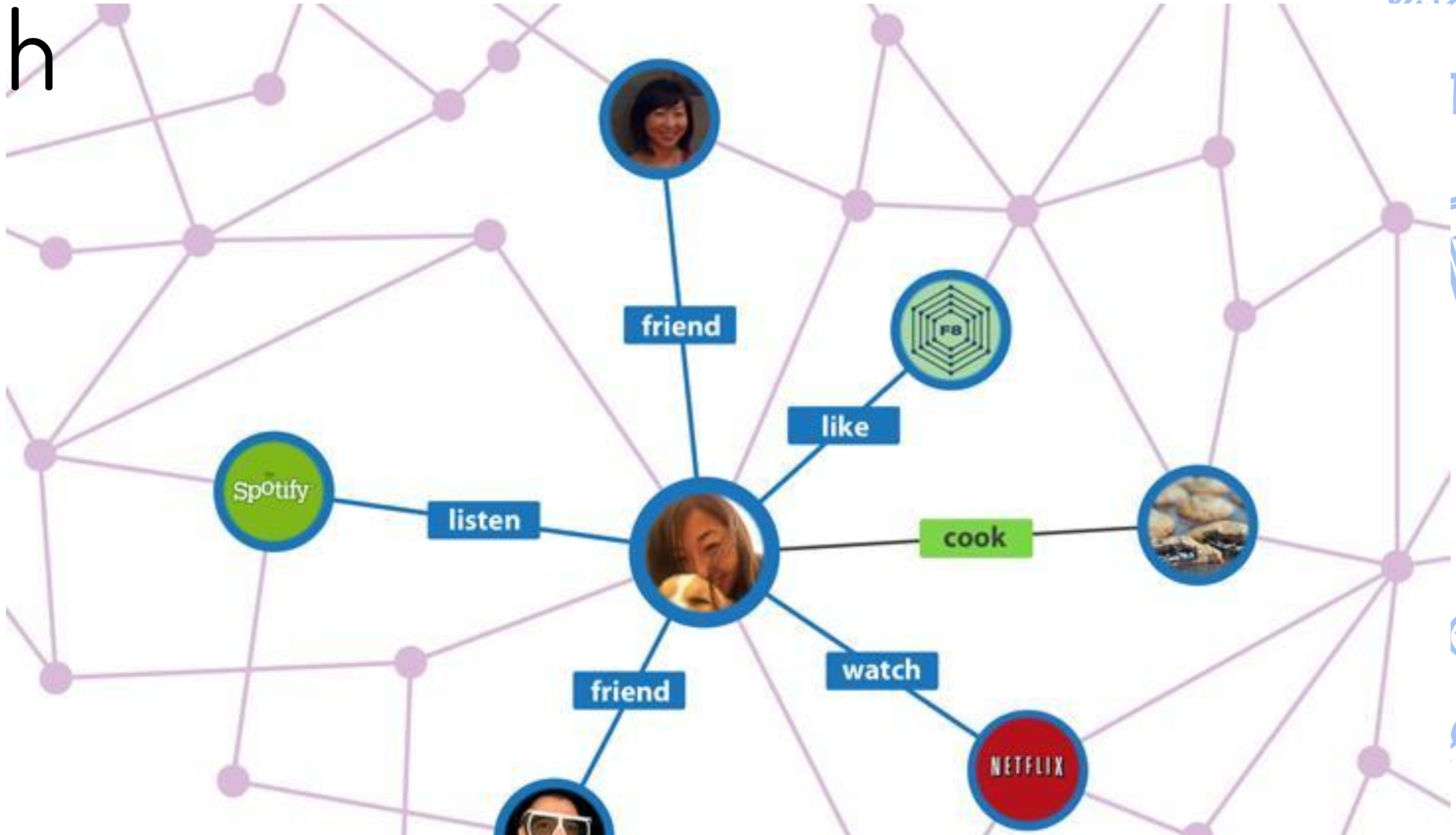
part 4: Application - Query embedding

PART ONE - A: INTRODUCTION - GRAPHS

When was the last time you ...

reconnected with a friend?

Facebook Social Graph



http://www.businessinsider.com/explainer-what-exactly-is-the-social-graph-2012-3

When was the last time you …

reconnected with a friend?

visited a doctor?

IBM Watson

Next comes the “ingestion” process: Watson preprocesses the information, building indices and other metadata that make the content more efficient to work with. It may also create a knowledge graph to represent and leverage key concepts and relationships within a domain.

https://www.ibm.com/think/marketing/how-watson-learns/

When was the last time you ...

reconnected with a friend?

visited a doctor?

browsed through products in a webshop?

Amazon Product Graph

When was the last time you ...

reconnected with a friend?

visited a doctor?

browsed through products in a webshop?

did a web search?

Google Knowledge Graph

https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html

When was the last time you ...

browsed through products in a webshop?

reconnected with a friend?

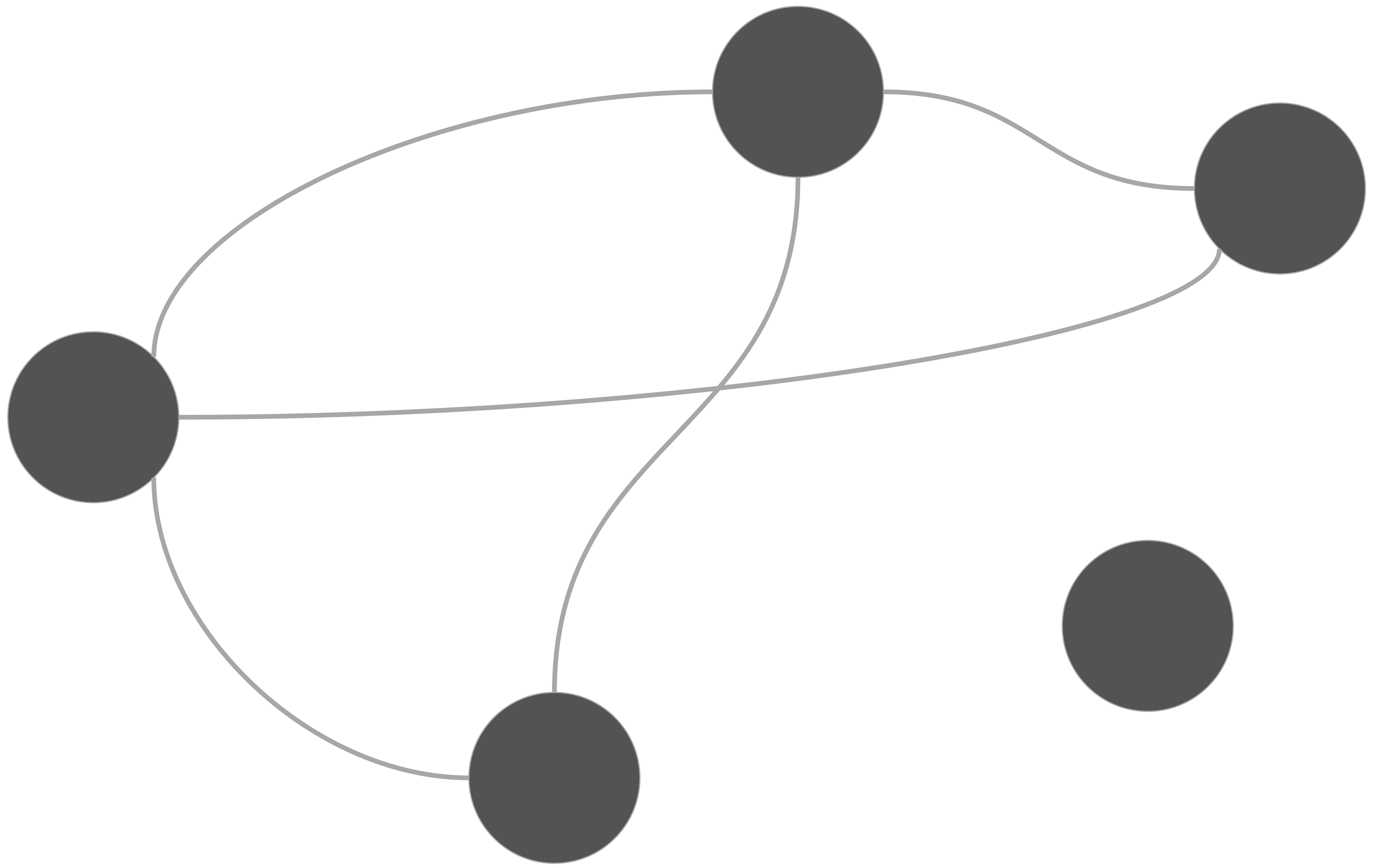
visited a doctor?

did a web search?

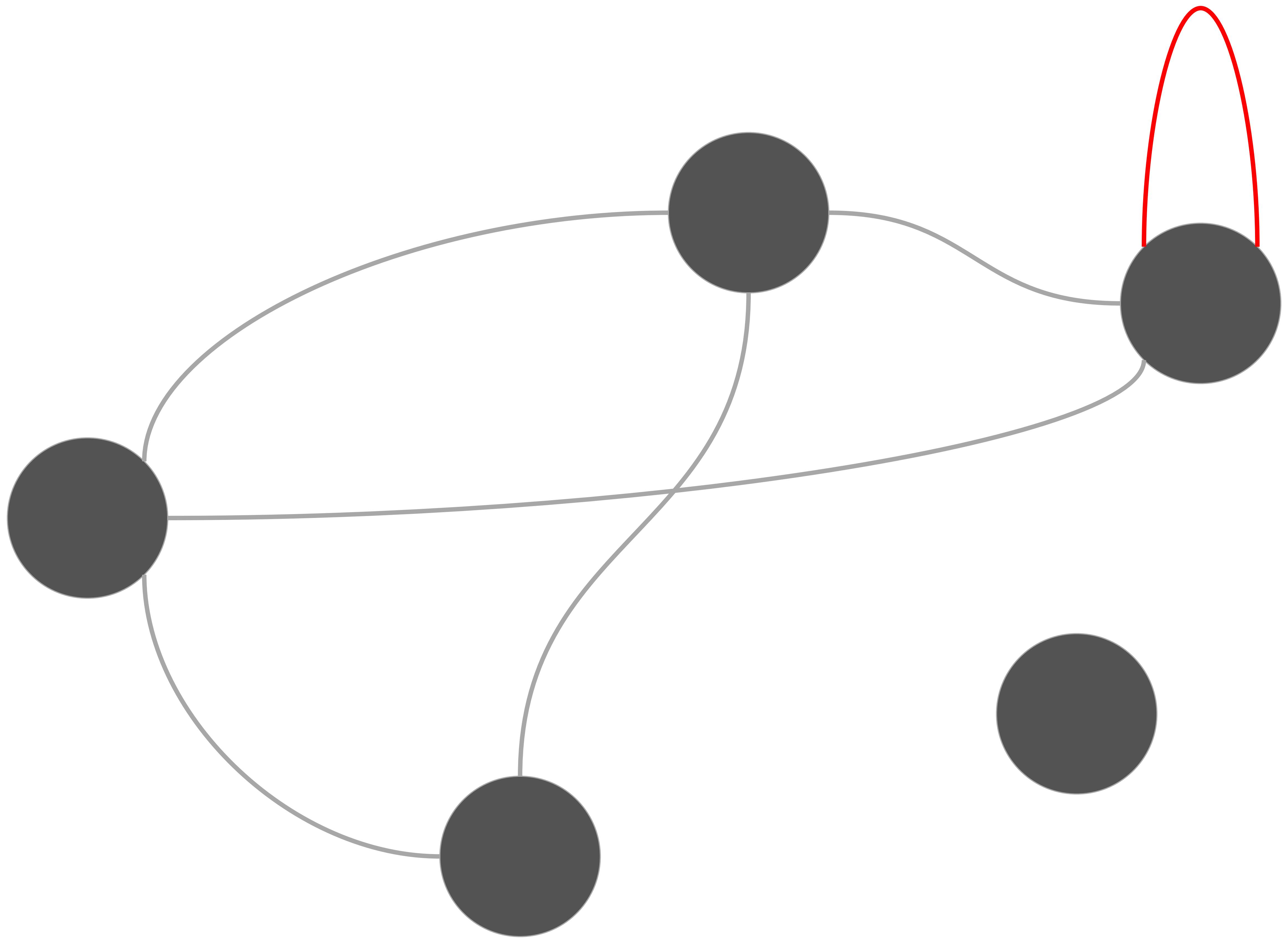
Knowledge graphs are all around us.

Other examples: Cyc, Freebase, DBPedia, Wikidata, YAGO, Thomson Reuters, Microsoft Satori, Yahoo KG, Springer, ...

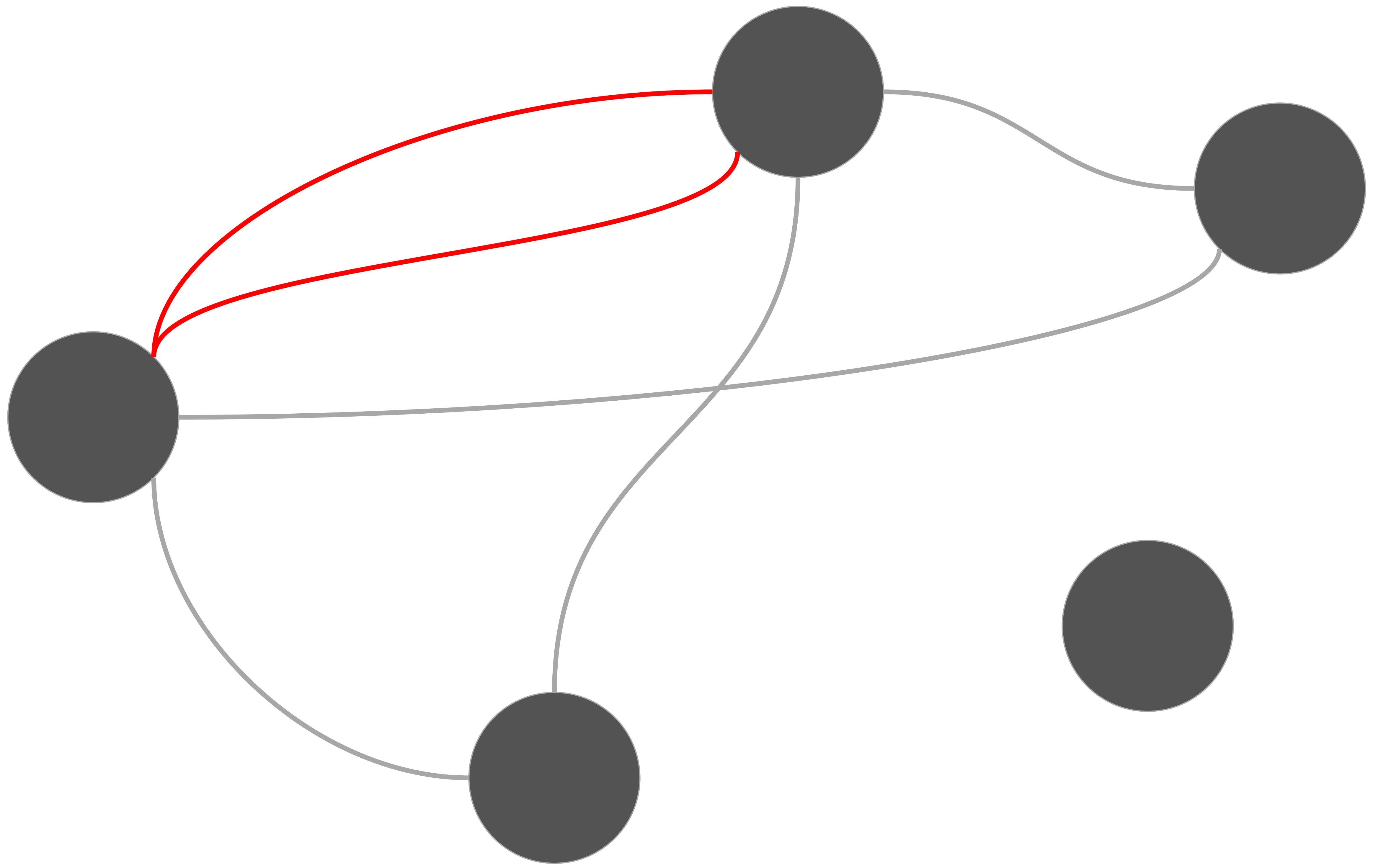
Graphs - Undirected - Simple Graphs



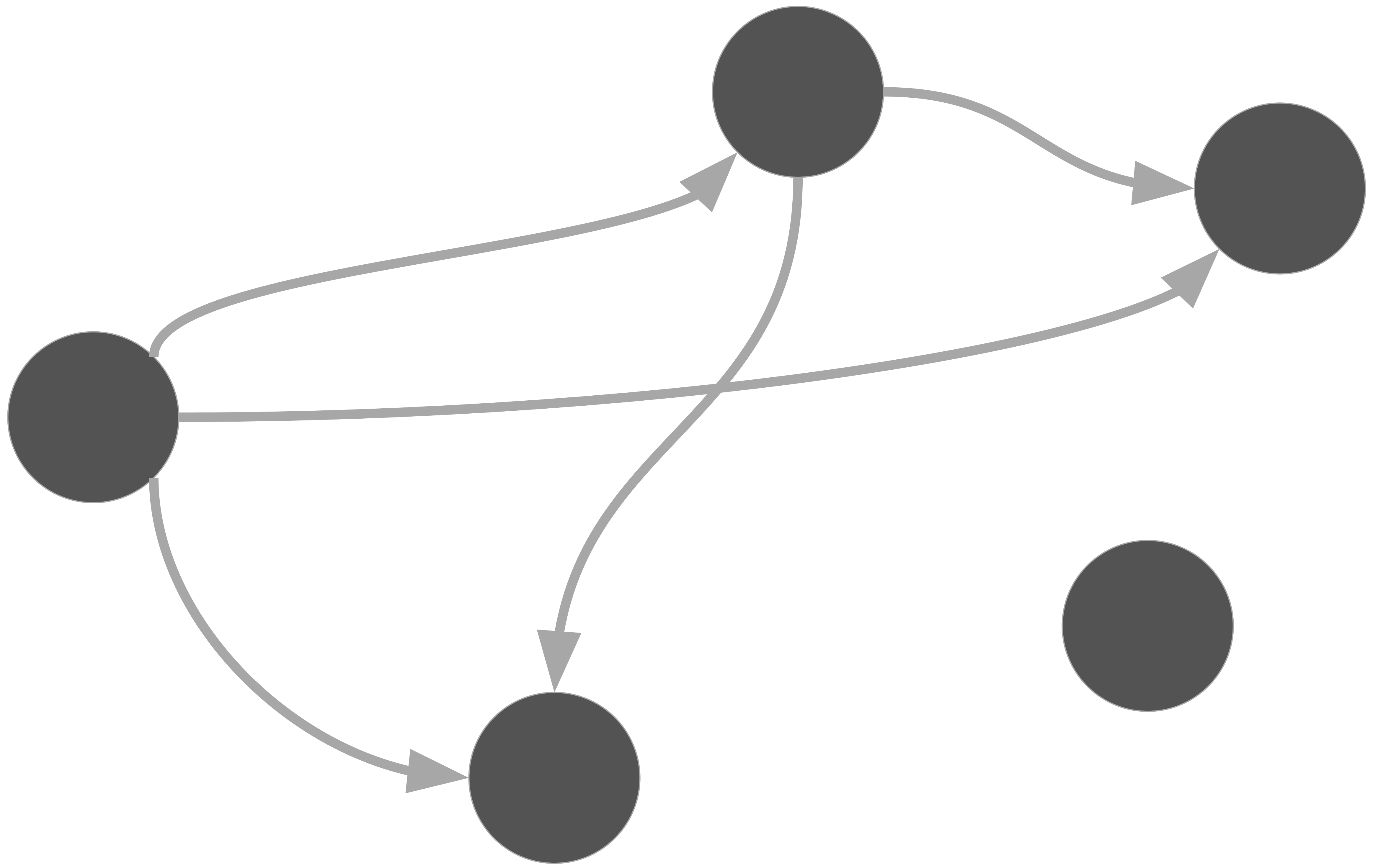
Graphs - Undirected - Self loop



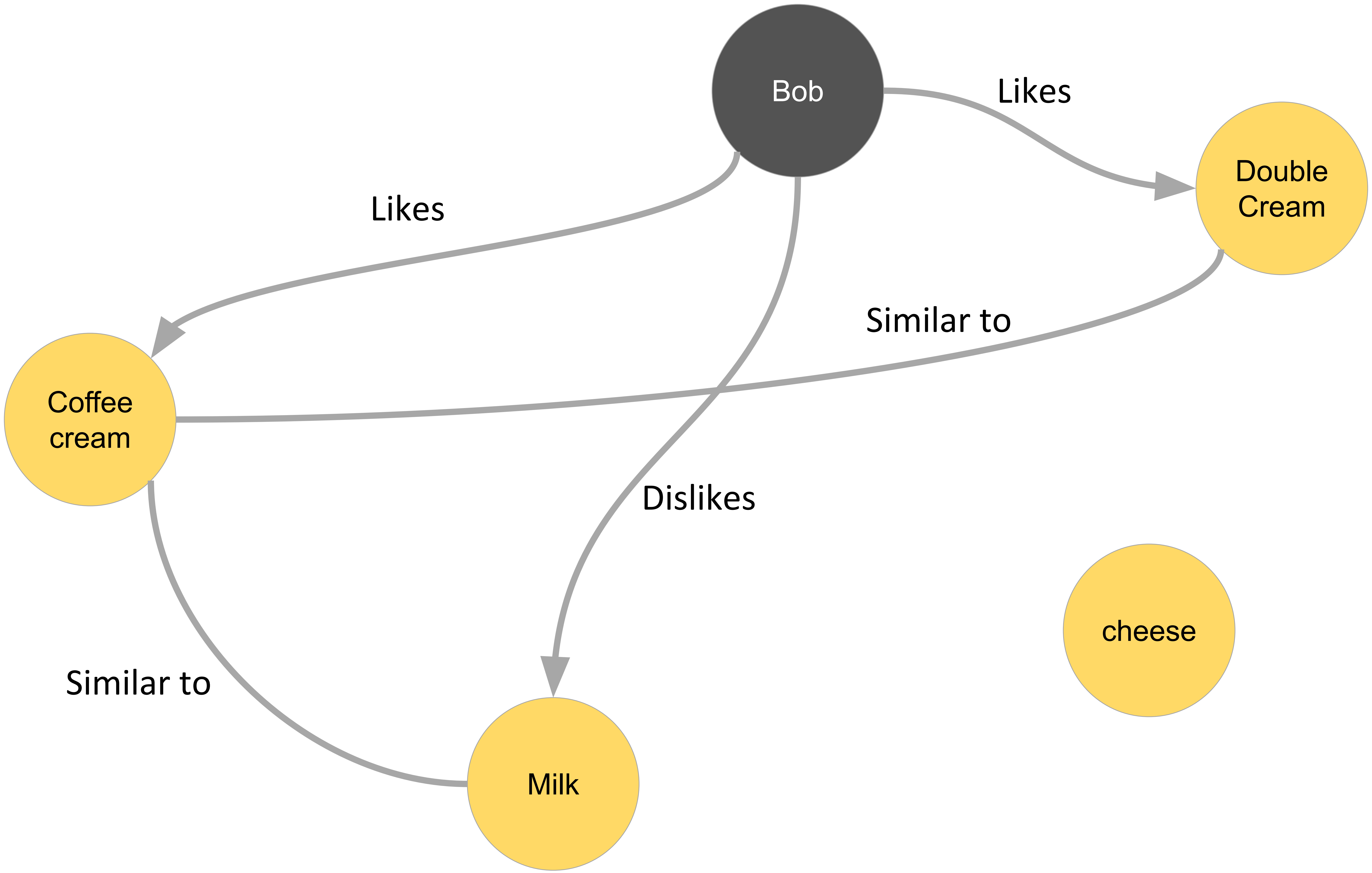
Graphs - Undirected - multigraph



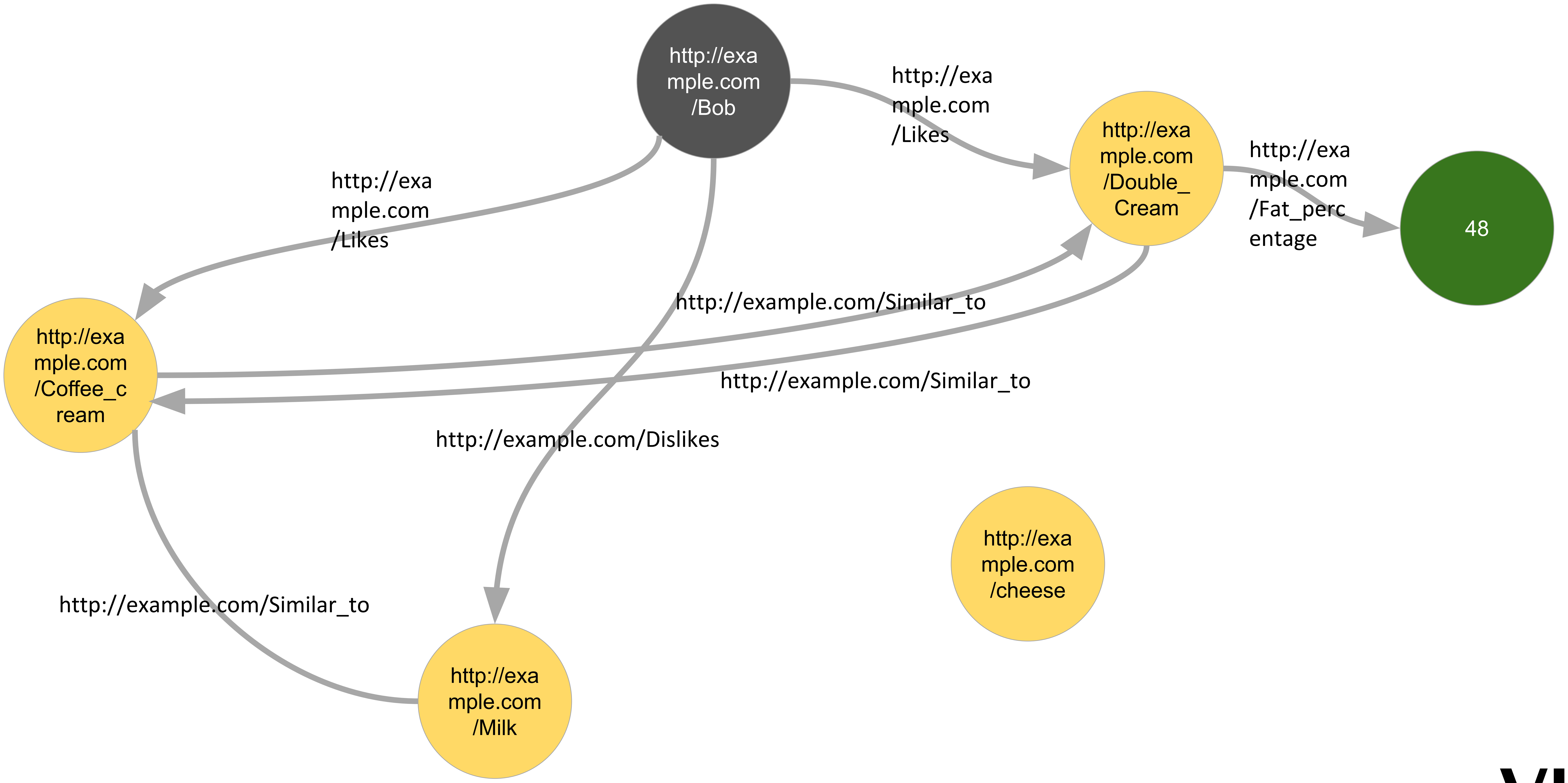
Graphs - Directed



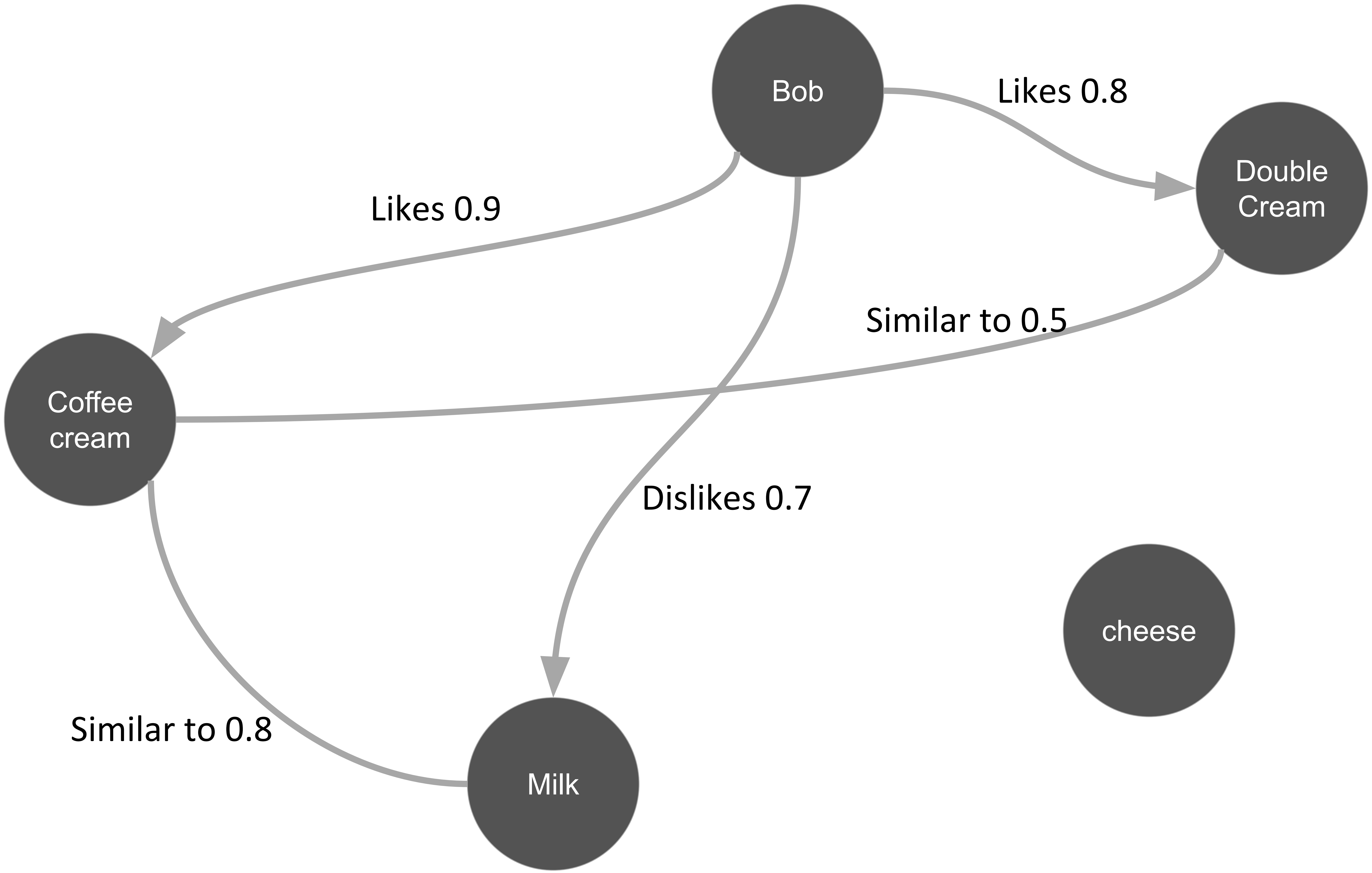
Graphs - Edge and node labels/types



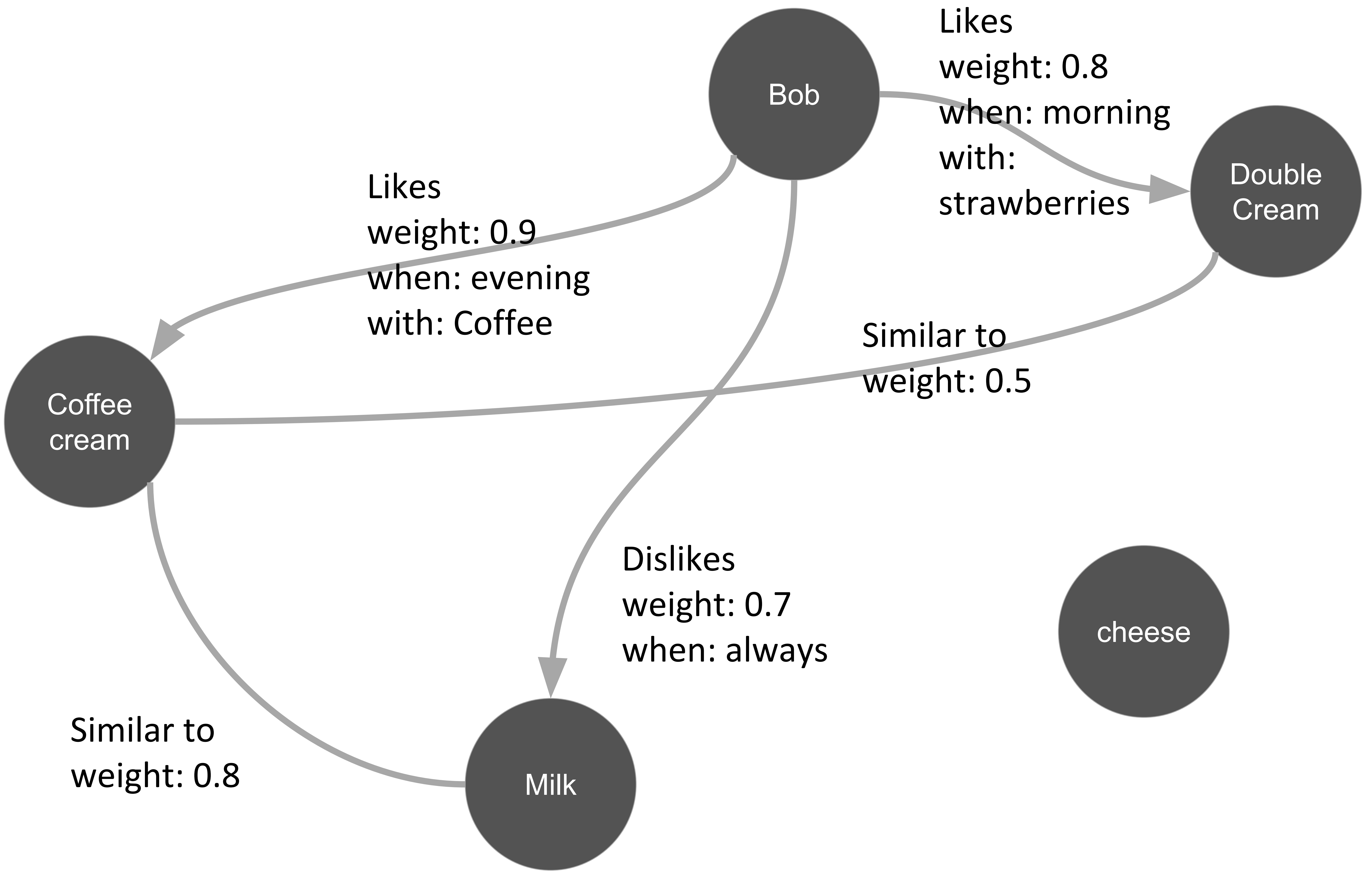
Graphs - Edge and node labels/types - RDF (simplified)



Graphs - Edge and node labels/types + weights



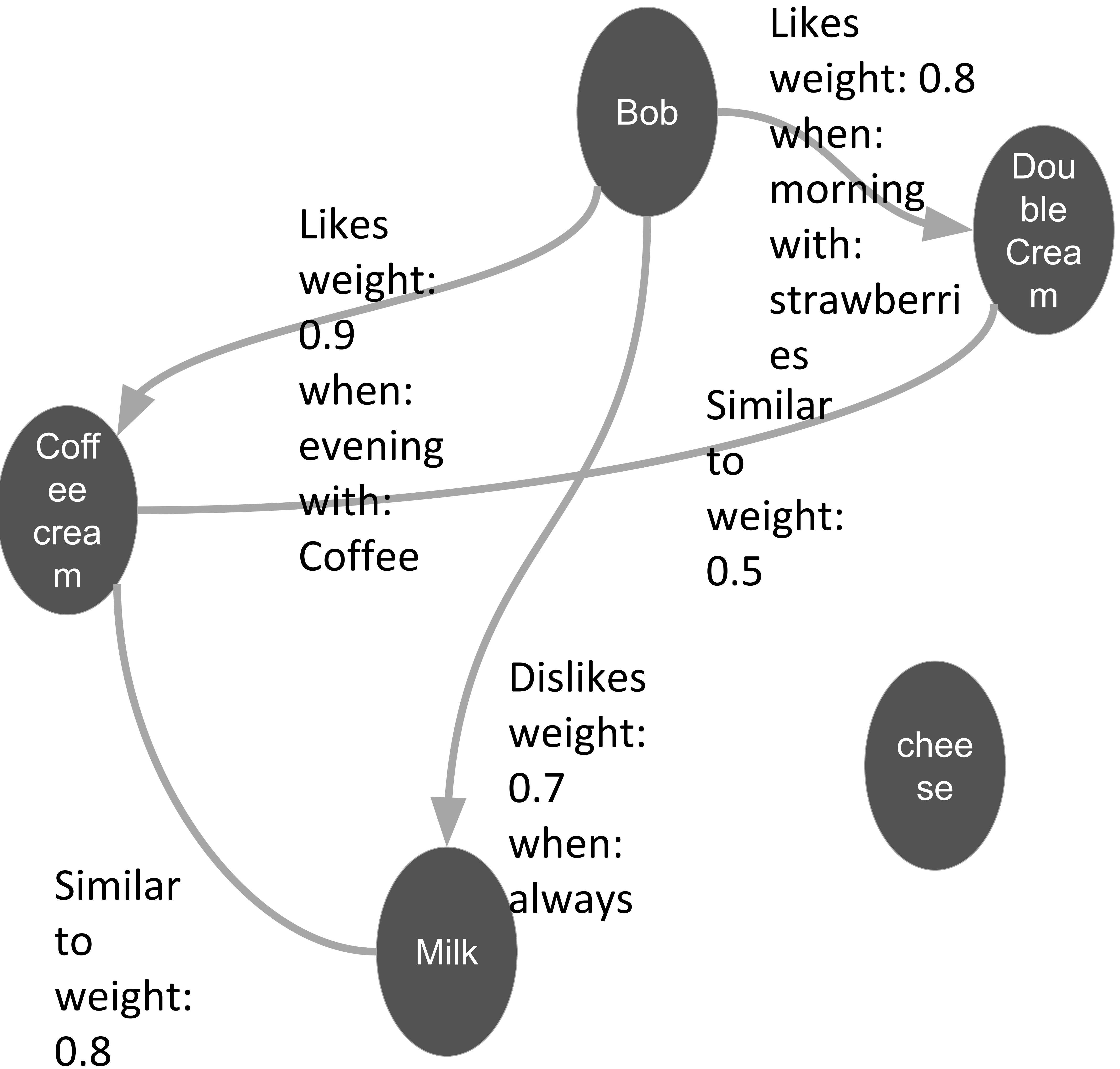
Graphs - Edge and node labels/types + weights

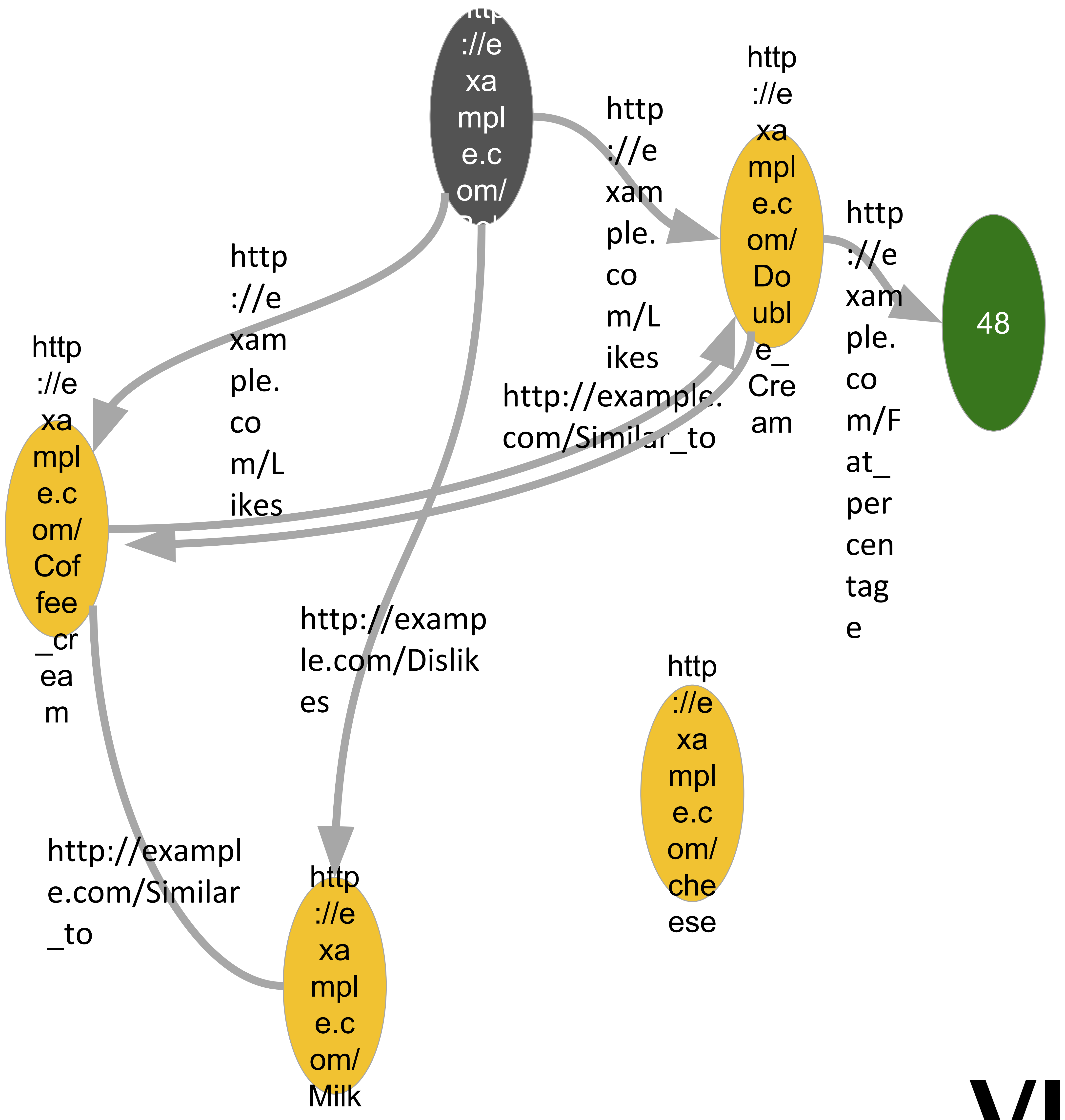


Graphs - summary

- For a given graph, you should know whether it has:
  - Self loops or not
  - Multigraph or not
  - Directed/undirected/mix
  - Edge labels (unique?)
  - Node labels (unique?)
  - Properties on edges (also called qualifiers)
    - Edge weights
  - Any combination of these is possible

What is now a knowledge graph?





PART ONE - B: INTRODUCTION - EMBEDDINGS

Embeddings

Embeddings are low dimensional representations of objects

- Low dimension: Much lower as the original size
- Representation: There is some meaning to it, a representation corresponds to something
- Objects: Words, sentences, images, audio, graphs, ….

Embedding of images

Input Volume (+pad 1) (7x7x3)

`x[:,:,0]`

0	0	0	0	0	0	0
0	1	1	0	2	1	0
0	2	1	0	2	0	0
0	2	2	1	2	2	0
0	0	2	2	1	2	0
0	2	1	0	0	2	0
0	0	0	0	0	0	0

`x[:,:,1]`

0	0	0	0	0	0	0
0	1	2	1	0	2	0
0	2	0	0	0	2	0
0	0	2	0	1	0	0
0	1	2	0	1	2	0
0	1	0	2	0	1	0
0	0	0	0	0	0	0

`x[:,:,2]`

0	0	0	0	0	0	0
0	2	0	0	0	1	0
0	0	0	0	1	0	0
0	1	1	0	2	1	0
0	0	0	0	2	1	0
0	0	1	1	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

`w0[:,:,0]`

-1	0	0
-1	-1	1
0	1	-1

`w0[:,:,1]`

-1	0	-1
1	-1	0
0	1	-1

`w0[:,:,2]`

0	-1	-1
1	1	-1
-1	-1	-1

Bias b0 (1x1x1)

`b0[:,:,0]`

1

Filter W1 (3x3x3)

`w1[:,:,0]`

0	1	1
1	1	-1
-1	1	-1

`w1[:,:,1]`

1	-1	0
1	1	1
-1	1	0

`w1[:,:,2]`

0	1	-1
0	1	1
1	0	0

Bias b1 (1x1x1)

`b1[:,:,0]`

0

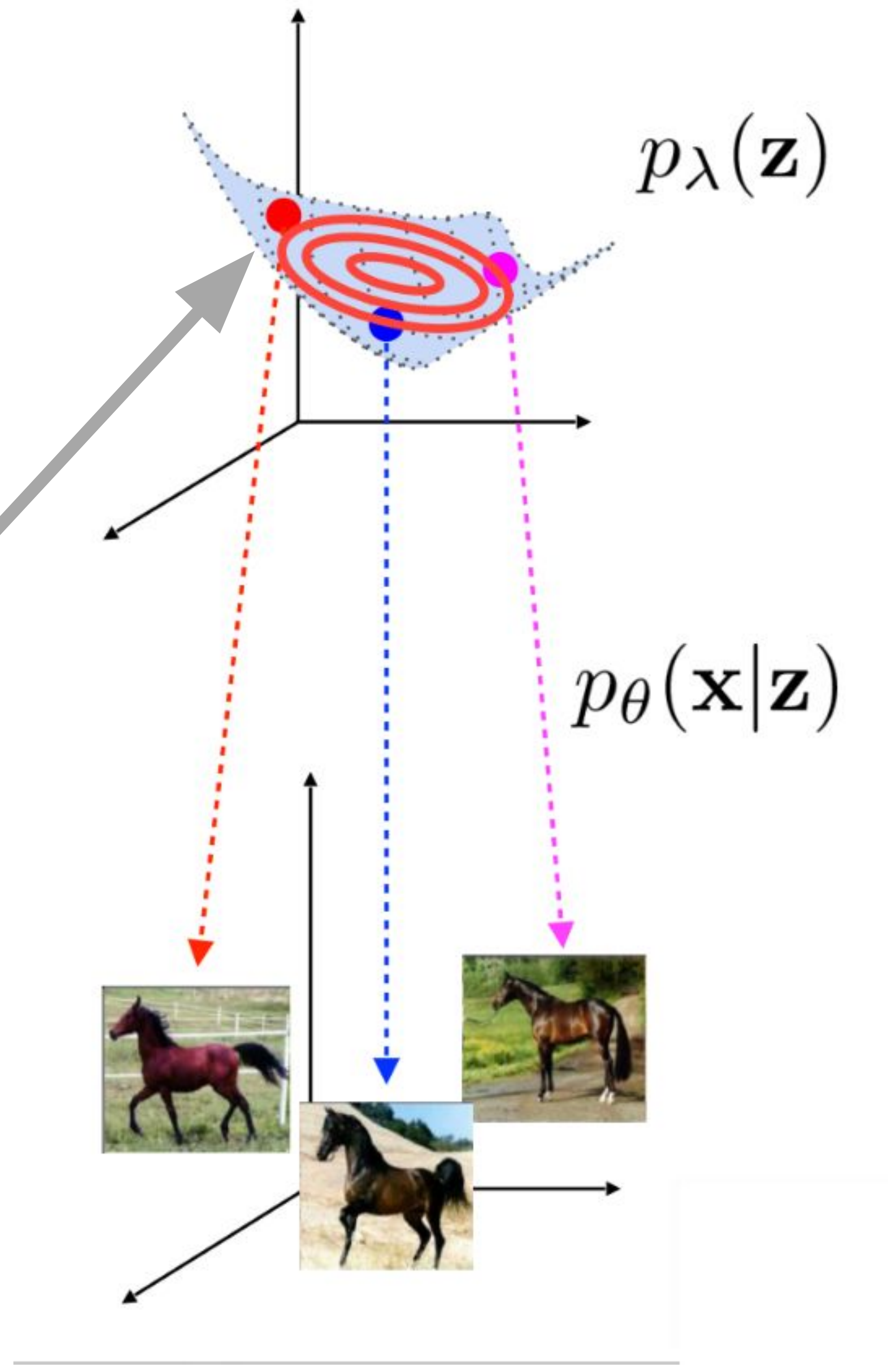
Output Volume (3x3x2)

`o[:,:,0]`

5	0	-2
-2	-3	0
-4	-9	-1

`o[:,:,1]`

8	-1	7
4	4	8
4	9	7



These are embeddings!

image source: https://cs231n.github.io/convolutional-networks/#fc

Embedding of images - navigable space

(a) Smile

(b) Background intensity

(c) Baldness

(d) Gender

https://arxiv.org/abs/1911.05627

Embeddings of words

Distributional hypothesis

- “You shall know a word by the company it keeps” Firth 1957
- “If units of text have similar vectors in a text frequency matrix, then they tend to have similar meaning” Turney and Pantel (2010)

For example, the word ‘cat’ occurs often in the context of the word ‘animal’, and so do words like ‘dog’ and ‘fish’.

But, the word ‘loudspeaker’ hardly ever co-occurs with ‘animal’

Embeddings of words

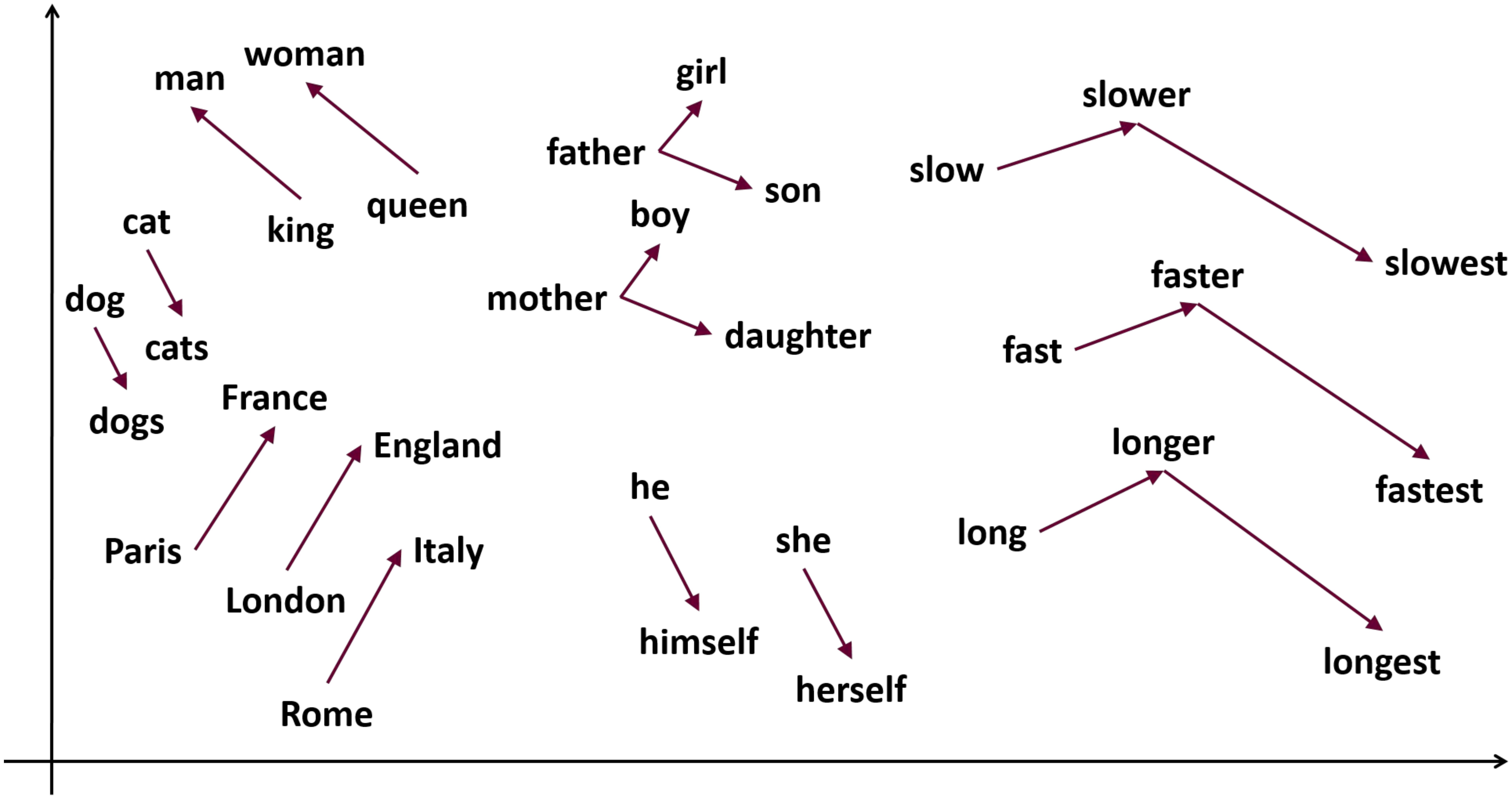
Distributional hypothesis

=> We can use context information to define embeddings for words.

One vector representation for each word

Generally, we can train them (see lecture on word2vec)

Embedding of words - navigation



https://samyzaf.com/ML/nlp/nlp.html

Graphs - why use them as input to machine learning?

Classification, Regression, Clustering of nodes/edges/whole graph

Recommender systems (who likes what)

Document modeling

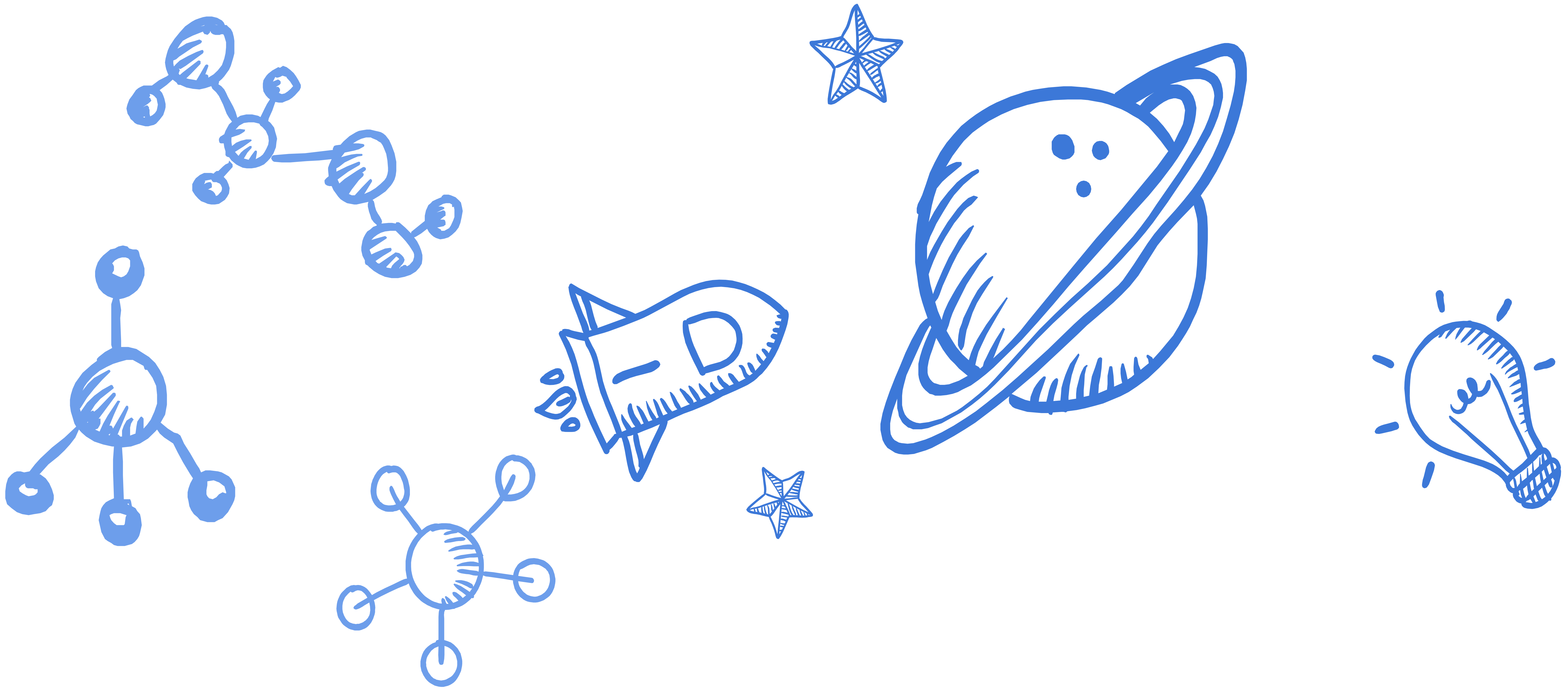
- Entity and Document similarity (Use concepts from a graph)

Alignment of graphs (which nodes are similar?)

Link prediction and error detection

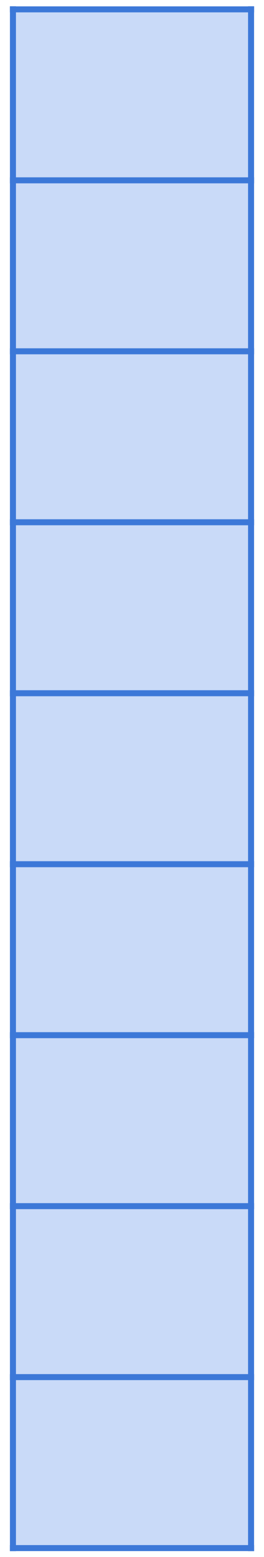
Linking text and semi-structured knowledge to graphs

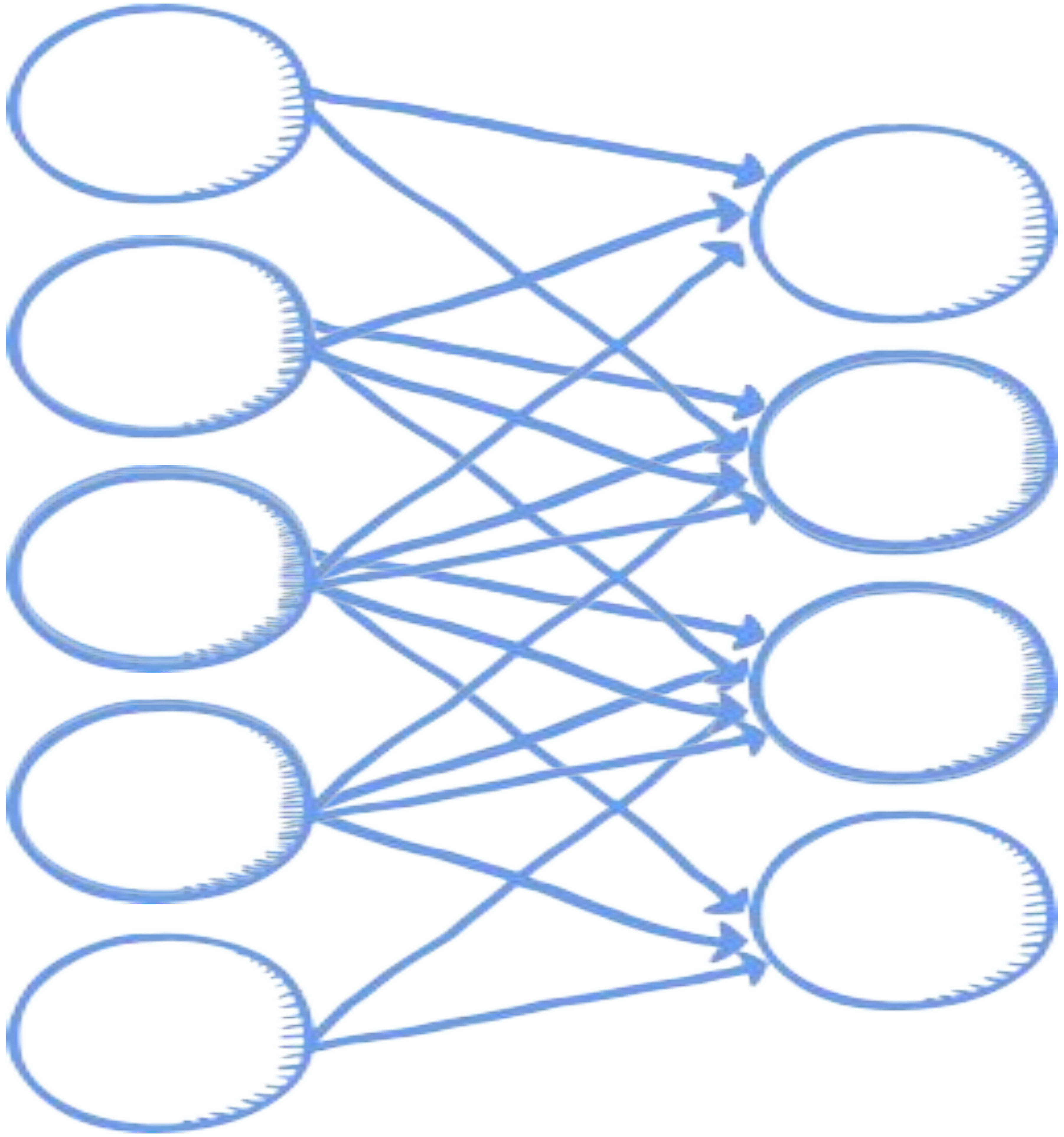
PART TWO: Graph Embedding Techniques



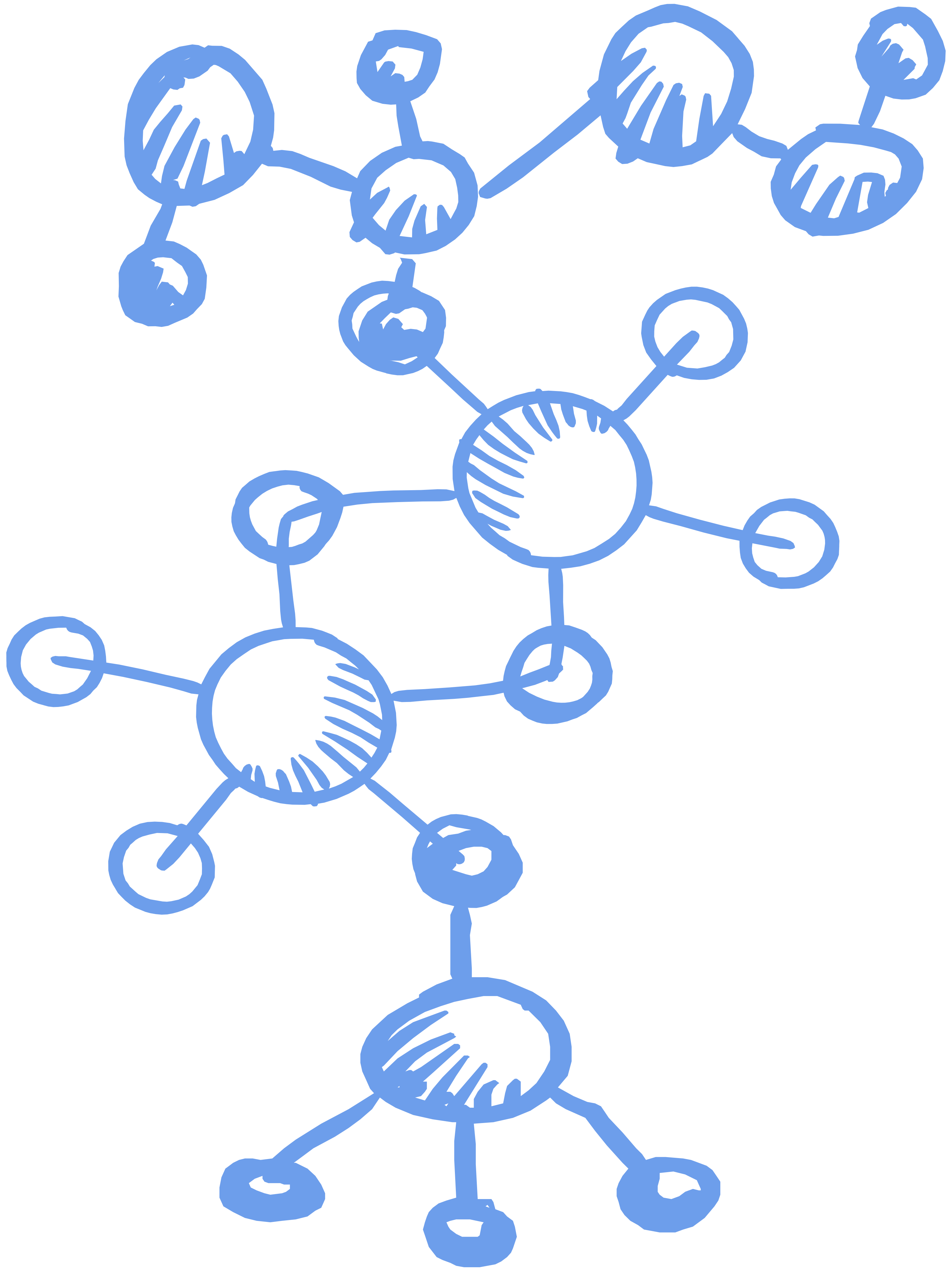
The Challenge

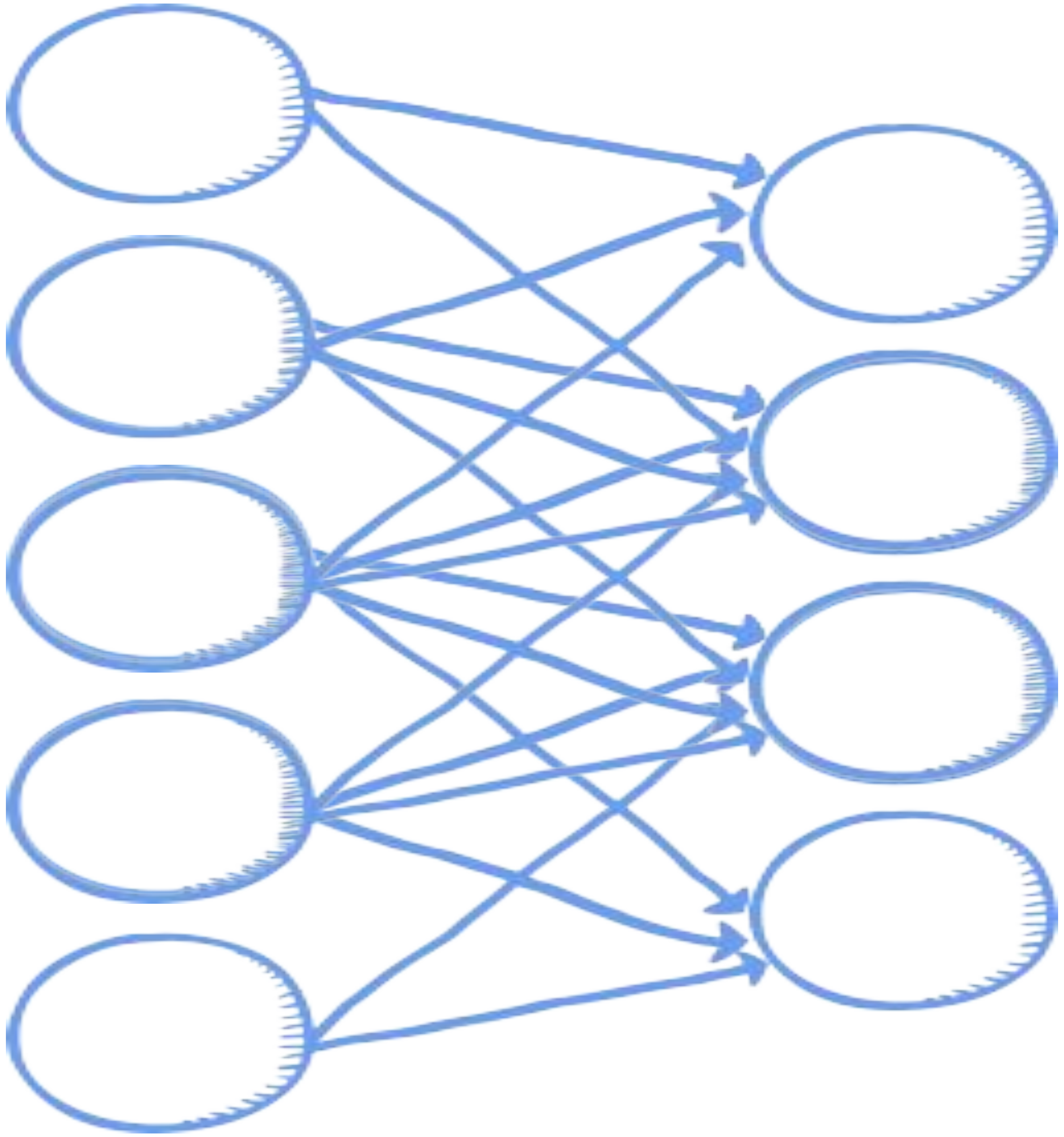
Graphs - model mismatch





Graphs - model mismatch

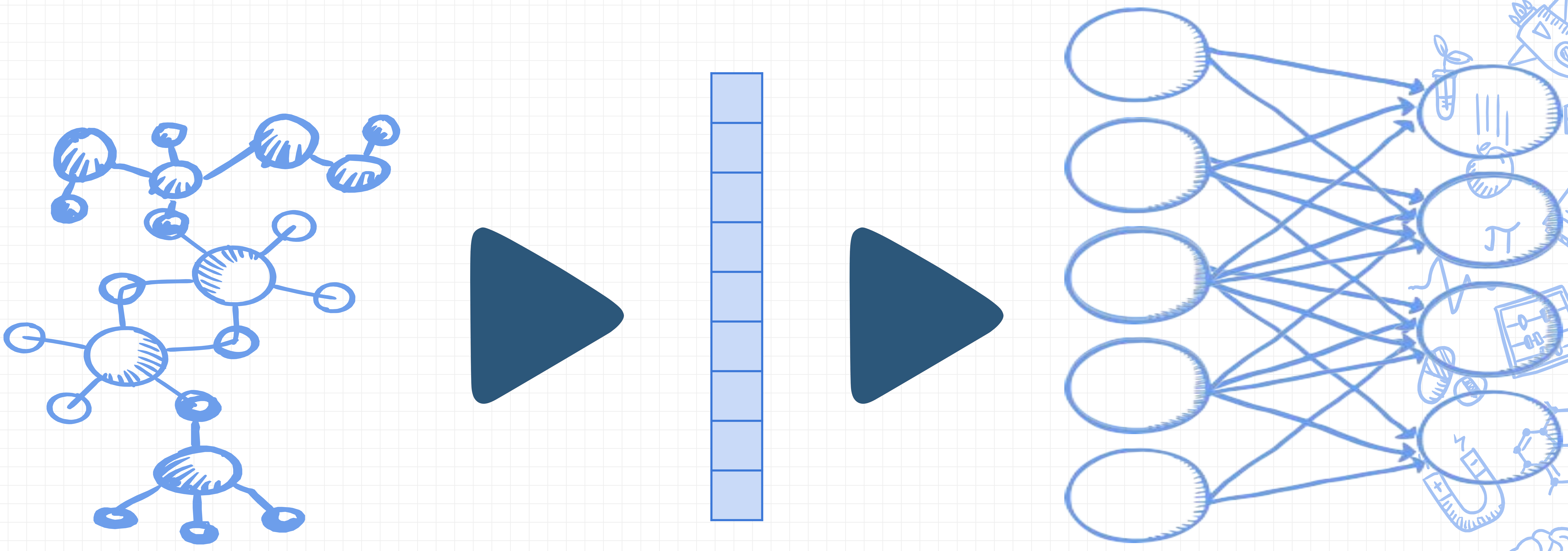




What we skip

- Traditional ML on graphs
    - Often have problems with scalability
    - Often need manual feature engineering
        - Task Specific

Embedding Knowledge Graphs in Vector Spaces



Embedding – propositionalization

- ✘ One vector for each entity
  - ✗ Compatible with traditional data mining algorithms and tools
- ✘ Preserve the information
- ✘ Unsupervised
  - ✗ task and dataset independent
- ✘ Efficient computation
- ✘ Low dimensional representation

Two Major Visions

Preserve Topology

- Keep neighbours close together

- Europe - Germany
- Africa - Algeria

Preserve Similarity

- Keep similar nodes close together

- Europe - Africa
- Germany - Algeria

Two Major Targets

Improve original data

- Knowledge Graph Completion
- Link Prediction
- Anomaly Detection

Downstream ML Tasks

- Classification/Regression/Clustering/K-NN
- Then used as part of a larger process
  - QA/dialog systems
  - Translation
  - Image segmentation

How ?

Three Major Approaches for Propositionalization

Translational

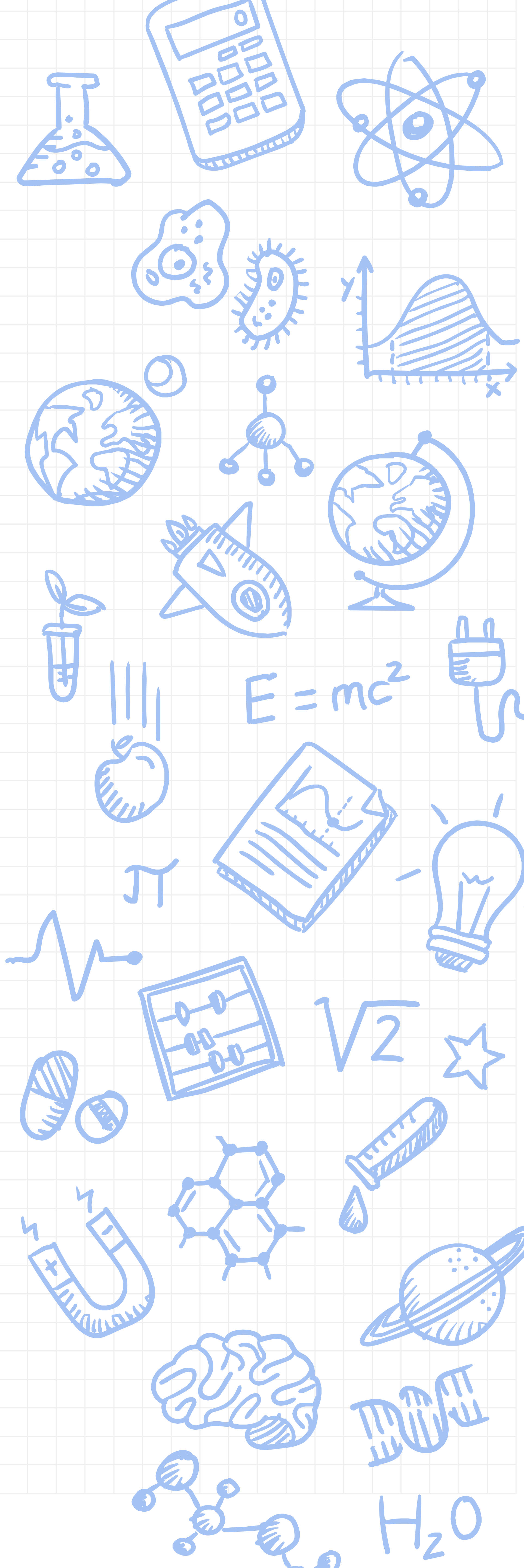
- Interpret relations as translations of concept in the embedded space

Tensor Factorization

- Make a 3D matrix and factorize it
- Reconstructing the original hints which edges were missing

Random Walk Based

- Use the context of a concept to embed it
- Use the distributional hypothesis

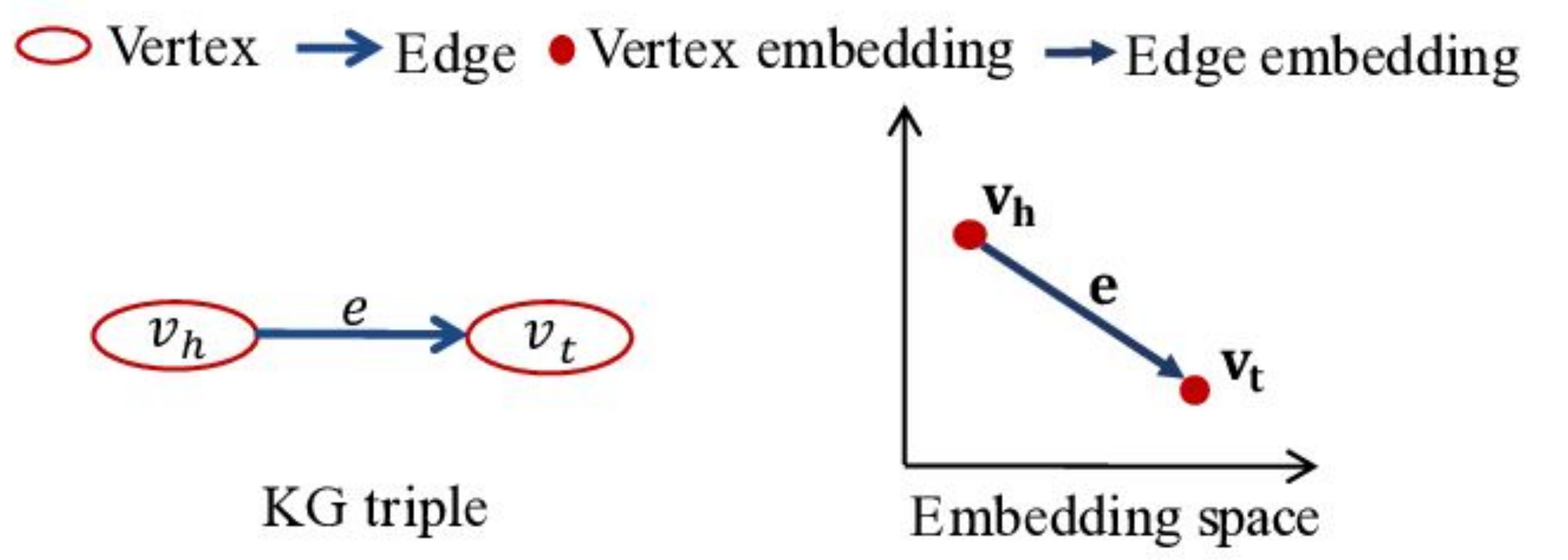


Translational

XTransX – translational embedding (Bordes et al. NIPS 2013 , Lin et al., AAAI'15)

- ✘ TransE
- ✘ TransH
- ✘ TransR
- ✘ CTransR
- ✘ PTransE
- ✘ …

TransE – translational embedding (Bordes et al. NIPS 2013)



(a) An illustration of TransE

Source: Structured query construction via knowledge graph embedding, 2019, Ruijie Wang et al.

$$\overrightarrow{V_h} + \vec{l} \simeq \overrightarrow{V_t}$$

$$\min \sum_{(h,l,t)\in S} d(\vec{h} + \vec{l}, \vec{t})$$

TransE – translational embedding (Bordes et al. NIPS 2013)

- TransE
  - Get h+l close to t
    - If (h,l,t) is a good triple
  - Get h+l far from t
    - If (h,l,t) is a bad triple

$$\mathcal{L} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} \left[\gamma + d(h+\ell,t) - d(h'+\ell,t')\right]_+$$

XTransX – translational embedding (Bordes et al. NIPS 2013 , Lin et al., AAAI'15)

- ✘ TransE
- ✘ TransH
- ✘ TransR
- ✘ CTransR
- ✘ PTransE
- ✘ …

Conceptually Easy

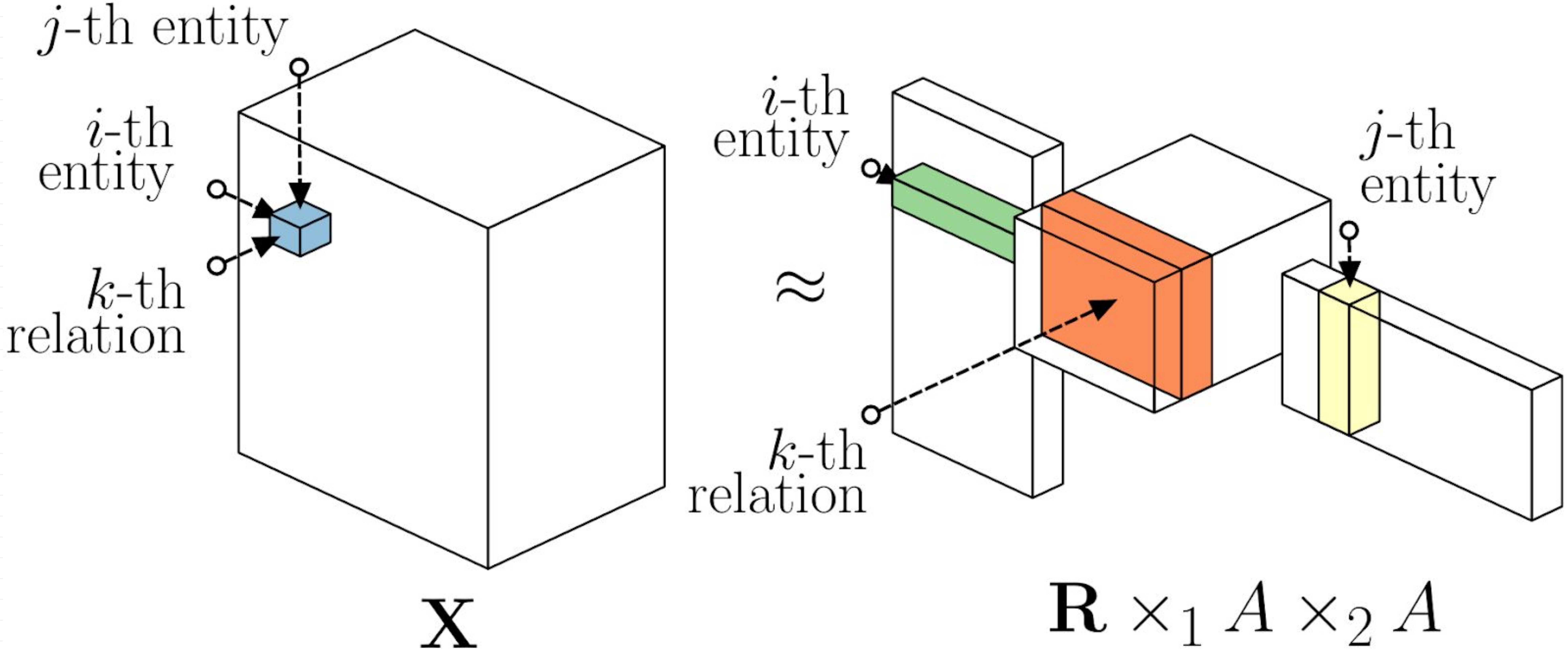
Embedding Quality

The better the model, the less scalable

One Hop

Matrix Factorization

RESCAL – Tensor Factorization
(Nickel and Tresp, ECML PKDD 2013)



Tensor Factorization

Conceptually Easy

Good for link prediction

Usually Scalable

Multi hop

Explainable

Numeric attributes can be included somehow

Embedding Quality for ML tasks

The better the model, the less scalable

Random Walk Based

Random Walk based methods
(Cochez, et al. , ISWC '17, Cochez, et al. WIMS'17, Ristoski et al. ISWC '16, Grover, Leskovec KDD '16)

- Use random walks to extract a context for each node
- Use this context as input to word embedding techniques

Random Walk based methods – RDF2Vec and Node2Vec (Cochez, et al. WIMS'17, Ristoski et al. ISWC '16, Grover, Leskovec KDD '16)

- Use random walks to create sequences on the graph.
- Feed these to word2vec

- Biasing the walks helps for specific cases
- Also some other graph kernels have been used

Global Embeddings – KGloVe
(Cochez, Ristoski, et al. , ISWC '17)

- Create co-occurrence stats using Personalized PageRank
  - All pairs PPR
- Apply the GloVe model
  - Similar things will have similar contexts
  - Optimizes for preserving analogy
    - King – Man + Woman ≃ Queen
    - Berlin – Germany + Austria ≃ Vienna
  - Complete context captured

Random walk based methods
(Cochez, et al. , ISWC '17, Cochez, et al. WIMS'17, Ristoski et al. ISWC '16, Grover, Leskovec KDD '16)

Deals with large Graphs

Good Embeddings

Good Training Time

Likely or partially Explainable

Larger Context Used

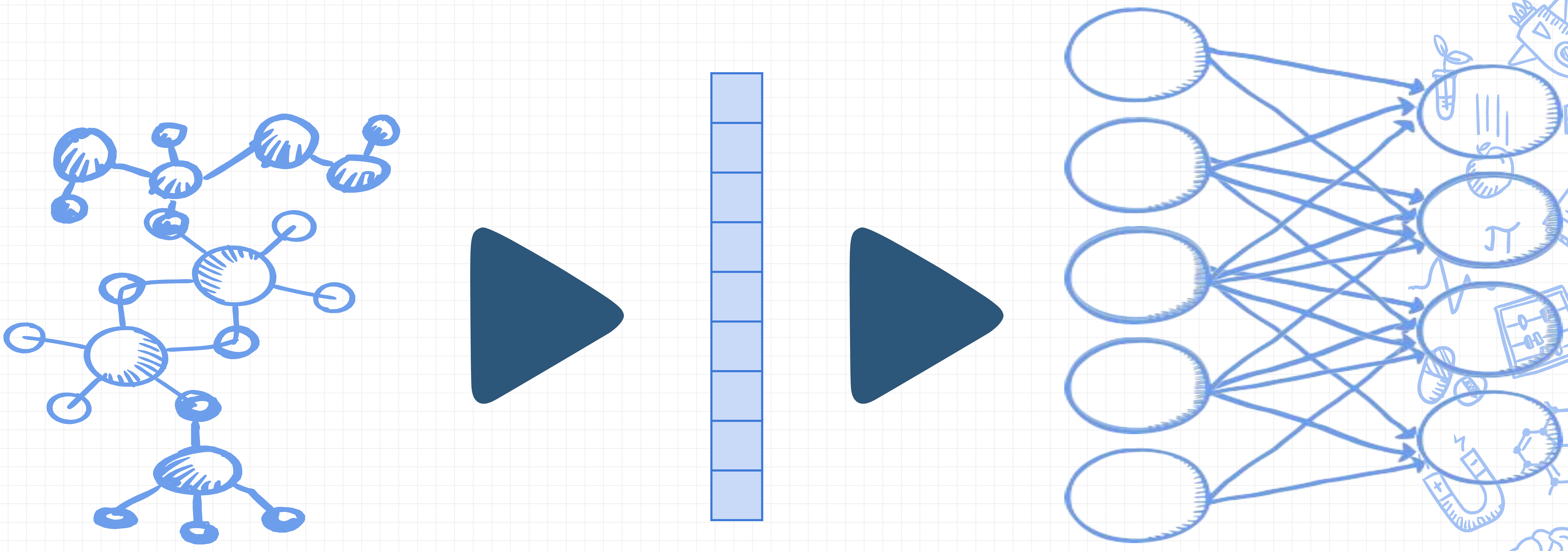
Link Prediction

PART THREE: Graph Neural Networks

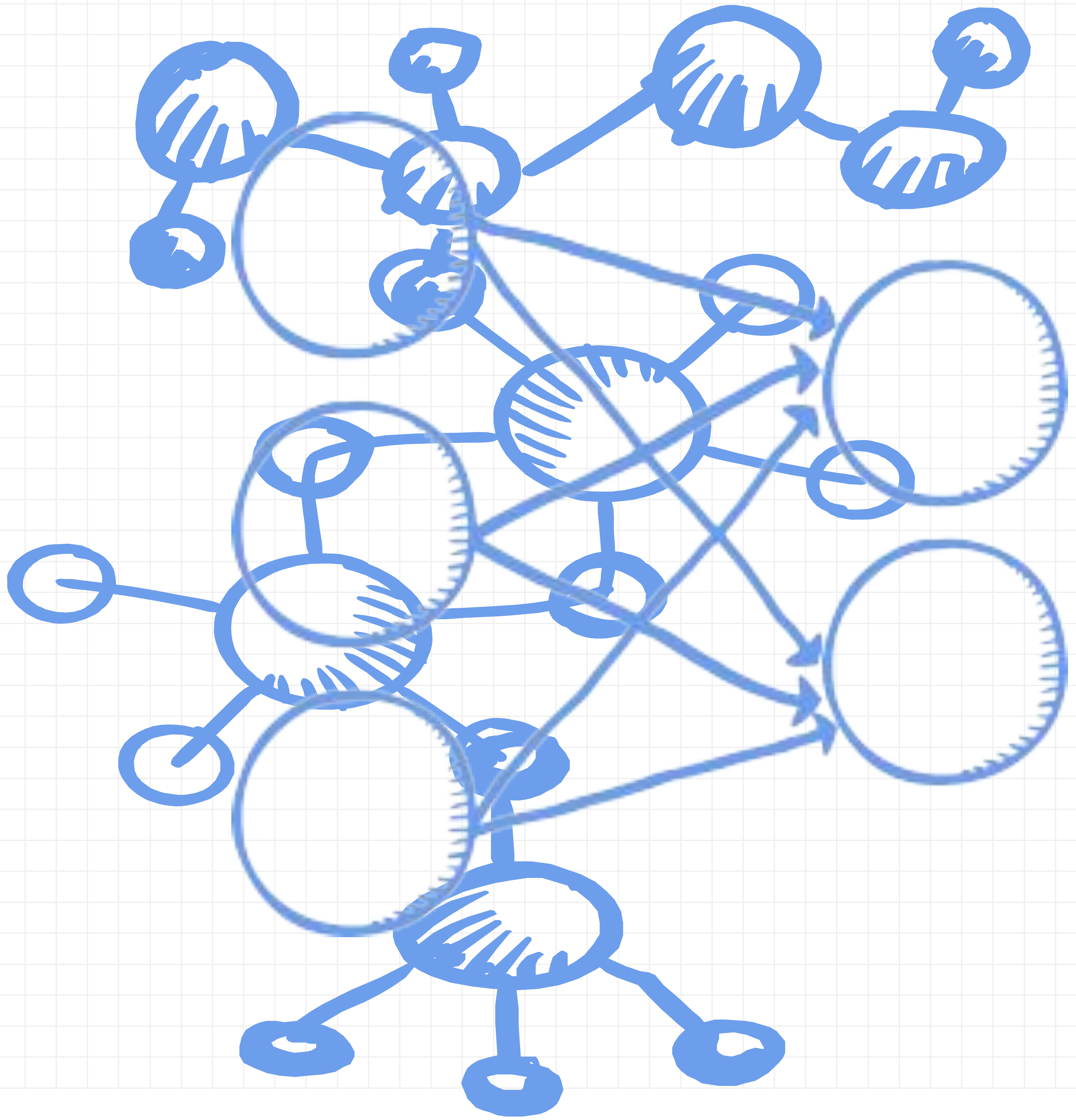
Graph Convolutional Networks (GCN: Kipf and Welling, ICLR'17, RGCN: Schlichtkrull et al. ESWC'18)

- How can we directly incorporate graph information into a machine learning algorithm?
  - Especially for end-to-end learning

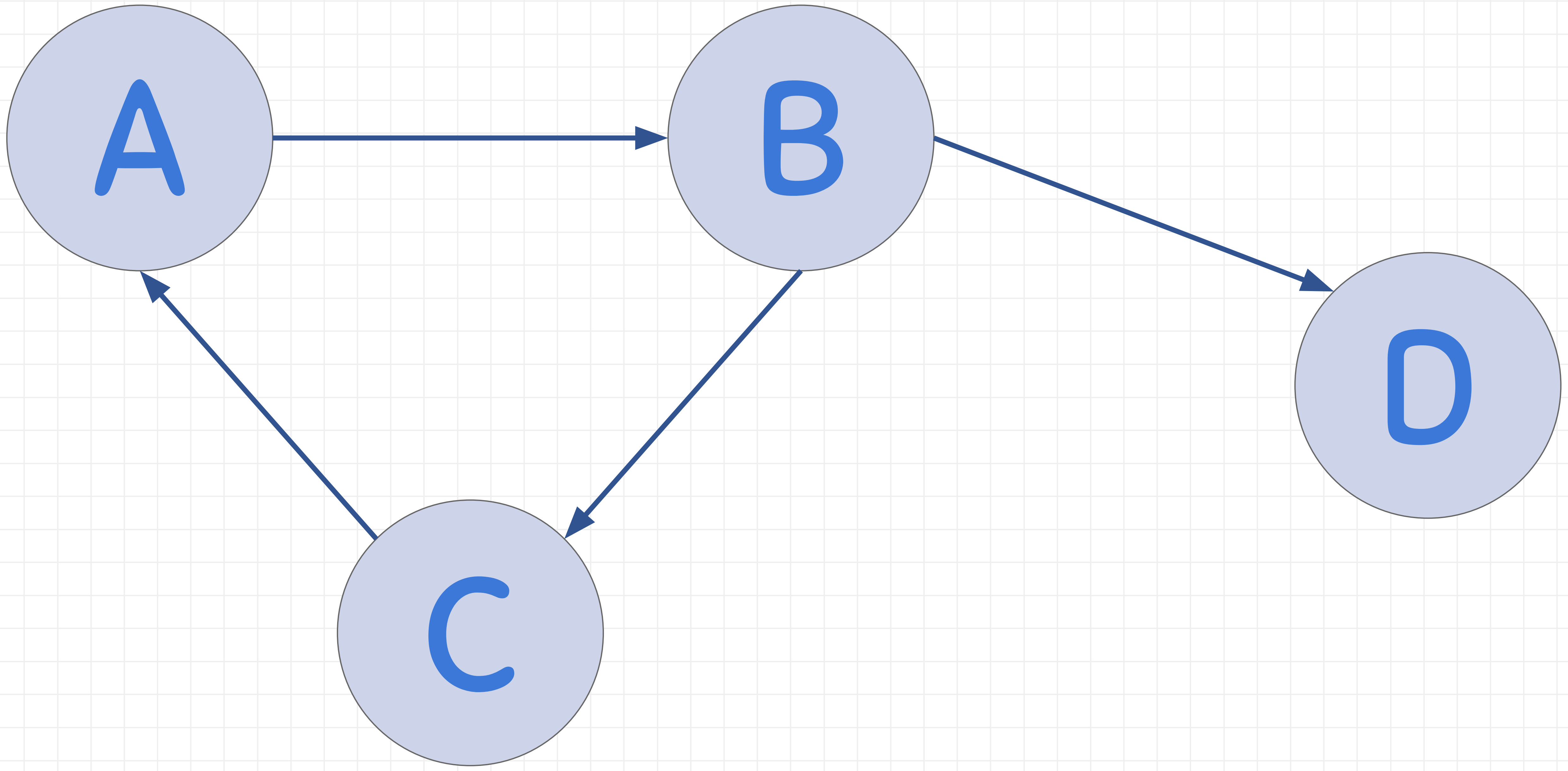
Embedding



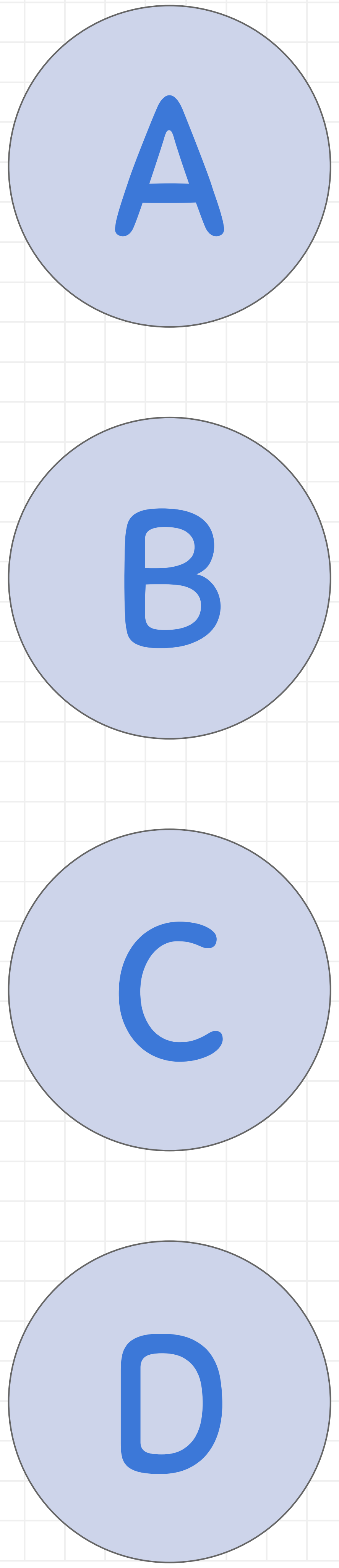
Graph Convolutional Network – Merge the two worlds

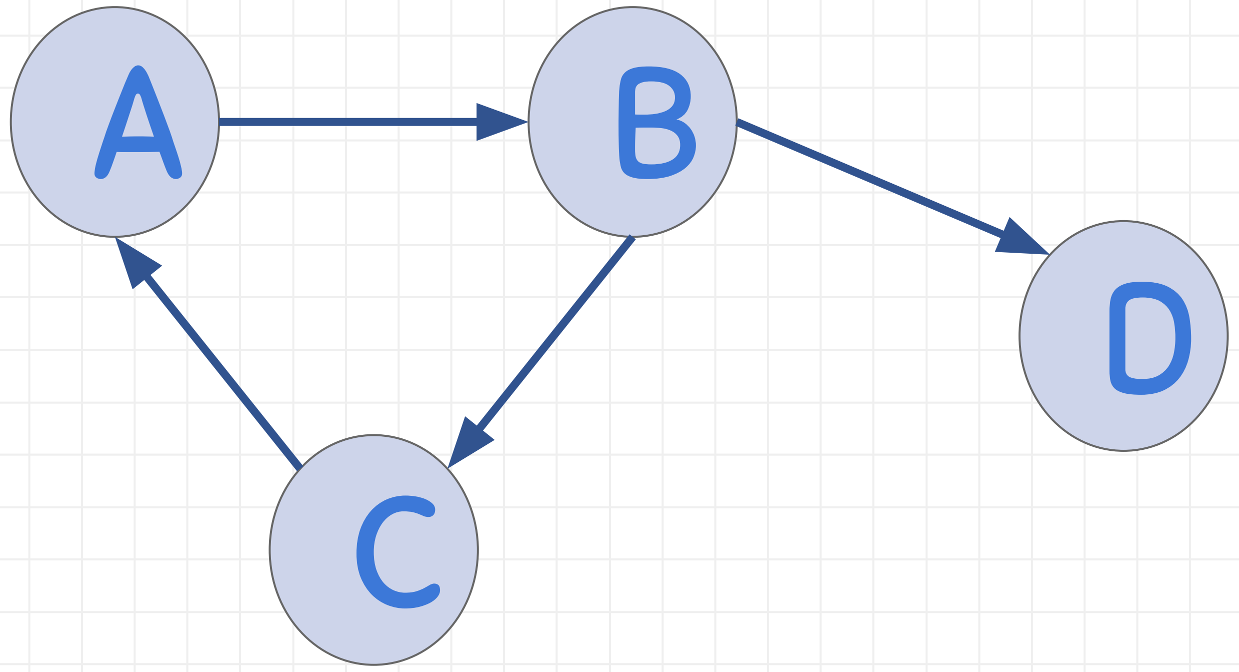


GCN – Example Graph

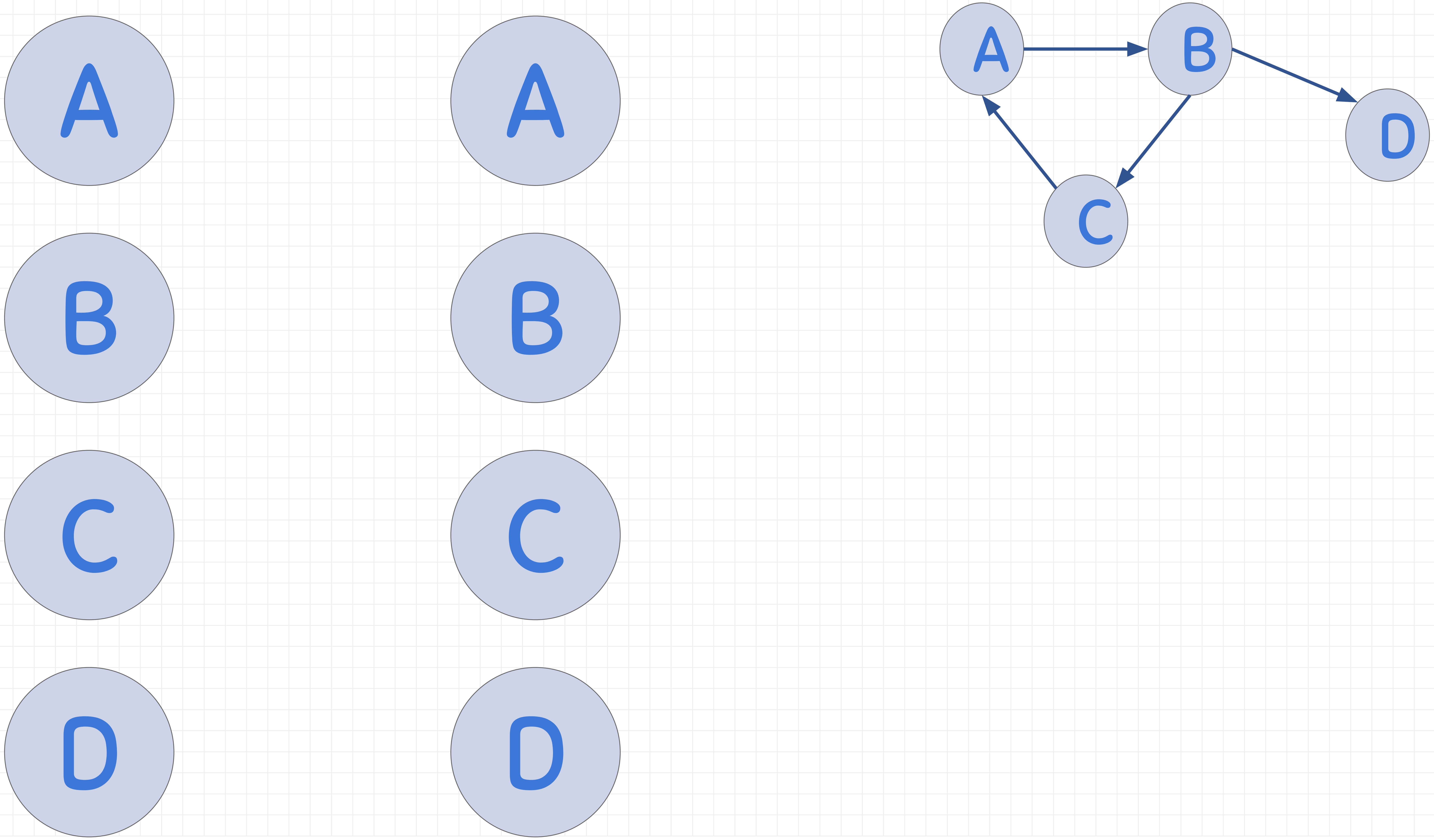


GCN – Example Graph – 1 Layer

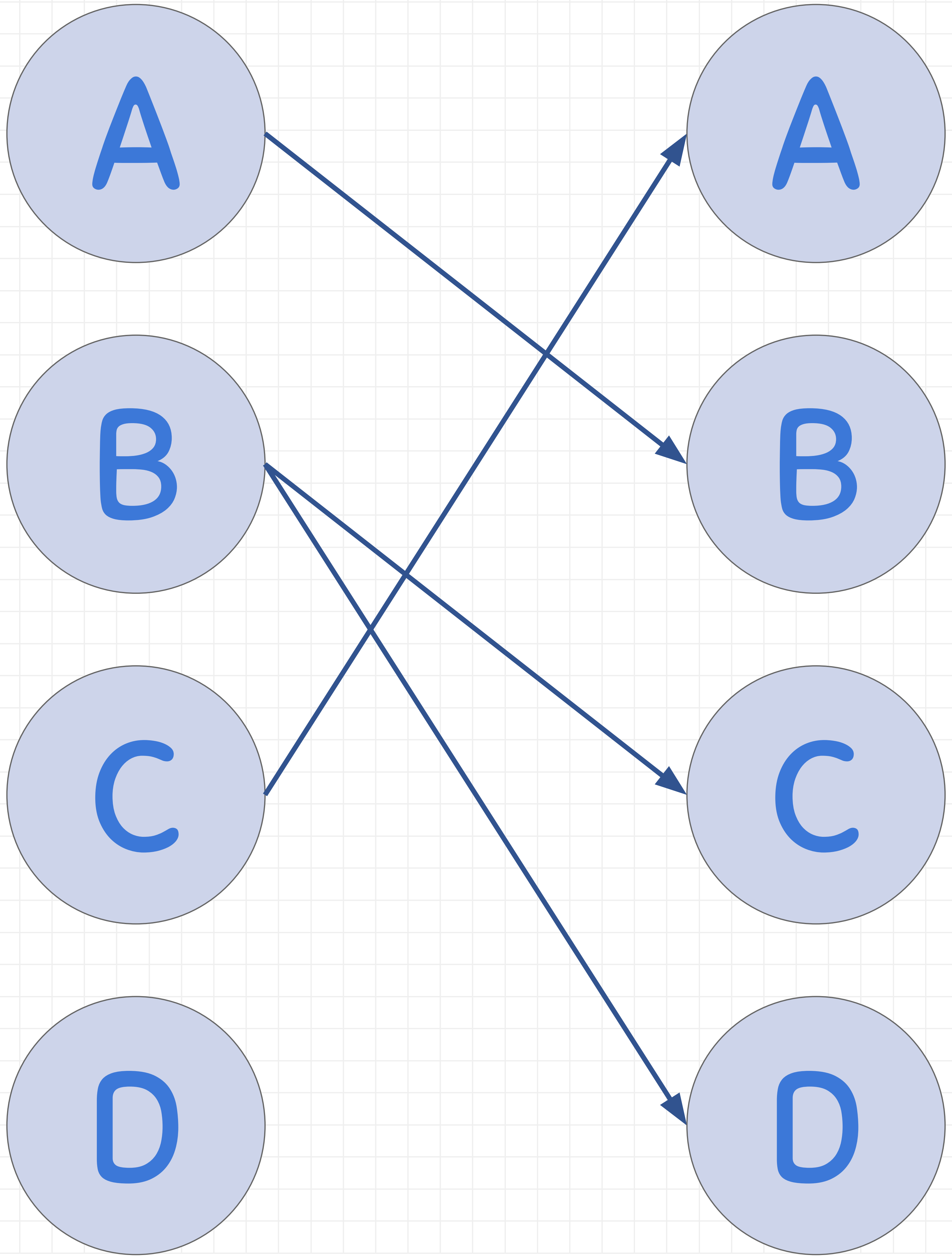


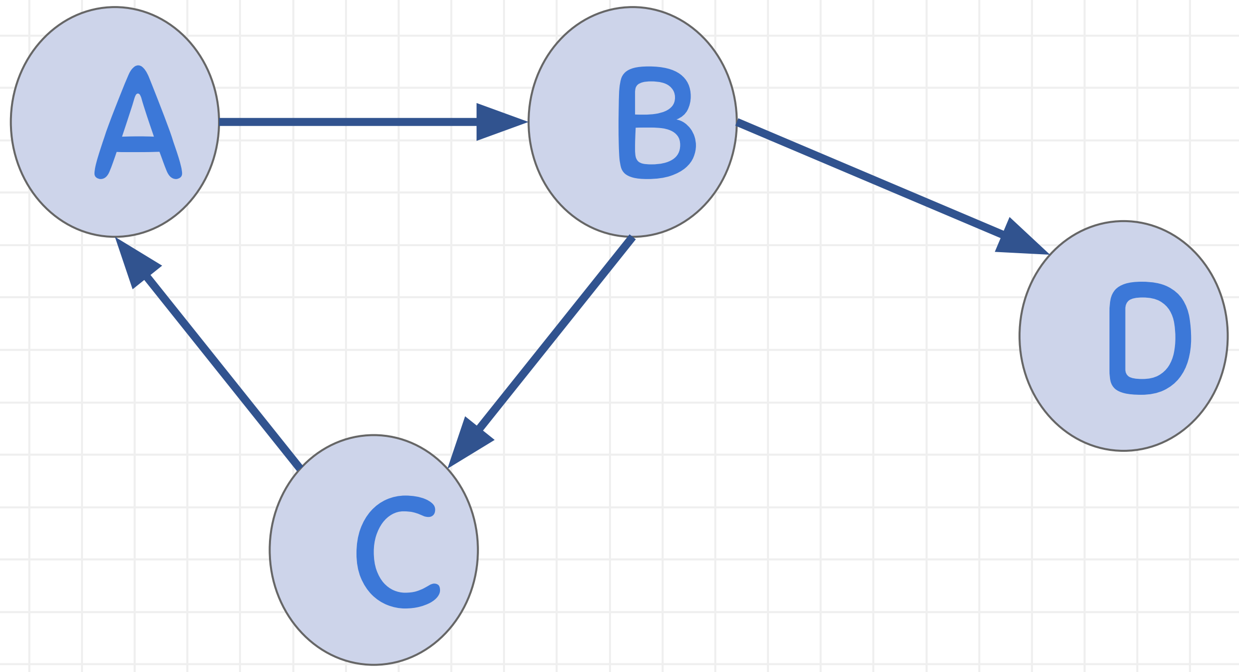


GCN – Example Graph – 2 Layer

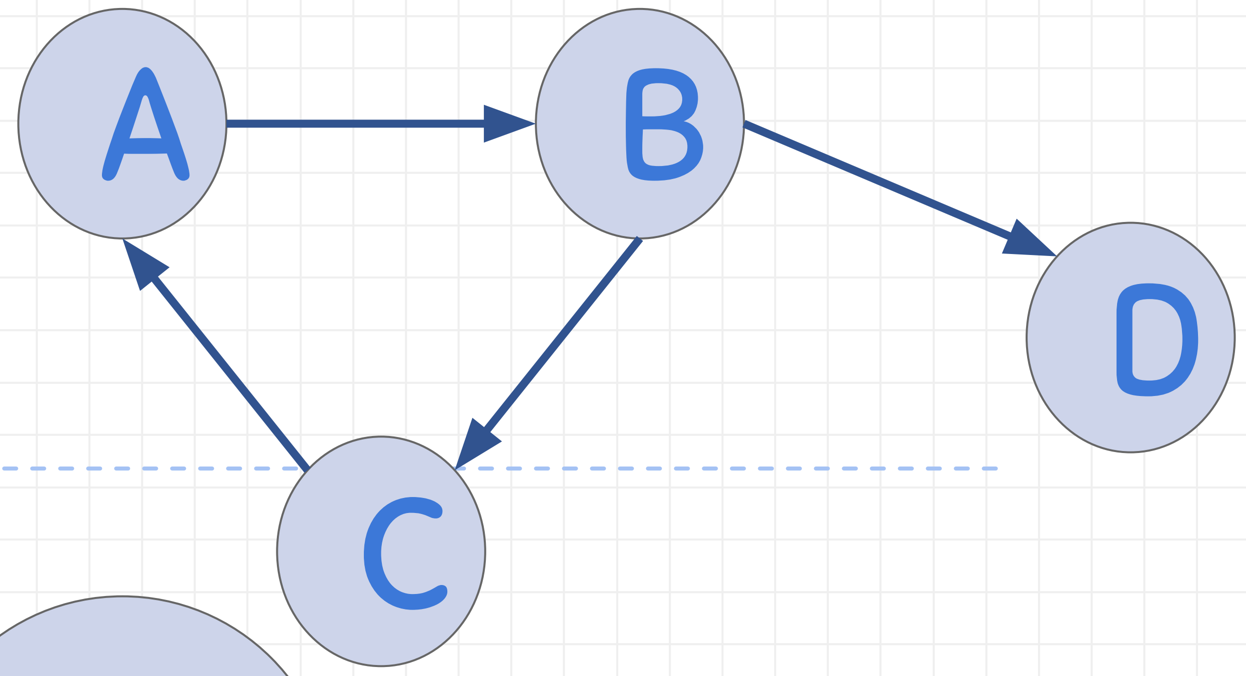


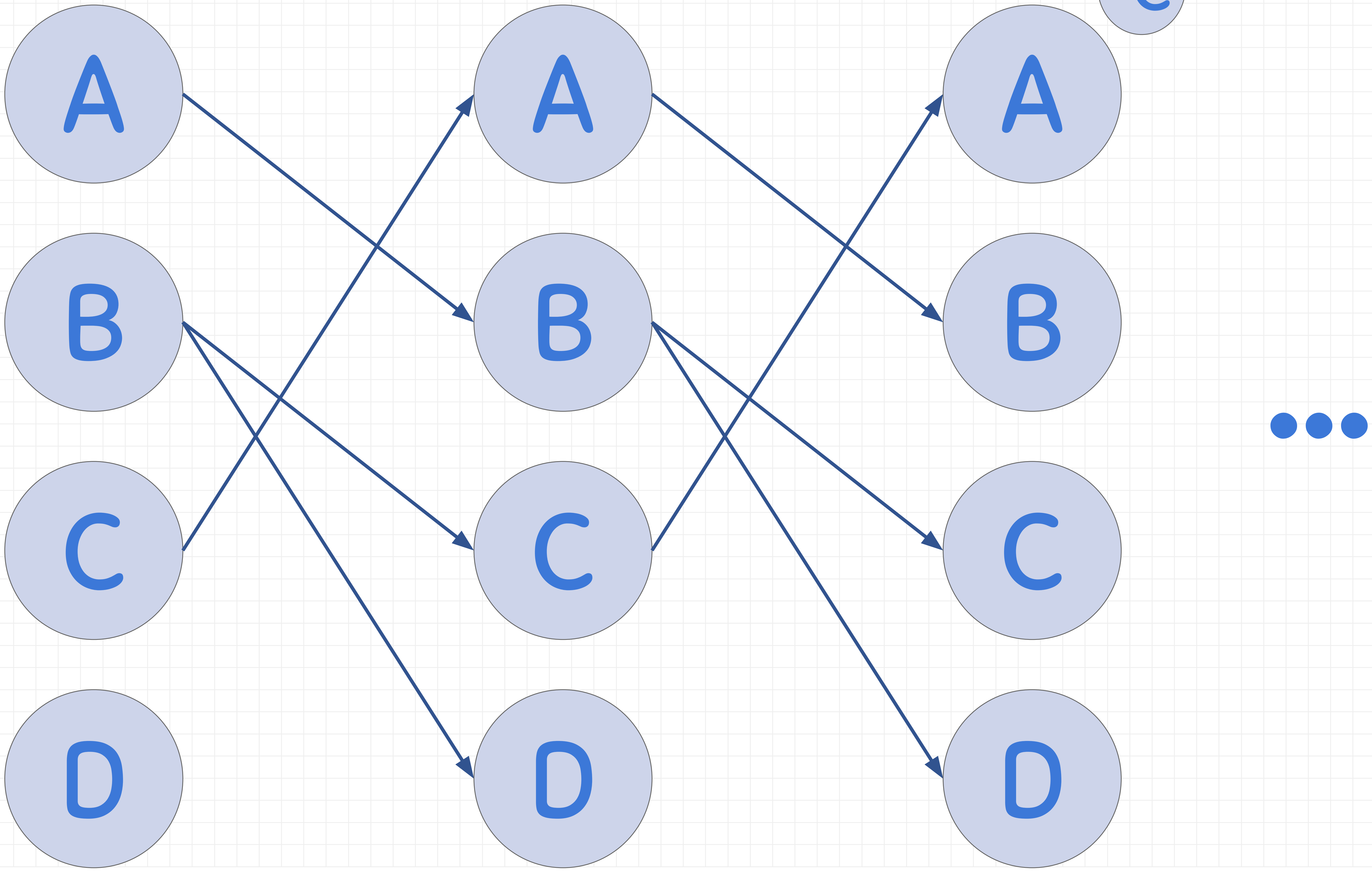
GCN – Example Graph – 2 Layer with Connections



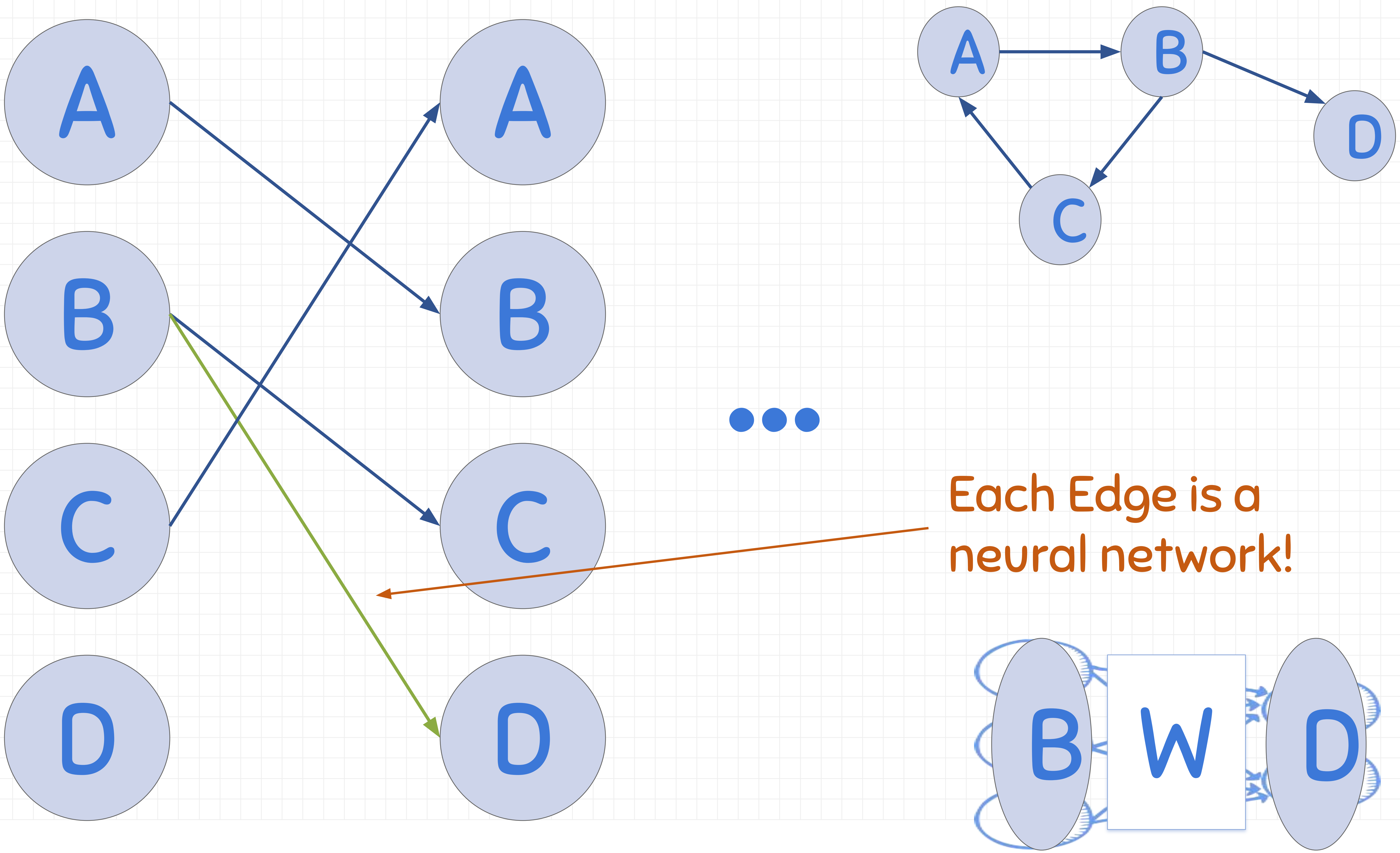


GCN – Example Graph – Multi Layer



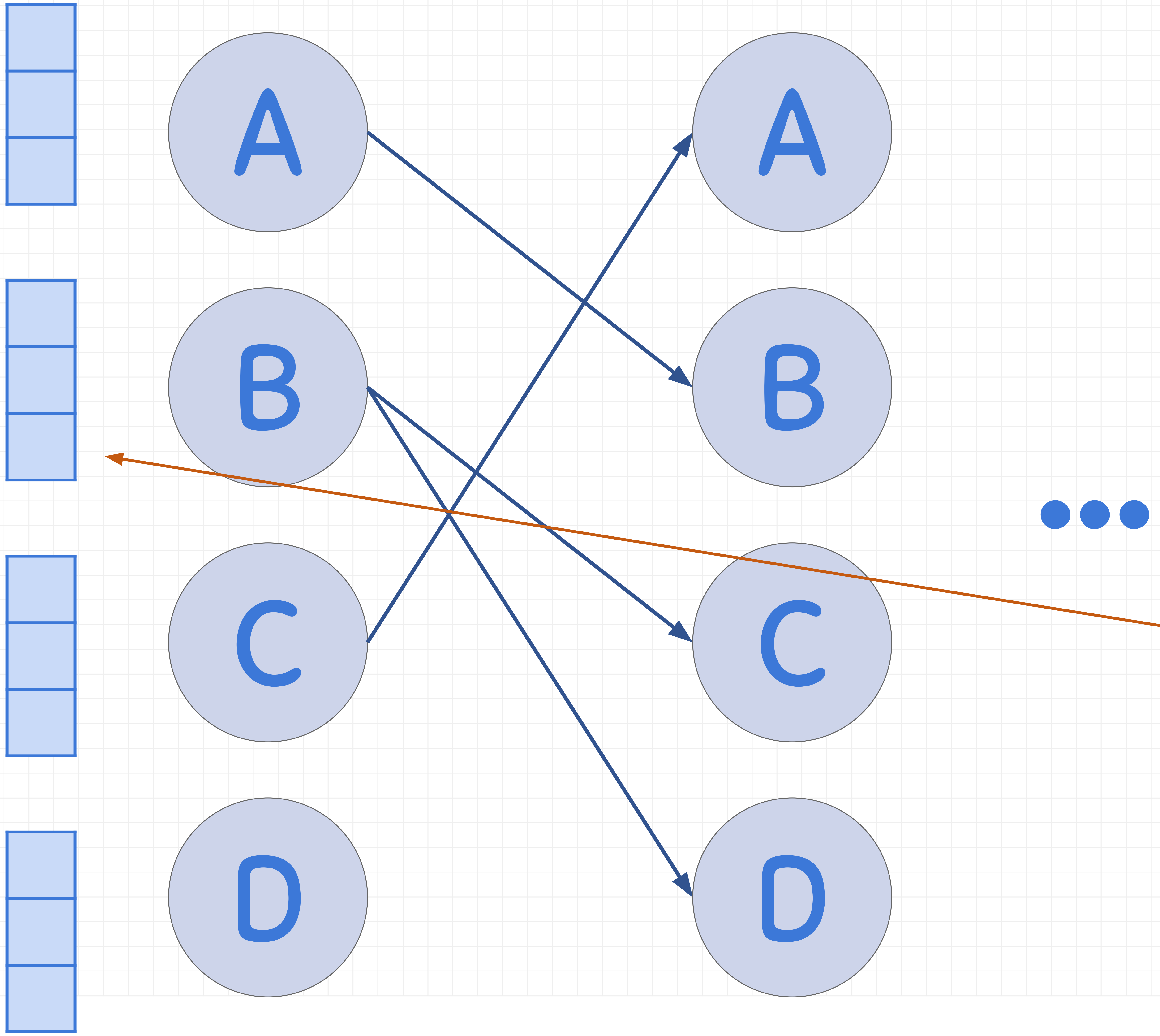


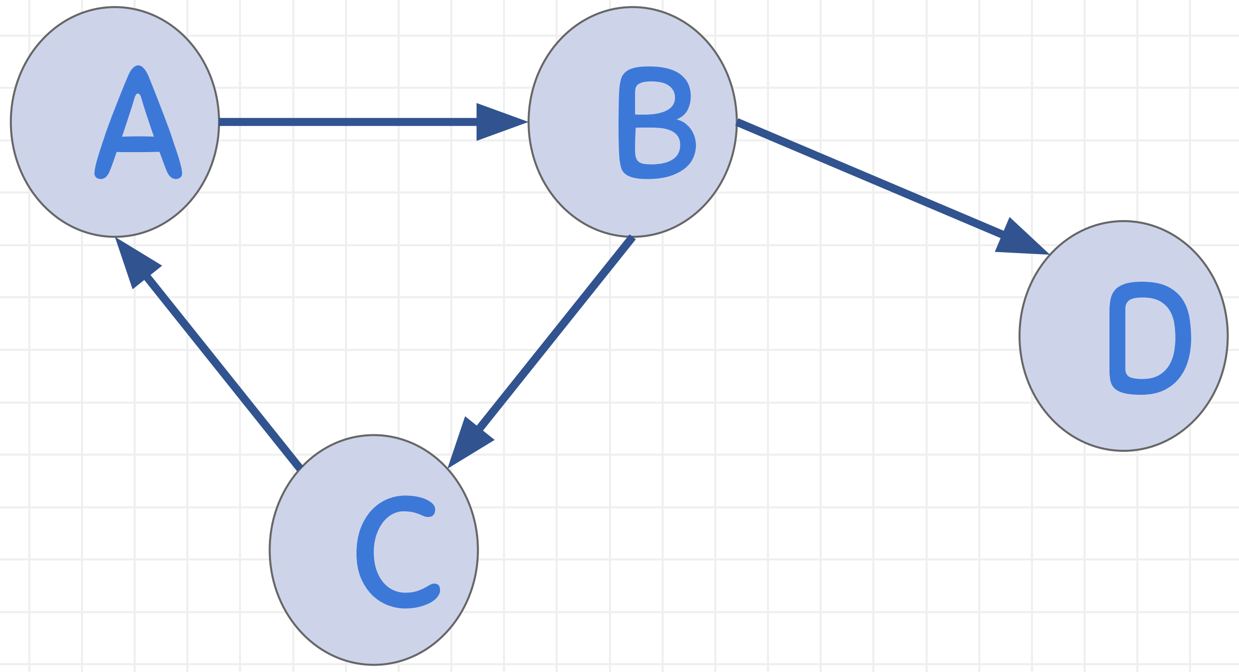
GCN – Example Graph – Weights



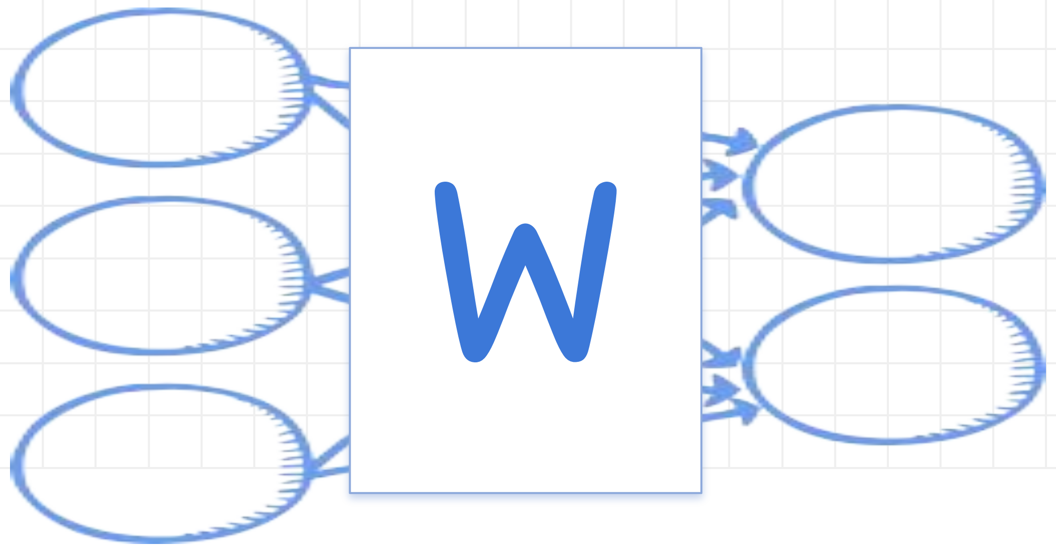
Each Edge is a neural network!

GCN – Example Graph – Weights

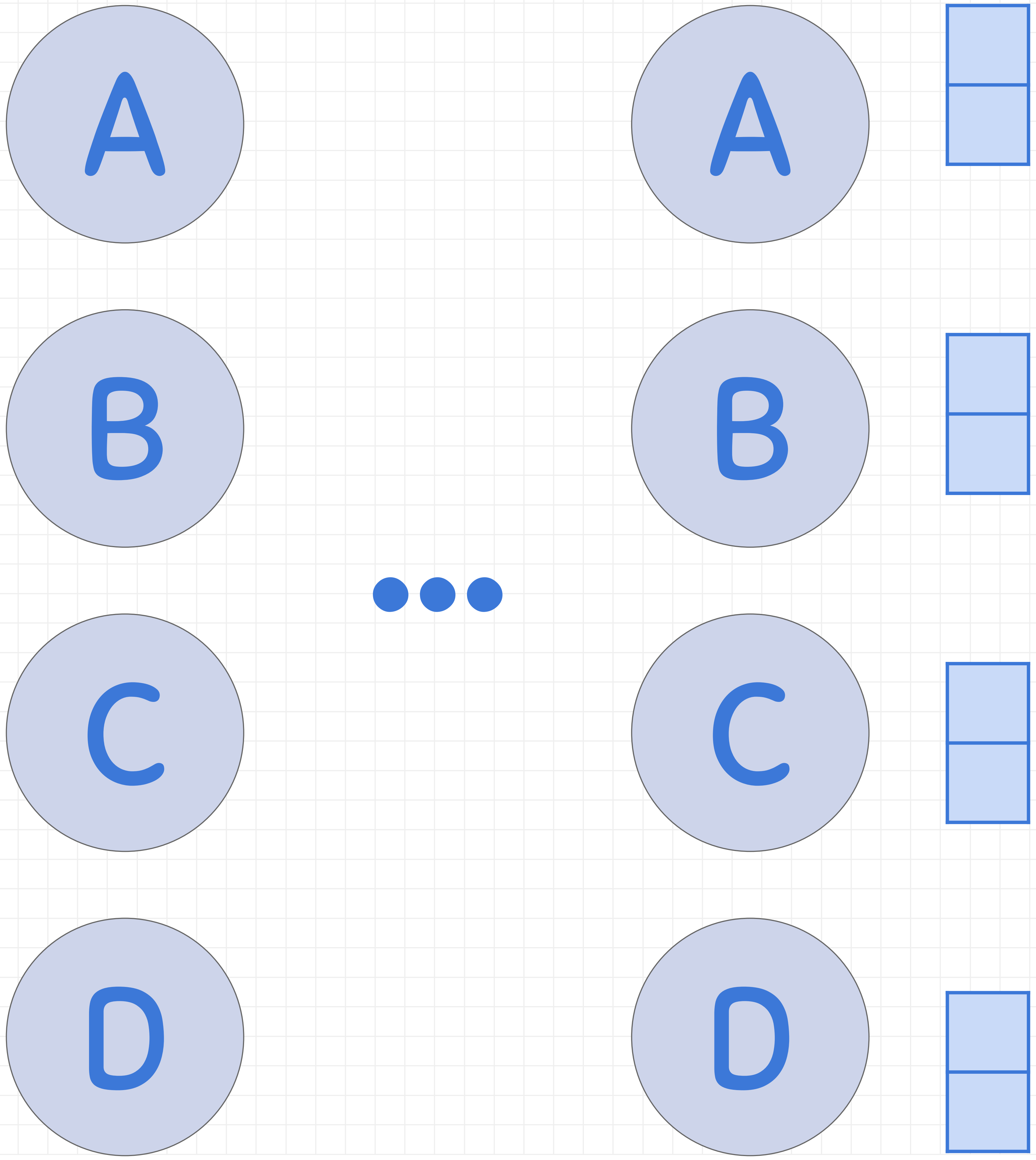


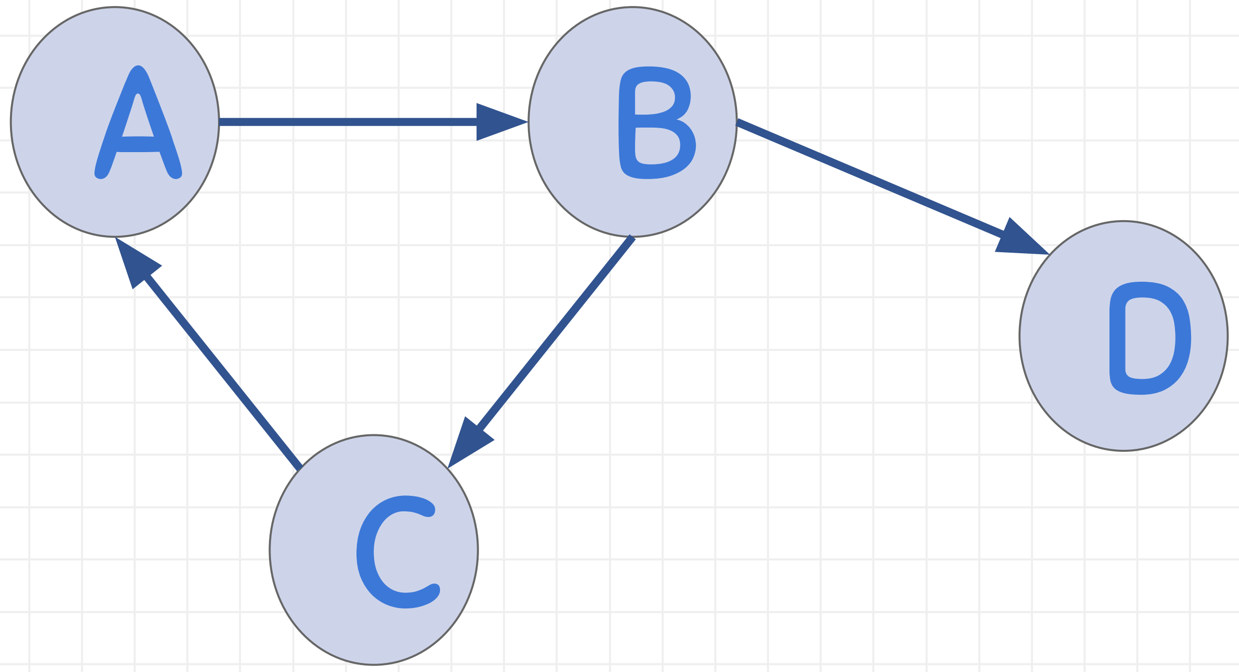


The input to EACH Node is a vector!

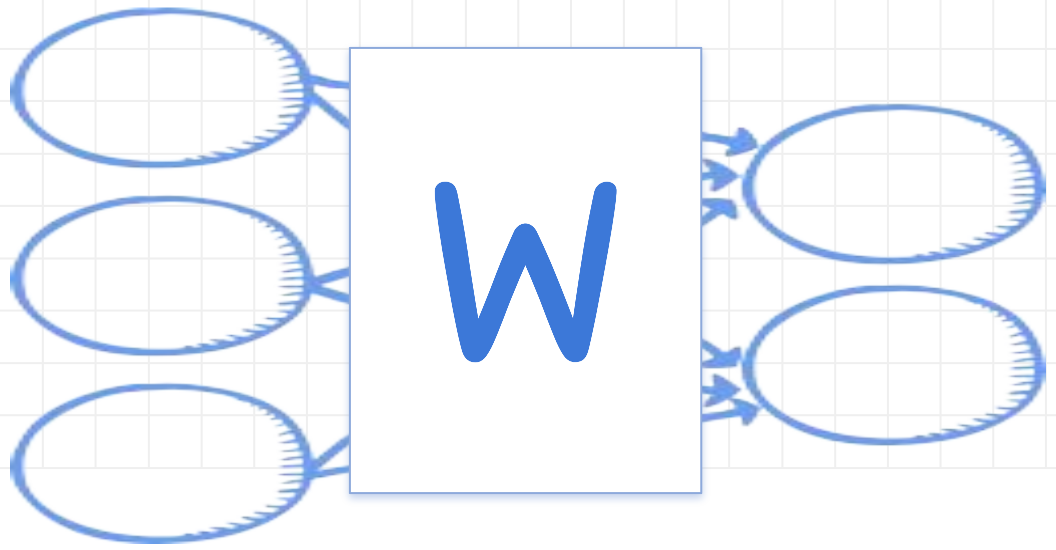


GCN – Example Graph – Weights

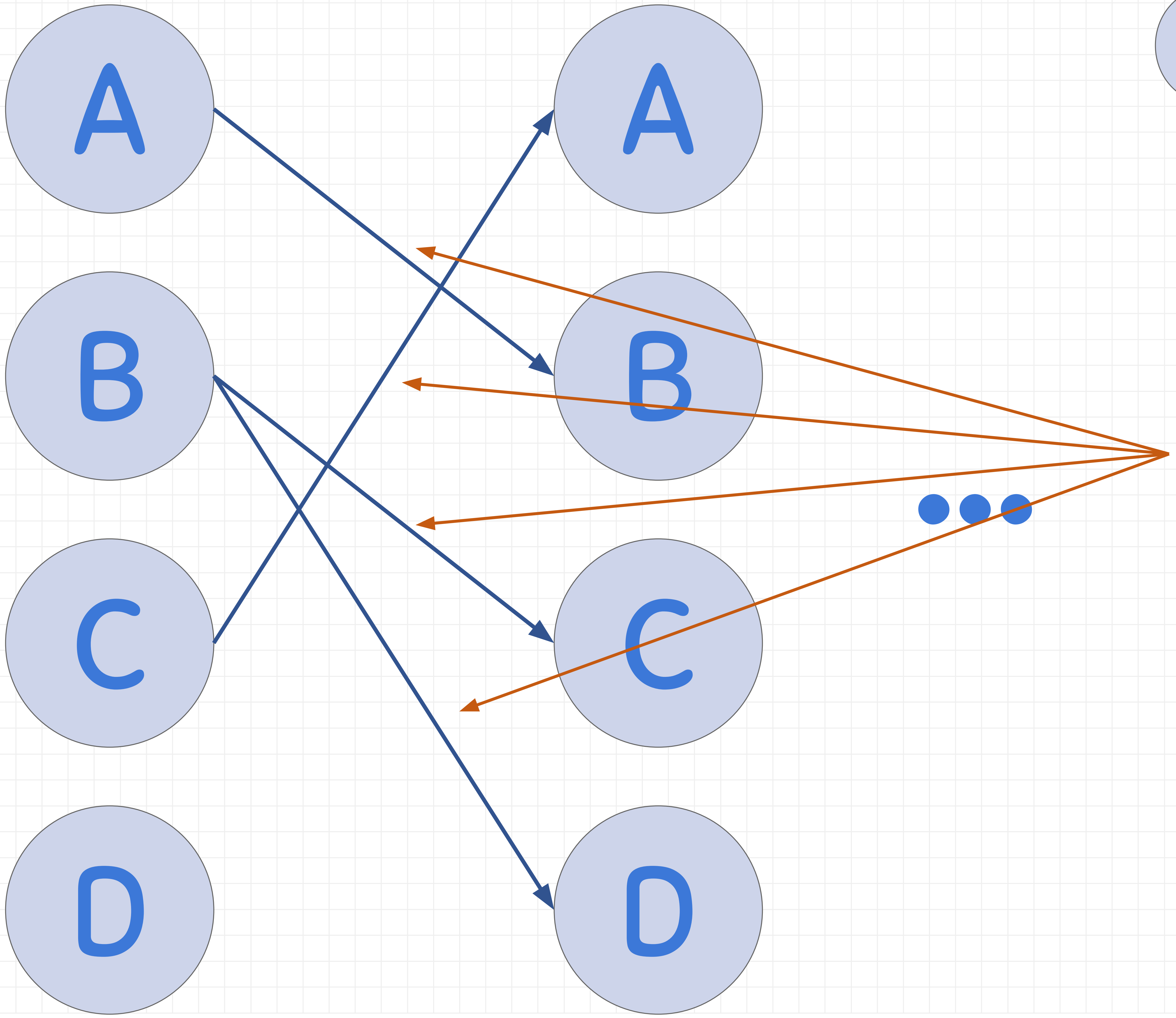


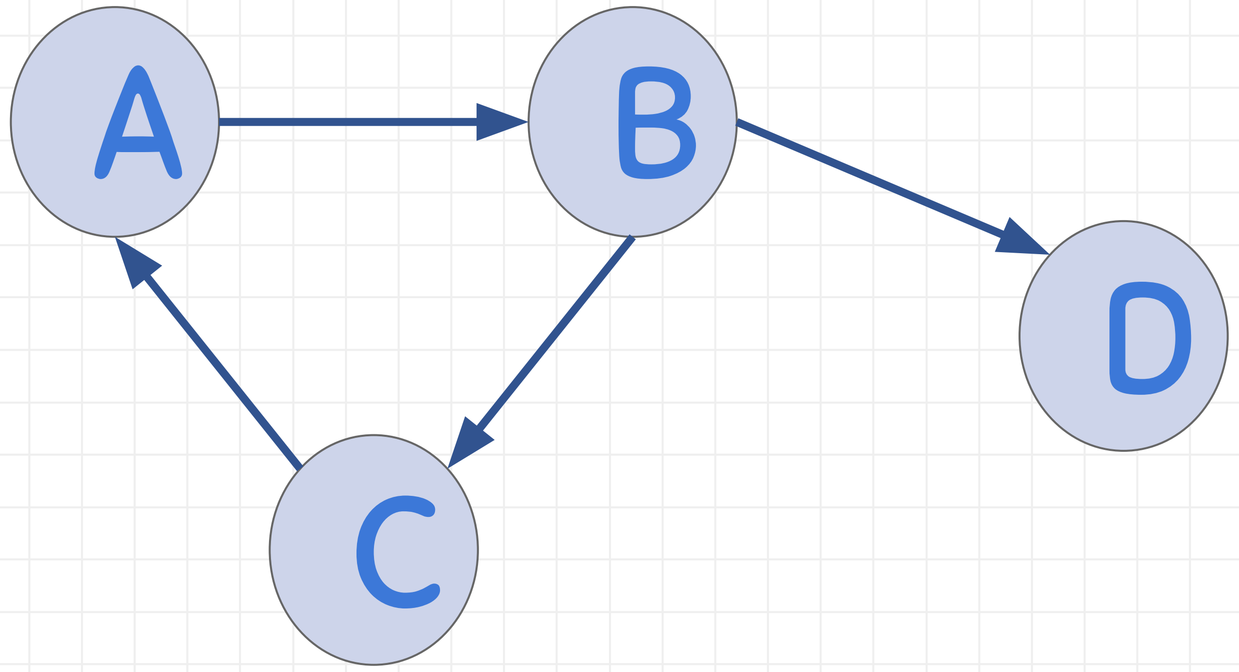


The output for EACH Node is a vector!

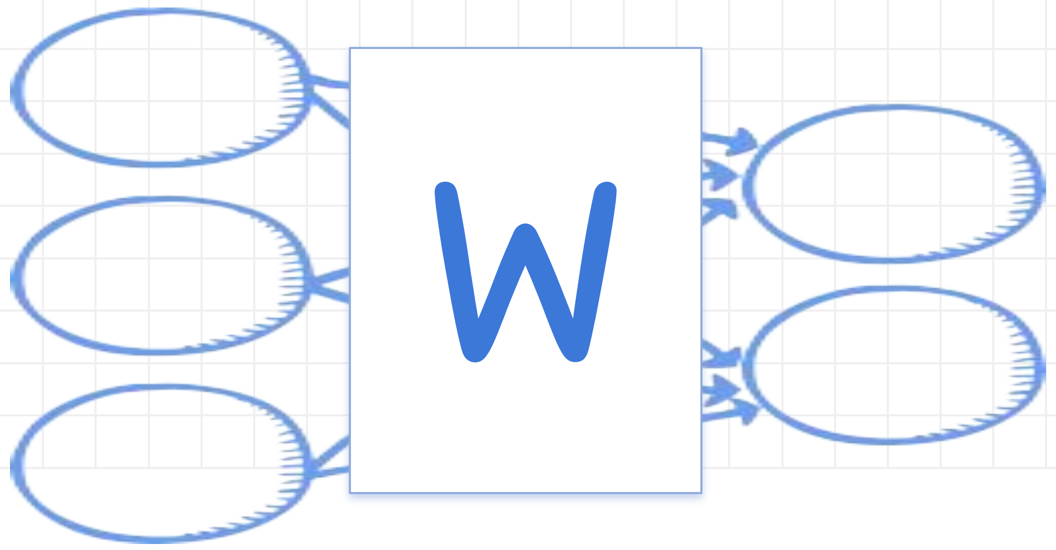


GCN – Example Graph – Weights Sharing

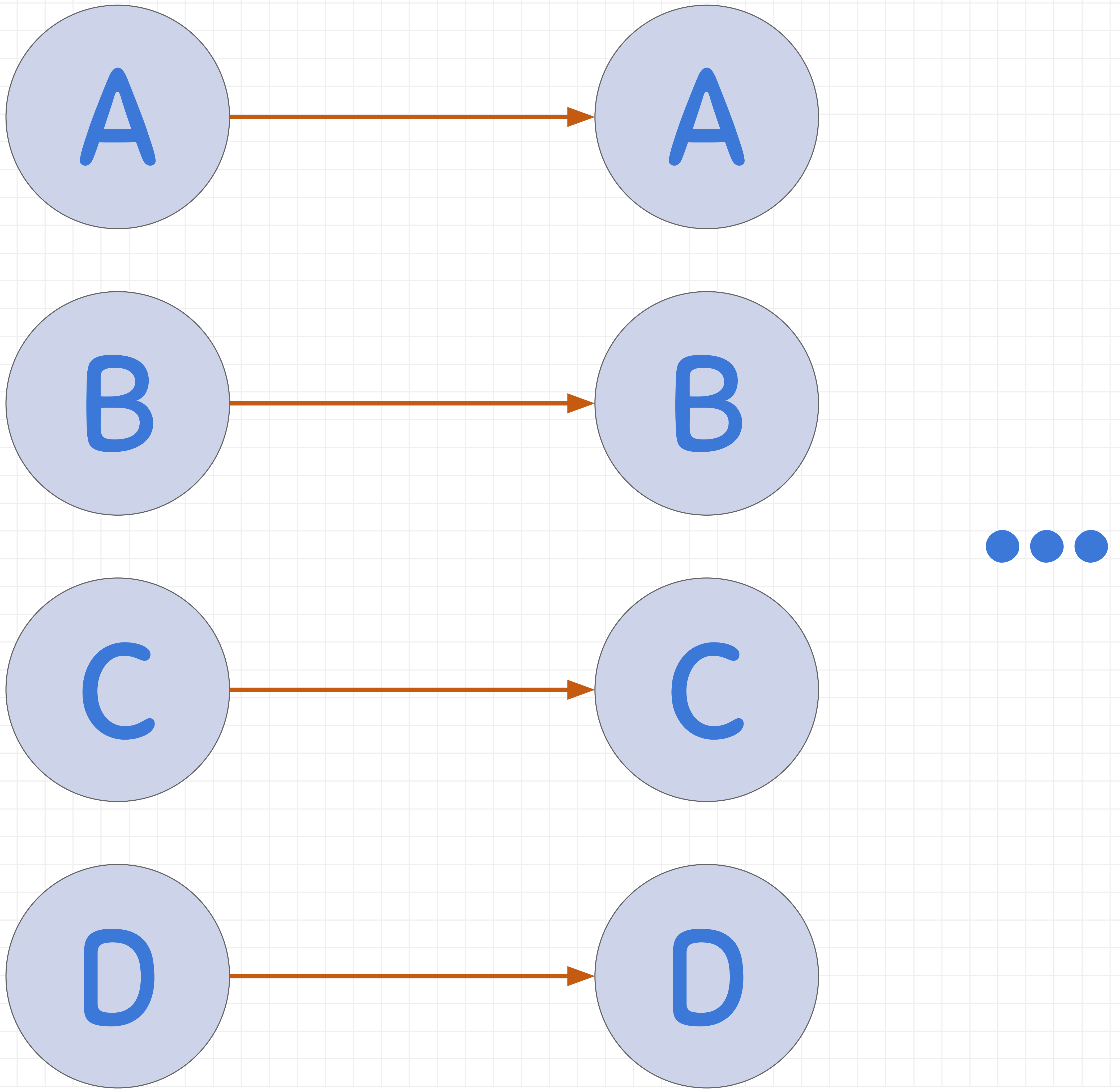


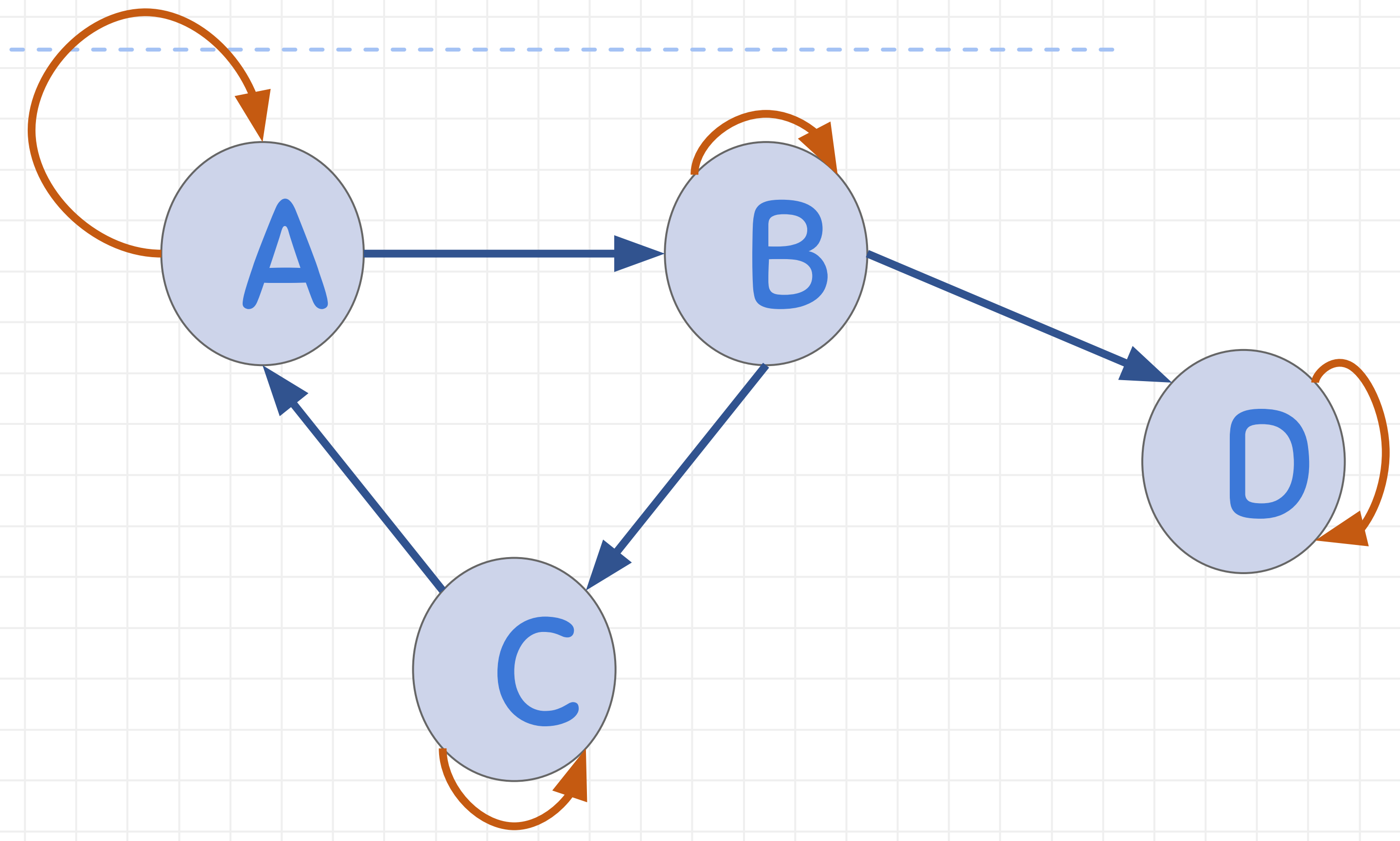


The weight matrix is shared!



GCN – Example Graph – Self Loops





... and self-loops are added

GCN - A different view - sparse matrix multiplications

In practise what was presented does not scale well

- (Except with clever engineering)

In practise more normalization is needed

GCN - A different view - sparse matrix multiplications

Reformulation:

$$H^{(l+1)} = \sigma\left(\qquad \tilde{A} \qquad H^{(l)} W^{(l)} \right)$$

$H^{(l)}$ is the l-th layer in the unrolled network (the l-th time-step)

A is the adjacency matrix, Ã is the same with also the diagonal set to 1

$W^{(l)}$ is a learnable weight matrix for layer l

GCN - A different view - sparse matrix multiplications

Reformulation:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

$H^{(l)}$ is the l-th layer in the unrolled network (the l-th time-step)

A is the adjacency matrix, Ã is the same with also the diagonal set to 1

$\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ ← Used for normalization

$W^{(l)}$ is a learnable weight matrix for layer l

That's it, we can include this structure into a larger network.

Examples of tasks

Node classification

- What is the type of a node?

Regression of attributes in the graph

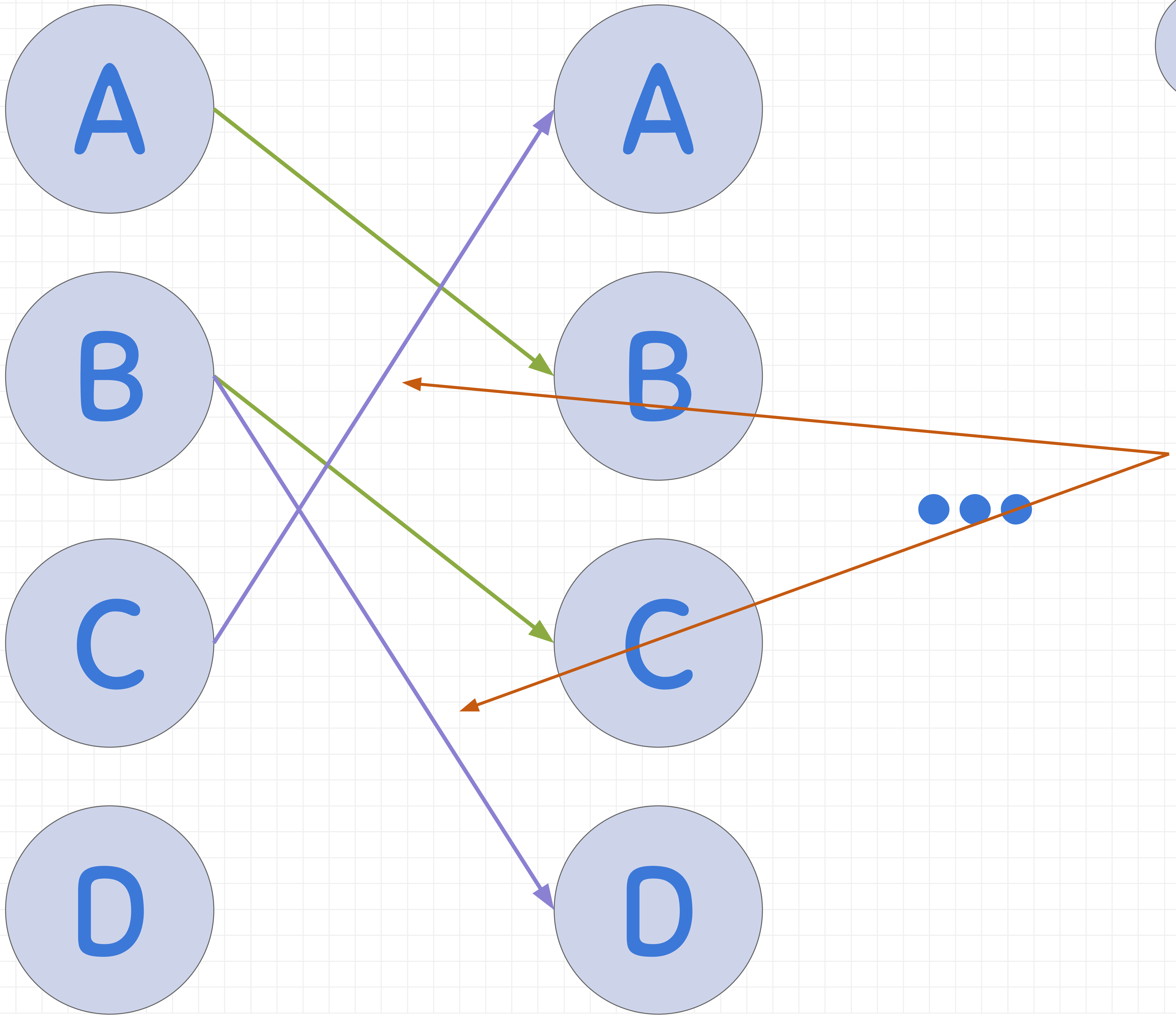
- What is the price of the product?

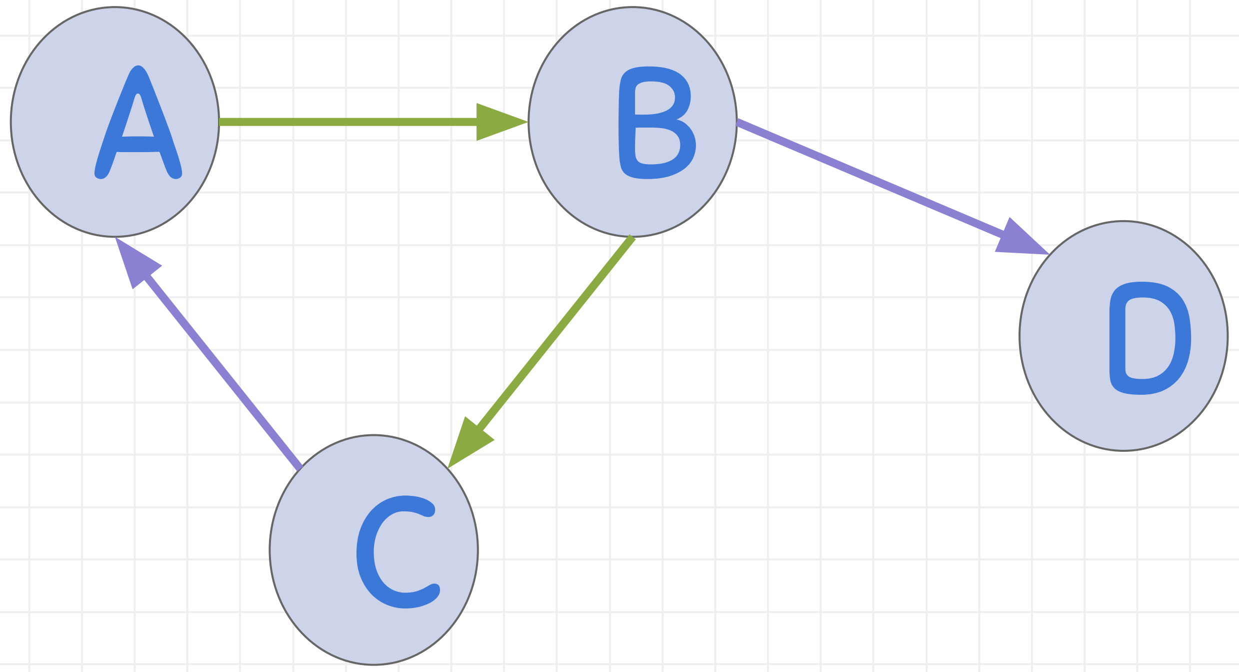
Regression/classification on the complete graph (by combining the output)

- What is the boiling point of a molecule?
- Is this molecule poisonous?

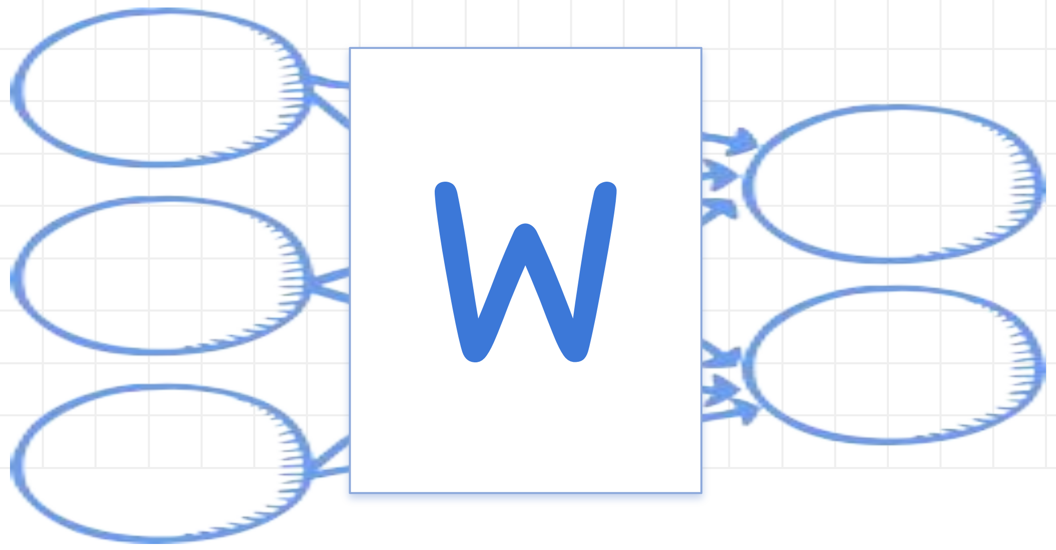
What if the graph has typed edges?

RGCN

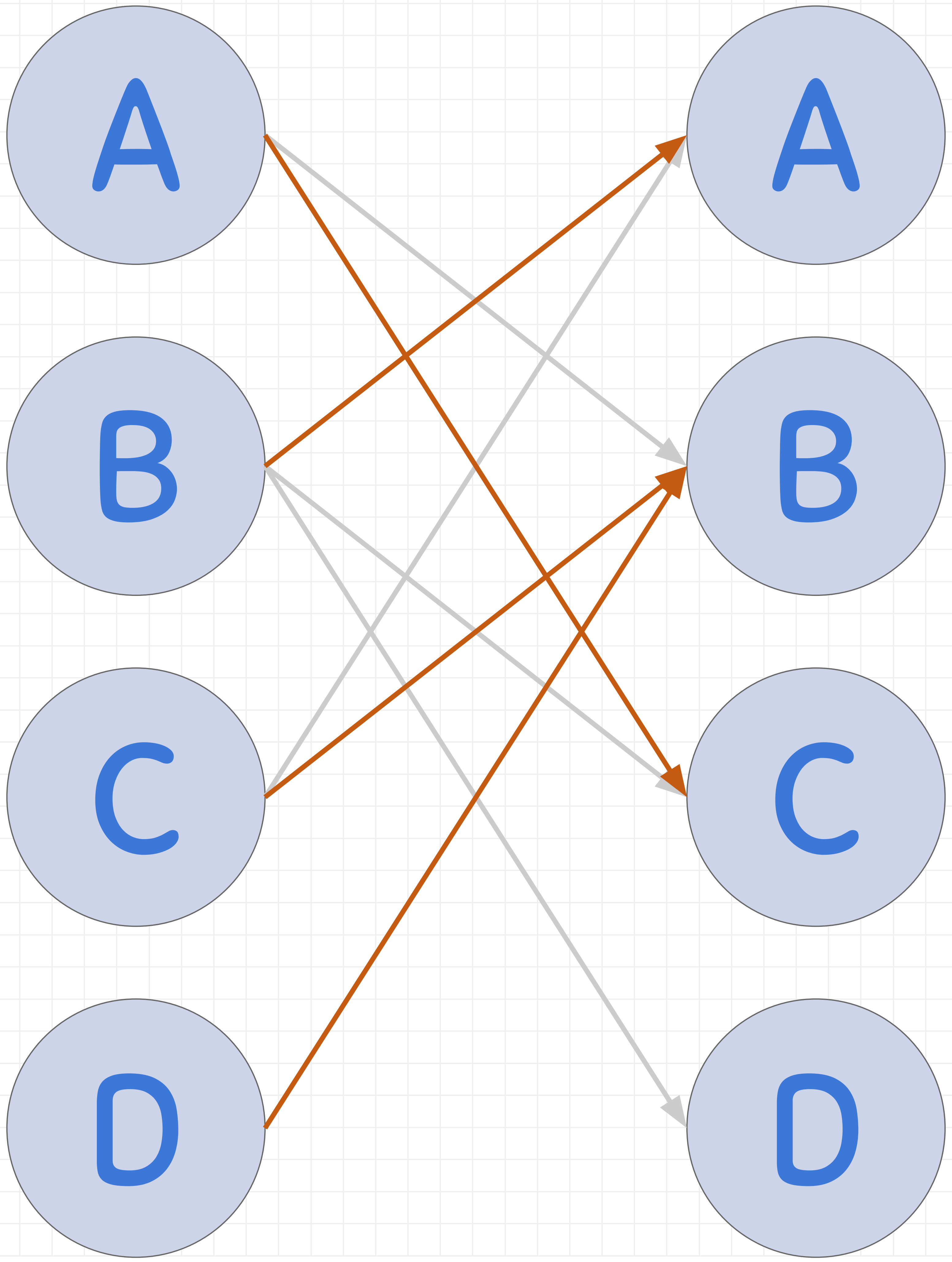


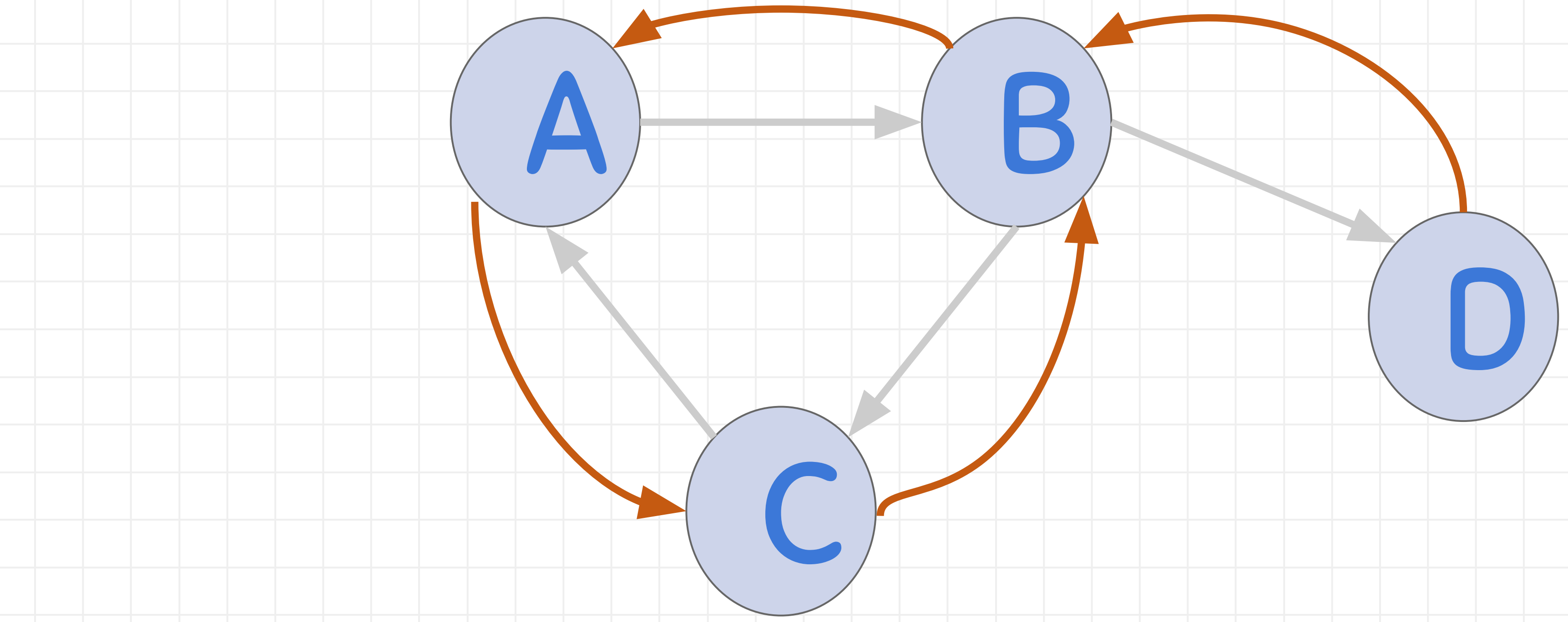


The weight matrix is shared per edge type!

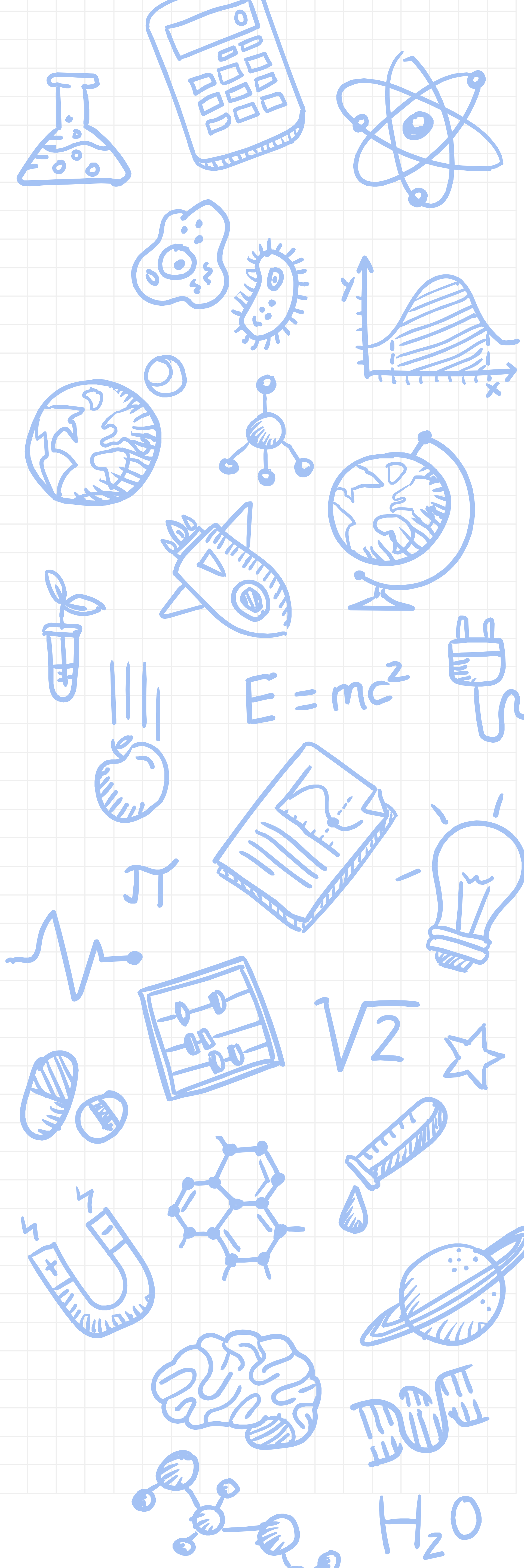


RGCN – Reverse edges





Also reverse edges (inverse relations) are added



RGCN - formally

In matrix multiplication form, the R-GCN is computed as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$

-> this formulation is per node in the graph, not for all at once, as was done in the GCN formulation!

RGCN - formally

In matrix multiplication form, the R-GCN is computed as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \qquad\qquad\qquad\qquad \right)$$

$h_i^{(l)}$ is the i-th node, in the l-th layer (=l-th message passing step)

(R is the set of all relations)

In matrix multiplication form, the R-GCN is computed as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \qquad\qquad\qquad\qquad \right)$$

$h_i^{(l)}$ is the i-th node, in the l-th layer (=l-th message passing step)

(R is the set of all relations)

N_i^r is the set of neighbours of node i with respect to relation r

RGCN - formally

In matrix multiplication form, the R-GCN is computed as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r} W_r^{(l)} h_j^{(l)}\right)$$

$h_i^{(l)}$ is the i-th node, in the l-th layer (=l-th message passing step)

$W_r^{(l)}$ is the weight matrix for relation r at layer l (R is the set of all relations)

N_i^r is the set of neighbours of node i with respect to relation r

RGCN - formally

In matrix multiplication form, the R-GCN is computed as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r} W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$

$h_i^{(l)}$ is the i-th node, in the l-th layer (=l-th message passing step)

$W_r^{(l)}$ is the weight matrix for relation r at layer l (R is the set of all relations)

W_0 is the weight matrix for the self loop

N_i^r is the set of neighbours of node i with respect to relation r

RGCN - formally

In matrix multiplication form, the R-GCN is computed as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r} \boxed{\frac{1}{c_{i,r}}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right)$$

$h_i^{(l)}$ is the i-th node in the l-th layer (=l-th message passing step)

$W_r^{(l)}$ is the we … the set of all relations)

W_0 is t…

$c_{i,r}$ is a normalization constant.
Usually $c_{i,r}$ is $|N_i^r|$

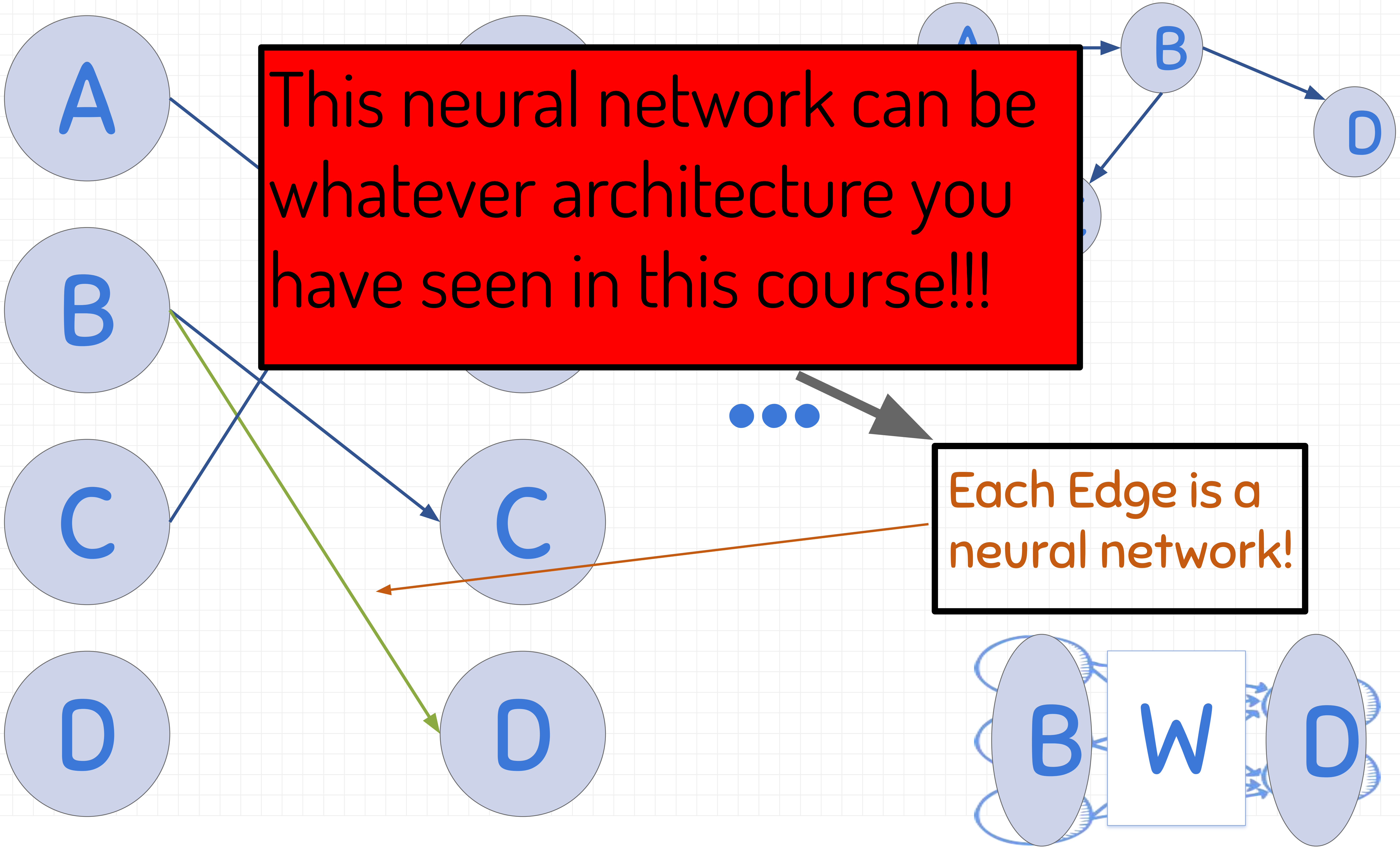
N_i^r is the set of neighbours of node i with respect to relation r

In general, we can do whatever we want in the unrolled view...

and see whether we can implement it somehow more efficiently...

GCN – We can do whatever we want

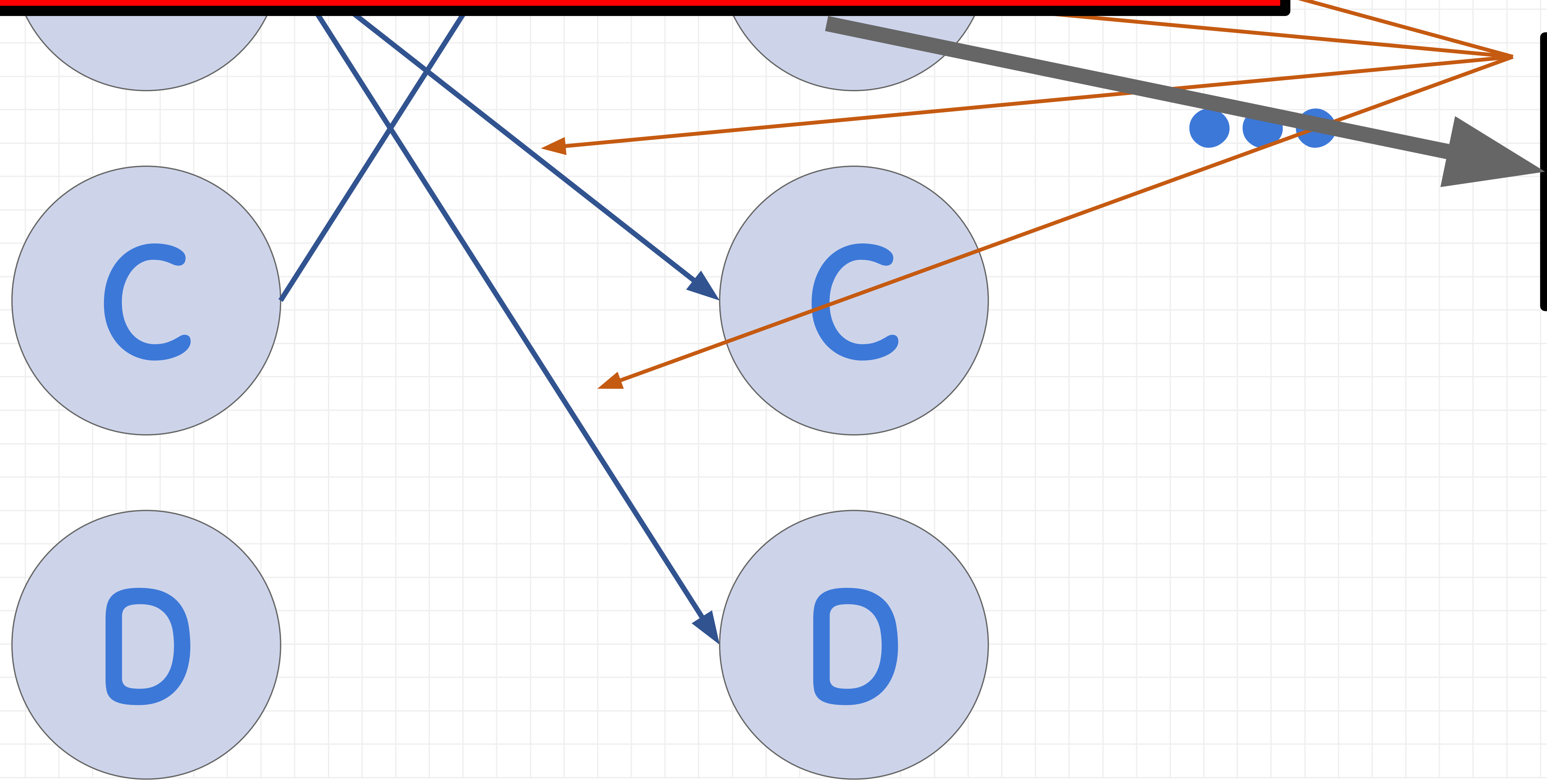
This neural network can be whatever architecture you have seen in this course!!!

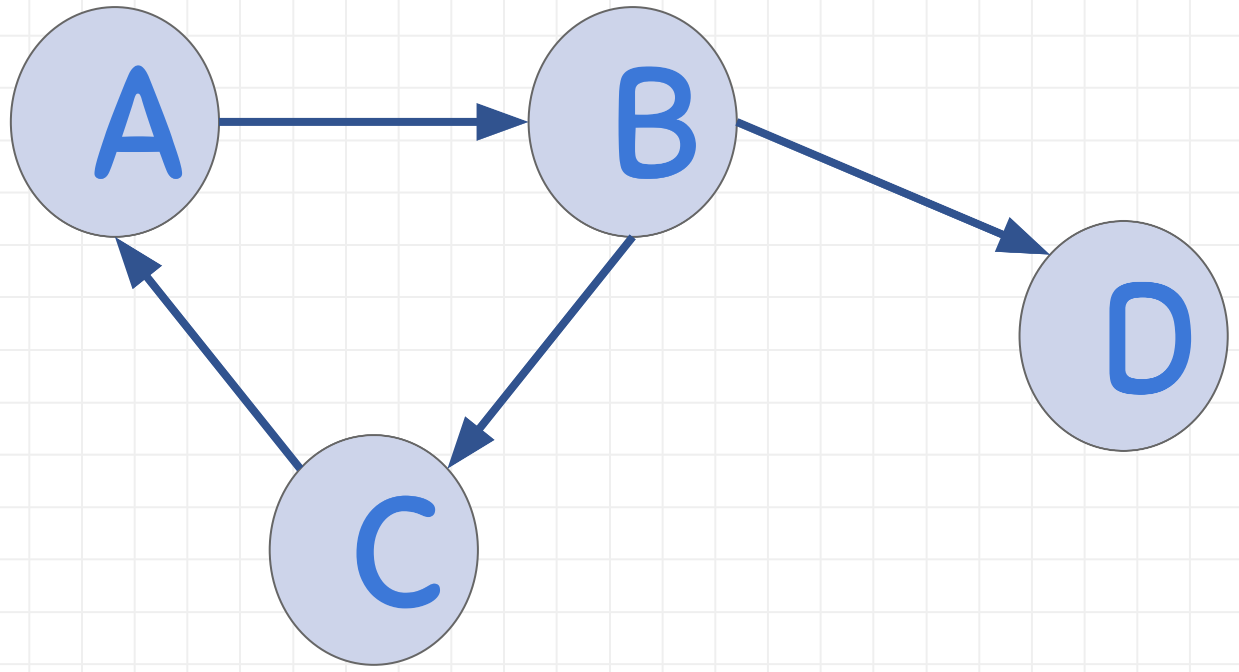


Each Edge is a neural network!

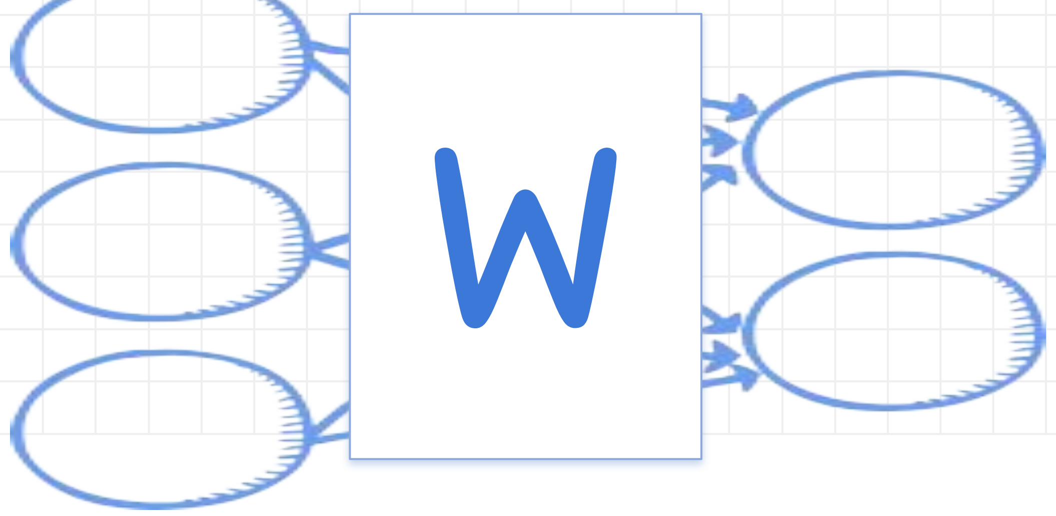
GCN – Example Graph – Weights Sharing

You can share wheights in whatever way seems to make sense.





The weight matrix is shared!

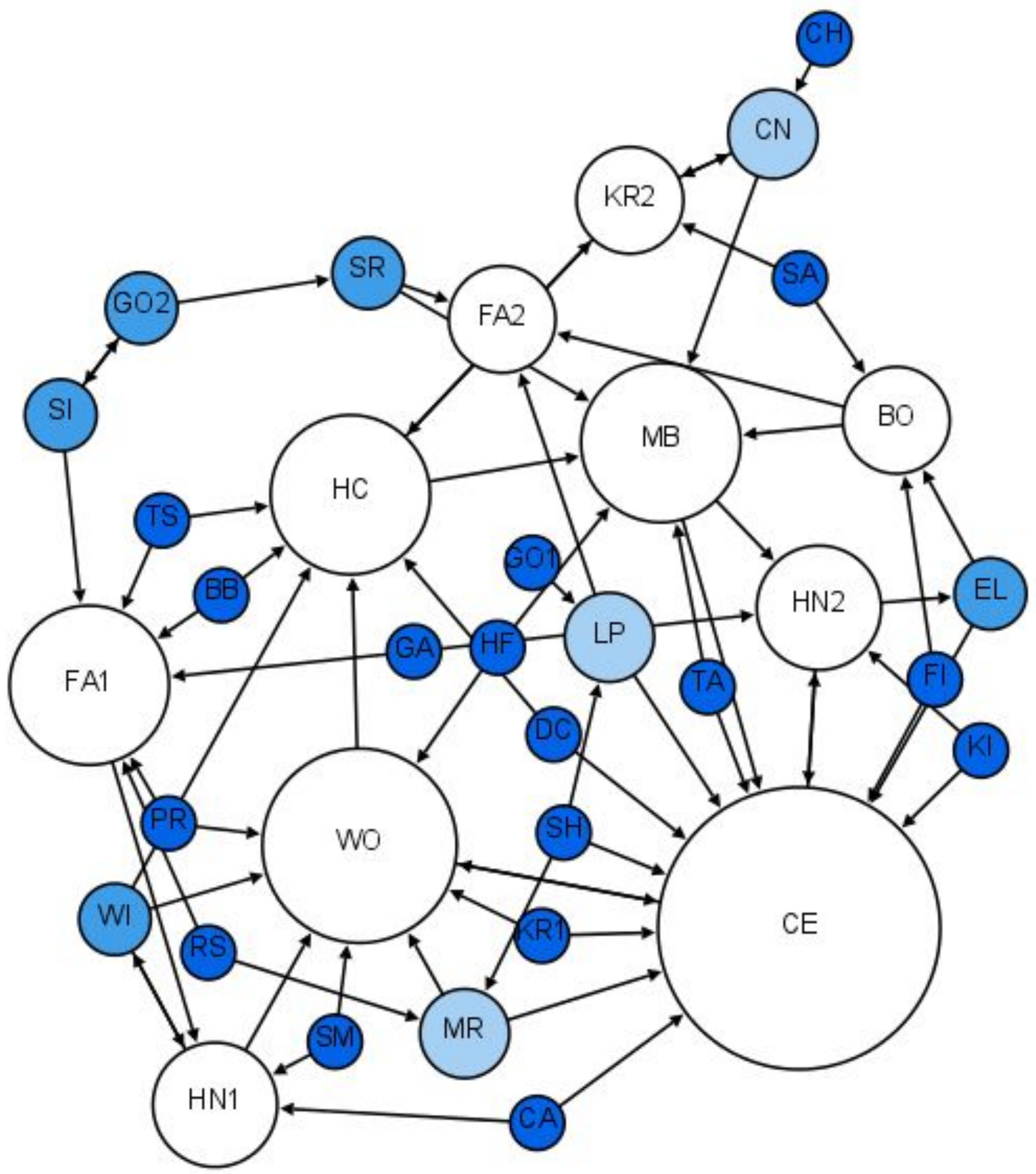


See also Scarselli, Franco, et al. "The graph neural network model." IEEE Transactions on Neural Networks 20.1 (2008): 61-80.

PART FOUR: Application - Query embedding

https://arxiv.org/abs/2002.02406

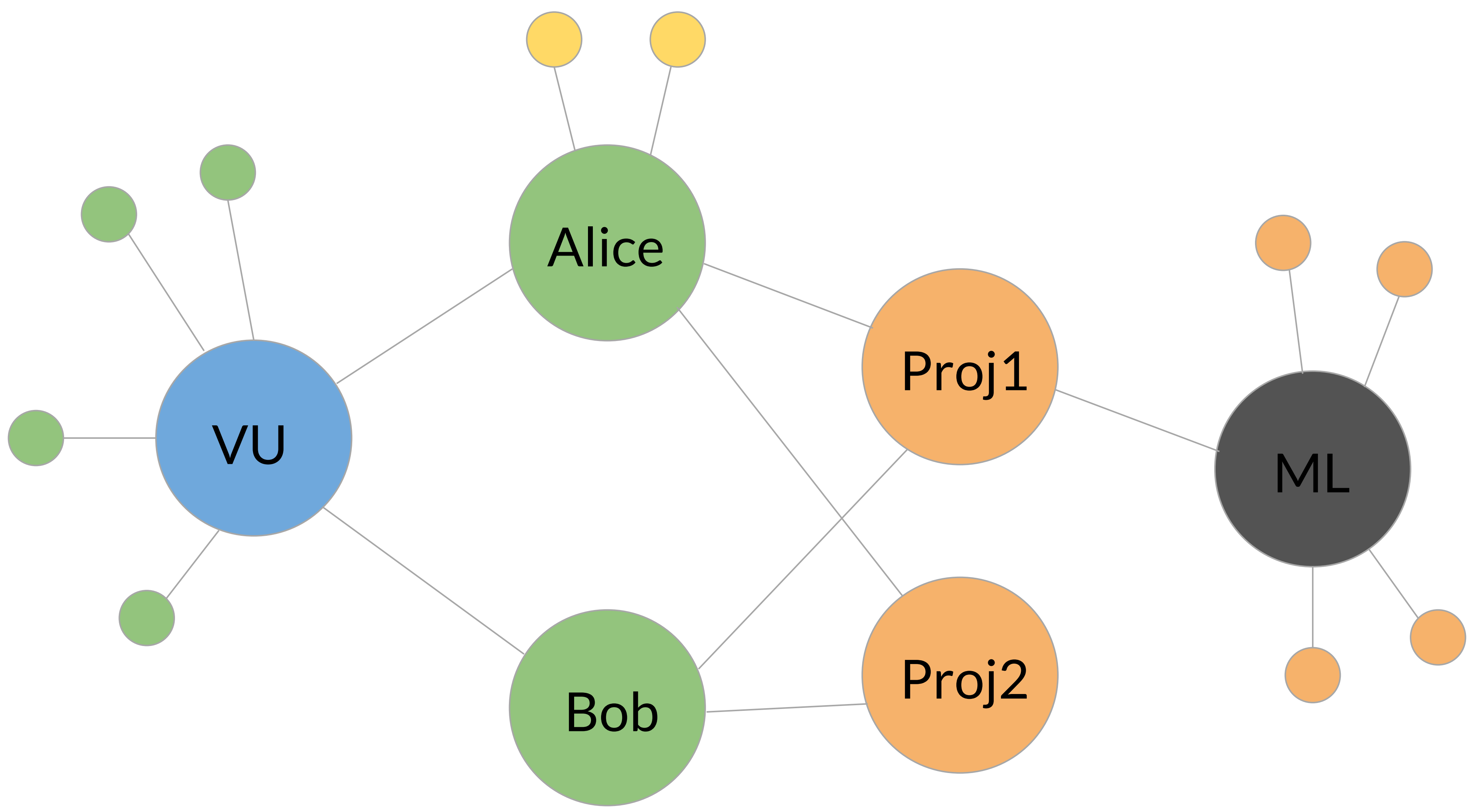
Knowledge graphs



- Can model interactions and properties
  - Medicine, biology, world facts, ...
- In general, useful for
  - Storing facts about entities and relations
  - **Answering questions about them**

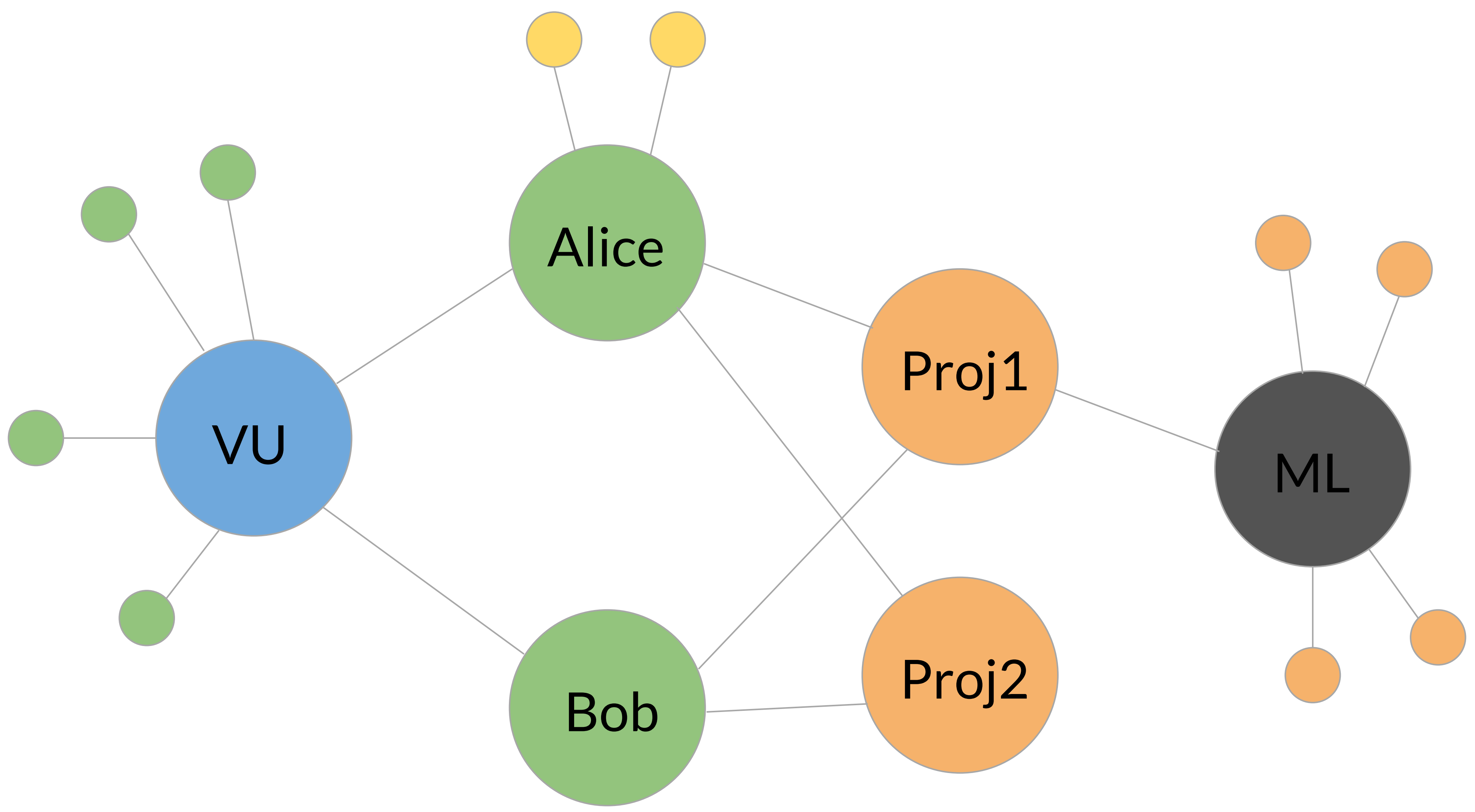
https://commons.wikimedia.org/wiki/File:Moreno_Sociogram_1st_Grade.png

Queries on knowledge graphs



- SPARQL queries operate on existing edges
  - Select all Projects, related to ML, on which Alice works
  - Answer: **Proj1**

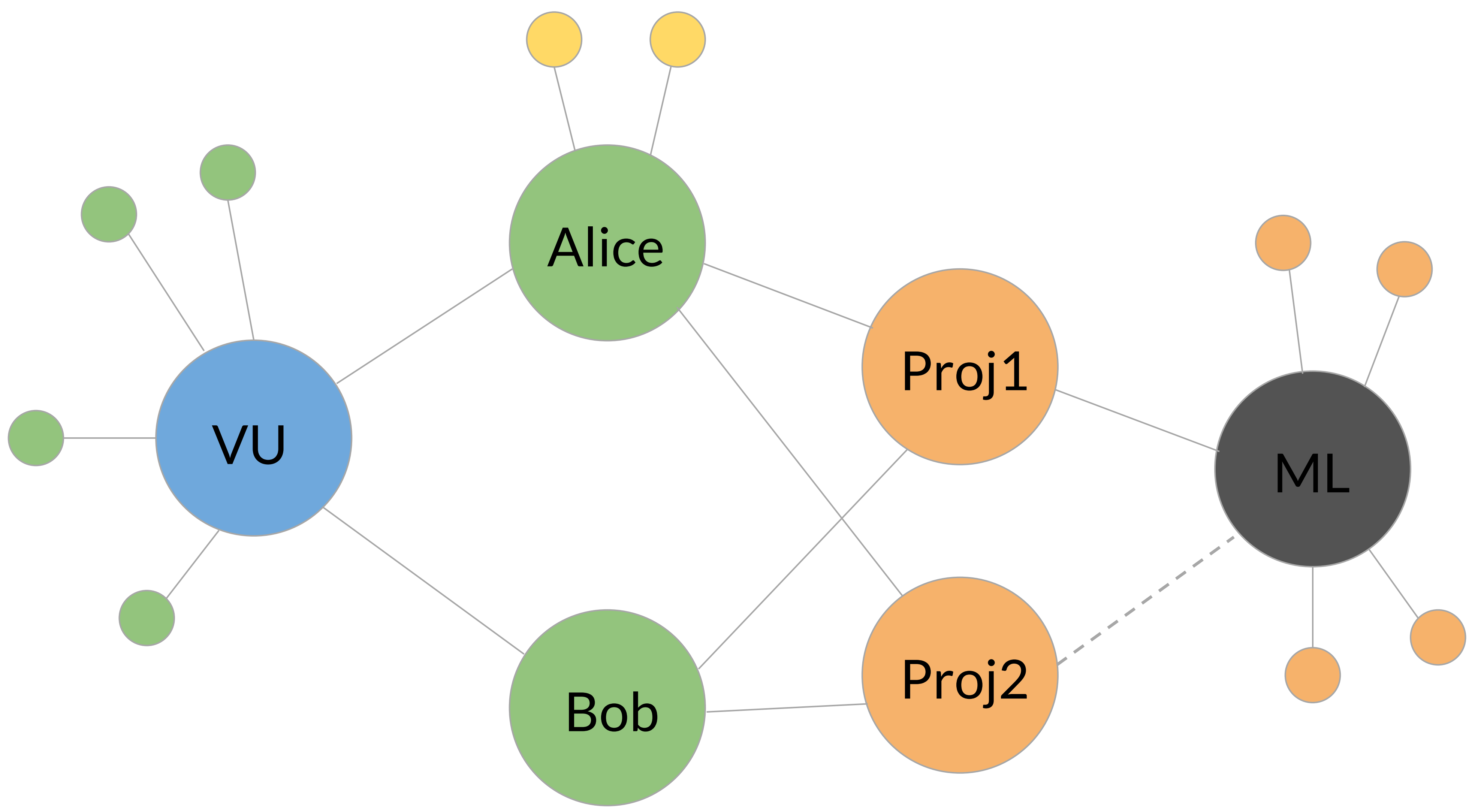
Queries on knowledge graphs



- SPARQL queries operate on existing edges
  - Select all Projects, related to ML, on which Alice works
  - Answer: **Proj1**
- Is **Proj2** a *likely* answer?

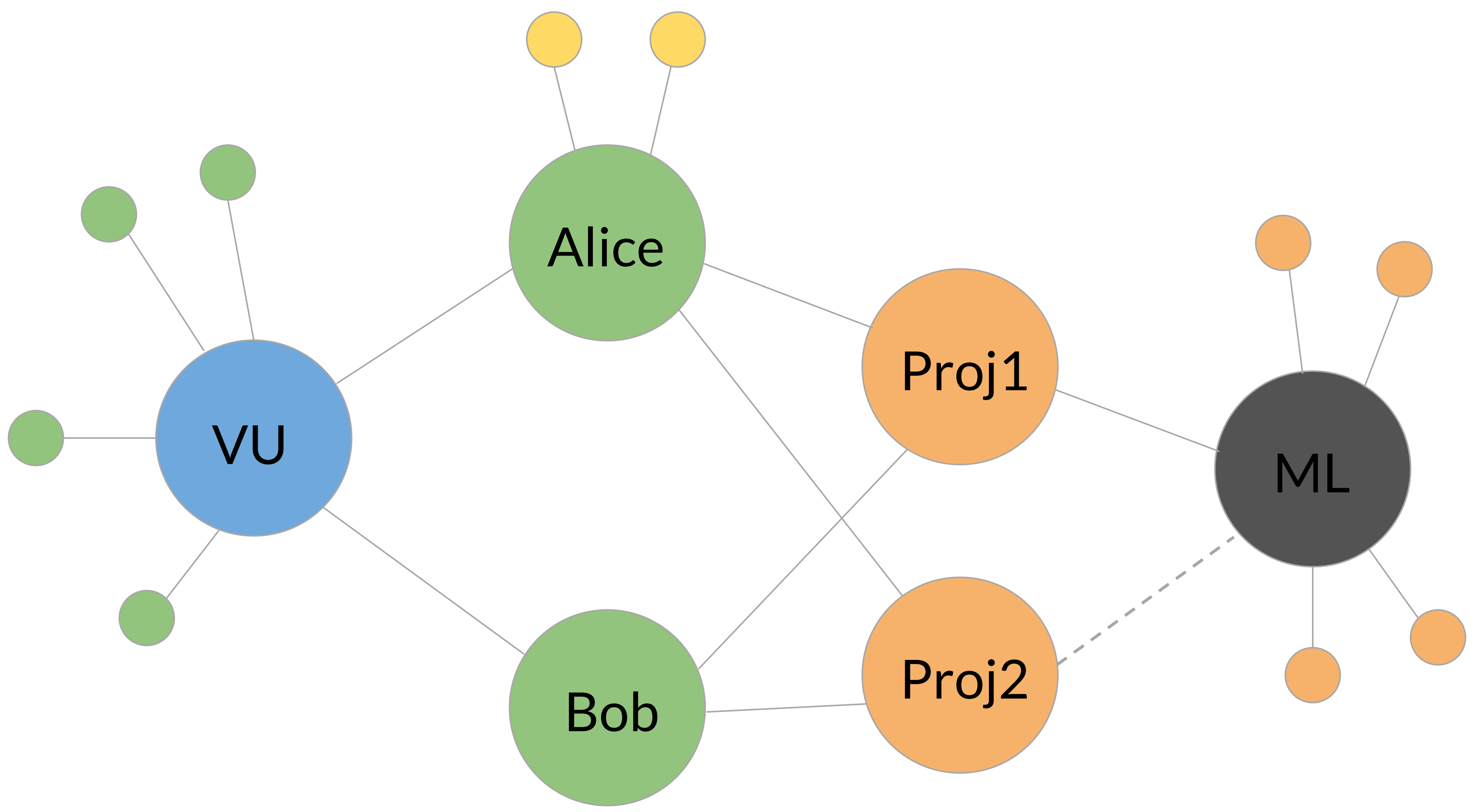
VU

Link prediction on knowledge graphs



- Assign a vector in $\mathbb{R}^d$ to every node: an ***embedding***
- The score of an edge is a function of the embeddings of entities involved
- Optimize:
  - Maximize scores of existing edges
  - Minimize scores of random edges
- Examples: TransE, DistMult, ComplEx

Link prediction for complex queries?



- Select all topics T,
- where T is related to a project P,
- and Alice and Bob work on P.

- Link prediction requires enumerating all possible T and P
  - Grows exponentially!

VU

Link prediction for complex queries?

A *subset* of Wikidata

- Select all topics T,
- where T is related to a project P,
- and Alice and Bob work on P.

- Link prediction requires enumerating all possible T and P
  - **Grows exponentially!**

VU

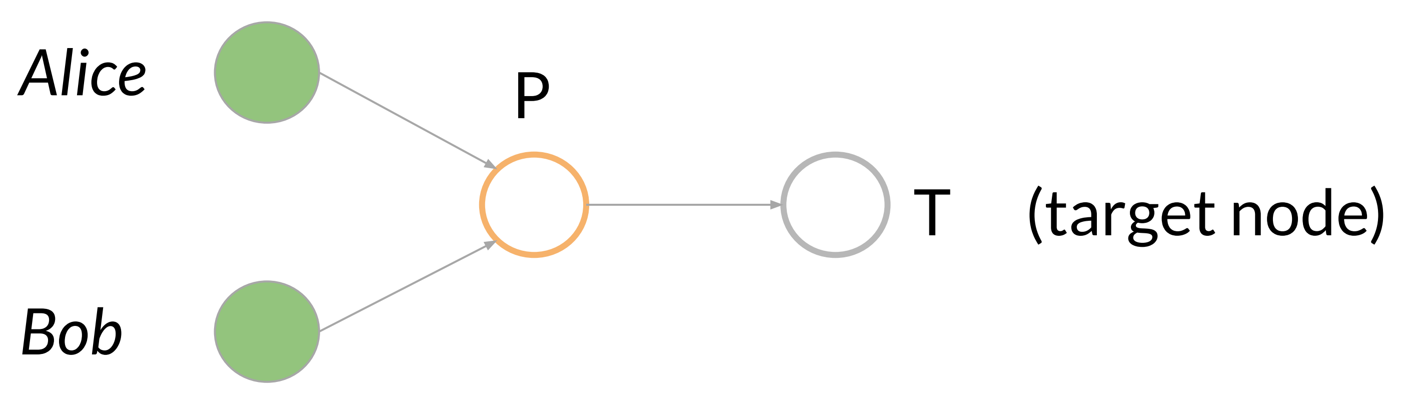
Queries are graphs too

- In particular, *Basic Graph Patterns*[1]

[1] Harris, S., Seaborne, A., Prud'hommeaux, E.: SPARQL 1.1 query language. W3C recommendation 21(10) (2013)

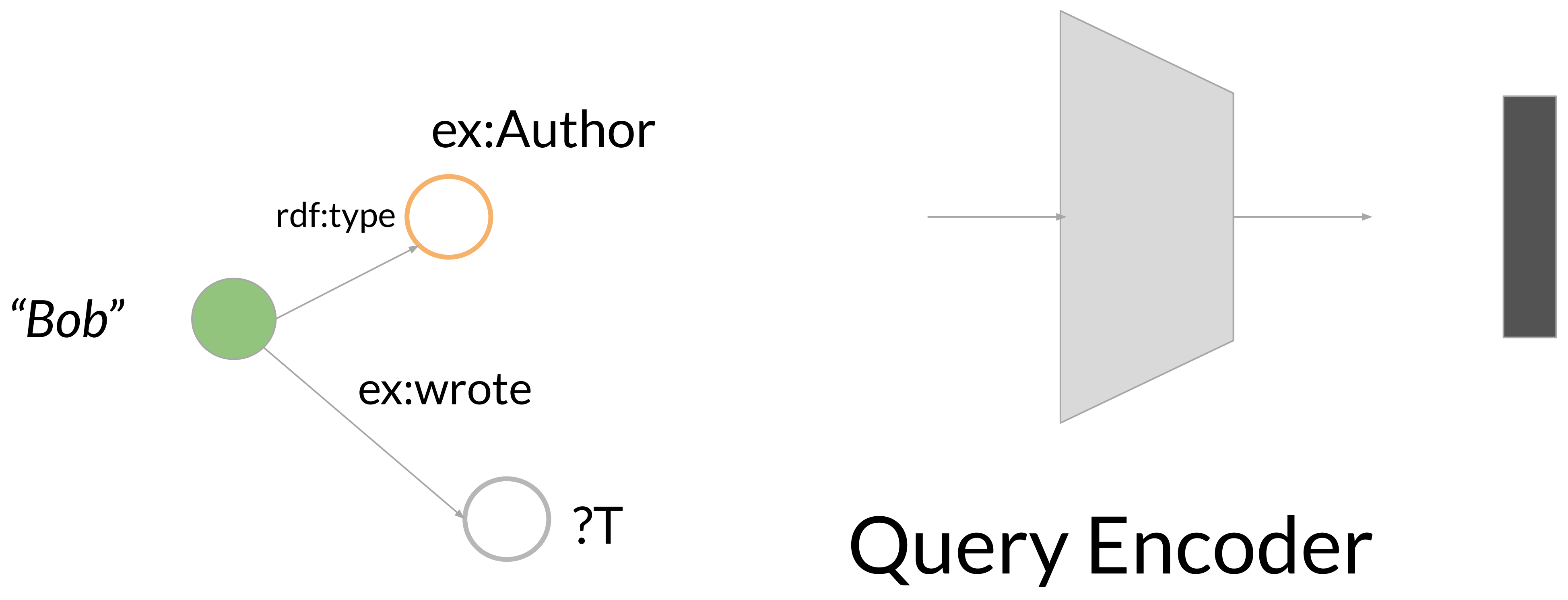
Queries are graphs too

- In particular, *Basic Graph Patterns*[1]
- Select all topics T where
    - T is related to project P
    - *Alice* works on P **and** *Bob* works on P

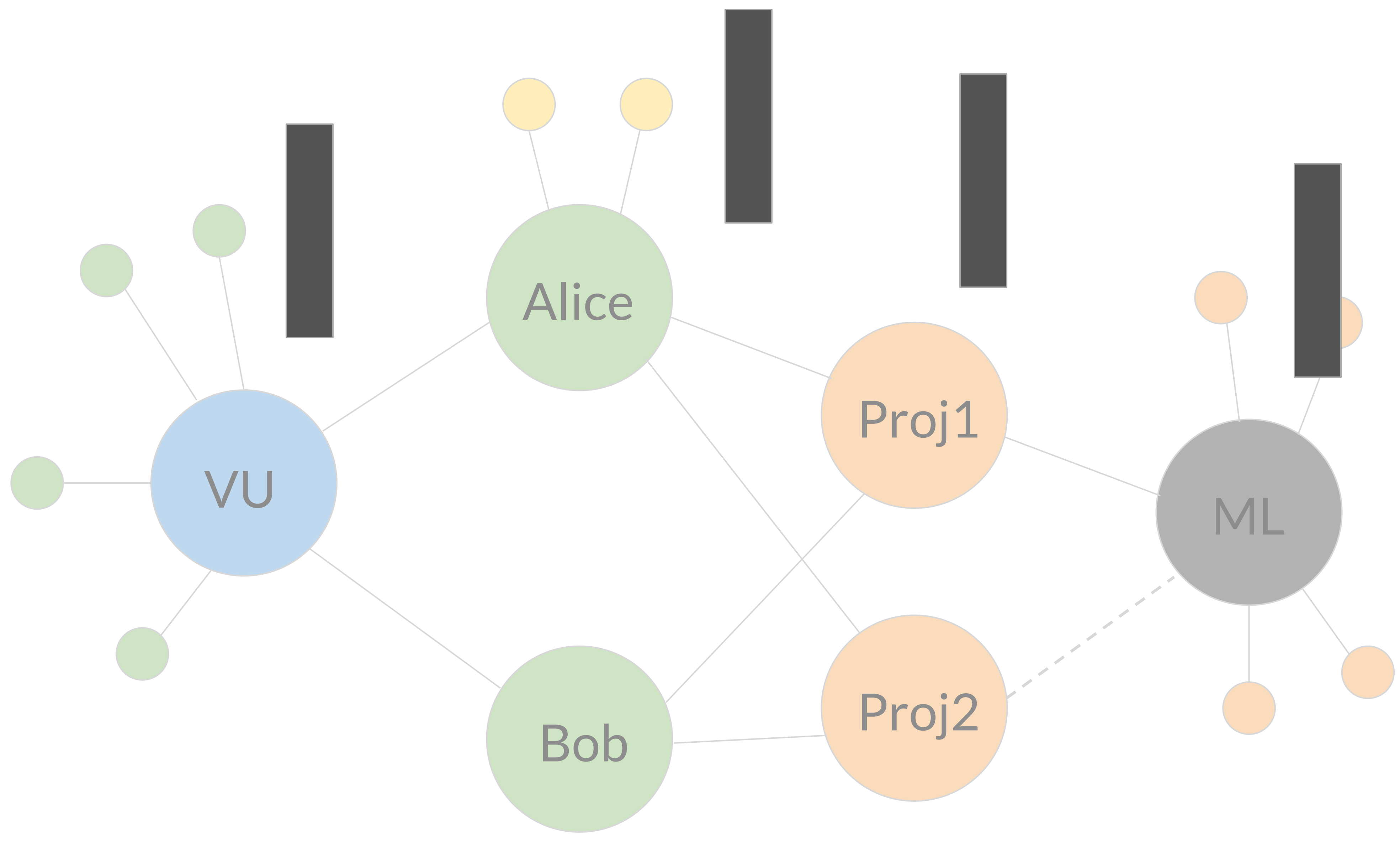


[1] Harris, S., Seaborne, A., Prud'hommeaux, E.: SPARQL 1.1 query language. W3C recommendation 21(10) (2013)

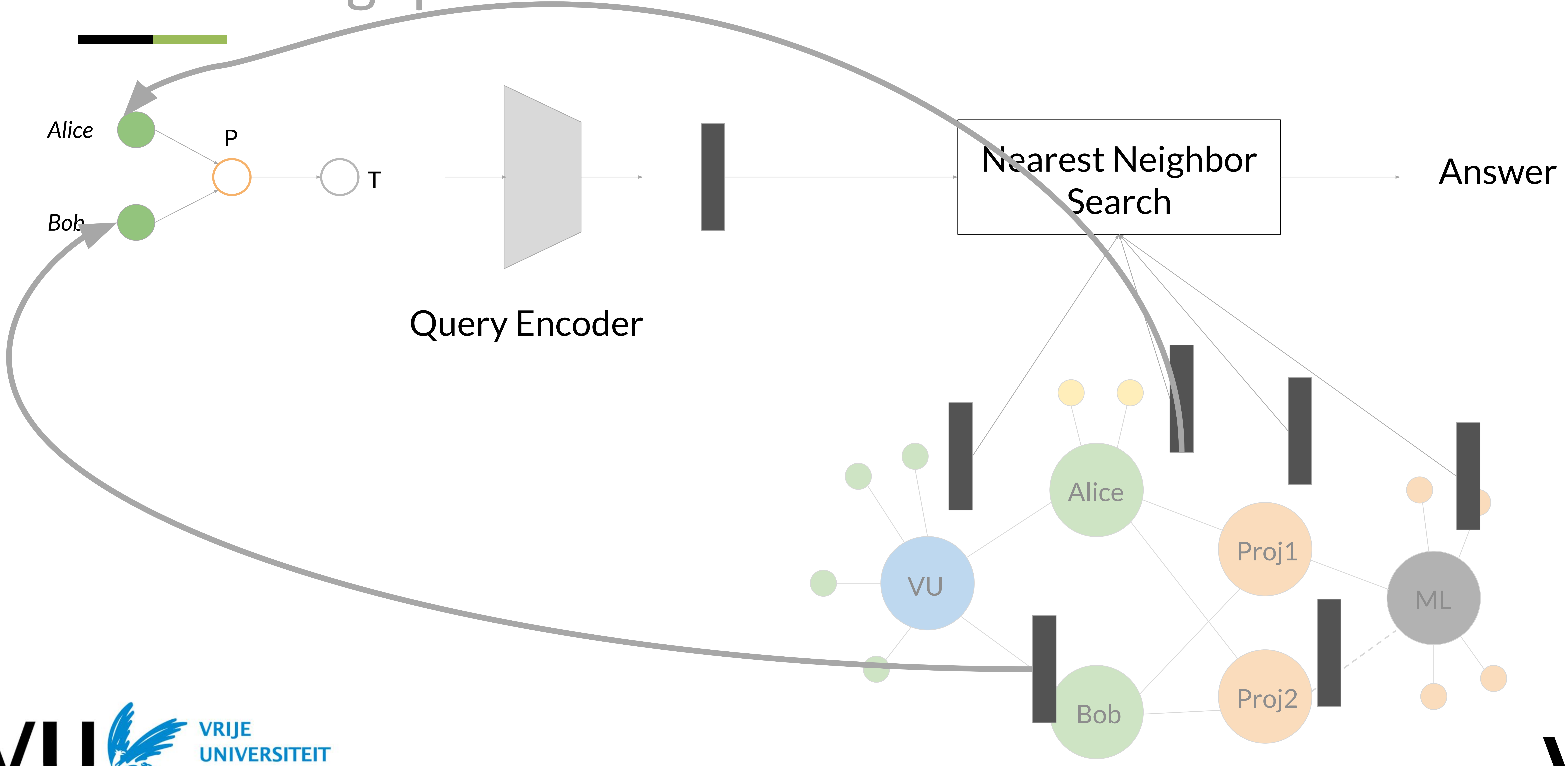
Embedding queries



:S1 hasSubject :Bob .
:S1 hasPredictate rdf:type .
:S1 hasObject ex:author .



Embedding queries

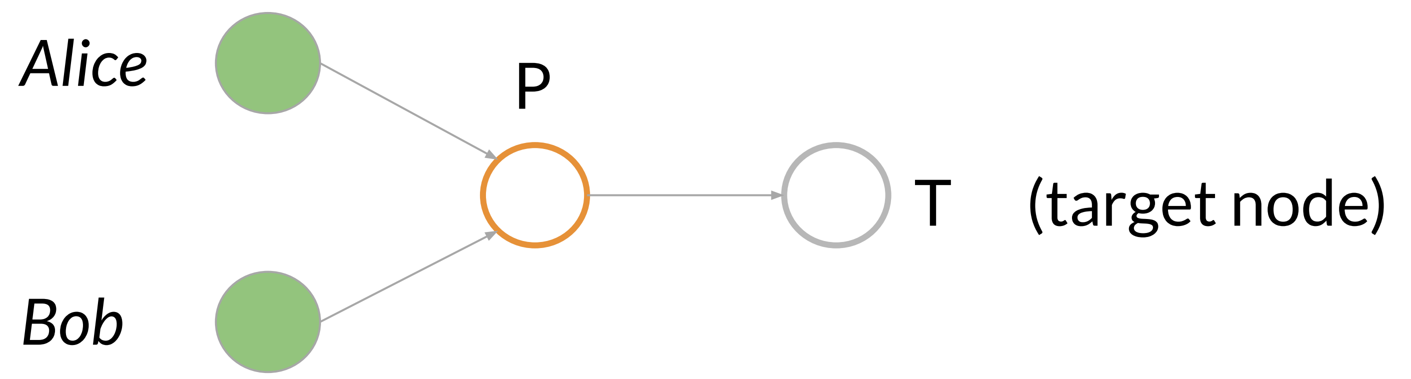


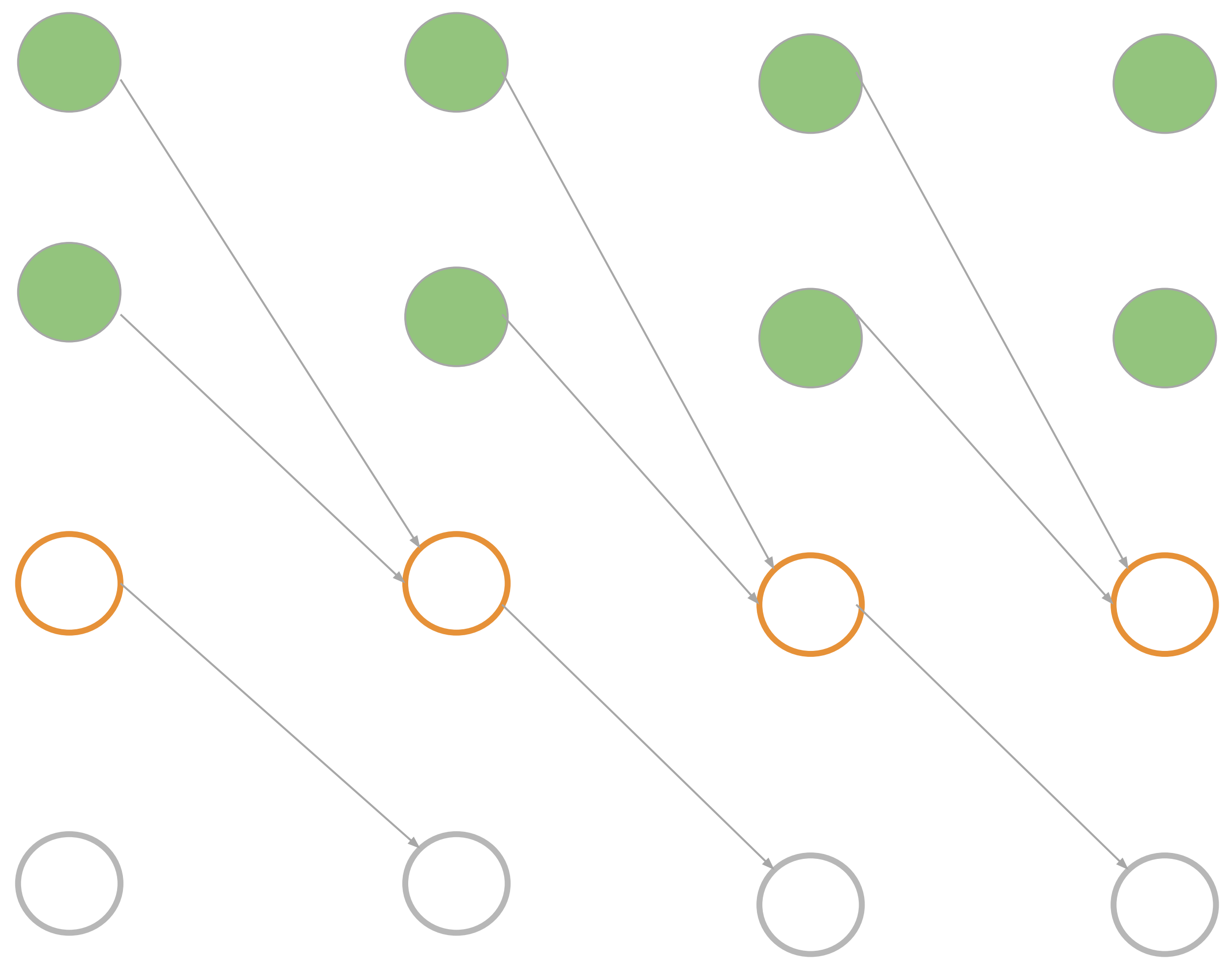
VU

The query encoder

- Graph Convolutional Networks operate on graphs, by applying message passing:
  - **Messages are vectors**

- **Message-Passing Query Embedding:**
  - Learnable parameters include both **entity** and **variable** node embeddings
  - Propagate messages across the BGP

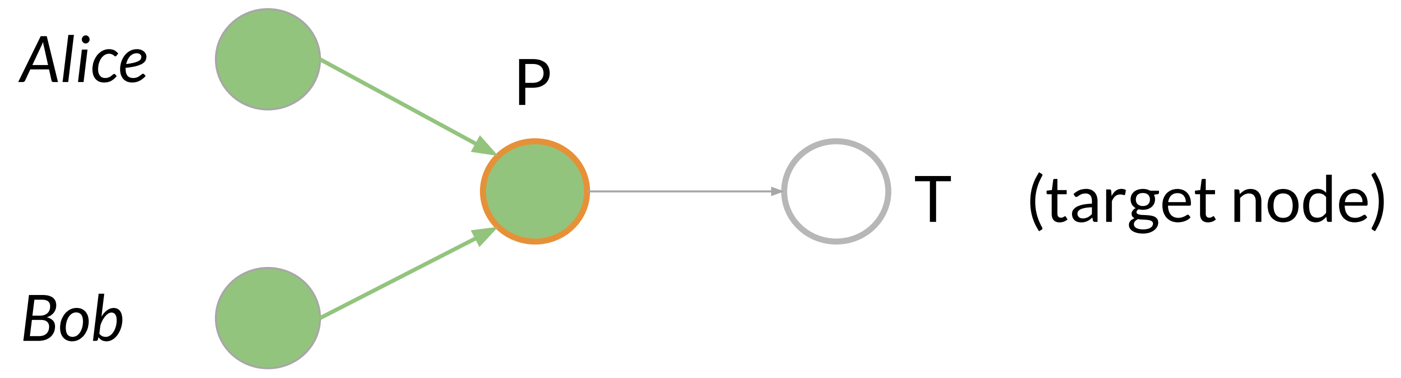




The query encoder

- Graph Convolutional Networks operate on graphs, by applying message passing:
    - **Messages are vectors**

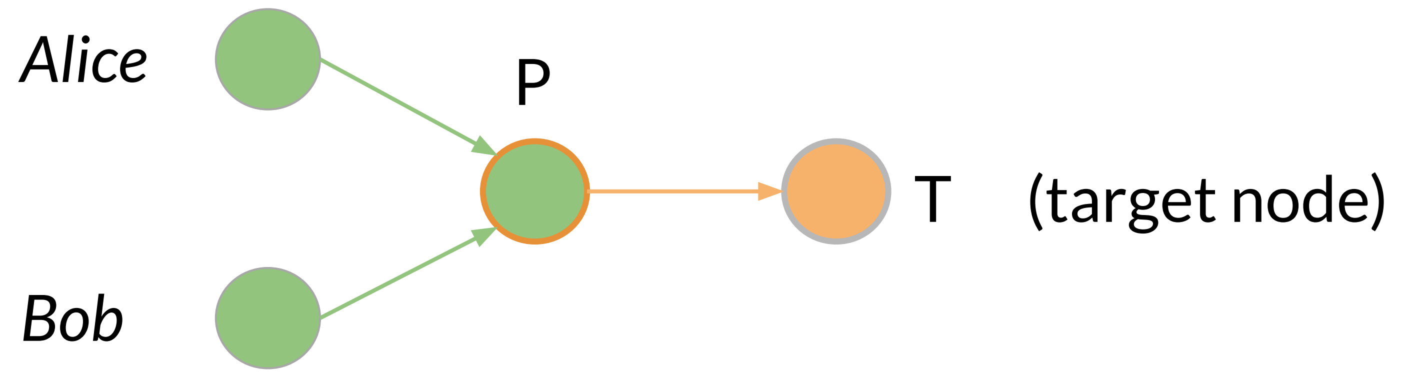
- **Message-Passing Query Embedding:**
    - Learnable parameters include both **entity** and **variable** node embeddings
    - Propagate messages across the BGP



The query encoder

- Graph Convolutional Networks operate on graphs, by applying message passing:
  - **Messages are vectors**

- **Message-Passing Query Embedding:**
  - Learnable parameters include both **entity** and **variable** node embeddings
  - Propagate messages across the BGP



VU VRIJE UNIVERSITEIT AMSTERDAM

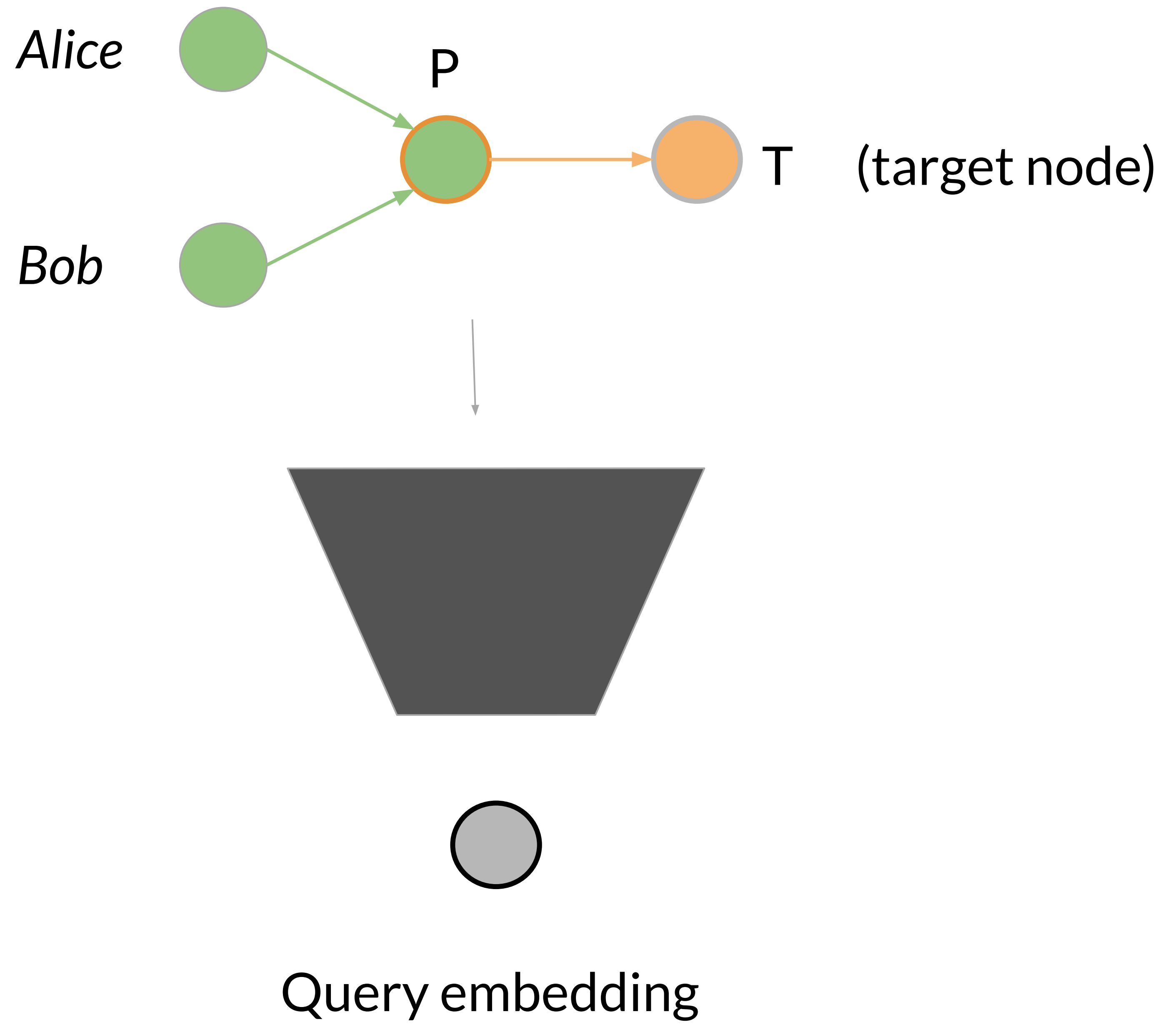
VU

The query encoder

- Graph Convolutional Networks operate on graphs, by applying message passing:
  - **Messages are vectors**

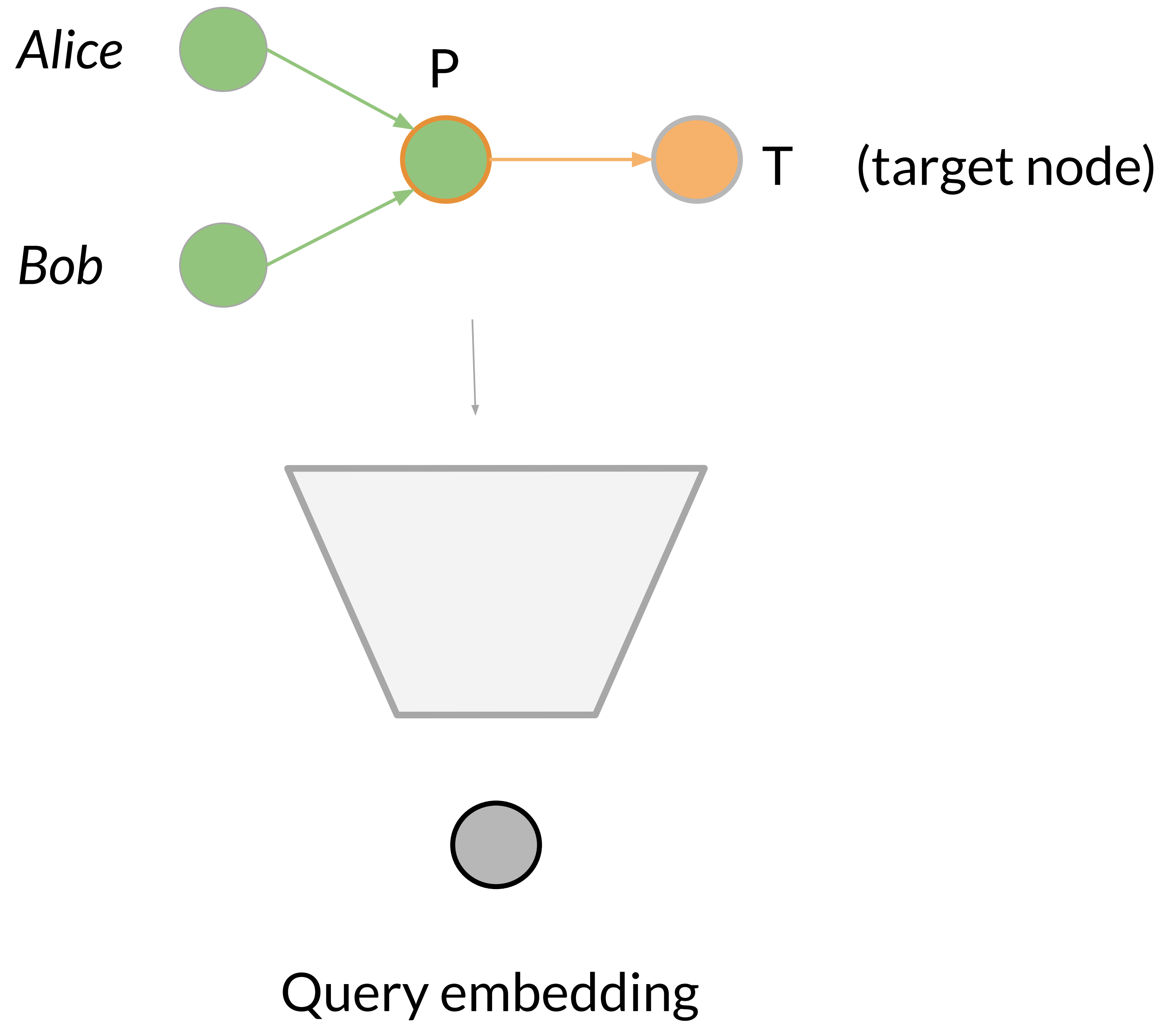
- **Message-Passing Query Embedding:**
  - Learnable parameters include both **entity** and **variable** node embeddings
  - Propagate messages across the BGP
  - After *k* steps of MP, map all node messages to a single query embedding

-

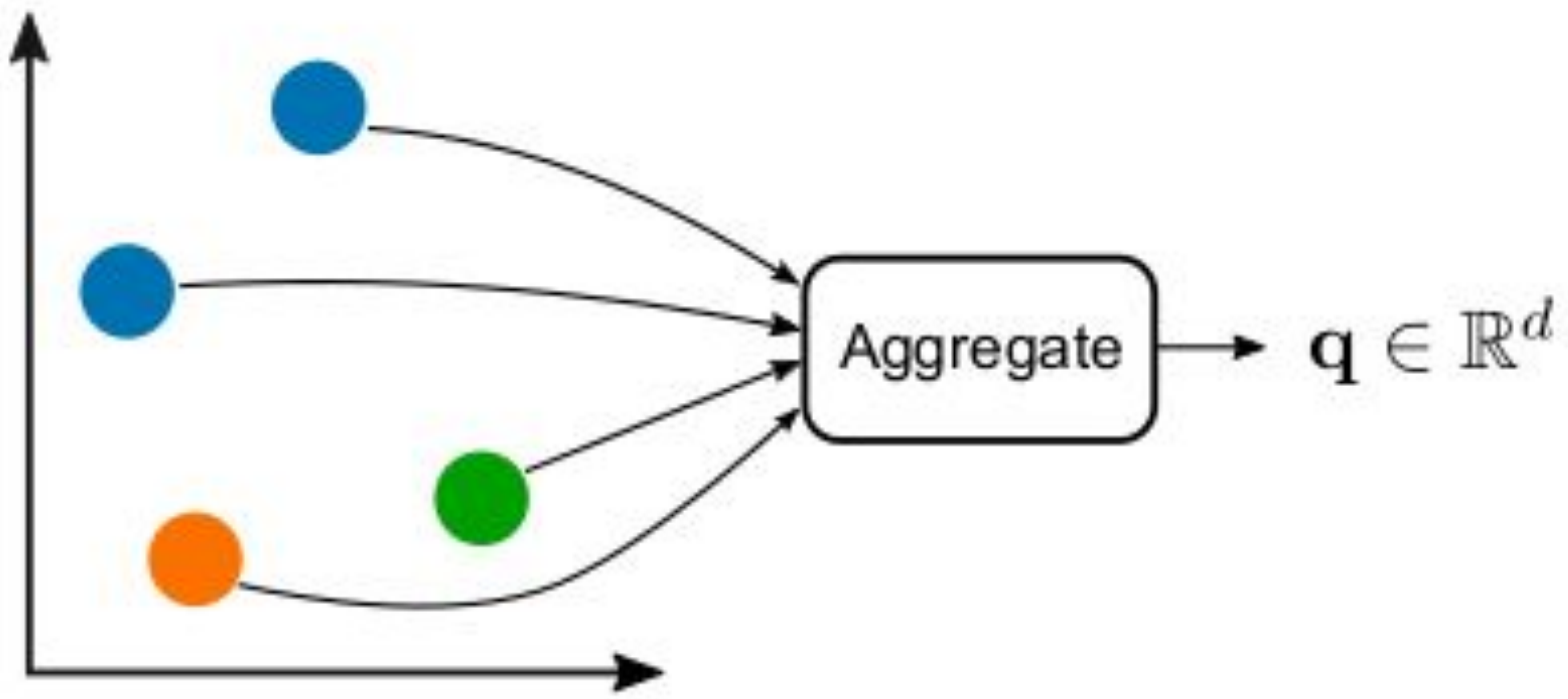


Graph aggregation functions

- Map node messages to query embedding
- Ideally permutation invariant
- Can contain learnable parameters for increased flexibility
- Simplest form: message at the target node



Graph aggregation functions



- Sum
- Max
- MLP

$$\mathbf{q} = \sum_{v \in \mathcal{V}_q} \mathrm{MLP}\left(\mathbf{h}_v^{(L)}\right)$$

- CMLP

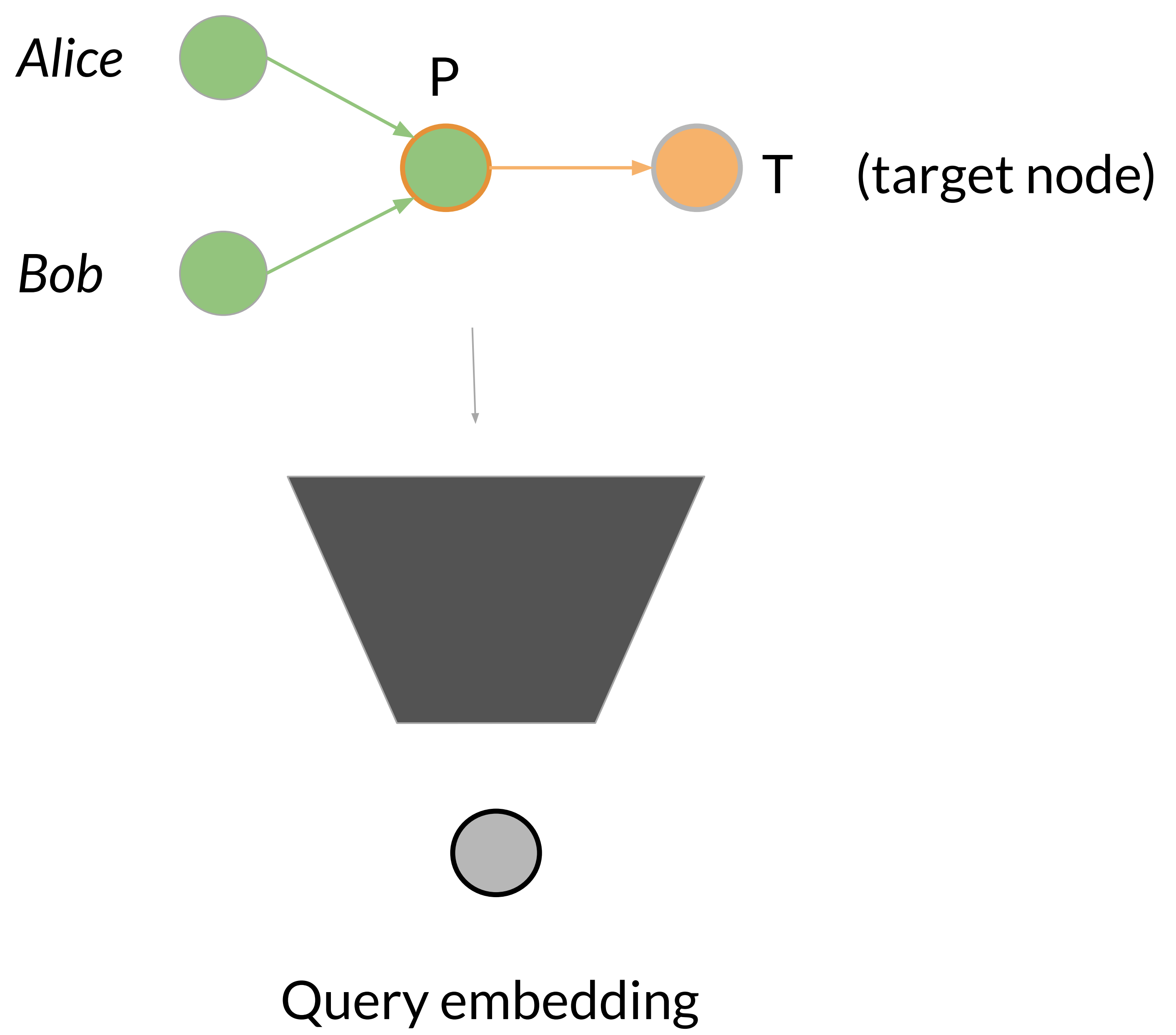
$$\mathbf{q} = \sum_{v \in \mathcal{V}_q} \mathrm{MLP}\left([\mathbf{h}_v^{(1)}, \ldots, \mathbf{h}_v^{(L)}]\right)$$

- TMLP

$$\mathbf{q} = \sum_{v \in \mathcal{V}_q} \mathrm{MLP}\left([\mathbf{h}_v^{(L)}, \mathbf{h}_{V_a}^{(L)}]\right)$$

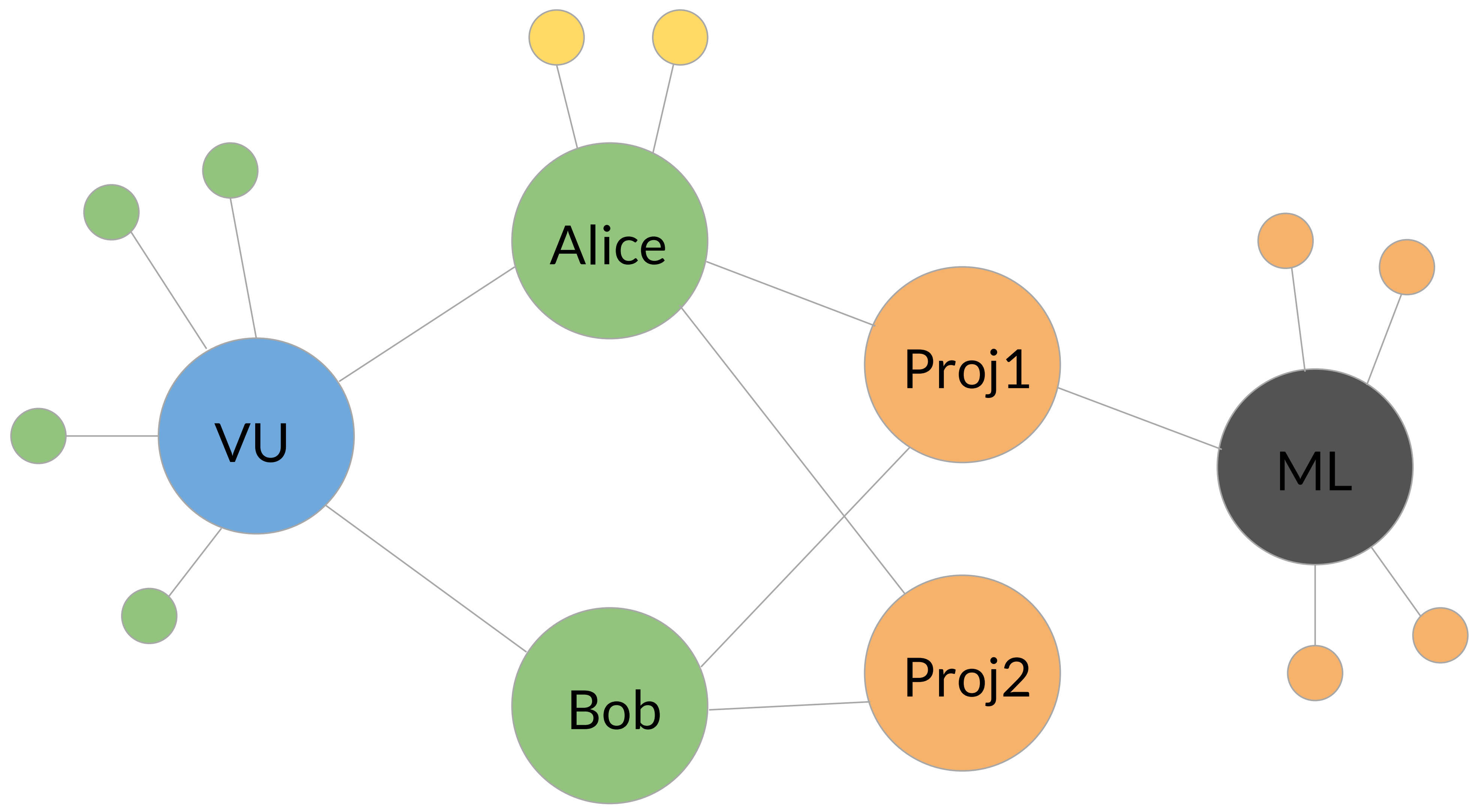
Why is this a good idea?

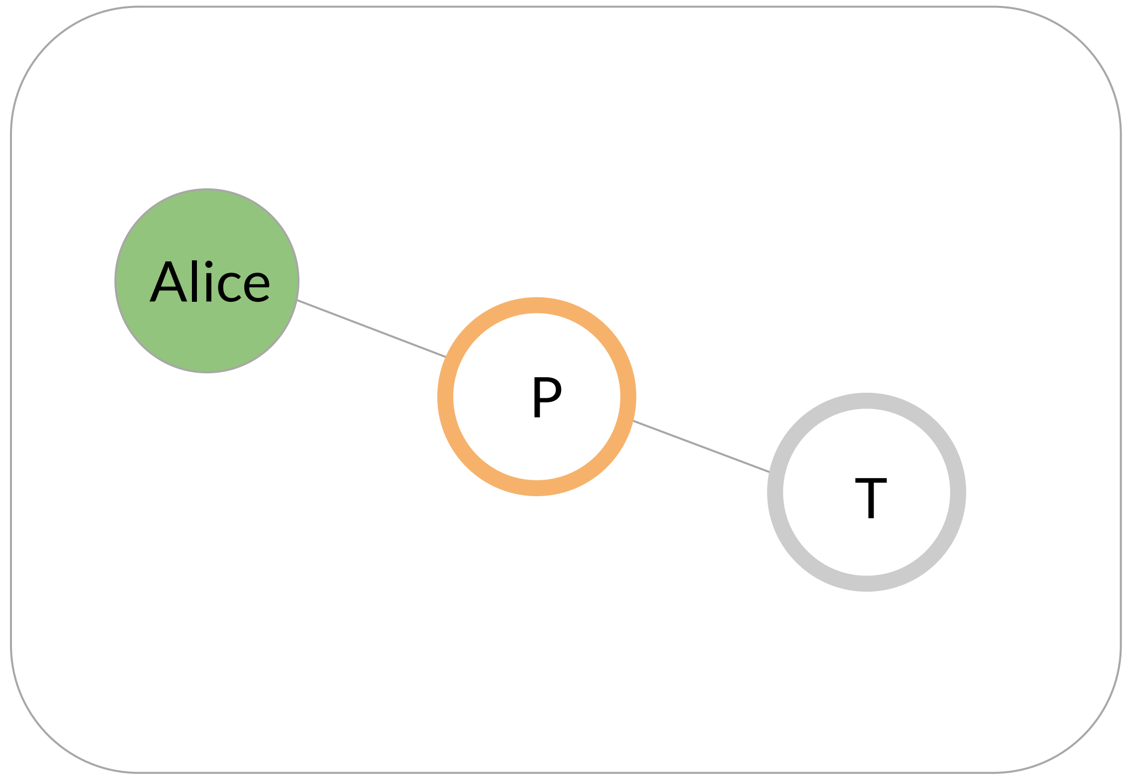
- Query encoded in embedding space before matching
- Answering is then O(n) instead of exponential
- MPQE encodes arbitrary queries

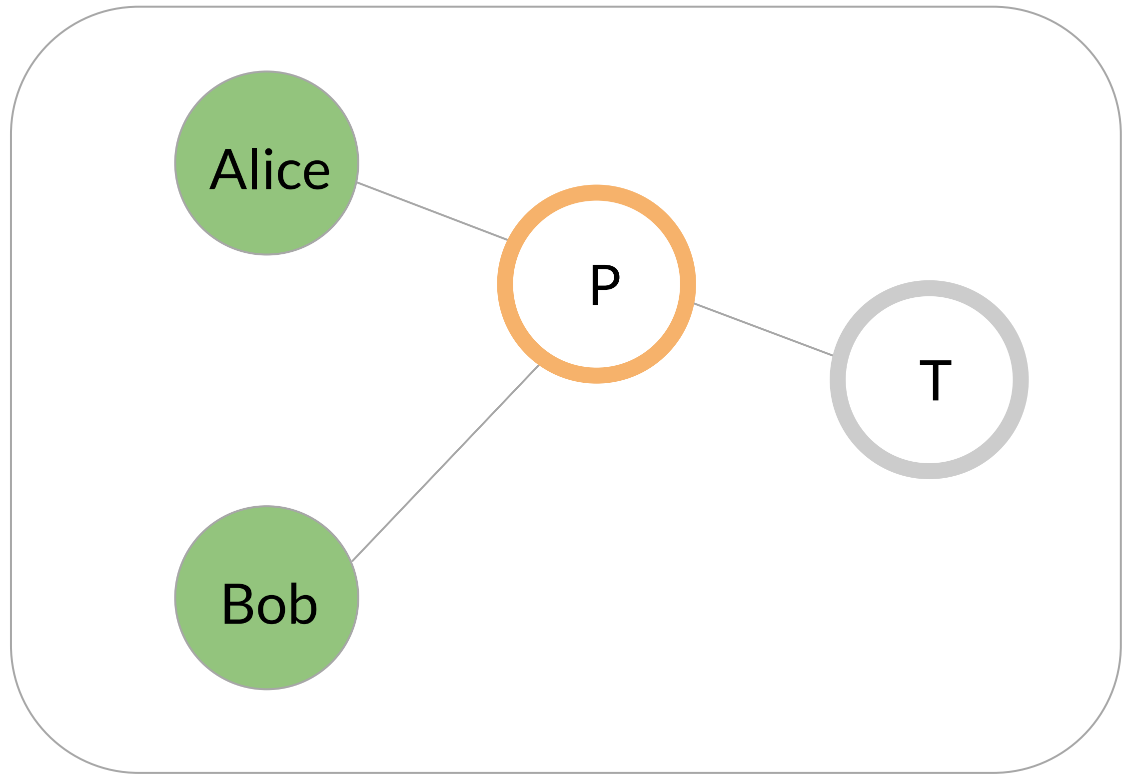


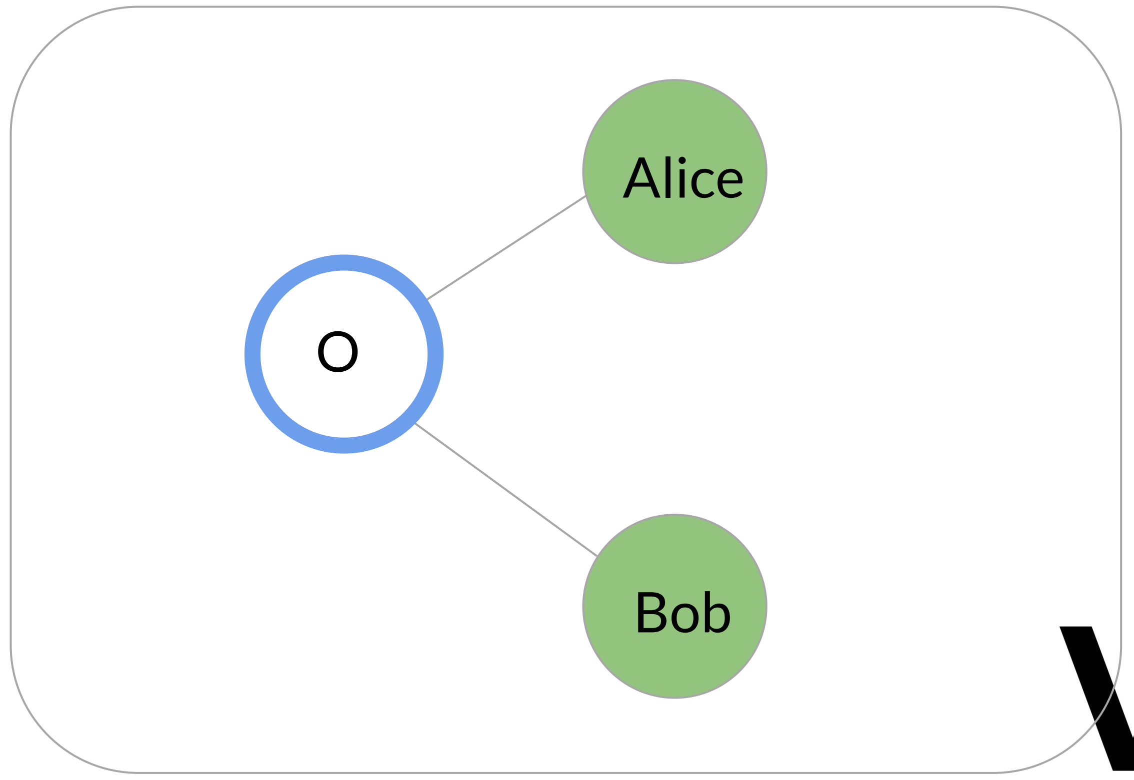
Evaluation

- Queries obtained from KG:
  - Sample subgraphs
  - Replace some entities by variables









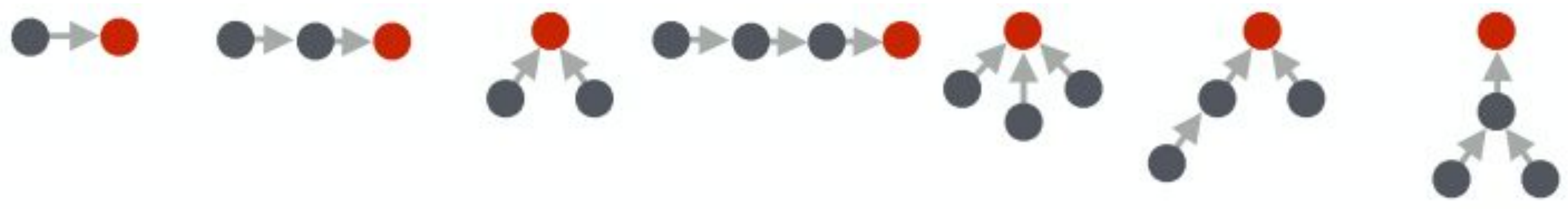
Evaluation

- Queries for training obtained after dropping some edges
- 4 knowledge graphs

	AIFB	MUTAG	AM	Bio
Entities	2,601	22,372	372,584	162,622
Entity types	6	4	5	5
Relations	39,436	81,332	1,193,402	8,045,726
Relation types	49	8	19	56

Evaluation

- Crucial question: how does a method generalize to unseen queries?
- Two scenarios:
  - Train on all 7 structures, evaluate on same structures
  - Train on **1-chain queries only**, evaluate on all 7 structures



VU

Results - all query types

	AIFB				MUTAG				AM				Bio			
	AUC		APR		AUC		APR		AUC		APR		AUC		APR	
Method	Base	All	Base	All	Base	All	Base	All	Base	All	Base	All	Base	All	Base	All
GQE-TransE	85.1	83.1	**87.9**	**86.7**	**94.5**	78.8	93.9	81.0	**92.4**	80.9	**92.1**	82.3	94.6	87.4	95.4	88.9
GQE-DistMult	85.1	**83.8**	86.6	86.0	81.3	**80.6**	81.8	**81.1**	83.9	**82.9**	84.8	83.2	97.0	90.0	96.5	90.3
GQE-Bilinear	**86.0**	83.4	84.0	83.3	94.0	78.5	**94.0**	79.7	91.0	80.7	91.5	**84.4**	**98.1**	**90.5**	**97.4**	**90.8**
RGCN-TM	89.3	84.9	90.0	87.4	91.2	**76.7**	90.9	**77.6**	**92.0**	**84.2**	**92.4**	**86.3**	**98.2**	88.8	**97.7**	89.8
RGCN-sum	88.1	84.7	88.7	86.8	**92.4**	74.6	90.9	73.1	90.1	80.9	91.0	83.6	98.1	90.0	97.3	90.5
RGCN-max	87.6	83.4	88.1	85.9	91.4	74.9	89.4	72.7	90.3	80.9	90.6	82.5	97.3	88.3	96.4	88.7
RGCN-MLP	89.2	85.8	90.7	87.3	90.9	73.7	90.9	74.8	92.0	82.9	91.7	84.1	97.8	89.9	97.2	90.0
RGCN-CMLP	**90.0**	**86.3**	**91.6**	**89.1**	92.0	74.3	**91.2**	72.5	91.9	82.5	92.3	85.5	98.0	90.1	97.3	90.2
RGCN-TMLP	89.3	85.5	90.2	87.4	91.7	74.4	90.7	73.6	91.1	83.3	91.4	84.9	98.0	**90.2**	97.6	**90.6**

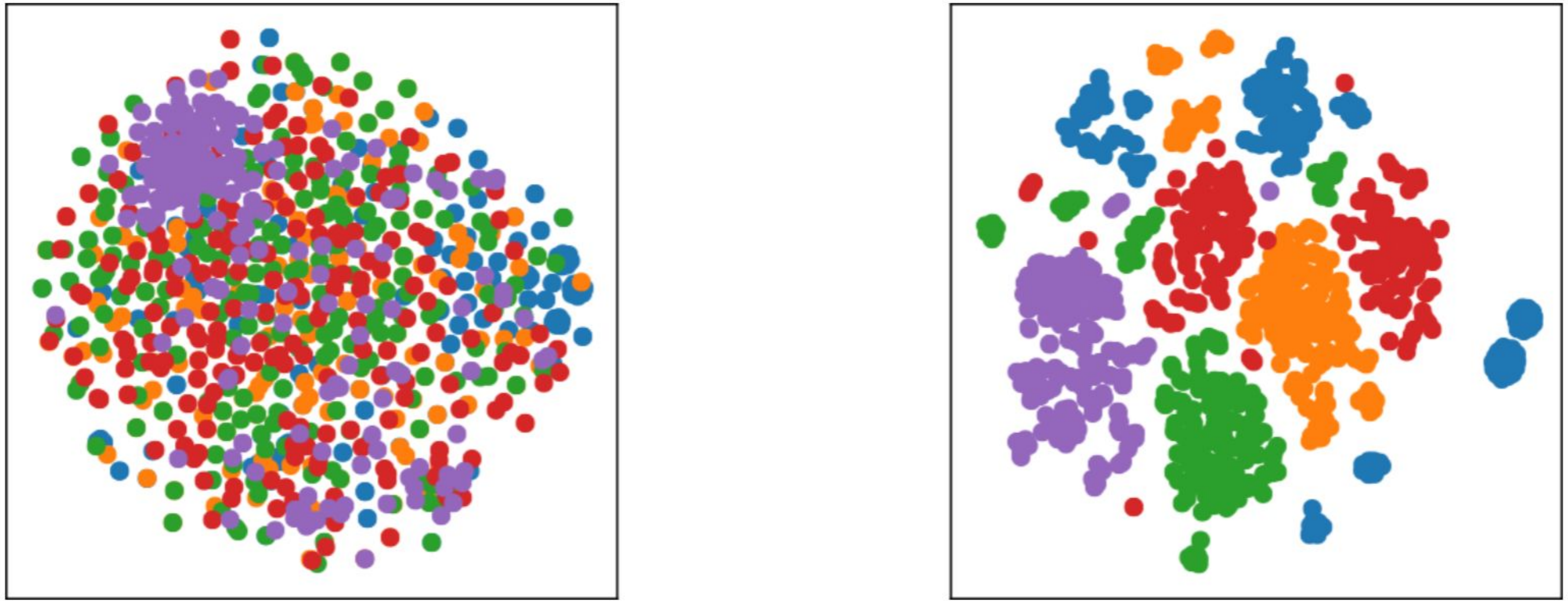
- We obtain competitive performance with previous work.
- Message-passing alone(RGCN-TM) is an effective mechanism

Results - 1-chain queries

Method	AIFB		MUTAG		AM		Bio	
	ch	all	ch	all	ch	all	ch	all
GQE-TransE	74.0	—	**89.4**	—	85.8	—	85.5	—
GQE-DistMult	72.8	—	85.4	—	82.4	—	95.9	—
GQE-Bilinear	72.7	—	89.1	—	**85.9**	—	85.8	—
RGCN-TM	**77.0**	**75.5**	86.8	**77.2**	**85.0**	**81.6**	**96.4**	**83.9**
RGCN-sum	69.8	69.6	82.8	74.0	52.5	53.9	92.4	80.0
RGCN-max	74.1	71.9	77.1	71.6	51.2	53.0	92.0	79.9
RGCN-MLP	69.1	68.0	76.0	70.0	51.3	53.8	90.7	78.7
RGCN-CMLP	69.7	69.1	84.6	74.2	51.5	53.8	89.8	78.3
RGCN-TMLP	75.0	75.4	80.1	71.9	53.1	53.5	91.4	79.4

- By training for **link prediction only**, our method generalizes to other 6, more complex query structures that were not seen during training

Learned representations



- Compared to previous methods (right), our method (left) learns embeddings that cluster according to the type of the entity.
- This points to future applications in learning better embeddings for KGs

Using R-GCN for Query embedding - Conclusion

- The proposed architecture is simple and learns entity and type embeddings useful for solving the task
- Our method allows encoding a general set of queries defined in terms of BGPs, by learning entity and variable embeddings and not constraining the query structure
- The message passing mechanism across the BGP exhibits superior generalization than previous methods
- Embeddings successfully capture the notion of entity types **without supervision**

THE PLAN

part 1: Introduction - Why graphs? What are embeddings?

part 2: Graph Embedding Techniques

part 3: Graph Neural Networks

part 4: Application - Query embedding