



# NODE EMBEDDINGS AND APPLICATION IN BIOMEDICAL ONTOLOGY

---

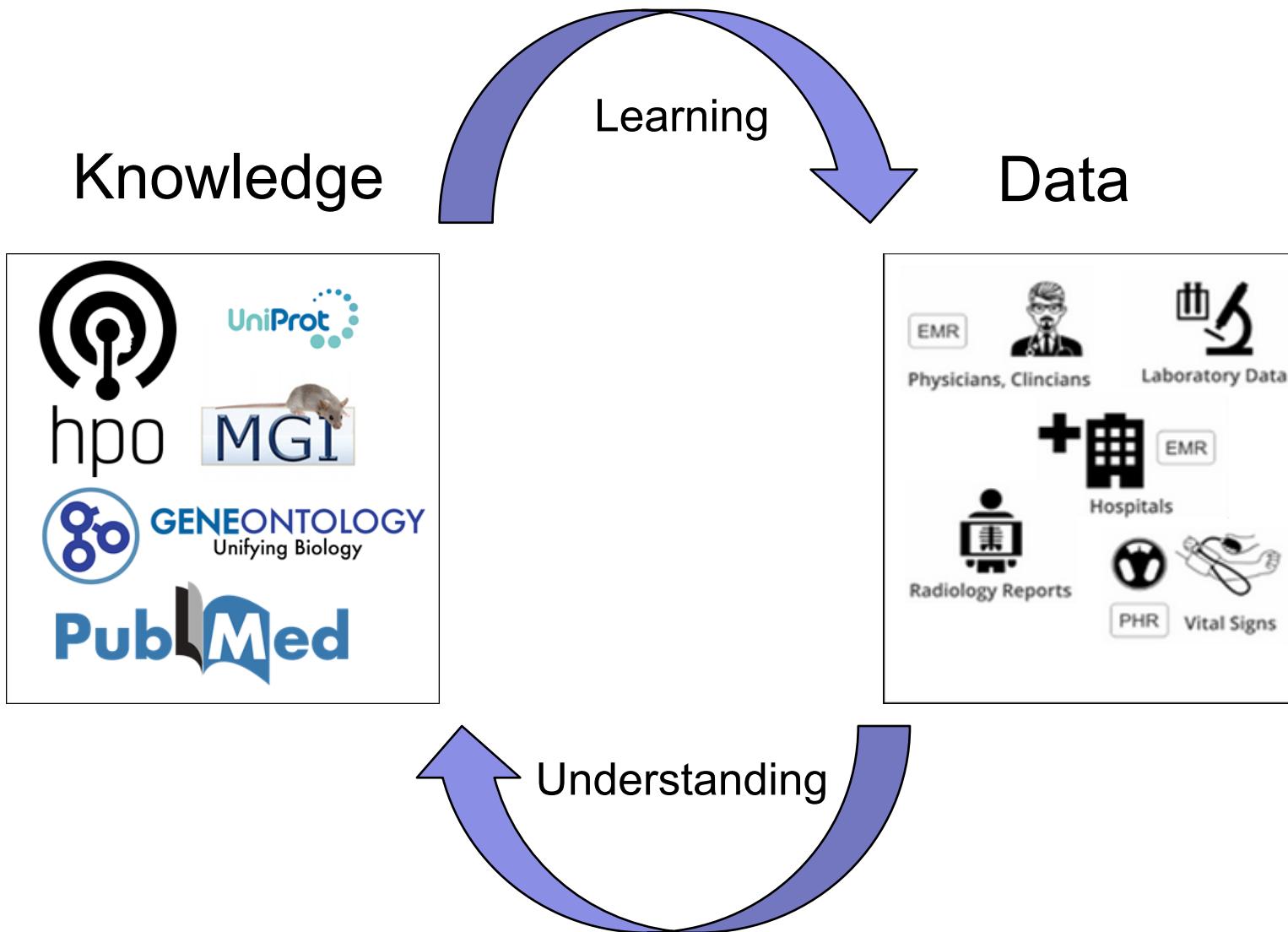
Feichen Shen, PhD  
Mayo Clinic, Division of Digital Health Sciences  
Shen.feichen@mayo.edu



# OUTLINES

- Introduction
- Word representation
- Node representation
- Application of constructing node embeddings for a biomedical ontology
- Implementation details

# INTRODUCTION



# WORD REPRESENTATION

# WORD REPRESENTATION

- Representing words using a vocabulary of words
- $V = [a, \text{apple}, \text{bee}, \dots, \text{zoo}]$
- Assume  $|V| = 10,000$
- one-hot encoding

Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$

# WORD REPRESENTATION

- Using one-hot encoding, it is hard to identify semantic relationship between two vectors.
- Apple and orange are much more similar than king and orange.
- Better to have a featurized representation for each of these words

# WORD REPRESENTATION

	Gender	Royal	Food	Cost	...
King	-1	0.93	0.02	0	...
Queen	1	0.95	0.01	0	...
Apple	0	-0.01	0.95	0.93	...
Orange	0.01	0	0.97	0.91	...

# WORD EMBEDDINGS

Distributed representations of text in an n-dimensional space

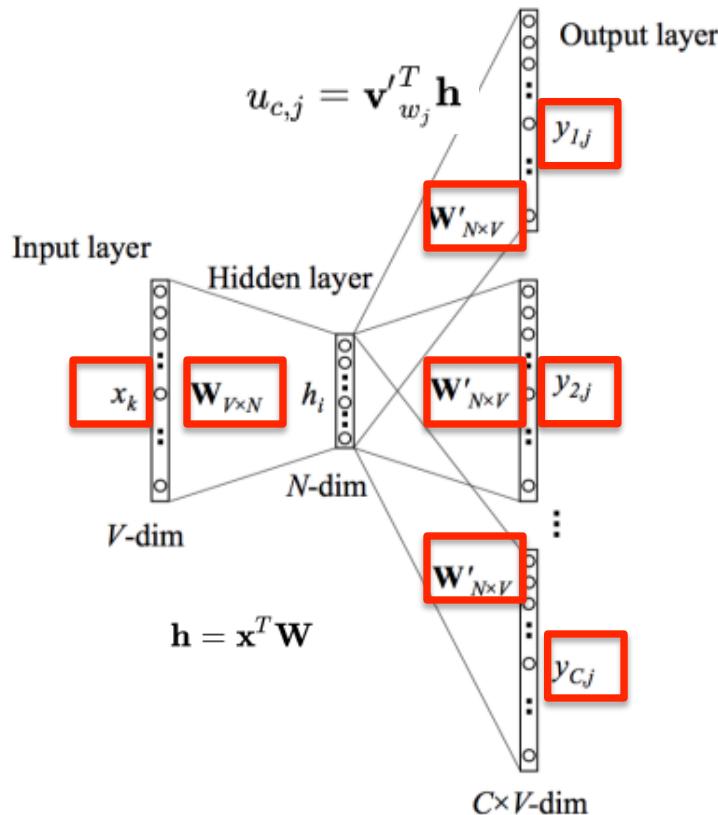
# WORD2VEC

- One of the models used to learn word embeddings
- Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems 2013 (pp. 3111-3119).
- Skip-Gram and Continuous Bag-of-Words (CBOW)

# WORD2VEC

Skip-Gram

$$\frac{\exp(u_{c,j})}{\sum_{j'=1}^V \exp(u_{j'})}$$



Predict context words given an input word

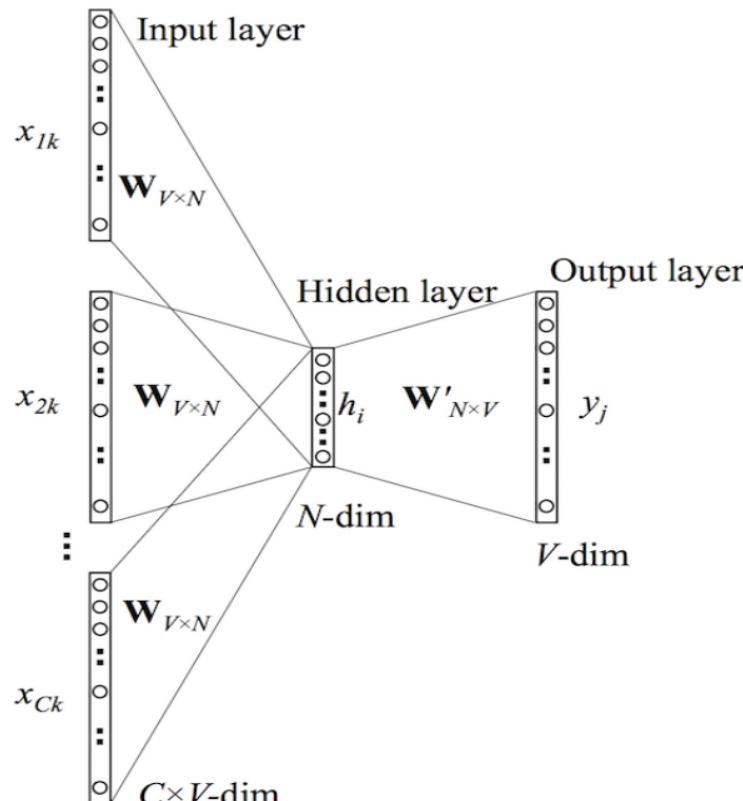
Loss function

$$\begin{aligned} E &= -\log p(w_{O,1}, w_{O,2}, \dots, w_{O,C} | w_I) \\ &= -\log \prod_{c=1}^C \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^V \exp(u'_{j'})} \\ &= -\sum_{c=1}^C u_{j_c^*} + C \cdot \log \sum_{j'=1}^V \exp(u'_{j'}) \end{aligned}$$

Learn the weights with Backpropagation

# WORD2VEC

## Continuous Bag-of-Words (CBOW)



Input: Context words with window size C

Output: A single word

Loss function

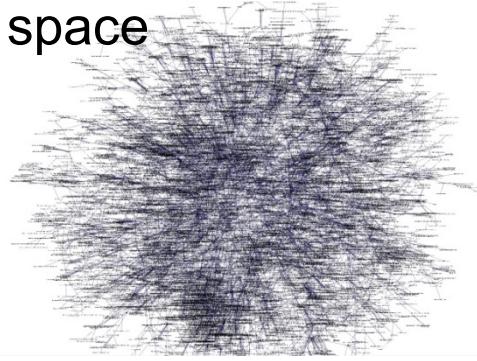
$$\begin{aligned} E &= -\log p(w_O | w_I) \\ &= -u_{j*} - \log \sum_{j'=1}^V \exp(u_{j'}) \\ &= -\mathbf{v}_{w_O}^T \cdot \mathbf{h} - \log \sum_{j'=1}^V \exp(\mathbf{v}_{w_{j'}}^T \cdot \mathbf{h}) \end{aligned}$$

Learn the weights with Backpropagation

# NODE EMBEDDINGS

- Similar to word embeddings
- Node→Word, Neighborhood→Context
- However, sliding window for text is not suitable for non-linear graph
- One solution is to use random walk to select “context” in graph

Graph space



Embedding space



**# -0.484338 0.85822 0.153959 -0.098285 -0.064248 0.235987 -0.189068 -0.308805 0.119551 0.173563 0.205180 -0.064880 -0.124346 -0.032367 0**  
**-0.272518 0.876239 0.161163 0.821788 -0.07625 0.278913 -0.195849 -0.394783 -0.128355 0.208492 0.445966 -0.061082 -0.165238 -0.057952 0**  
**0.188285 0.857289 0.153959 0.821788 -0.07625 0.278913 -0.195849 -0.394783 -0.128355 0.208492 0.445966 -0.061082 -0.165238 -0.057952 0**  
**0.1833 -0.259797 0.857397 0.163549 -0.082269 -0.077151 0.241342 -0.191172 -0.379289 -0.126249 0.182148 0.415996 -0.061236 -0.177352 -0.078984 0**  
**0.989 -0.258128 0.855444 0.166560 0.814121 -0.078908 0.248623 -0.191175 -0.365271 -0.127958 0.181381 0.414394 -0.059354 -0.177944 -0.058668 0**  
**-0.202385 0.855444 0.166560 0.814121 -0.078908 0.248623 -0.191175 -0.365271 -0.127958 0.181381 0.414394 -0.059354 -0.177944 -0.058668 0**  
**0.838 -0.242548 0.849378 0.152868 0.804368 -0.085052 0.225471 -0.169433 -0.322209 -0.111762 0.173882 0.217144 -0.056781 -0.128984 -0.056819 0**  
**-0.127256 0.856669 0.152185 0.804085 -0.085240 0.219648 -0.157795 -0.317989 -0.108145 0.163389 0.255088 -0.054571 -0.118908 -0.054513 0**  
**0.327 -0.228755 0.854887 0.141803 0.812875 -0.085397 0.219548 -0.155488 -0.317698 -0.108549 0.162329 0.244095 -0.057285 -0.122585 -0.053337 0**  
**0.927 -0.225669 0.856669 0.152185 0.804085 -0.085240 0.219648 -0.157795 -0.317989 -0.108145 0.163389 0.255088 -0.054571 -0.118908 -0.054513 0**  
**0.182385 0.857289 0.153959 0.821788 -0.07625 0.278913 -0.195849 -0.394783 -0.128355 0.208492 0.445966 -0.061082 -0.165238 -0.057952 0**  
**0.1759 -0.212364 0.852816 0.135453 0.805083 -0.062068 0.205461 -0.162629 -0.308093 -0.111027 0.161258 0.351261 -0.050571 -0.115397 -0.051769 0**  
**0.1461 -0.222394 0.853975 0.132621 0.814579 -0.057988 0.215109 -0.153418 -0.313987 -0.101387 0.159752 0.352478 -0.051913 -0.115986 -0.056767 0**  
**0.1785 -0.217818 0.858482 0.134099 0.812892 -0.066442 0.205347 -0.155849 -0.311955 -0.118612 0.159860 0.345328 -0.054969 -0.121382 -0.047748 0**  
**0.1759 -0.212364 0.852816 0.135453 0.805083 -0.062068 0.205461 -0.162629 -0.308093 -0.111027 0.161258 0.351261 -0.050571 -0.115397 -0.051769 0**  
**1.061 -0.223488 0.852431 0.140303 0.802554 -0.063774 0.206661 -0.162022 -0.321131 -0.111048 0.155479 0.353826 -0.053930 -0.112905 -0.051936 0**  
**0.989 -0.217818 0.858482 0.134099 0.812892 -0.066442 0.205347 -0.155849 -0.311955 -0.118612 0.159860 0.345328 -0.054969 -0.121382 -0.047748 0**  
**0.677 -0.214353 0.852189 0.134712 0.810339 -0.052341 0.208794 -0.155481 -0.302341 -0.108049 0.160988 0.344085 -0.055916 -0.116665 -0.057178 0**  
**0.989 -0.217818 0.858482 0.134099 0.812892 -0.066442 0.205347 -0.155849 -0.311955 -0.118612 0.159860 0.345328 -0.054969 -0.121382 -0.047748 0**  
**0.677 -0.214353 0.852189 0.134712 0.810339 -0.052341 0.208794 -0.155481 -0.302341 -0.108049 0.160988 0.344085 -0.055916 -0.116665 -0.057178 0**  
**0.323 -0.215450 0.857118 0.154687 0.819977 -0.059549 0.210862 -0.167736 -0.313218 -0.103820 0.156158 0.346836 -0.054856 -0.127748 -0.051816 0**  
**0.259 -0.212409 0.849853 0.131934 0.801548 -0.061963 0.199641 -0.158844 -0.317426 -0.109323 0.149482 0.340803 -0.058074 -0.122382 -0.050471 0**  
**0.989 -0.217818 0.858482 0.134099 0.812892 -0.066442 0.205347 -0.155849 -0.311955 -0.118612 0.159860 0.345328 -0.054969 -0.121382 -0.047748 0**  
**0.4387 -0.213621 0.854541 0.135453 0.811804 -0.059481 0.205574 -0.154599 -0.313588 -0.108101 0.154249 0.345728 -0.056127 -0.113311 -0.051572 0**  
**0.989 -0.217818 0.858482 0.134099 0.812892 -0.066442 0.205347 -0.155849 -0.311955 -0.118612 0.159860 0.345328 -0.054969 -0.121382 -0.047748 0**  
**0.696 -0.215350 0.841268 0.132584 0.812829 -0.058399 0.208397 -0.155883 -0.305718 -0.108320 0.155889 0.339807 -0.054132 -0.116824 -0.046223 0**  
**1.021 -0.215350 0.841268 0.132584 0.812829 -0.058399 0.208397 -0.155883 -0.305718 -0.108320 0.155889 0.339807 -0.054132 -0.116824 -0.046223 0**  
**0.2802 -0.215393 0.852397 0.141803 0.805185 -0.061963 0.199641 -0.158844 -0.317426 -0.109323 0.149482 0.340803 -0.058074 -0.122382 -0.050471 0**  
**0.332 -0.218970 0.839871 0.131613 0.809148 -0.048731 0.206661 -0.162545 -0.384422 -0.108703 0.156964 0.345495 -0.059513 -0.116919 -0.048142 0**  
**0.989 -0.217818 0.858482 0.134099 0.812892 -0.066442 0.205347 -0.155849 -0.311955 -0.118612 0.159860 0.345328 -0.054969 -0.121382 -0.047748 0**  
**0.884 -0.218952 0.859433 0.129253 0.812169 -0.051514 0.198383 -0.144688 -0.396767 0.146931 0.228899 -0.058625 -0.138852 -0.047189 0**  
**0.989 -0.217818 0.858482 0.134099 0.812892 -0.066442 0.205347 -0.155849 -0.311955 -0.118612 0.159860 0.345328 -0.054969 -0.121382 -0.047748 0**  
**1.061 -0.219930 0.841820 0.132277 0.809562 -0.065625 0.190352 -0.156987 -0.291991 -0.095114 0.152518 0.330234 -0.053432 -0.111469 -0.051905 0**  
**1.075 -0.207383 0.841820 0.132277 0.809562 -0.065625 0.190352 -0.156987 -0.291991 -0.095114 0.152518 0.330234 -0.053432 -0.111469 -0.051905 0**  
**1.023 -0.219930 0.841820 0.132277 0.809562 -0.065625 0.190352 -0.156987 -0.291991 -0.095114 0.152518 0.330234 -0.053432 -0.111469 -0.051905 0**

# NODE EMBEDDINGS

## Node Embeddings with Random Walk

GE Algorithm	Ransom Walk Methods	Preserved Proximity	DL Model
DeepWalk[17] [34]	truncated random walk	$2^{nd}$	
GenVector [66]	truncated random walk	$2^{nd}$ (word-image)	
Constrained DeepWalk [25]	truncated random walk sampling with edge weight	$2^{nd}$ (user-user & concept-concept) $2^{nd}$	SkipGram with hierarchical softmax (Eq. (11))
DDRW [47]	truncated random walk	$2^{nd}$ + class identity	
TriDNR [73]	truncated random walk	$2^{nd}$ (among node, word & label)	
node2vec [28]	BFS + DFS	$2^{nd}$	
UPP-SNE [113]	truncated random walk	$2^{nd}$ (user-user & profile-profile)	SkipGram with
Planetoid [62]	sampling node pairs by labels and structure	$2^{nd}$ + label identity	negative sampling
NBNE [19]	sampling direct neighbours of a node	$2^{nd}$	(Eq. (12))
DGK [93]	graphlet kernel: Random sampling [114]	$2^{nd}$ (by graphlet)	SkipGram (Eqs. (11)–(12))
metapath2vec [46]	meta-path based random walk	$2^{nd}$	heterogeneous SkipGram
ProxEmbed [44]	truncate random walk	node ranking tuples	
HSNL [29]	truncate random walk	$2^{nd}$ + QA ranking tuples	LSTM
RMNL [30]	truncated random walk	$2^{nd}$ + user-question quality ranking	
DeepCas [63]	Markov chain based random walk	information cascade sequence	GRU
MRW-MN [36]	truncated random walk	$2^{nd}$ + cross-modal feature difference	DCNN+ SkipGram

# **NODE REPRESENTATION: NODE2VEC**

# NODE2VEC

- Scalable feature learning for networks.
- Grover A, Leskovec J.
- In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining 2016 Aug 13 (pp. 855-864). ACM.

# NODE2VEC

## 2 steps

- Sampling strategy

Apply random walk on graph to prepare input data

- Node embeddings

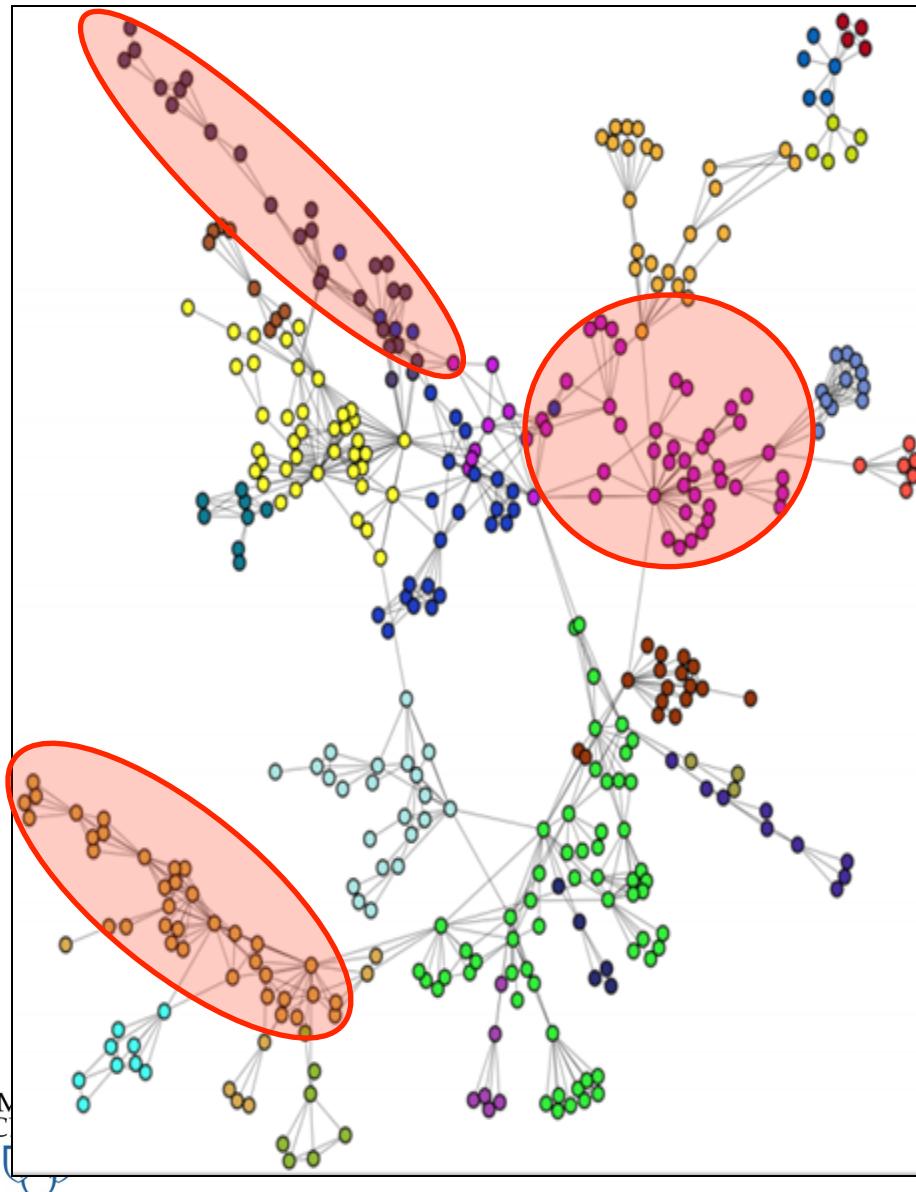
Apply word2vec on prepared input data to generate embeddings for node

# APPROACH

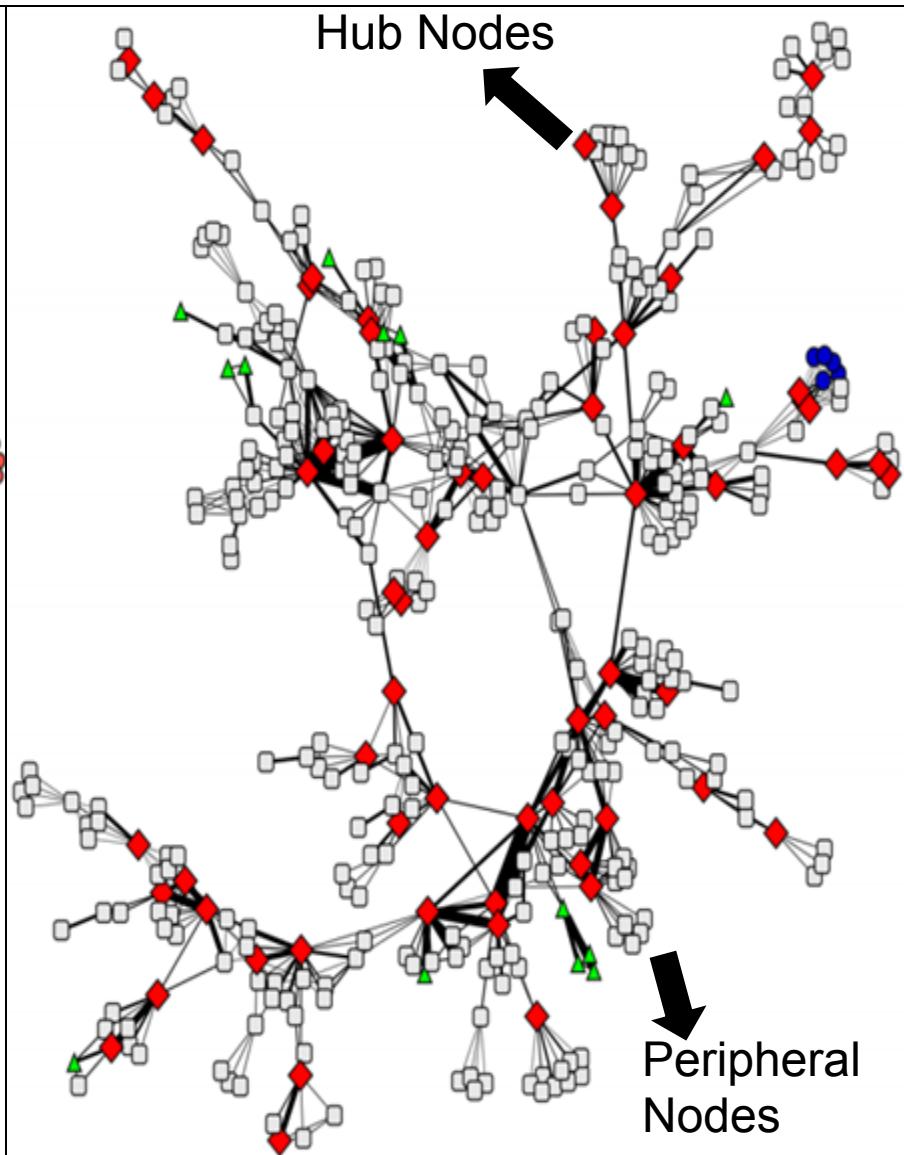
## -1. SAMPLING STRATEGY

- 1. **Homophily equivalence**: embed nodes from the same network community closely together
- 2. **Structural equivalence**: nodes share similar roles have similar embeddings

## Homophily equivalence



## Structural equivalence



# APPROACH

## -1. SAMPLING STRATEGY

- Classic search strategies

- Breadth-first sampling (BFS)
  - Depth-first sampling (DFS)

# APPROACH

## -1. SAMPLING STRATEGY

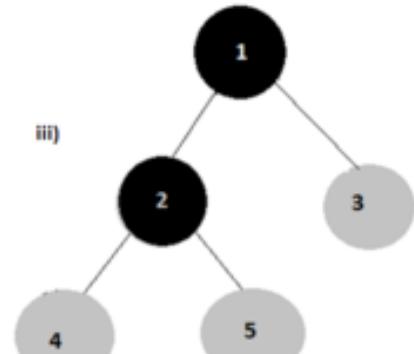
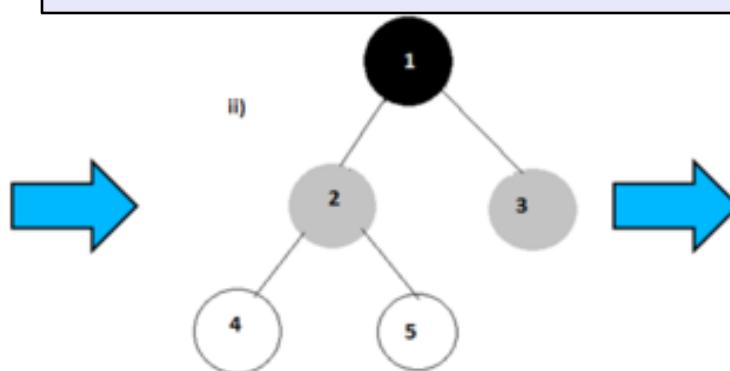
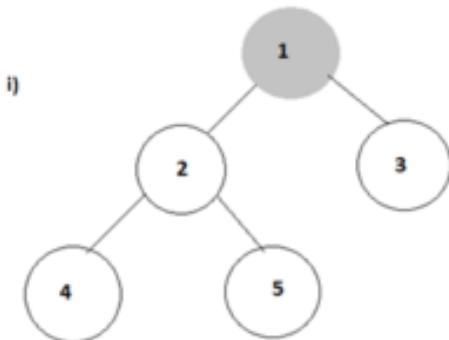
### Breadth-first sampling (BFS)

- The goal is to traverse all nodes in the graph.
- Starts at the top level of a graph and explores the neighbor nodes first before moving to the next level neighbors.
- Iterative, FIFO queue

1. Start from node 1

2. Find node 1 and continue to look at node 1's neighbor 2 and 3, now 2 and 3 has the highest priority to be searched

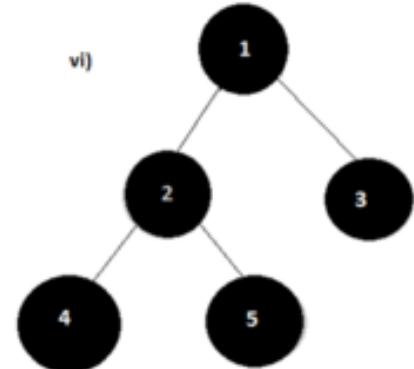
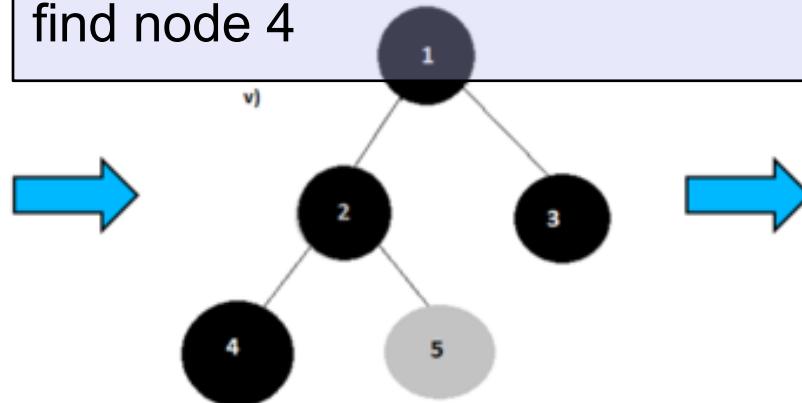
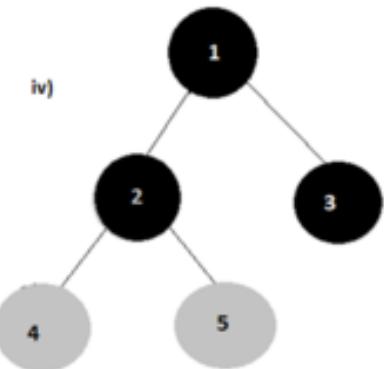
3. Find node 2



4. Find node 3

5. Check 2's neighbor 4 and 5, find node 4

6. Find node 5



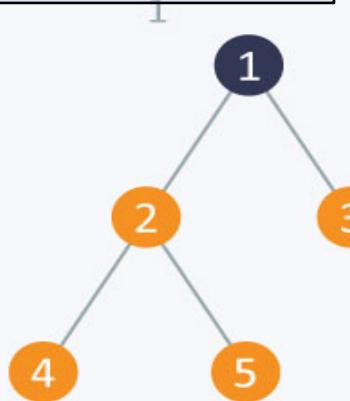
# APPROACH

## -1. SAMPLING STRATEGY

### Depth-first sampling (DFS)

- It starts from the top level and explores as far as possible along each branch before backtracking
- Recursive, LIFO stack

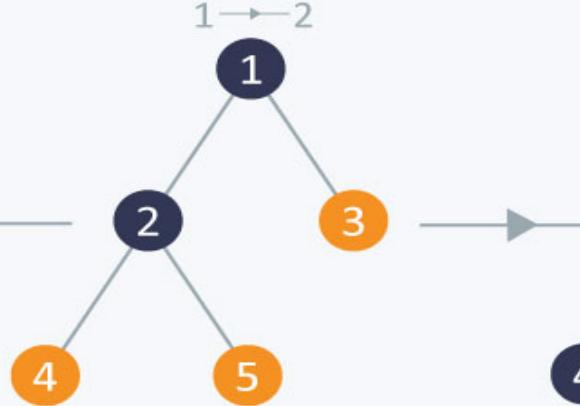
1. Start from node 1



2. Look at 1's neighbor in the left branch, and find node 2

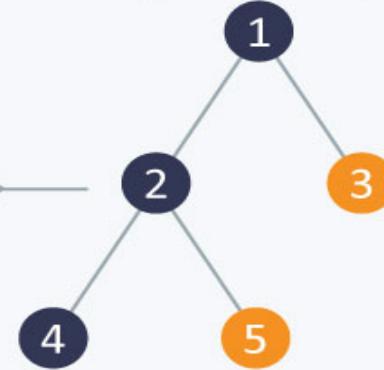
DFS

1 → 2

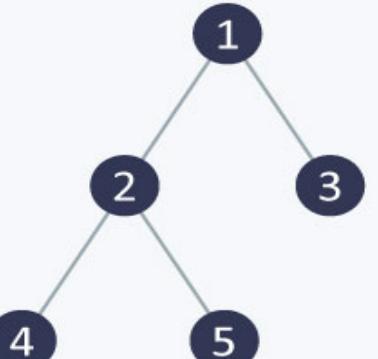
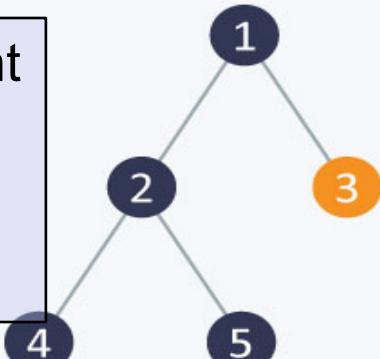


3. Find 2's left branch neighbor, node 4

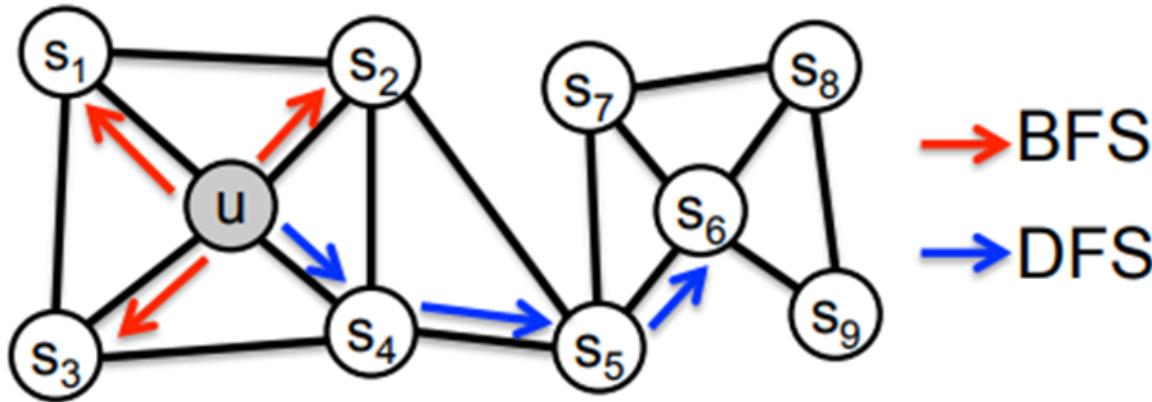
1 → 2 → 4



4. Go back to right branch, find node 5



5. All left branch has been searched. Go back to right branch and found node 3



- **BFS: structural analysis, microscopic view of neighborhood of every node**
- **DFS: homophily analysis, macro-view of the neighborhood among different communities**
- Real-world network has a lot of such mixture of BFS and DFS

# APPROACH

## -1. SAMPLING STRATEGY

- Flexible neighborhood sampling strategy to smoothly interpolate between BFS and DFS

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

Random Walk

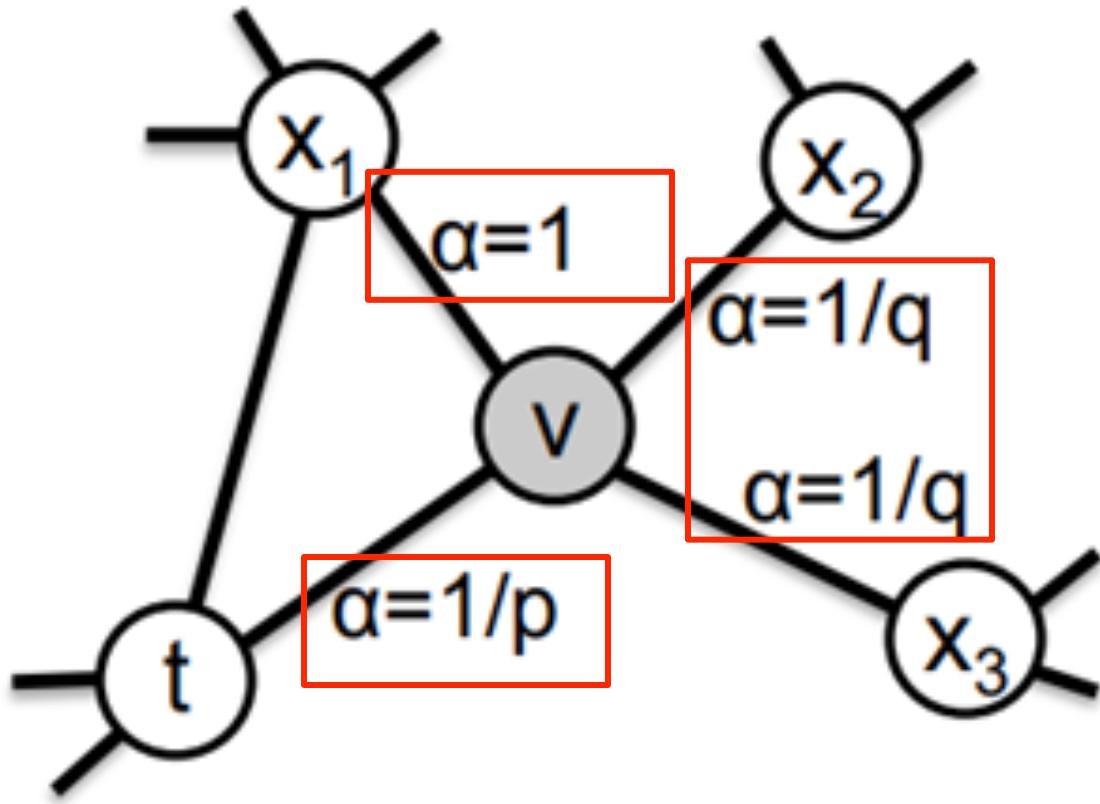
$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$$

Transition Probability

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

Bias Term

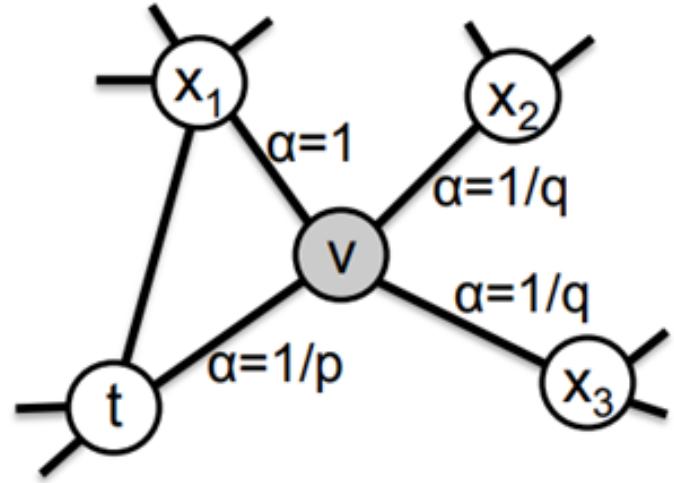
$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$



the current scenario is that the walk just transitioned from  $t$  to  $v$  and is now evaluating its next step out of node  $v$ .

# APPROACH

## -1. SAMPLING STRATEGY

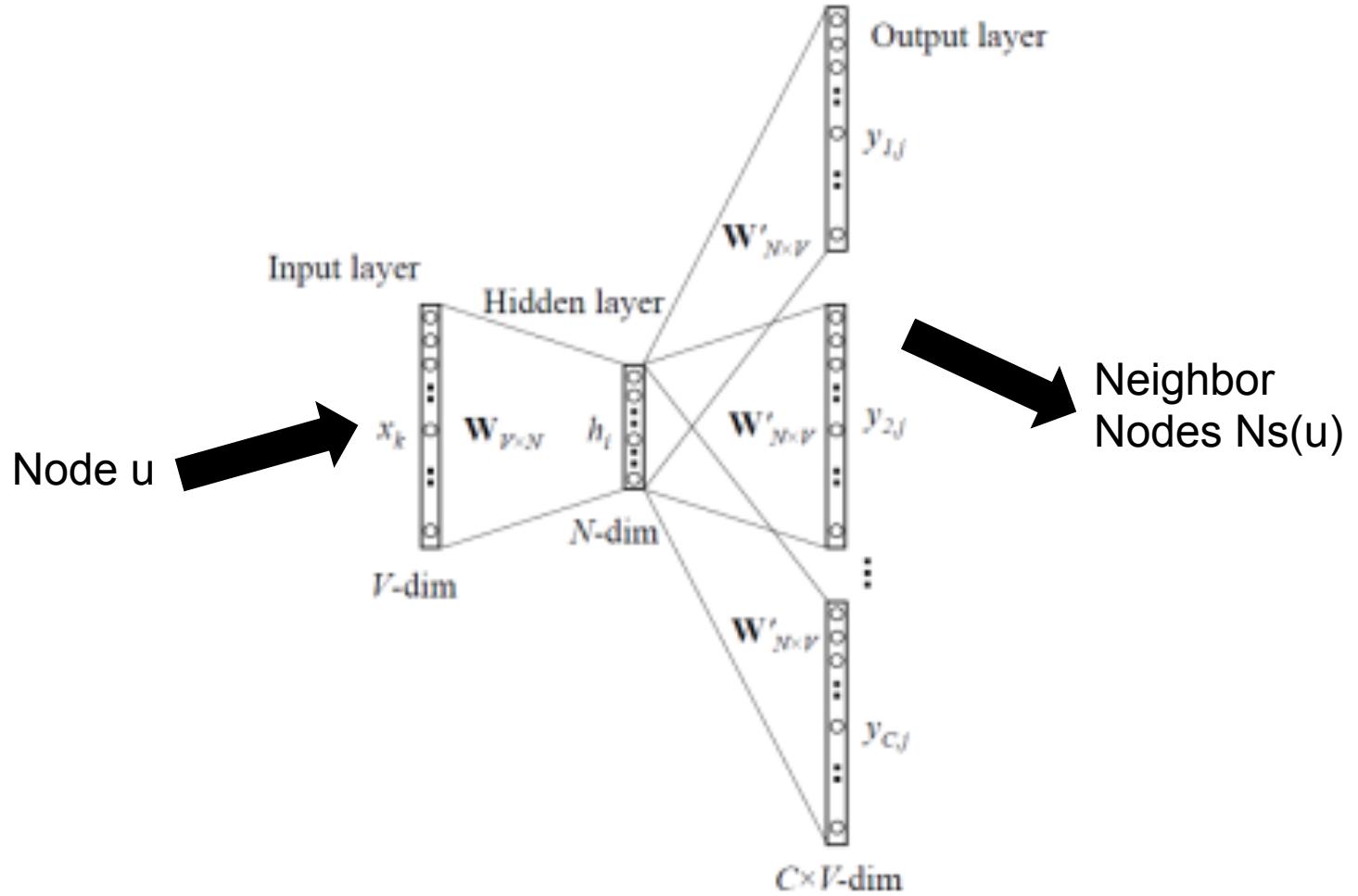


- $p$  controls the return step, and  $q$  controls the step of walk to outside world
- $p$ : return parameter. Control the likelihood of revisit. Large  $p$  ensure less revisit. Small  $p$  lead a back step
- $q$ : in-out parameter. BFS while  $q>1$  and DFS while  $q<1$
- $p$  and  $q$  are controller to balance between BFS and DFS

# APPROACH

## -2. WORD2VEC

Skip-gram



# APPROACH

## -2. WORD2VEC

- The purpose is to maximize the log-probability of  $N_s(u)$  given the input – feature representation of node  $u$

$$\max_f \sum_{u \in V} \log \Pr(N_s(u) | f(u))$$

- Optimize weight matrix with SGD

# **APPLY NODE2VEC ON BIOMEDICAL ONTOLOGY**

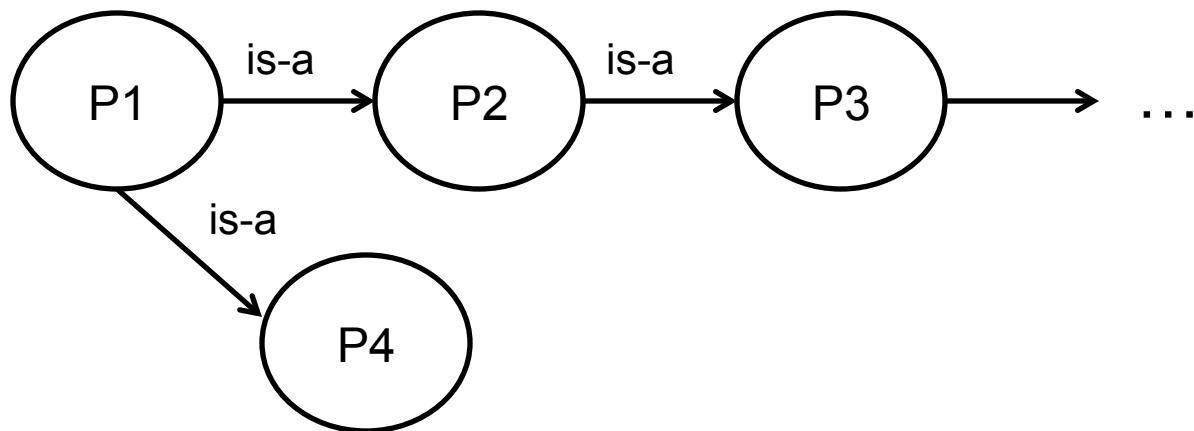
# ONTOLOGY

Representation, formal naming, and definition of the categories, properties, and relations between concepts, data and entities within one domain

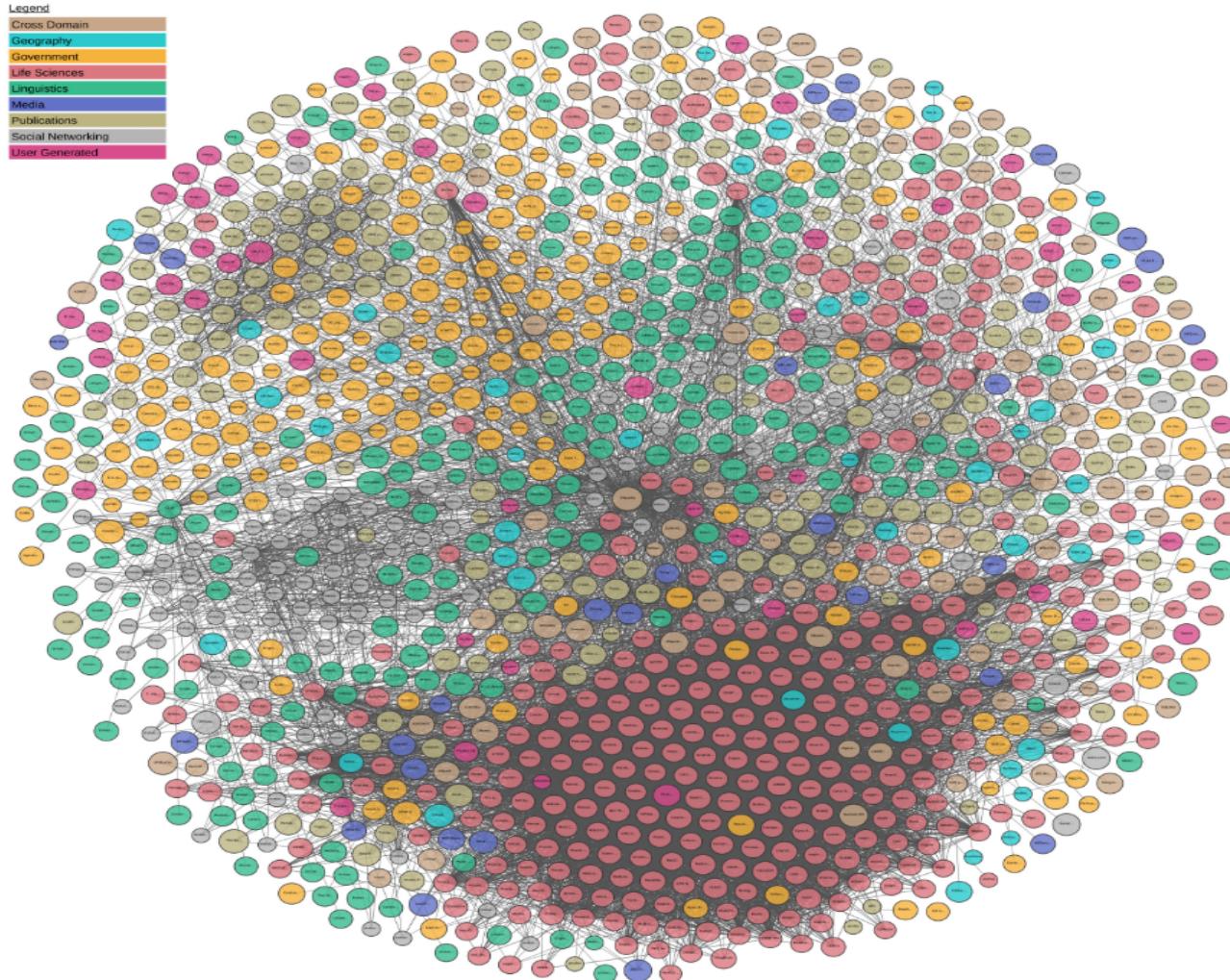
# ONTOLOGY

## Resource Description Framework (RDF)

- {Subject, Predict, Object} triplet is the smallest unit to build the RDF graph



# ONTOLOGY



# ONTOLOGY

- The Web Ontology Language (OWL) is used to describe ontology

.owl format

Java: Jena, OWL API

Python: OWLReady2, Ontospy

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://purl.obolibrary.org/obo/hp.owl#"
    xmlns:base="http://purl.obolibrary.org/obo/hp.owl"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:oboInOwl="http://www.geneontology.org/formats/oboInOwl#"
    xmlns:hsapdv="http://purl.obolibrary.org/obo/hsapdv#"
    xmlns:hp="http://purl.obolibrary.org/obo/hp#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:hp2="http://purl.obolibrary.org/obo/hp.owl#"
    xmlns:obo="http://purl.obolibrary.org/obo/"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
<owl:Ontology rdf:about="http://purl.obolibrary.org/obo/hp.owl">
    <oboInOwl:saved-by rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Peter Robinson, Sebastian Koehler, Sandra Doelken, Chris Mungall, Melissa Haendel, 1
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Please see license of HPO at http://www.human-phenotype-ontology.org</rdfs:comment>
    <oboInOwl:logical-definition-view-relation rdf:datatype="http://www.w3.org/2001/XMLSchema#string">has_part</oboInOwl:logical-definition-view-relation>
    <oboInOwl:default-namespace rdf:datatype="http://www.w3.org/2001/XMLSchema#string">human_phenotype</oboInOwl:default-namespace>
    <dc:creator>Peter N Robinson</dc:creator>
    <dc:contributor>Mark Engelstad</dc:contributor>
    <dc:contributor>Sandra Doelken</dc:contributor>
    <dc:rights>Peter Robinson, Sebastian Koehler, The Human Phenotype Ontology Consortium, and The Monarch Initiative</dc:rights>
    <dc:creator>Sebastian Koehler</dc:creator>
    <dc:contributor>Chris Mungall</dc:contributor>
    <dc:subject>Phenotypic abnormalities encountered in human disease</dc:subject>
    <dc:contributor>Joie Davis</dc:contributor>
    <dc:contributor>Courtney Hum</dc:contributor>
    <dc:contributor>Melissa Haendel</dc:contributor>
    <dc:contributor>Nicole Vasilevsky</dc:contributor>
    <dc:creator>The Monarch Initiative</dc:creator>
    <dc:creator>The Human Phenotype Ontology Consortium</dc:creator>
    <dc:license>see http://www.human-phenotype-ontology.org</dc:license>
```

# ONTOLOGY

.nt format

## Subject

<http://www.skeim.org#002>  
<http://www.skeim.org#0021>  
<http://www.skeim.org#0021>  
<http://www.skeim.org#0021>  
<http://www.skeim.org#0021105770>  
<http://www.skeim.org#0021105770>  
<http://www.skeim.org#0021105770>  
<http://www.skeim.org#0021105770>  
<http://www.skeim.org#0021105770>  
<http://www.skeim.org#0021105770>  
<http://www.skeim.org#0021105770>  
<http://www.skeim.org#0021105770>

## Predicate

<http://www.skeim.org#hasLevel>  
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  
< http://www.w3.org/2000/01/rdf-schema#Comment>  
< http://www.w3.org/2000/01/rdf-schema#subClassOf>  
< http://www.w3.org/ns/org#memberOf>  
< http://www.skeim.org#hasEIN>  
<http://www.w3.org/2000/01/rdf-schema#label>  
< http://www.skeim.org#division>  
<http://www.skeim.org#district>  
< http://www.skeim.org#thana>  
<http://www.skeim.org#postOffice>  
< http://www.w3.org/ns/org#site>

## Object

<http://www.skeim.org#Secondary>  
<http://www.skeim.org#School>  
"About High School Institutions@en".  
<http://www.skeim.org#002>.  
"http://www.skeim.org#0021".  
"105770@en".  
"Comilla Zilla School@en".  
"Chittagong @en".  
"Comilla@en".  
"Adarsha Sadar @en".  
"Comilla@en".  
"Kandirpar@en".

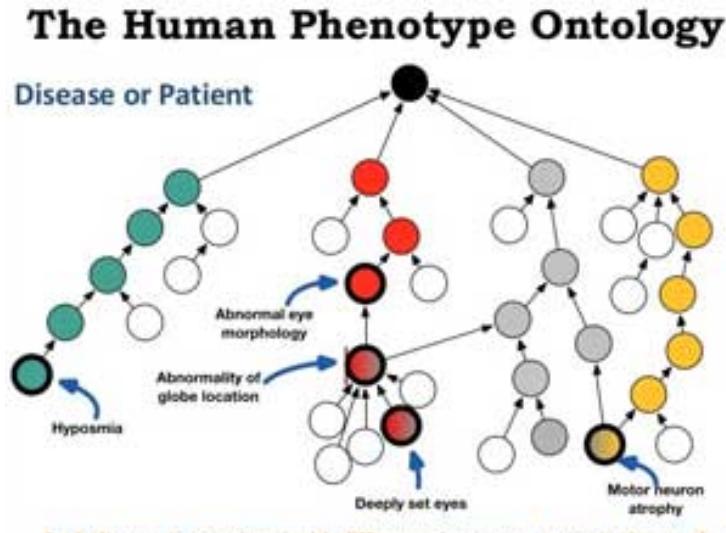
# ONTOLOGY

.ttl format

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
@prefix ex: <http://example.org/stuff/1.0/> .  
  
<http://www.w3.org/TR/rdf-syntax-grammar>  
    dc:title "RDF/XML Syntax Specification (Revised)" ;  
    ex:editor [  
        ex:fullname "Dave Beckett";  
        ex:homePage <http://purl.org/net/dajobe/>  
    ] .
```

# HUMAN PHENOTYPE ONTOLOGY (HPO)

- A standardized vocabulary of phenotypic abnormalities encountered in human disease
- The HPO has been used in many different tools and algorithms for clinical diagnostics, phenotype-driven genomic diagnostics, translational bioinformatics, and more
- The HPO is currently being developed using the medical literature, Orphanet, DECIIPHER, and OMIM.
- Over 13,000 terms in total, maintained in a hierarchical structure



# HUMAN PHENOTYPE ONTOLOGY (HPO)

The screenshot shows the HPO homepage with a dark teal header bar. On the left is the HPO logo (a stylized 'P' icon inside a circle). To its right are three dropdown menus: 'Tools ▾', 'Downloads ▾', and 'Help ▾'. Below the header is a search bar with a dropdown menu set to 'All' and a placeholder text 'Search for phenotypes, diseases, genes...'. Below the search bar, there is an example search query: 'e.g. Arachnodactyly | Marfan syndrome | FBN1'.

human  
phenotype  
ontology

Tools ▾

Downloads ▾

Help ▾

All ▾

Search for phenotypes, diseases, genes...

e.g. Arachnodactyly | Marfan syndrome | FBN1

# HUMAN PHENOTYPE ONTOLOGY (HPO)

human phenotype ontology

Tools ▾ Downloads ▾ Help ▾ All ▾ Search for phenotypes, diseases, genes...

Hierarchy

- Abnormal atrial septum morphology 8
- Abnormal cardiac atrium morphology 28
- Atrial septal defect 6
  - Primum atrial septal defect
  - Patent foramen ovale
  - Sinus venosus atrial septal defect
  - Swiss cheese atrial septal defect
  - Unroofed coronary sinus
  - Secundum atrial septal defect

Atrial septal defect HP:0001631 ⓘ

*Atrial septal defect (ASD) is a congenital abnormality of the interatrial septum that enables blood flow between the left and right atria via the interatrial septum.*

**Synonyms:** Defect in the atrial septum, Atrial septal defect, ASD, Hole in heart wall separating two upper heart chambers, An opening in the wall separating the top two chambers of the heart, Atria septal defect

**Xrefs:** SNOMEDCT\_US:253366007, ICD-10:Q21.1, MSH:D006344, Fyler:2050, SNOMEDCT\_US:405752007, SNOMEDCT\_US:70142008, UMLS:C0018817

**Export Associations**

Disease Associations Gene Associations

Disease Id	Disease Name	Associated Genes
ORPHA:261190	15q14 Microdeletion Syndrome	MEIS2 [4212]
ORPHA:261236	16p13.11 Microdeletion Syndrome	
ORPHA:261243	16p13.11 Microduplication Syndrome	
ORPHA:261272	17q12 Microduplication Syndrome	TBX4 [9496]
ORPHA:261279	17q23.1q23.2 Microdeletion Syndrome	

# HUMAN PHENOTYPE ONTOLOGY (HPO)

## Download Ontology

### hp.obo

The [OBO flat file format](#) is a simple way of representing an ontology that models represent a subset of the concepts in the OWL description logic language. The OBO format was designed to maximize human readability and has been widely used in the field of bio-ontologies and was probably one of the many factors responsible for the success of the [Gene Ontology](#). The HPO file in OBO format can be downloaded here:

- <http://purl.obolibrary.org/obo/hp.obo>

### hp.owl

OWL, which stands for [Web Ontology Language](#), is an ontology language for the semantic web. The OWL version of the HPO (which is the primary version) contains some features that the OBO version does not, including the logical definitions of HPO classes. Consider for instance the following definition of the HPO term for [Astigmatism](#).

```
'has part' some
  ('asymmetrically curved'
   and ('inheres in' some cornea)
   and ('has modifier' some abnormal))
```

The logical makes use of terms from the [PATO](#) Phenotype And Trait Ontology as well as the [UBERON](#) cross-species anatomy ontology in order to define [Astigmatism](#) as an abnormal asymmetric curvature of the cornea. A number of HPO-based algorithms make use of these definitions to link to definitions of mouse phenotypes at the [Mouse Genome Informatics](#) resource, for quality control, and other purposes. For many of the HPO algorithms that are primarily intended for clinical phenotype matching, these features are not relevant and either the OBO or the OWL format are suitable. The HPO file in OWL format can be downloaded here:

- <http://purl.obolibrary.org/obo/hp.owl>

## EXPERIMENTS

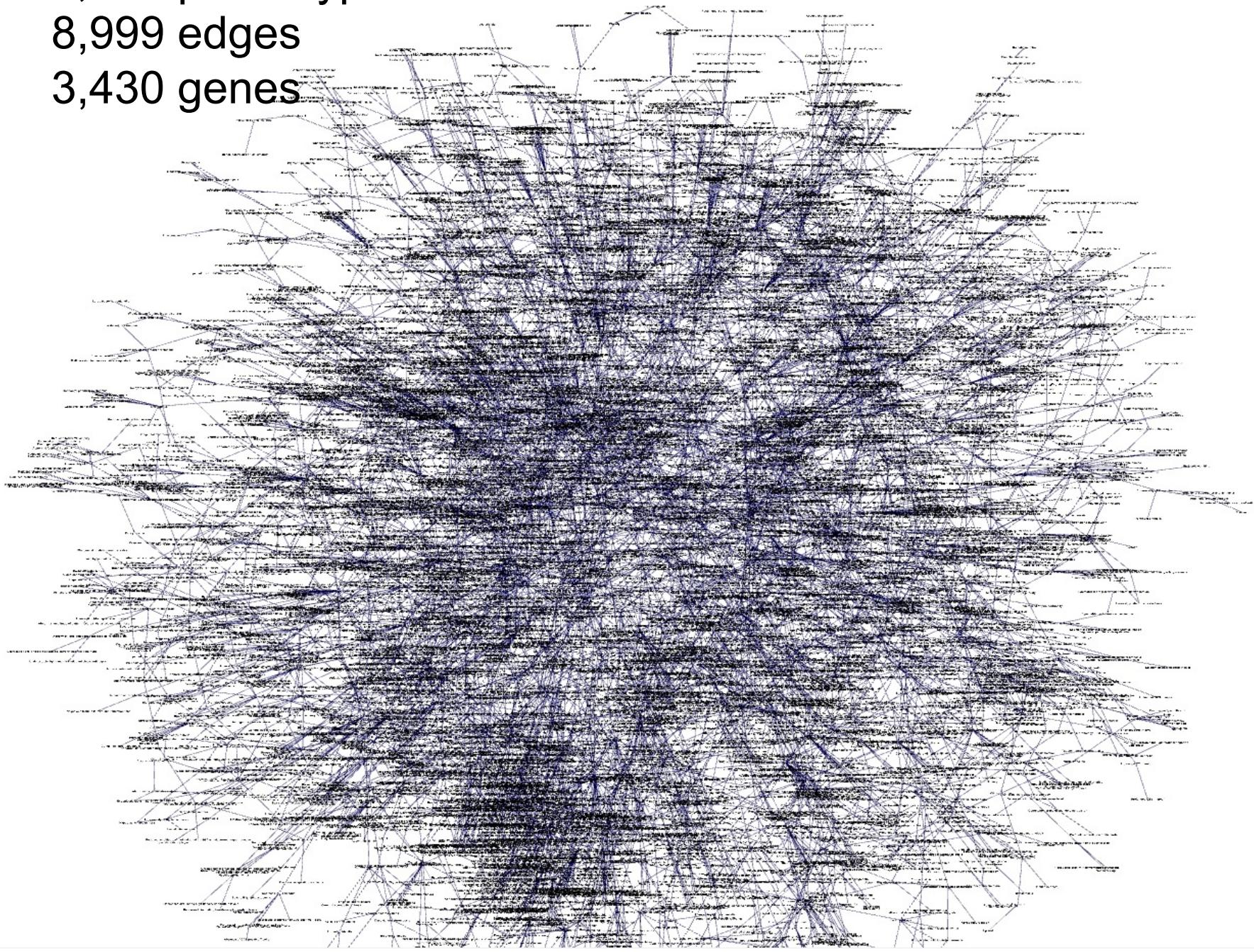
### Human Phenotype Ontology

- ❑ Annotate 7,280 phenotypes with gene information (entrez ID)
- ❑ Sub-graph, nodes with annotated gene information
- ❑ Prune isolated nodes to make all nodes connected

7,258 phenotype nodes

8,999 edges

3,430 genes



# RESULTS

## Link prediction

- Prepare positive examples:
  - Randomly used 60%, 10%, and 30% of entire edges for training, validation, and testing purposes.
- Prepare negative examples:
  - Randomly used 60%, 10%, and 30% of entire edges for training, validation, and testing purpose

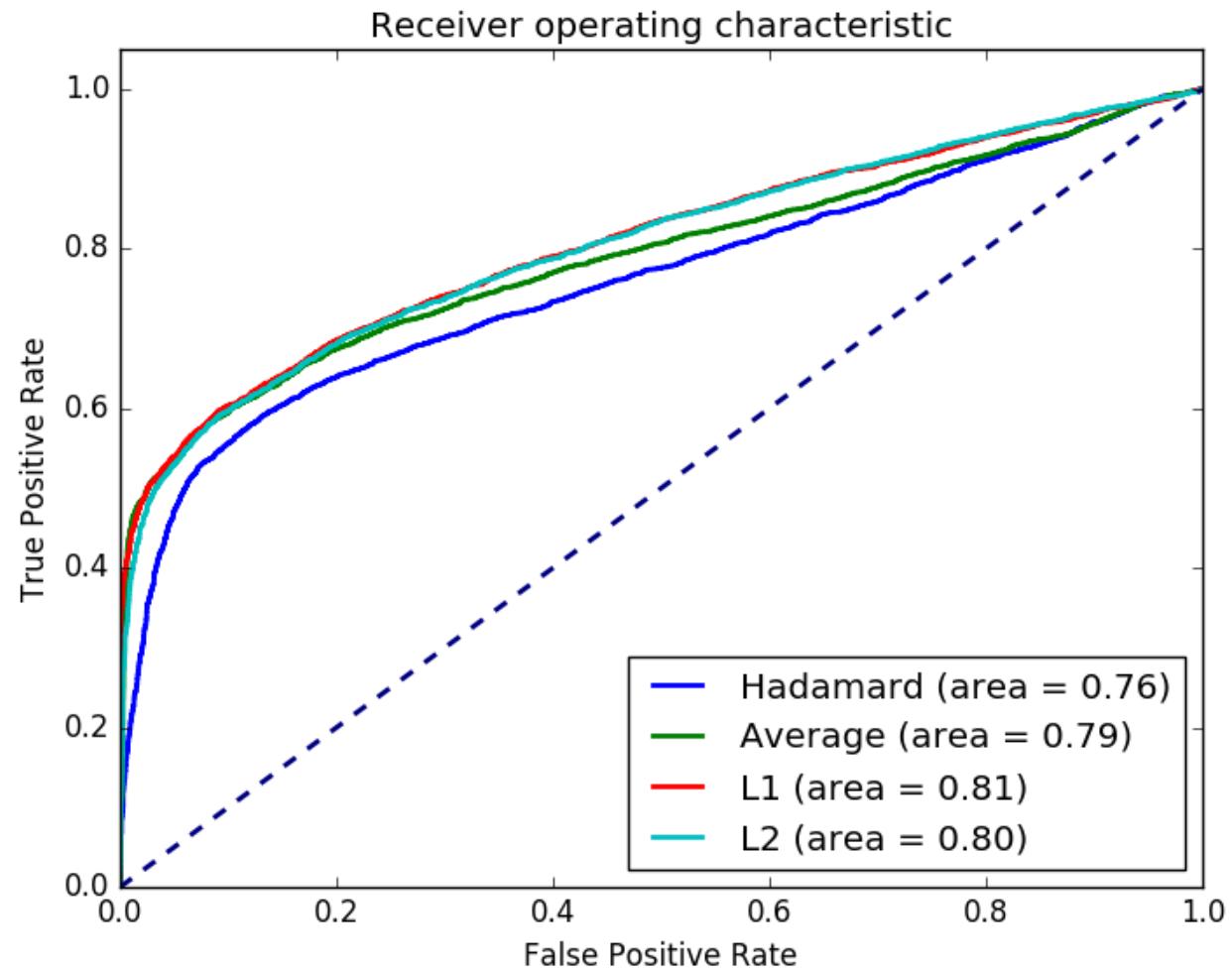
# RESULTS

- Generate edge embedding based on node embedding
- Given nodes  $u$  and  $v$ ,  $f(u)$  and  $f(v)$  denotes their feature representations
  - LogisticRegression(edge\_emb, edge\_label)

Operator	Definition
Average	$\frac{f(u) + f(v)}{2}$
Hadamard	$f(u) * f(v)$
Weighted-L1	$ f(u) - f(v) $
Weighted-L2	$ f(u) - f(v) ^2$

# RESULTS

$p=1, q=0.05$



# RESULTS

Operator	Precision	Recall	F-measure
Average	0.76	0.75	0.74
Hadamard	0.74	0.73	0.72
Weighted-L1	0.76	0.75	0.75
Weighted-L2	0.78	0.75	0.74

# RESULTS

- Comparing with baseline scoring methods:

Scores	Definition
Jaccard's Coefficient	$\frac{N(u) \cap N(v)}{N(u) \cup N(v)}$
Adamic-Adar Score	$\sum_{t \in N(u) \cap N(v)} \frac{1}{\log  N(t) }$

$N(u)$ : direct neighbor of node u

$N(v)$ : direct neighbor of node v

$N(t)$ : direct neighbor of node t

# RESULTS

Methods	AUROC	Average Precision
Node2vec (L1)	0.81	0.85
Adamic-Adar	0.63	0.67
Jaccard	0.5	0.46

# RESULTS

■ (7258, 128)

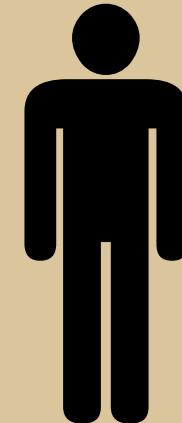
p	-0.240330	0.058522	0.153959	-0.002036	-0.064240	0.235507	-0.186960	-0.338655	-0.119517	0.175563	0.391702	-0.066000	-0.124363	-0.063367	0.
3098	-0.272518	0.076228	0.161163	0.021788	-0.076025	0.278918	-0.195849	-0.394783	-0.128165	0.200492	0.445066	-0.061802	-0.165236	-0.087852	
1369	-0.266067	0.057190	0.168280	0.006649	-0.070766	0.245432	-0.184228	-0.374658	-0.125654	0.183740	0.413518	-0.069982	-0.153400	-0.066426	
1033	-0.259970	0.057397	0.163549	-0.002428	-0.077161	0.241342	-0.191172	-0.370289	-0.126249	0.182140	0.415996	-0.066236	-0.137282	-0.070804	
989	-0.250128	0.055444	0.166569	0.014121	-0.078800	0.240832	-0.191375	-0.365271	-0.127956	0.181881	0.414304	-0.058934	-0.137044	-0.058468	0
2081	-0.246176	0.047528	0.161245	0.009027	-0.060807	0.230705	-0.185488	-0.356683	-0.124391	0.176036	0.388074	-0.061950	-0.130584	-0.059033	
830	-0.224584	0.049370	0.152805	0.004360	-0.063052	0.225471	-0.169435	-0.322209	-0.111722	0.172882	0.371441	-0.056873	-0.126904	-0.056810	0
1872	-0.223858	0.028879	0.134200	0.009756	-0.053364	0.234226	-0.187693	-0.329627	-0.123021	0.177479	0.365998	-0.049611	-0.123004	-0.062653	
327	-0.226755	0.054007	0.141038	0.012875	-0.058397	0.198548	-0.155008	-0.317698	-0.105430	0.162329	0.344005	-0.057285	-0.122505	-0.063517	0
92	-0.225669	0.050669	0.132185	0.000865	-0.052302	0.219648	-0.157705	-0.317097	-0.104145	0.161302	0.355088	-0.054497	-0.118933	-0.052913	0
496	-0.211478	0.042819	0.136933	0.004403	-0.050679	0.208560	-0.158854	-0.306980	-0.100467	0.150951	0.343334	-0.057042	-0.113162	-0.047319	0
1759	-0.215291	0.050190	0.135453	0.005083	-0.062060	0.205241	-0.160292	-0.308093	-0.111927	0.162504	0.351751	-0.060556	-0.115892	-0.058769	
2150	-0.224865	0.041183	0.145084	0.005946	-0.057137	0.219195	-0.160681	-0.338186	-0.102507	0.167785	0.370981	-0.058028	-0.129208	-0.059591	
1461	-0.222304	0.053975	0.132621	0.014579	-0.057800	0.215194	-0.163410	-0.313907	-0.101367	0.158752	0.352470	-0.061913	-0.116908	-0.056670	
1026	-0.234396	0.045661	0.148524	0.007663	-0.052798	0.224044	-0.171789	-0.334390	-0.112462	0.170467	0.378480	-0.064150	-0.127602	-0.059090	
3039	-0.217818	0.050422	0.134099	0.012842	-0.066442	0.205347	-0.155844	-0.310195	-0.110612	0.159405	0.345328	-0.054990	-0.113828	-0.047740	
1115	-0.223488	0.052431	0.148303	0.002554	-0.063774	0.206668	-0.162822	-0.321313	-0.111940	0.155479	0.353809	-0.053989	-0.120905	-0.055936	
190	-0.237800	0.055537	0.145510	0.002572	-0.054028	0.230477	-0.169945	-0.331388	-0.111711	0.167688	0.379870	-0.063387	-0.129694	-0.057227	0
677	-0.214353	0.052189	0.134712	0.010339	-0.052341	0.208794	-0.155401	-0.303248	-0.106807	0.160090	0.344848	-0.055918	-0.114665	-0.057178	0
987	-0.197369	0.045039	0.125478	0.009690	-0.046102	0.184292	-0.136240	-0.272959	-0.097918	0.140693	0.304756	-0.043673	-0.105947	-0.043855	0
323	-0.215443	0.051718	0.145657	0.010977	-0.059549	0.210862	-0.160736	-0.313210	-0.103026	0.156158	0.346836	-0.060588	-0.122748	-0.051498	0
259	-0.212490	0.049053	0.139140	0.001940	-0.061963	0.199641	-0.158844	-0.317382	-0.105923	0.149482	0.346885	-0.060574	-0.123020	-0.050471	0
880	-0.214039	0.048417	0.130425	0.009720	-0.056724	0.203819	-0.158062	-0.312044	-0.109439	0.150580	0.346755	-0.062049	-0.113490	-0.045735	0
4307	-0.213021	0.045414	0.135402	0.010044	-0.059401	0.206574	-0.154959	-0.313658	-0.106180	0.154264	0.345726	-0.061927	-0.113313	-0.052572	
1008	-0.207112	0.050631	0.133816	0.002981	-0.050075	0.196813	-0.148940	-0.285359	-0.104198	0.141817	0.330695	-0.046988	-0.112498	-0.048617	
996	-0.215358	0.041260	0.132543	0.012829	-0.050389	0.203927	-0.155883	-0.305718	-0.108328	0.155589	0.339067	-0.054132	-0.116024	-0.046225	0
1224	-0.217495	0.048022	0.140306	0.011062	-0.056477	0.209850	-0.168410	-0.310982	-0.104730	0.159963	0.355919	-0.061271	-0.119244	-0.056927	
2002	-0.216393	0.052970	0.141161	0.004912	-0.056215	0.209660	-0.162081	-0.310780	-0.102078	0.147595	0.344340	-0.059995	-0.109240	-0.057045	
332	-0.209703	0.039071	0.131613	0.009140	-0.048731	0.206831	-0.162645	-0.304422	-0.100736	0.156964	0.345549	-0.056915	-0.110919	-0.048142	0
2955	-0.205782	0.045253	0.129652	0.010800	-0.048560	0.199658	-0.152954	-0.287979	-0.099875	0.152010	0.323420	-0.048287	-0.113029	-0.050371	
804	-0.203928	0.039433	0.129529	0.012169	-0.051414	0.198383	-0.144608	-0.287990	-0.096271	0.146931	0.320889	-0.050625	-0.108652	-0.047109	0
1020	-0.205594	0.051070	0.136965	0.000829	-0.056967	0.198051	-0.151474	-0.300690	-0.108494	0.142082	0.335021	-0.052939	-0.108617	-0.047203	
1361	-0.199030	0.036256	0.135451	0.010022	-0.056347	0.200584	-0.157369	-0.295043	-0.100084	0.156233	0.331045	-0.050897	-0.110129	-0.055913	
1765	-0.207103	0.041020	0.132215	0.009502	-0.056525	0.196352	-0.156987	-0.291991	-0.095114	0.152518	0.336234	-0.051432	-0.111460	-0.053085	
1625	-0.205050	0.041435	0.137897	0.005170	-0.059694	0.199951	-0.160806	-0.297397	-0.099223	0.157134	0.337223	-0.057472	-0.116203	-0.056040	
400	-0.200612	0.045511	0.132215	0.003550	-0.057074	0.199915	-0.140557	-0.290111	-0.092000	0.144271	0.325762	-0.056004	-0.106010	-0.050371	

# RESULTS

- Idiopathic pulmonary fibrosis (IPF) is a type of lung diseases that results in fibrosis of the lungs for an unknown reason
- It is estimated that up to 3 million worldwide have IPF

Prevalence of IPF is estimated as

**$\frac{14-42.7}{100,000}$**



**Persons**

IPF-related Phenotype	Similar Phenotypes	Similarity Score	IPF-related Phenotype	Similar Phenotypes	Similarity Score
cough	Neonatal breathing dysregulation	0.93	gastroesophageal reflux (GERD)	Prominent interphalangeal joints	0.56
	Abnormal breath sound	0.92		Abnormality of body height	0.53
	Snoring	0.92		Depigmented fundus	0.52
	Abnormal blood gas level	0.92		Wrist swelling	0.52
	Restrictive ventilatory defect	0.91		Facial wrinkling	0.51
IPF-related Phenotype	Similar Phenotypes	Similarity Score	IPF-related Phenotype	Similar Phenotypes	Similarity Score
pulmonary fibrosis	Atelectasis	0.9	exertional dyspnea	Dyspnea	0.98
	Pulmonary hypoplasia	0.89		Respiratory distress	0.94
	Abnormal lung morphology	0.89		Hypoplastic distal radial epiphyses	0.47
	Alveolar proteinosis	0.88		Spastic dysarthria	0.46
	Unilateral primary pulmonary dysgenesis	0.88		Nonketotic hyperglycinemia	0.45
IPF-related Phenotype	Similar Phenotypes	Similarity Score	IPF-related Phenotype	Similar Phenotypes	Similarity Score
bronchiectasis	Abnormal bronchus morphology	0.96	pulmonary insufficiency	Abnormal pulmonary valve physiology	0.97
	Bronchomalacia	0.93		Pulmonic stenosis	0.93
	Recurrent bronchitis	0.89		Renal steatosis	0.5
	Recurrent upper respiratory tract infections	0.79		Calf muscle hypertrophy	0.48
	Recurrent lower respiratory tract infections	0.77		Prominent coccyx	0.47

# **IMPLEMENTATION**

# TOOLS

- Python 3.6
- Jupyter Notebook
- Anaconda 3.7

1. Install Anaconda 3.7 version
2. Create a running env with necessary libraries

```
conda create --name NEEEnv python=3.6 networkx pandas scipy scikit-learn gensim matplotlib
```

3. install ontospy

```
pip install ontospy
```

4. Activate your running env

```
conda activate NEEEnv
```

5. Run the Jupyter Notebook

```
jupyter notebook
```

Notice:

For Mac, if you face some troubles opening NEEEnv in you jupyter notebook, you explicitly install the new kernel using the following command in terminal

```
ipython kernel install --user --name=NEEEnv
```

After this, the system will create a new kernel directory under "/Users/you\_account/Library/Jupyter/kernels/NEEEnv. You can then open notebook and select NEEEnv as the environment

# SOURCES

- Reference implementation of Ontospy. <http://lambdamusic.github.io/Ontospy/>
- Reference implementation of node2vec. Author: Aditya Grover  
For more details, refer to the paper: node2vec: Scalable Feature Learning for Networks Aditya Grover and Jure Leskovec Knowledge Discovery and Data Mining (KDD), 2016. <https://github.com/aditya-grover/node2vec>
- Reference implementation of link prediction task using fb data: Lucas Hu <https://github.com/lucashu1>

# STEPS

- Ontology exploration
- Graph generation
- Node embeddings generation
- t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization
- Similarity measurement
- Edge embeddings generation
- Link prediction evaluation

# ONTOLOGY EXPLORATION

OntologyPractice.ipynb

```
import ontospy
```

```
model = ontospy.Ontospy("http://xmlns.com/foaf/0.1/", verbose=True)
```

```
Reading: <http://xmlns.com/foaf/0.1/>
.. trying rdf serialization: <xml>
..... success!
-----
Loaded 631 triples.
-----
RDF sources loaded successfully: 1 of 1.
..... 'http://xmlns.com/foaf/0.1/'
```

```
-----
Scanning entities...
-----
Ontologies.....: 1
Classes.....: 15
Properties.....: 67
..annotation....: 7
..datatype.....: 26
..object.....: 34
Concepts (SKOS)....: 0
Shapes (SHACL)....: 0
```

# ONTOLOGY EXPLORATION

```
model.all_classes
```

```
[<Class *http://xmlns.com/foaf/0.1/Agent*>,
 <Class *http://xmlns.com/foaf/0.1/Document*>,
 <Class *http://xmlns.com/foaf/0.1/Group*>,
 <Class *http://xmlns.com/foaf/0.1/Image*>,
 <Class *http://xmlns.com/foaf/0.1/LabelProperty*>,
 <Class *http://xmlns.com/foaf/0.1/OnlineAccount*>,
 <Class *http://xmlns.com/foaf/0.1/OnlineChatAccount*>,
 <Class *http://xmlns.com/foaf/0.1/OnlineEcommerceAccount*>,
 <Class *http://xmlns.com/foaf/0.1/OnlineGamingAccount*>,
 <Class *http://xmlns.com/foaf/0.1/Organization*>,
 <Class *http://xmlns.com/foaf/0.1/Person*>,
 <Class *http://xmlns.com/foaf/0.1/PersonalProfileDocument*>,
 <Class *http://xmlns.com/foaf/0.1/Project*>,
 <Class *http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing*>,
 <Class *http://www.w3.org/2004/02/skos/core#Concept*>]
```

# ONTOLOGY EXPLORATION

```
model.all_properties_object
```

```
[<Property *http://xmlns.com/foaf/0.1/account*>,
 <Property *http://xmlns.com/foaf/0.1/accountServiceHomepage*>,
 <Property *http://xmlns.com/foaf/0.1/based_near*>,
 <Property *http://xmlns.com/foaf/0.1/currentProject*>,
 <Property *http://xmlns.com/foaf/0.1/depiction*>,
 <Property *http://xmlns.com/foaf/0.1/depicts*>,
 <Property *http://xmlns.com/foaf/0.1/focus*>,
 <Property *http://xmlns.com/foaf/0.1/fundedBy*>,
 <Property *http://xmlns.com/foaf/0.1/holdsAccount*>,
 <Property *http://xmlns.com/foaf/0.1/homepage*>,
 <Property *http://xmlns.com/foaf/0.1/img*>,
 <Property *http://xmlns.com/foaf/0.1/interest*>,
 <Property *http://xmlns.com/foaf/0.1/isPrimaryTopicOf*>,
 <Property *http://xmlns.com/foaf/0.1/knows*>,
 <Property *http://xmlns.com/foaf/0.1/logo*>,
 <Property *http://xmlns.com/foaf/0.1/made*>,
 <Property *http://xmlns.com/foaf/0.1/maker*>,
```

# ONTOLOGY EXPLORATION

```
model.printClassTree()
```

```
foaf:Agent
----foaf:Group
----foaf:Organization
----foaf:Person
foaf:Document
----foaf:Image
----foaf:PersonalProfileDocument
foaf:LabelProperty
foaf:OnlineAccount
----foaf:OnlineChatAccount
----foaf:OnlineEcommerceAccount
----foaf:OnlineGamingAccount
foaf:Project
http://www.w3.org/2003/01/geo/wgs84\_pos#SpatialThing
----foaf:Person
skos:Concept
```

# ONTOLOGY EXPLORATION

```
model.toplayer_classes
```

```
[<Class *http://xmlns.com/foaf/0.1/Agent*>,
 <Class *http://xmlns.com/foaf/0.1/Document*>,
 <Class *http://xmlns.com/foaf/0.1/LabelProperty*>,
 <Class *http://xmlns.com/foaf/0.1/OnlineAccount*>,
 <Class *http://xmlns.com/foaf/0.1/Project*>,
 <Class *http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing*>,
 <Class *http://www.w3.org/2004/02/skos/core#Concept*>]
```

# ONTOLOGY EXPLORATION

```
model.get_class('document')
```

```
[<Class *http://xmlns.com/foaf/0.1/Document*>,
 <Class *http://xmlns.com/foaf/0.1/PersonalProfileDocument*>]
```

```
c1 =_[1]
```

```
print(c1.rdf_source())
```

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
```

# ONTOLOGY EXPLORATION

```
c1.parents()
```

```
[<Class *http://xmlns.com/foaf/0.1/Document*>]
```

```
c1.children()
```

```
[]
```

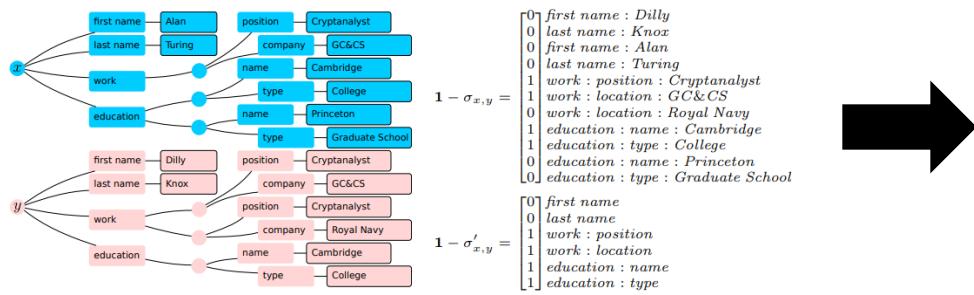
# DATASET INFORMATION

## Facebook anonymous user data

- This dataset consists of 'circles' (or 'friends lists') from Facebook. The dataset includes node features (profiles), circles, and ego networks.
- A small network with annotated features will be demonstrated

## DATASET INFORMATION

## ■ Feature matrix



- Edges data, represented by source-target pair

236 186

122 285

24 346

271 304

176 9

<http://i.stanford.edu/~julian/pdfs/nips2012.pdf>

<http://snap.stanford.edu/data/egonets-Facebook.html>

# GRAPH GENERATION

## - READ INPUT DATA

graphGeneration.ipynb

```
import networkx as nx
import pandas as pd
import numpy as np
import pickle
```

```
edges_dir = ' testData/0.edges'
feats_dir = ' testData/0.allfeat'
```

```
# Read edge list
g = nx.read_edgelist(edges_dir,nodetype=int)
```

# GRAPH GENERATION

## - READ INPUT DATA

```
# Read feature list
df = pd.read_csv(feats_dir, sep=' ', header=None, index_col=0)
```

```
g.node[node_index]['features'] = features_series.values
```

```
# Make sure the graph is connected
nx.is_connected(g)
```

# GRAPH GENERATION

## -DUMP TO PICKLE FILE

```
# Save adj, features in pickle file
network_tuple = (adj, features)
```

```
with open("test-adj-feat.pkl", "wb") as f:
    pickle.dump(network_tuple, f)
```

# GRAPH GENERATION

## ■ Some upgrades from Python 2 to Python 3

1.

```
f = open(edges_dir)
g = nx.read_edgelist(f, nodetype=int)    ➔   g = nx.read_edgelist(edges_dir, nodetype=int)
```

2.

```
features_series.as_matrix() ➔ features_series.values
```

3.

```
pd.read_table ➔ pd.read_csv
```

## GRAPH LOADING

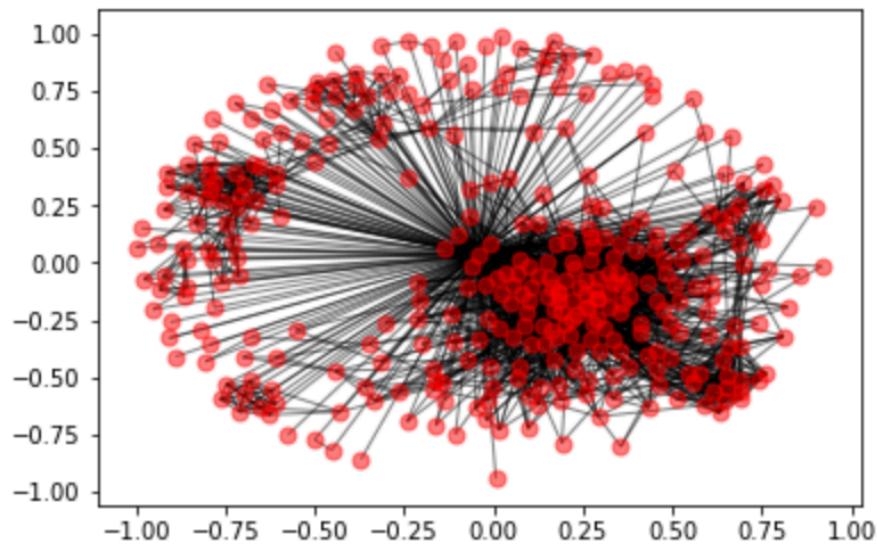
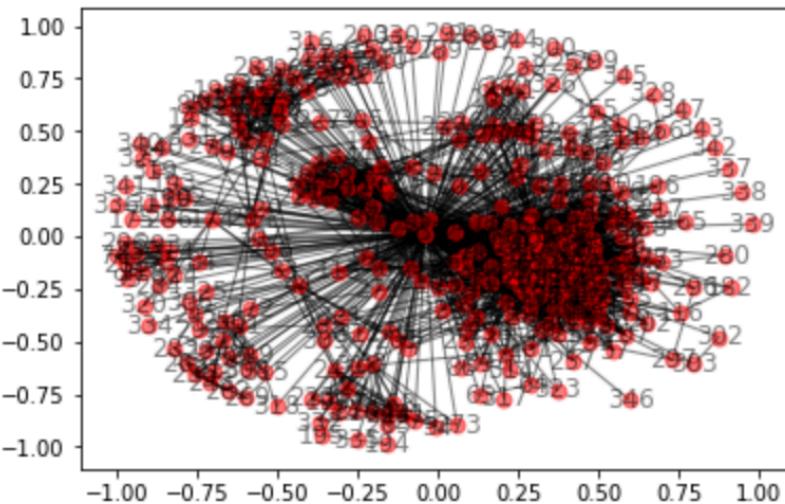
```
# Load pickled (adj, feat) tuple
network_dir = 'test-adj-feat.pkl'
with open(network_dir, 'rb') as f:
    adj, features = pickle.load(f)

g = nx.Graph(adj)
```

# GRAPH VISUALIZATION

```
# visualize network
# plt.figure(figsize=(16,16)) # set up canvas size
nx.draw_networkx(g, alpha=0.5, with_labels=True, node_size=50, node_color='r')
plt.show()
```

# GRAPH VISUALIZATION



# NODE EMBEDDINGS GENERATION

## -TRAIN AND TEST SPLIT

Node2vecTest.ipynb

```
adj_sparse = nx.to_scipy_sparse_matrix(g)

adj_train, train_edges, train_edges_false, val_edges, val_edges_false, \
test_edges, test_edges_false = mask_test_edges(adj_sparse, test_frac=.3, val_frac=.1)

g_train = nx.from_scipy_sparse_matrix(adj_train)
```

# NODE EMBEDDINGS GENERATION -HYPERPARAMETERS

```
P = 1 # Return hyperparameter
Q = 0.25 # In-out hyperparameter
WINDOW_SIZE = 10 # Context size for optimization
NUM_WALKS = 10 # Number of walks per source
WALK_LENGTH = 80 # Length of walk per source
DIMENSIONS = 128 # Embedding dimension
DIRECTED = False # Graph directed/undirected
WORKERS = 8 # Num. parallel workers
ITER = 1 # SGD epochs
```

# NODE EMBEDDINGS GENERATION

## -RANDOM WALK

```
import node2vec
```

```
# Preprocessing, generate walks
g_n2v = node2vec.Graph(g_train, DIRECTED, P, Q)
g_n2v.preprocess_transition_probs()
walks = g_n2v.simulate_walks(NUM_WALKS, WALK_LENGTH)
walks = [list(map(str, walk)) for walk in walks]
```

# NODE EMBEDDINGS GENERATION

## -RANDOM WALK

```
# Preprocessing, generate walks
g_n2v = node2vec.Graph(g_train, DIRECTED, P, Q)
g_n2v.preprocess_transition_probs()
walks = g_n2v.simulate_walks(NUM_WALKS, WALK_LENGTH)
walks = [list(map(str, walk)) for walk in walks]
```

# NODE EMBEDDINGS GENERATION -RANDOM WALK

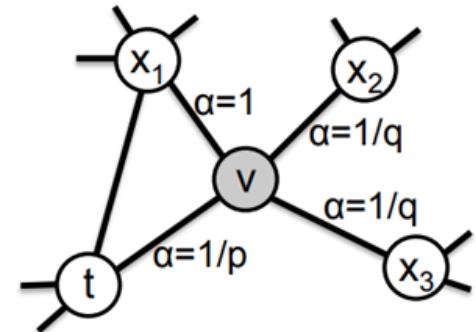
## Node2vec.py

### ■ Preprocessing\_transition\_probs:

```
for dst_nbr in sorted(G.neighbors(dst)):
    if dst_nbr == src:
        unnormalized_probs.append(G[dst][dst_nbr]['weight']/p)
    elif G.has_edge(dst_nbr, src):
        unnormalized_probs.append(G[dst][dst_nbr]['weight'])
    else:
        unnormalized_probs.append(G[dst][dst_nbr]['weight']/q)
```

```
norm_const = sum(unnormalized_probs)
normalized_probs = [float(u_prob)/norm_const for u_prob in unnormalized_probs]
```

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$



# NODE EMBEDDINGS GENERATION

## -RANDOM WALK

```
# Preprocessing, generate walks
g_n2v = node2vec.Graph(g_train, DIRECTED, P, Q)
g_n2v.preprocess_transition_probs()
walks = g_n2v.simulate_walks(NUM_WALKS, WALK_LENGTH)
walks = [list(map(str, walk)) for walk in walks]
```

# NODE EMBEDDINGS GENERATION -WORD2VEC

```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
```

```
# Train skip-gram model
model = Word2Vec(walks, size=DIMENSIONS, window=WINDOW_SIZE,
                  min_count=0, sg=1, workers=WORKERS, iter=ITER)

# Store embeddings mapping
emb_mappings = model.wv

model.wv.save_word2vec_format('fb_test.emd')
```

# NODE EMBEDDINGS VISUALIZATION

```
# Open embedding file
with open('fb_test.emd') as f:
    next(f)
    for line in f:
        splits = line.strip().split()
        label = splits[0]
        vec = [float(v) for v in splits[1:]]

        x.append(vec)
        y.append(label)
```

# NODE EMBEDDINGS VISUALIZATION

```
from sklearn.manifold import TSNE

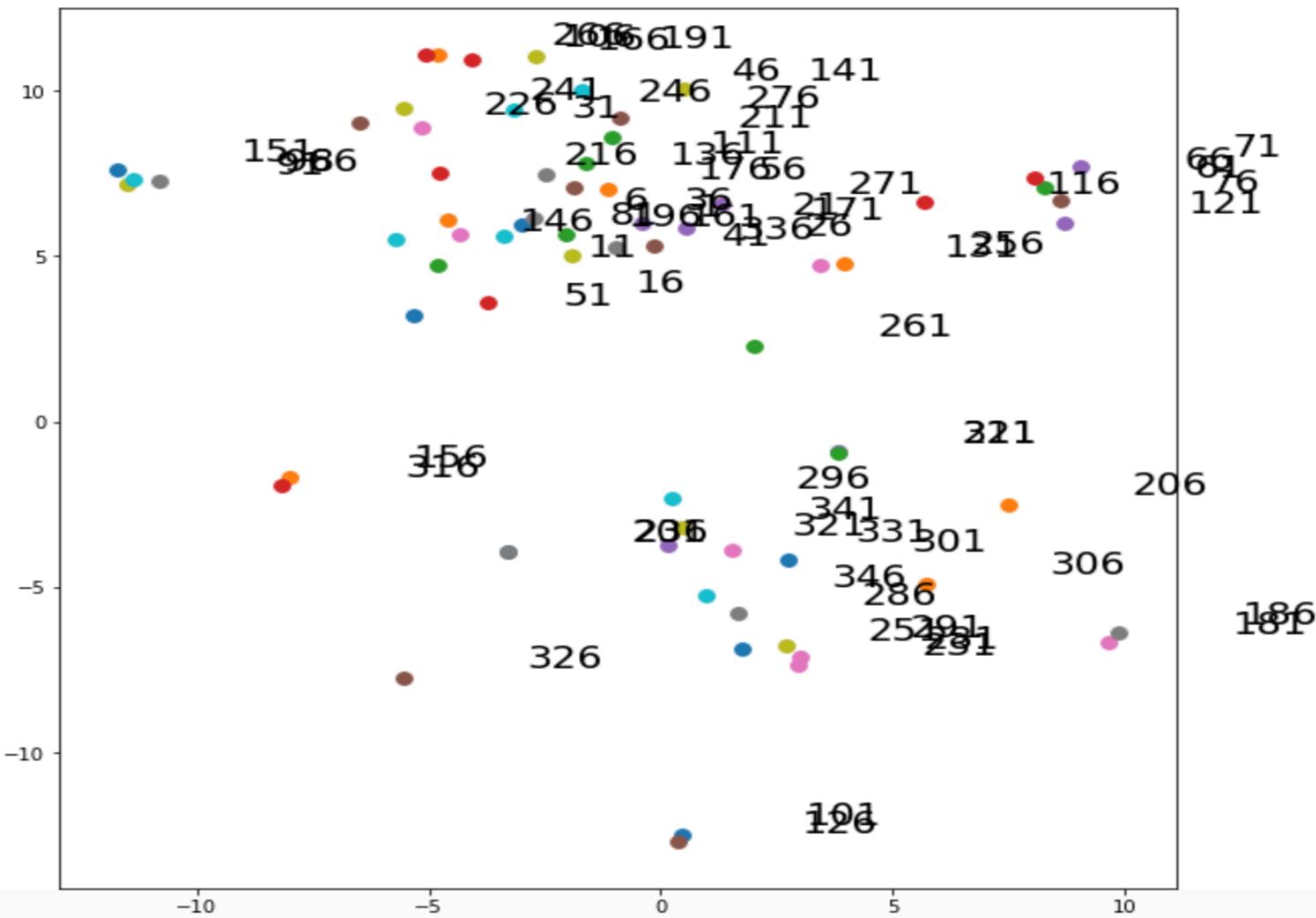
tsne = TSNE(n_components=2,perplexity=50.0)
X_tsne = tsne.fit_transform(X)

plt.figure(figsize=(10,10))

for i in range(1, len(X), 5):
    plt.scatter(X_tsne[i, 0], X_tsne[i, 1], s=60)
    plt.annotate(str(i),
                 xy=(X_tsne[i, 0], X_tsne[i, 1]),
                 xytext=(100, 2),
                 textcoords='offset points',
                 ha='right',
                 va='bottom', fontsize=20)

plt.show()
```

# NODE EMBEDDINGS VISUALIZATION



# SIMILARITY MEASUREMENT

```
s1 = emb_mappings.similarity('156', '316')

print(s1)
```

```
s2 = emb_mappings.most_similar('156')

print(s2)
```

# GENERATE EDGE EMBEDDINGS

```
func_list = [lambda x1, x2: np.multiply(x1, x2),  
             lambda x1, x2: (x1 + x2) / 2,  
             lambda x1, x2: np.absolute(x1 - x2),  
             lambda x1, x2: np.square(x1 - x2),  
             ]
```

Operator	Definition
Average	$\frac{f(u) + f(v)}{2}$
Hadamard	$f(u) * f(v)$
Weighted-L1	$ f(u) - f(v) $
Weighted-L2	$ f(u) - f(v) ^2$

# EVALUATE LINK PREDICTION PERFORMANCE

```
train_edge_embs = np.concatenate([pos_train_edge_embs[i], neg_train_edge_embs[i]]))  
train_edge_labels = np.concatenate([np.ones(len(train_edges)), np.zeros(len(train_edges_false))])  
  
test_edge_embs = np.concatenate([pos_test_edge_embs[i], neg_test_edge_embs[i]]))  
test_edge_labels = np.concatenate([np.ones(len(test_edges)), np.zeros(len(test_edges_false))])  
  
edge_classifier = LogisticRegression(random_state=0,solver='lbfgs')  
edge_classifier.fit(train_edge_embs, train_edge_labels)
```

# EVALUATE LINK PREDICTION PERFORMANCE

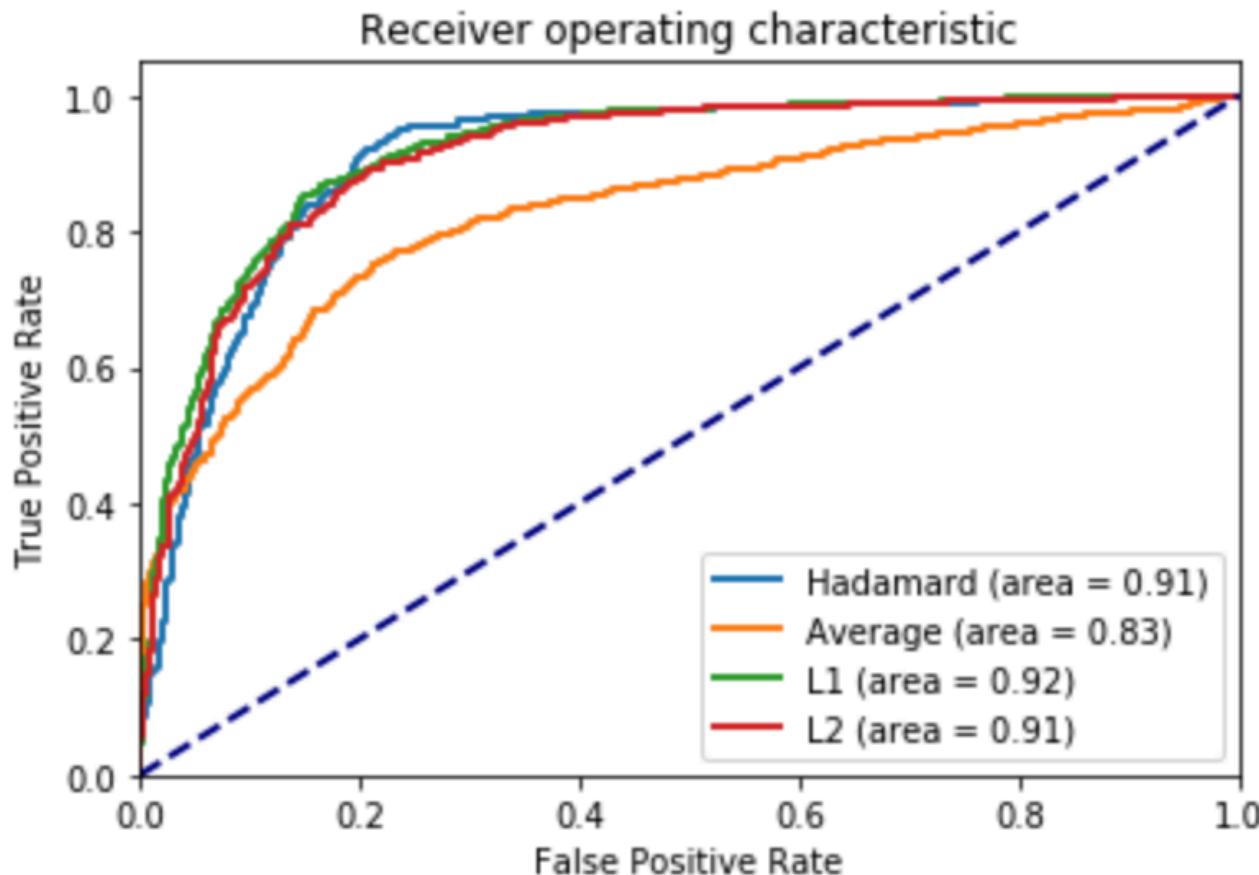
```
test_preds = edge_classifier.predict_proba(test_edge_embs)[:, 1]
test_roc = roc_auc_score(test_edge_labels, test_preds)
test_ap = average_precision_score(test_edge_labels, test_preds)
```

# EVALUATE LINK PREDICTION PERFORMANCE

```
fpr = dict()
tpr = dict()
roc_auc = dict()
fpr, tpr, _ = roc_curve(test_edge_labels, test_preds)
roc_auc = auc(fpr, tpr)
lw = 2

plt.plot(fpr, tpr,
          lw=lw, label='%s (area = %0.2f)' % (labels[i], roc_auc))
```

# EVALUATE LINK PREDICTION PERFORMANCE



**THANK YOU!**