Graph Theory

Dr. Irfan Yousuf
Department of Computer Science (New Campus)
UET, Lahore

(Week 17; May 05 - 09, 2025)

Outline

- Graph Embedding
- Knowledge Graph

Graph Embedding

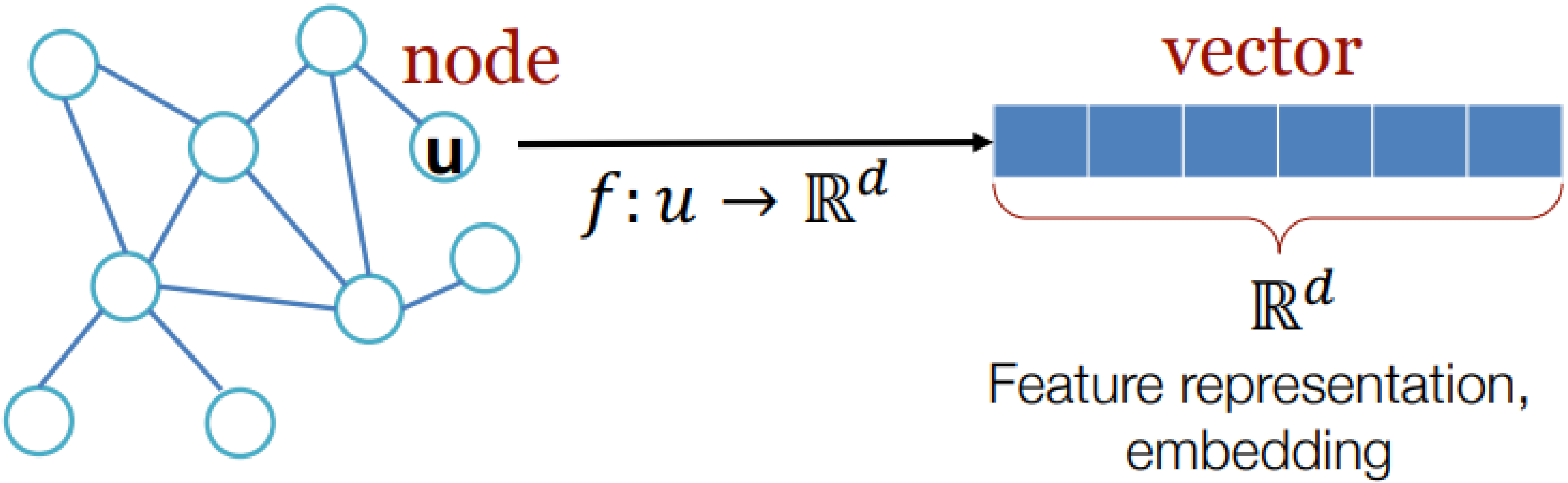
- Graph embeddings are the transformation of graphs to a vector or a set of vectors.
- They encode the structural and semantic information of the graph, enabling downstream machine learning algorithms to operate on them.
- The primary objective of graph embeddings is to capture the underlying patterns and properties of the graph, facilitating various tasks.
- Graph embeddings provide a solution by transforming the nodes and edges of a graph into low-dimensional vectors while preserving important structural information.

Graph Embedding

- In essence, graph embeddings transform the graph data from a <u>discrete, symbolic representation into a continuous, numerical representation</u>, which traditional machine learning models or deep learning architectures can efficiently process.

- The main advantage of graph embeddings lies in their ability to preserve both the local neighborhood information (i.e., the direct connections of each node) and the global structural patterns of the entire graph.

Graph Embedding



	x0	x1	x2	x3	x4	x5	x6
Node							
0	0.102280	-1.166695	2.670930	1.390916	-0.402797	1.636759	-1.846049
1	0.762967	-0.440913	-0.605629	0.149154	-0.091160	0.443734	0.581106
2	-0.515594	1.069805	0.822512	-0.850004	1.564274	1.069993	0.482808
3	-0.361819	0.245590	-0.768354	0.580419	1.026706	1.327916	-0.143739
4	0.510549	-0.642235	0.195326	-1.243745	-0.513322	1.483265	0.331653
5	0.464585	0.972432	-0.521167	0.778889	-0.742073	1.062958	0.565665
6	0.797623	-0.690840	-2.337762	0.077373	-0.199774	-1.680805	-0.684513
7	0.443615	1.486821	0.596929	-1.507983	0.171009	1.208194	-0.472783
8	-0.462106	-1.926415	-0.058096	0.009180	0.353436	-0.144805	-0.757387
9	0.308620	0.844023	0.425973	-0.377976	-1.307095	2.064543	-1.198032
10	-0.656677	-1.013211	-1.415589	-1.255749	-0.324081	0.979972	-0.090685

Word Embedding

- Word Embeddings in NLP is a technique where individual words are represented as real-valued vectors in a lower-dimensional space and captures inter-word semantics.
- Each word is represented by a real-valued vector with tens or hundreds of dimensions.
- Word embeddings give us a way to use an efficient, dense representation in which similar words have a similar encoding.
- An embedding is a dense vector of floating point values (the length of the vector is a parameter you specify).

Word Embedding: One Hot Encoding

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets a 1x9 vector representation

Word Embedding: Custom Encoding

Try to build a lower dimensional embedding

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

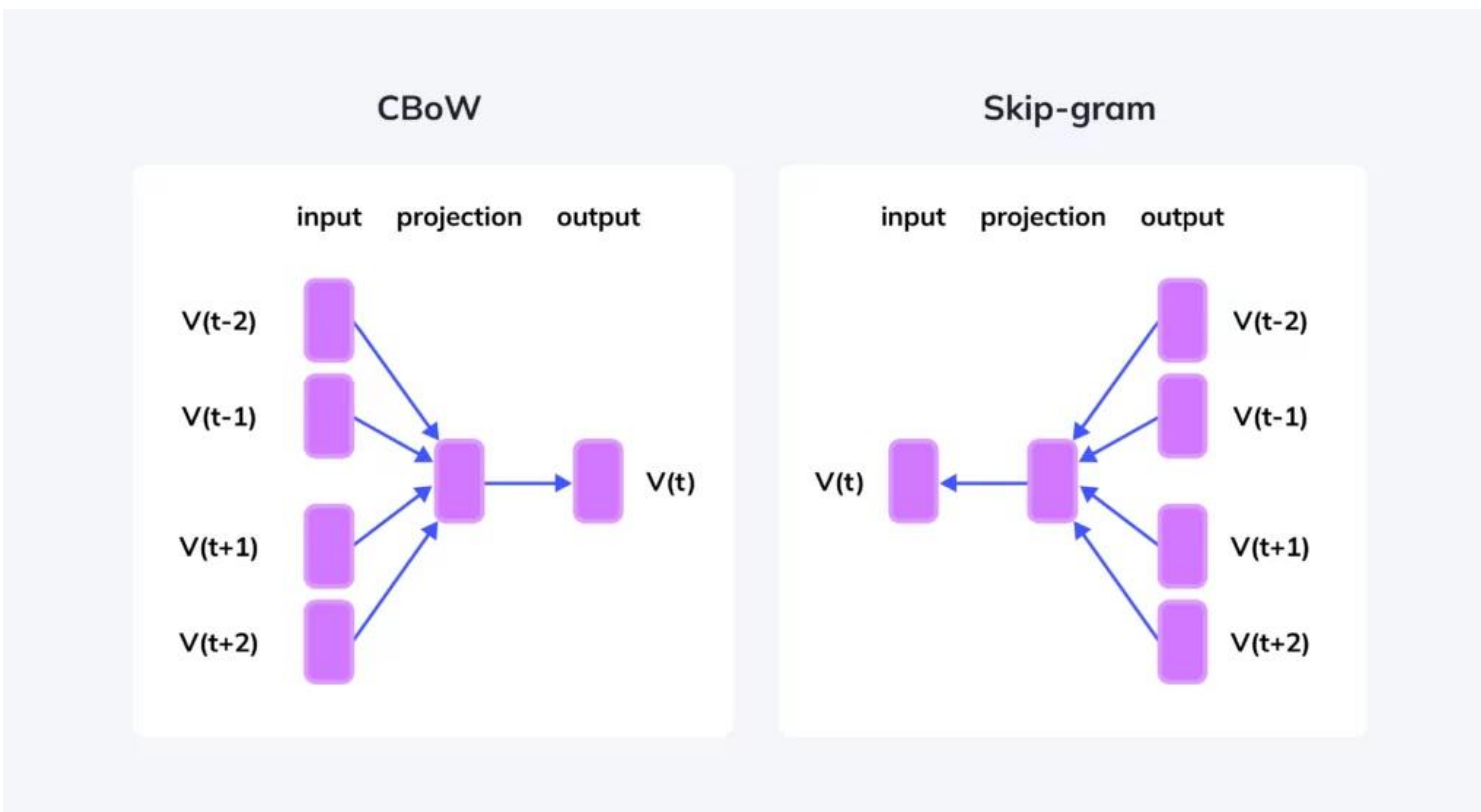
	Femininity	Youth	Royalty
Man	0	0	0
Woman	1	0	0
Boy	0	1	0
Girl	1	1	0
Prince	0	1	1
Princess	1	1	1
Queen	1	0	1
King	0	0	1
Monarch	0.5	0.5	1

Each word gets a 1x3 vector

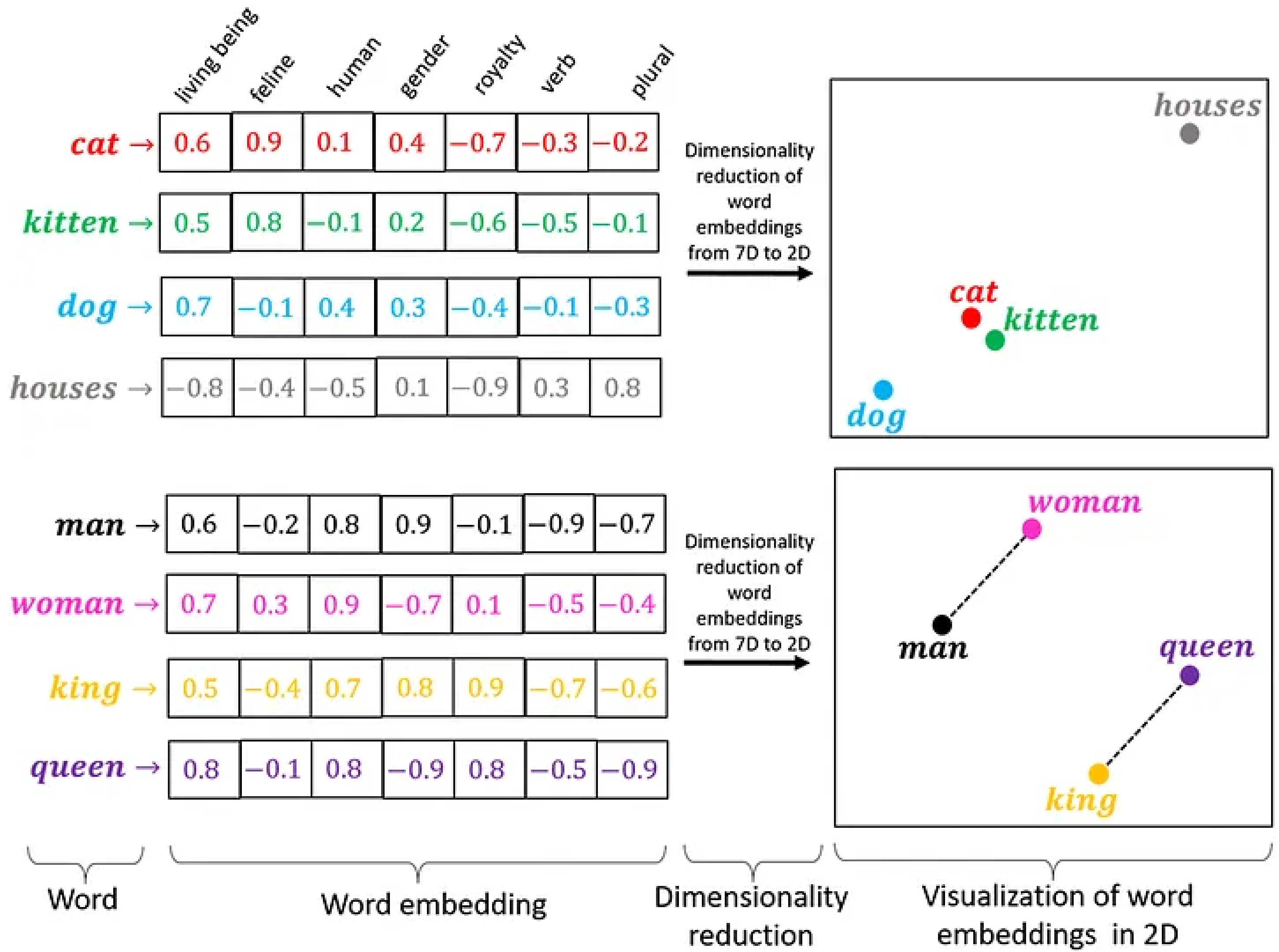
Similar words... similar vectors

@shane_a_lynn | @TeamEdgeTier

Word Embedding



Word Embedding: Word2Vec



Graph Representation Learning

The process of learning graph embeddings is referred to as **graph representation learning**. It involves learning a mapping function $f : G \rightarrow R^n$, where f learns the graph's local and global geometric properties in the form of a low-dimensional representation with a size of n. There are two main types of graph embeddings:

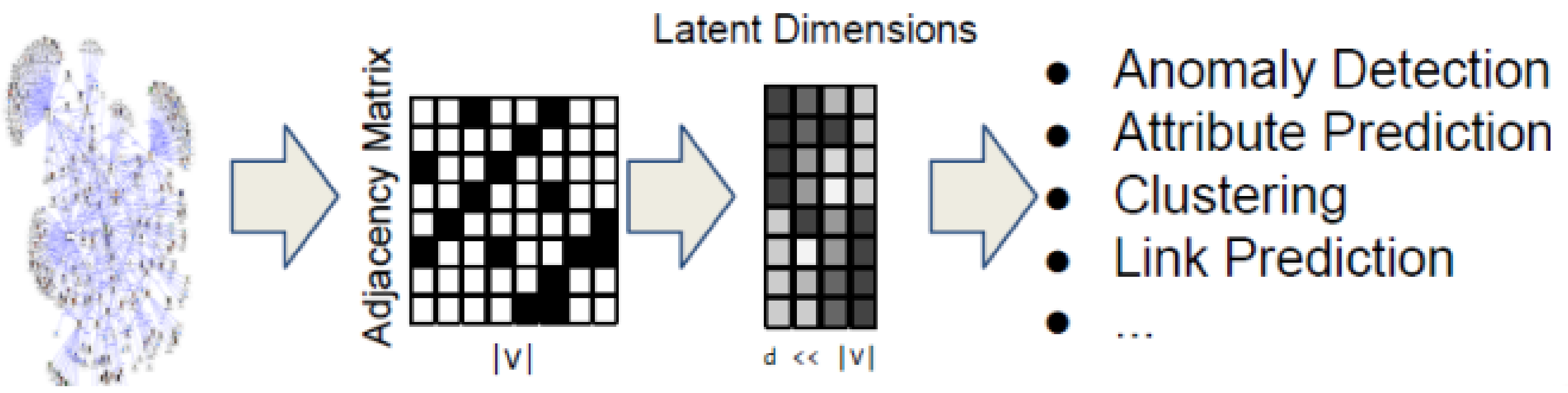
- node embeddings ($f : V \rightarrow R^n$)
- edge embeddings ($f : E \rightarrow R^n$)

The choice between these embeddings depends on the specific machine learning task.

Graph Representation Learning

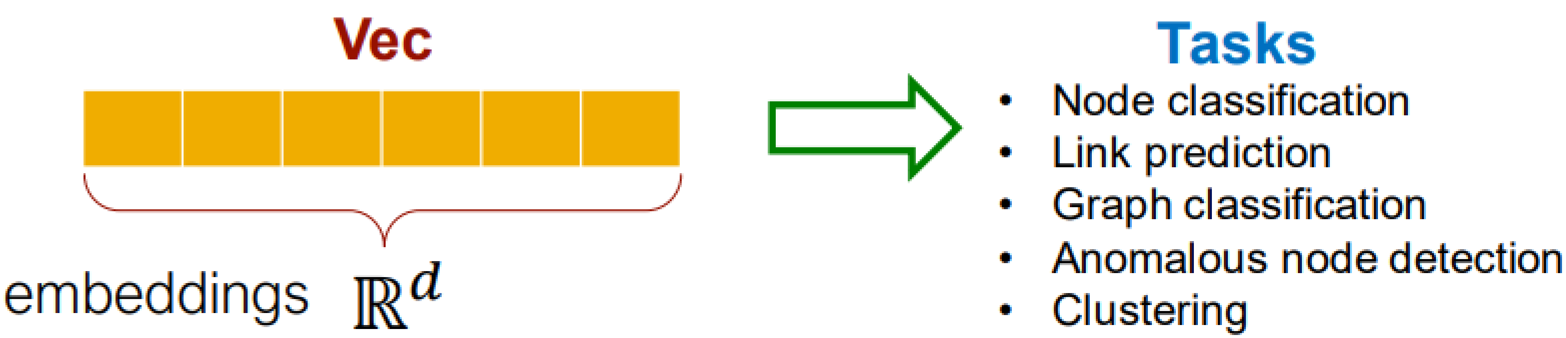
The goal is to map each node into a low-dimensional space

- Distributed representation for nodes
- Similarity between nodes indicates link strength
- Encodes network information and generate node representation



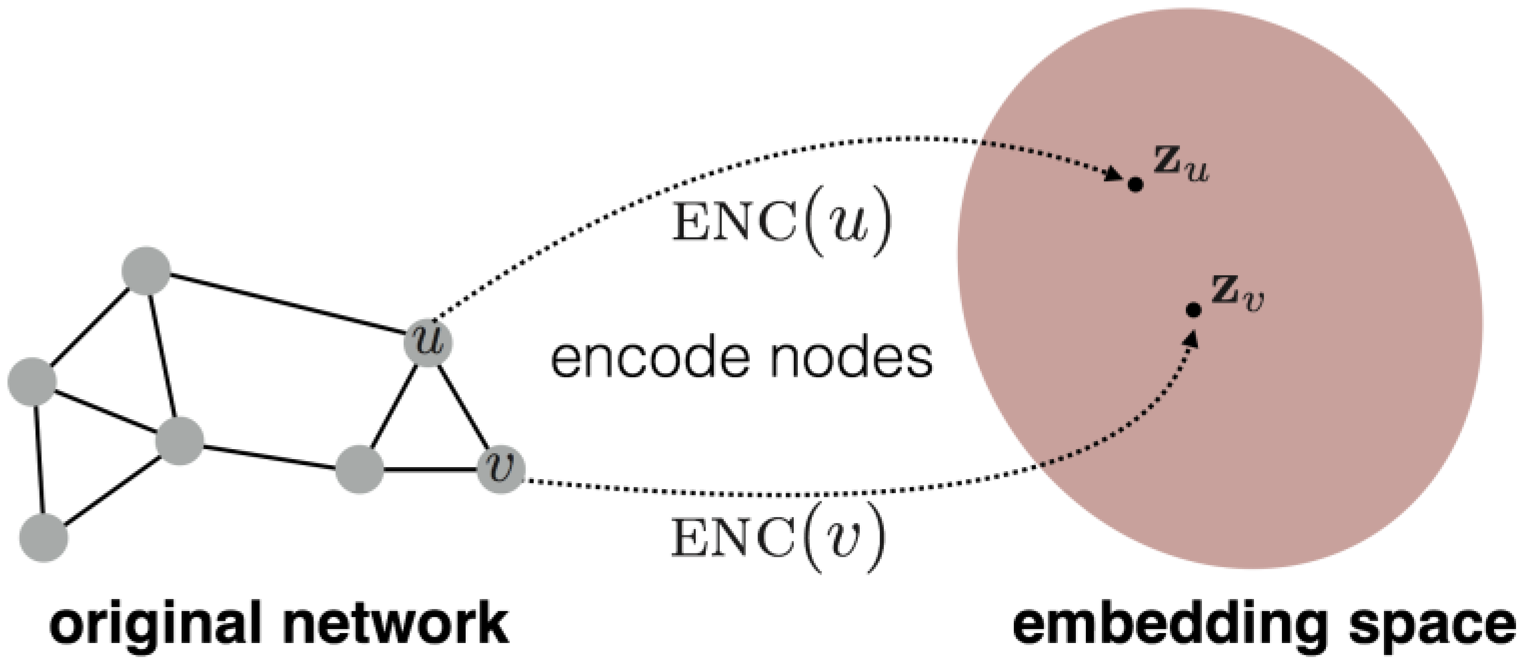
Graph Representation Learning

- **Task: Map nodes into an embedding space**
  - Similarity of embeddings between nodes indicates their similarity in the network. For example:
    - Both nodes are close to each other (connected by an edge)
  - Encode network information
  - Potentially used for many downstream predictions



Node Embedding

- Goal is to encode nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the graph

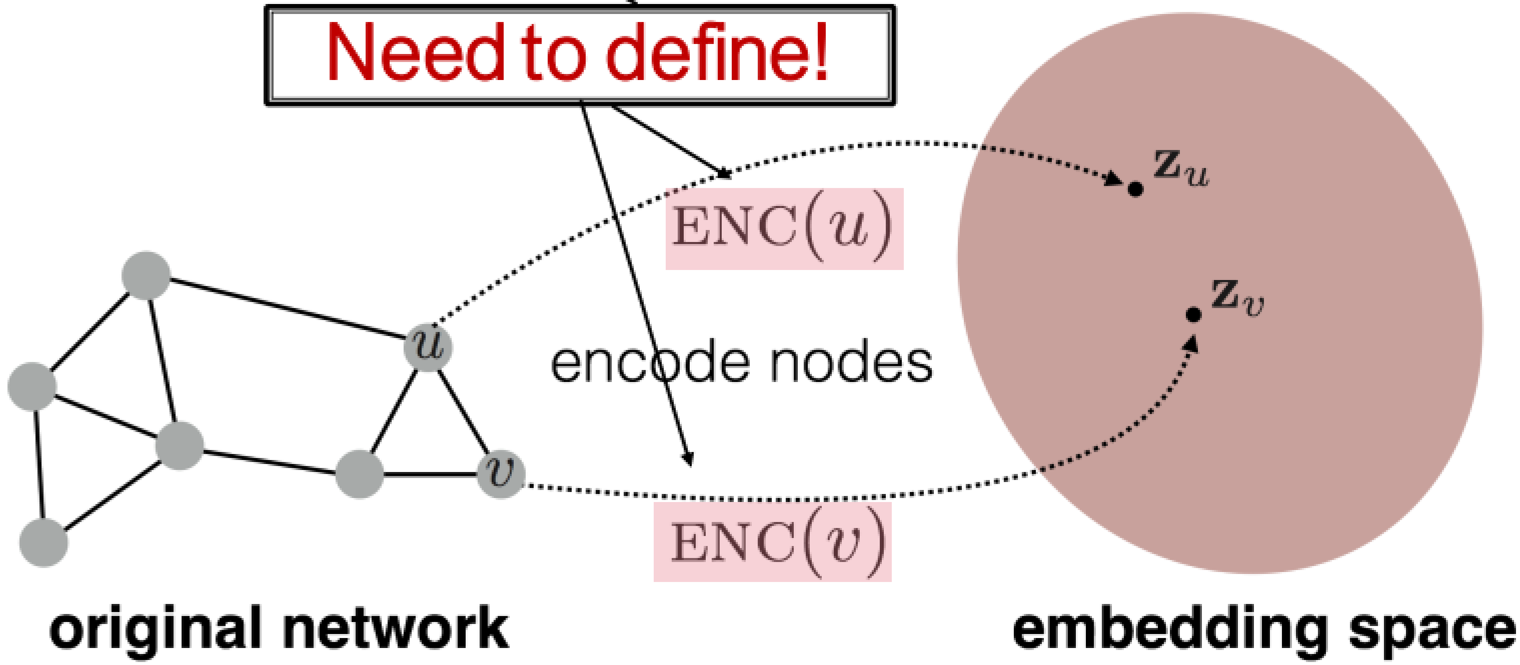


Node Embedding

Goal: $\text{similarity}(u, v) \approx \mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u$

in the original network — Similarity of the embedding

Need to define!



Node Embedding

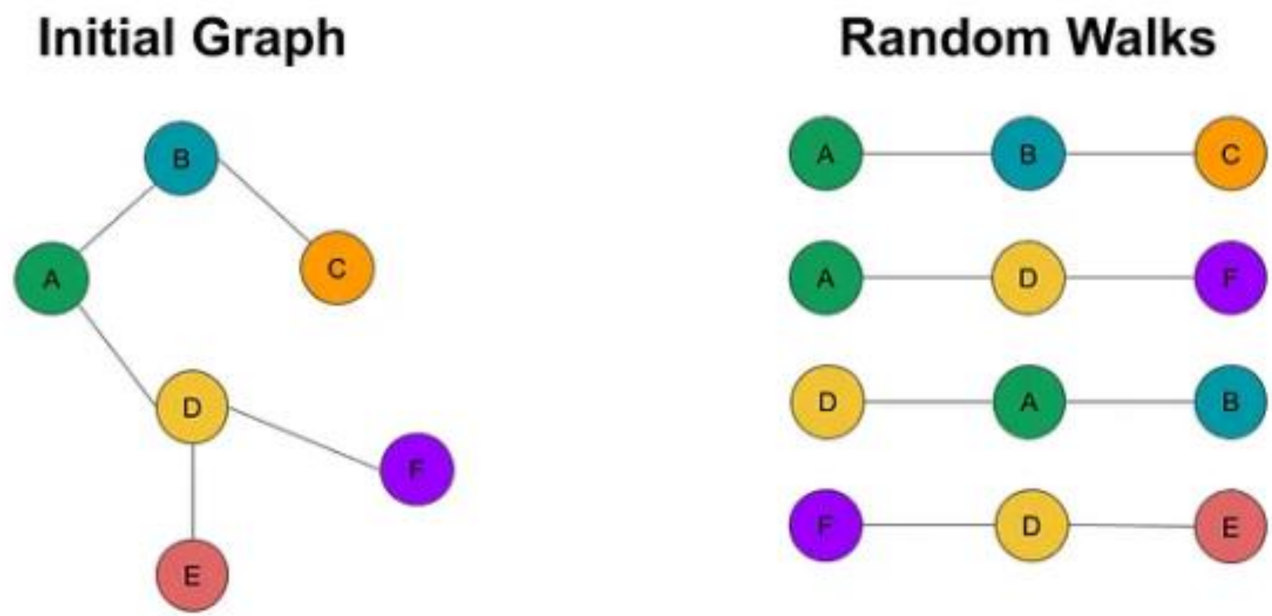
- Key choice of methods is **how they define node similarity.**
- Should two nodes have a similar embedding if they…
  - are linked?
  - share neighbors?
  - have similar “structural roles”?
- We will now learn node similarity definition that uses **random walks**, and how to optimize embeddings for such a similarity measure.

Graph Traversal and Random Walk

- Intuition: Find embedding of nodes to d-dimensions that preserves similarity
- Idea: Learn node embedding such that nearby nodes are close together
- Given a node u, how do we define nearby nodes?
  - $N_S(u)$ … neighbourhood of u obtained by some strategy S

Graph Traversal and Random Walk

Similarly, a random walk creates “sentences” by performing random walks through the graph.



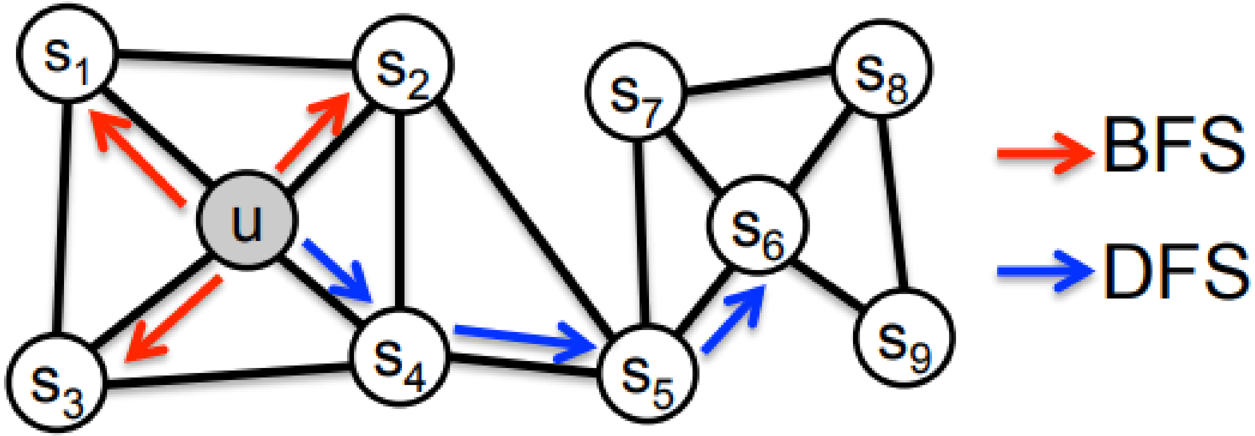
Graph Traversal

- Given $G = (V, E)$
- Goal is to learn $f: u \to \mathbb{R}^d$
  - where f is a table lookup
    - We directly "learn" coordinates $f(u)$ of u
- Given node u, we want to learn feature representation $f(u)$ that is predictive of nodes in u's neighborhood $N_S(u)$

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

Graph Traversal: BFS vs. DFS

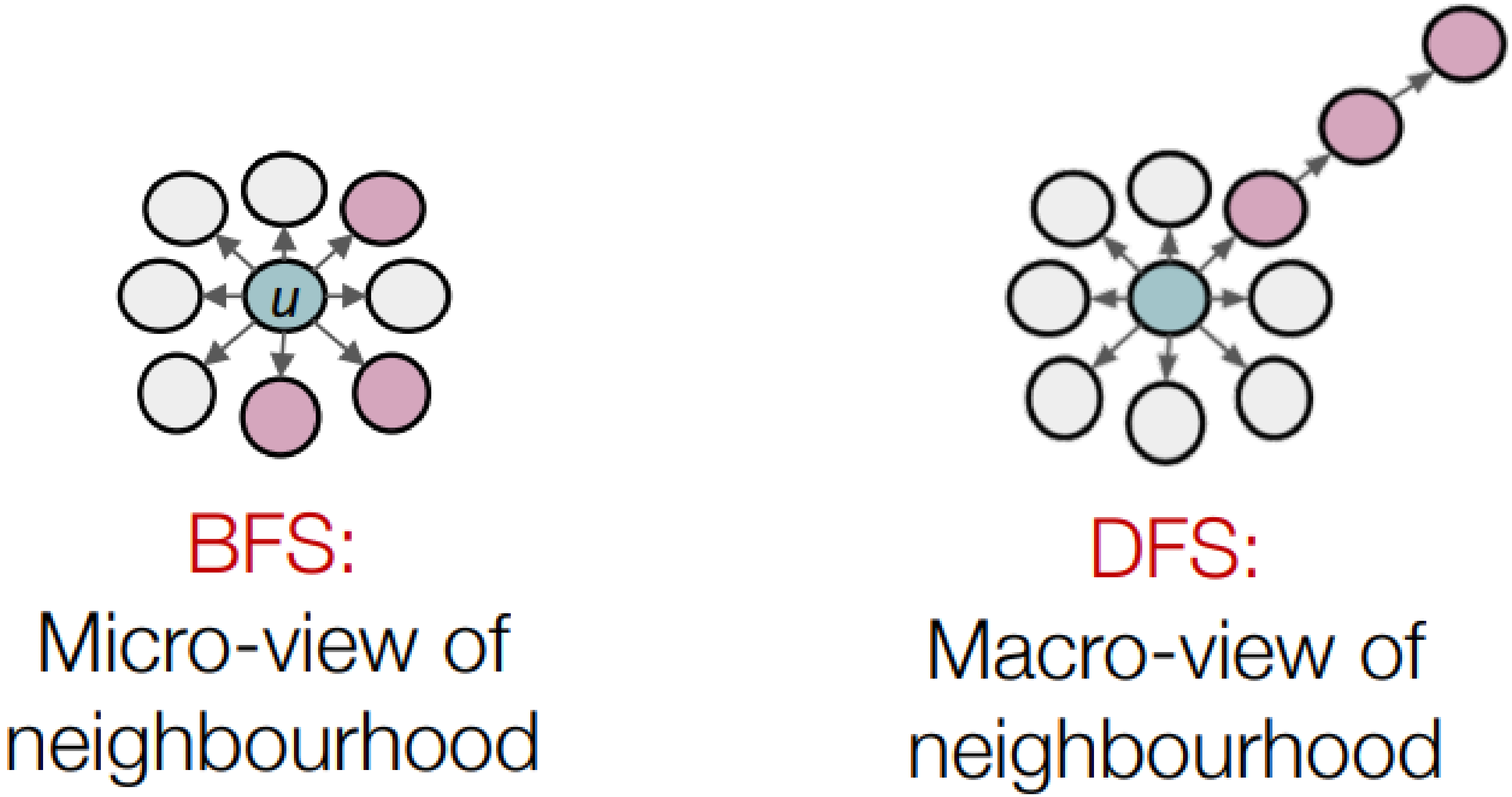
Two classic strategies to define a neighborhood $N_S(u)$ of a given node u:



$N_{BFS}(u) = \{ s_1, s_2, s_3 \}$ Local microscopic view

$N_{DFS}(u) = \{ s_4, s_5, s_6 \}$ Global macroscopic view

Graph Traversal: BFS vs. DFS



node2vec algorithm

1) Simulate r random walks of length l starting from each node u

2) Optimize the node2vec objective using Stochastic Gradient Descent

node2vec algorithm

Task-independent feature learning in networks:

- An explicit locality preserving objective for feature learning
- Biased random walks capture diversity of network patterns
- Scalable and robust algorithm

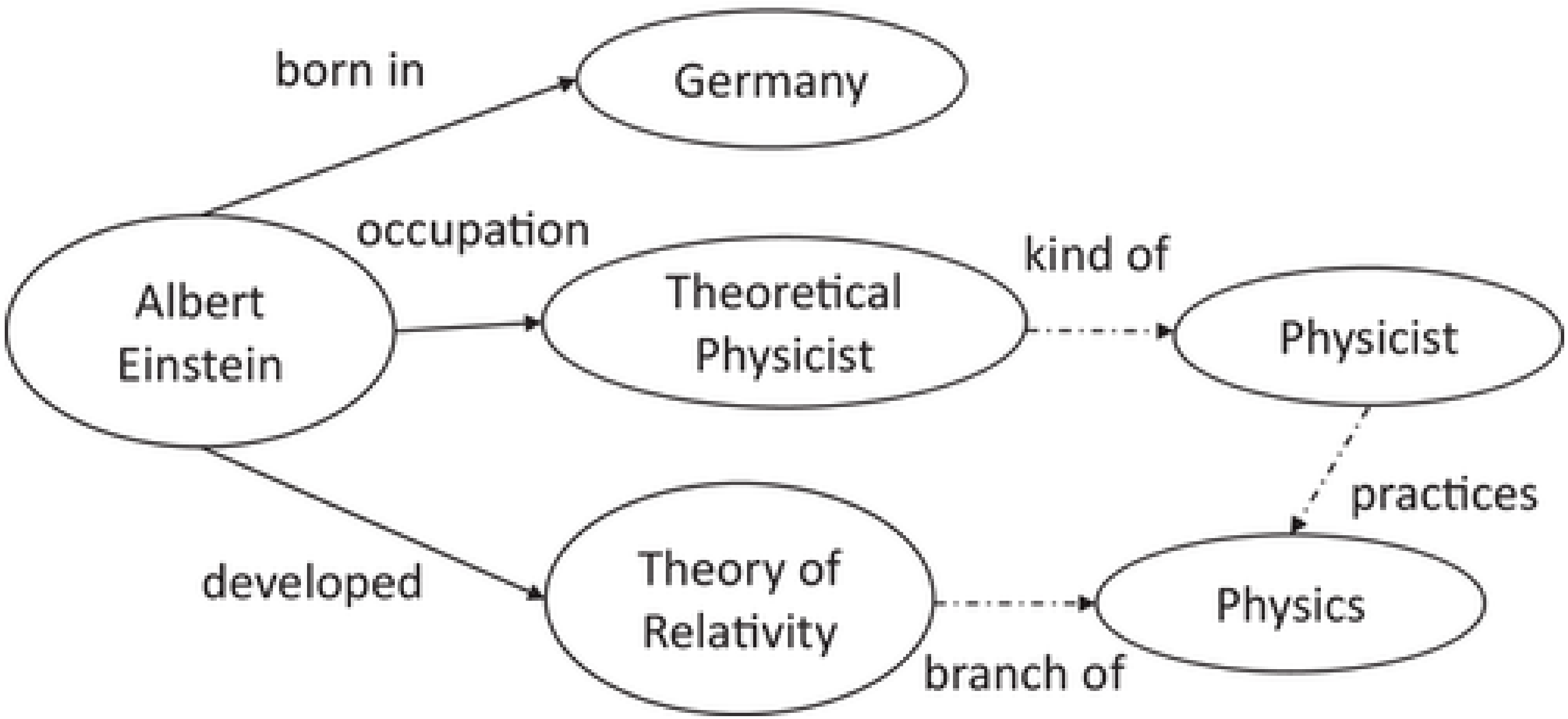
Knowledge Graph

Knowledge Graph

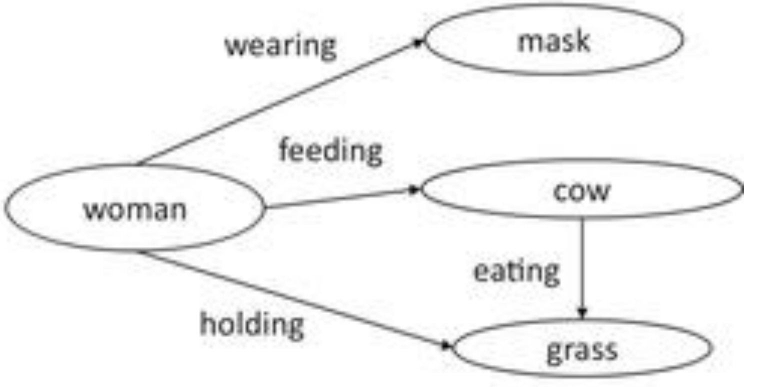
- A Knowledge Graph is a (mostly directed) labeled graph in which domain-specific meanings are associated with nodes and edges.
- A **node** could represent any real-world entity, for example, people, companies, and computers.
- An **edge** label captures the relationship of interest between the two nodes.
- For example, a friendship relationship between two people; a customer relationship between a company and person; or a network connection between two computers.

Knowledge Graph

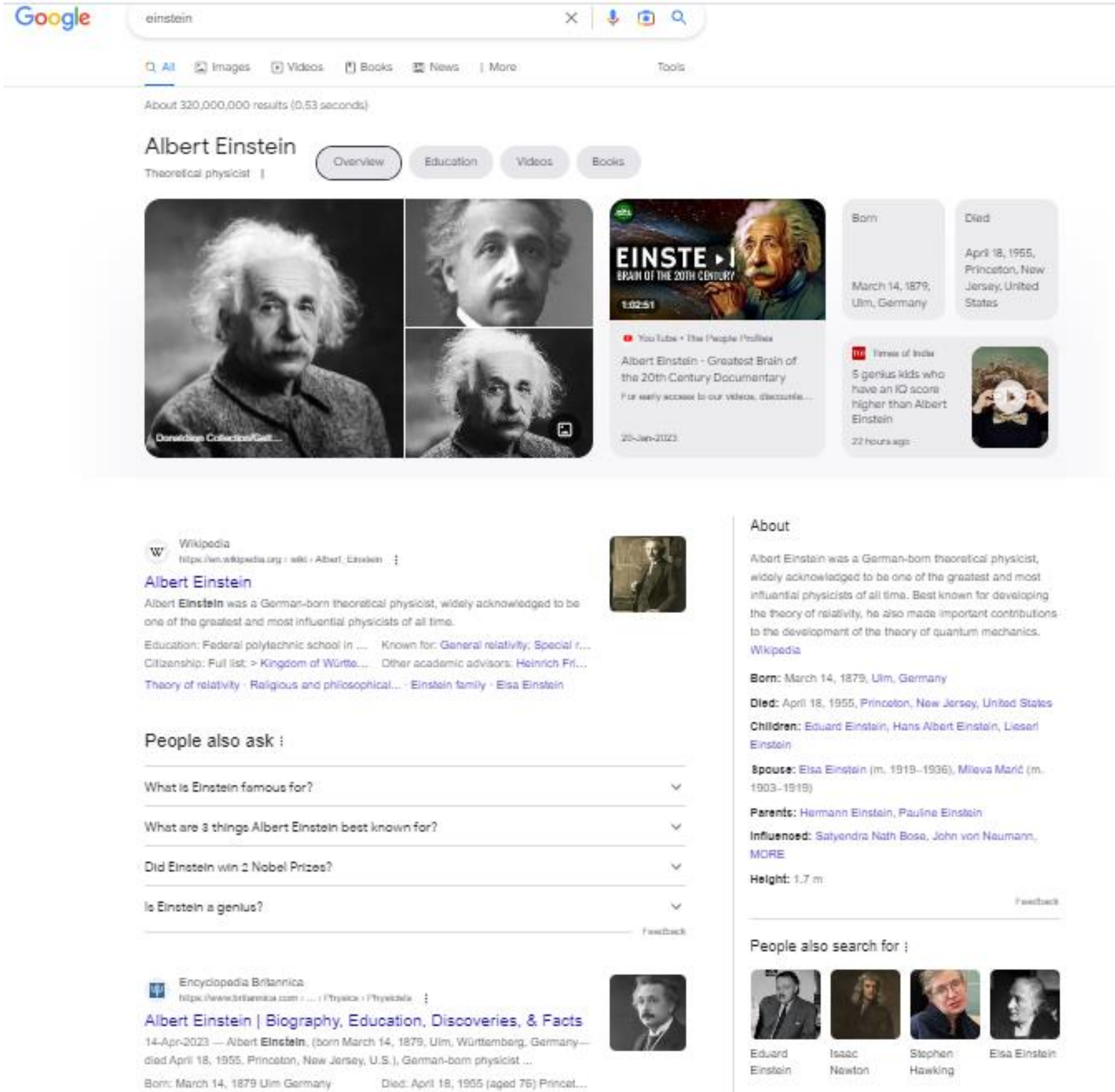
Albert Einstein was a **German-born theoretical physicist** who developed the **theory of relativity**.



Knowledge Graph



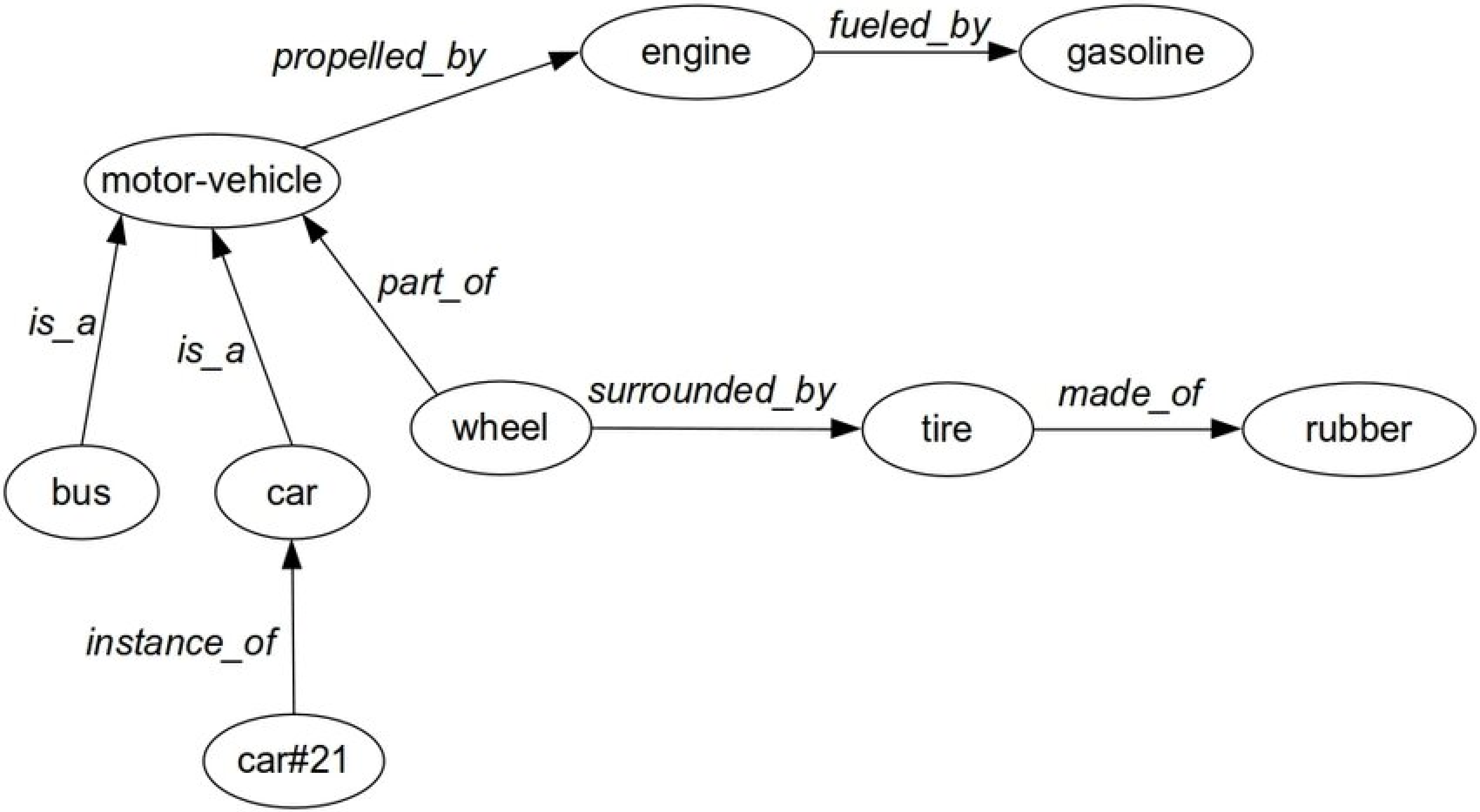
Google Knowledge Graph

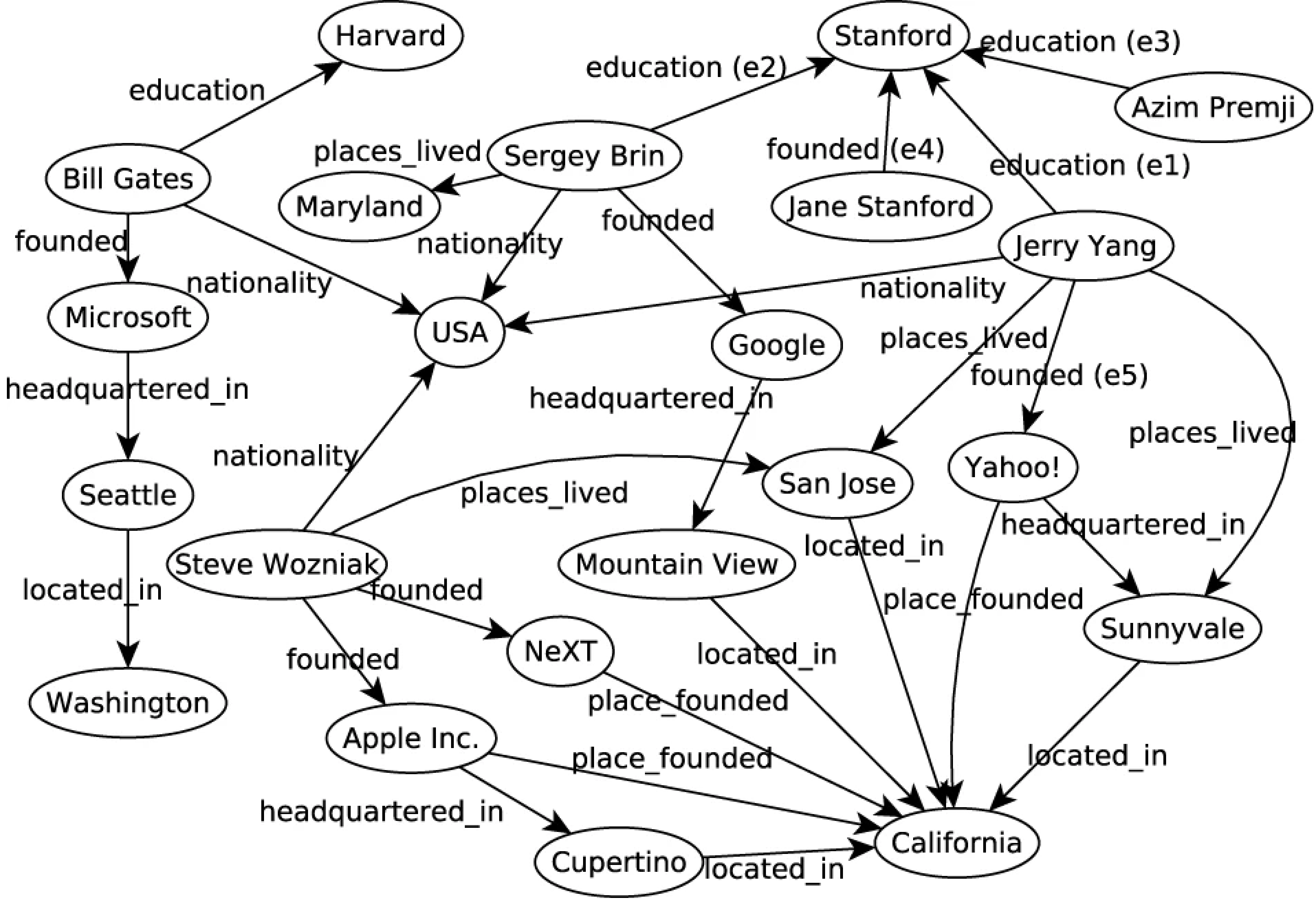


Google Knowledge Graph

- Google's goal is to understand the meaning behind “real-world entities and their relationships to one another.”

- As opposed to just showing results based on the strings of words used in the search query.


propelled_by
engine
fueled_by
gasoline
motor-vehicle
part_of
is_a
is_a
surrounded_by
made_of
wheel
tire
rubber
bus
car
instance_of
car#21

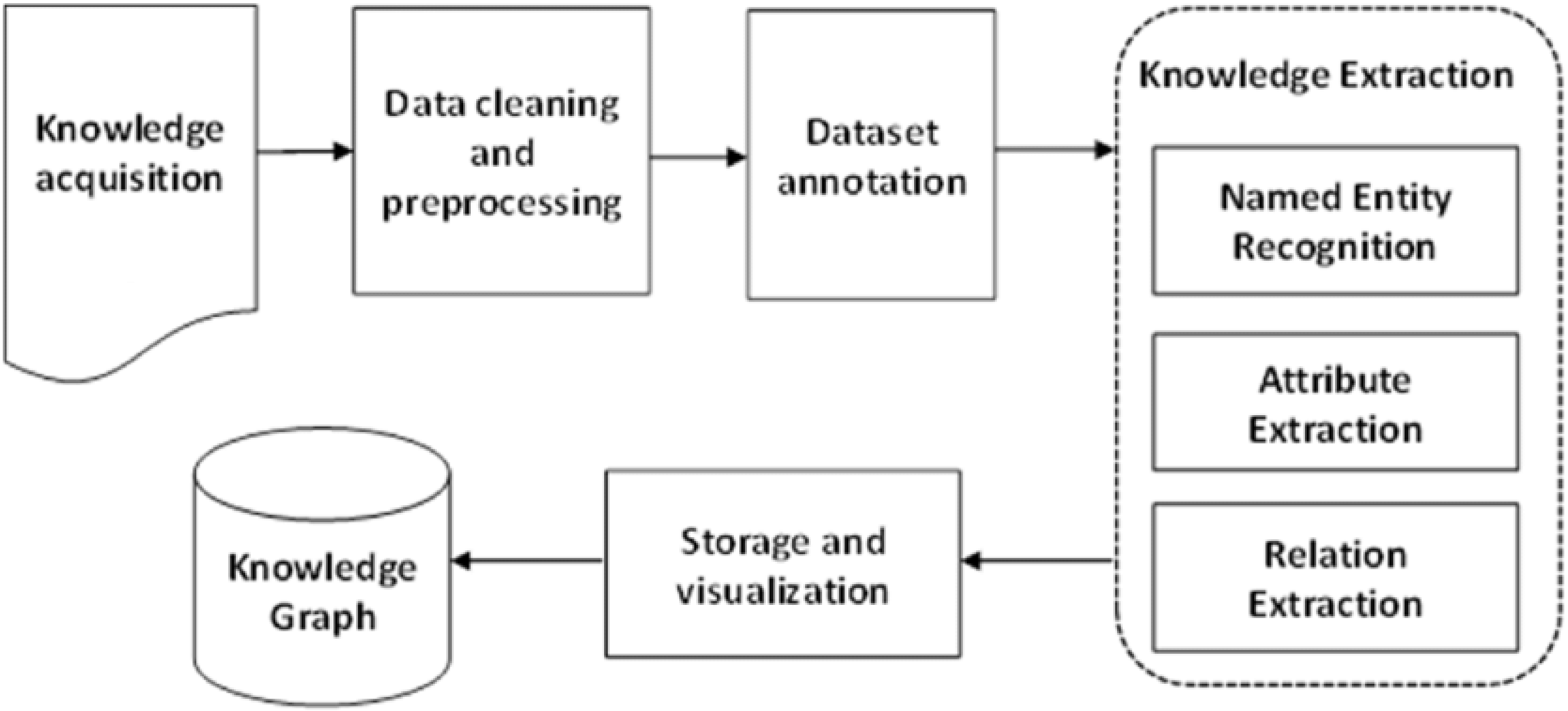

Harvard
Stanford
education (e3)
education
education (e2)
Azim Premji
places_lived
Sergey Brin
founded (e4)
education (e1)
Bill Gates
Maryland
Jane Stanford
founded
founded
nationality
Jerry Yang
nationality
nationality
Microsoft
USA
Google
places_lived
founded (e5)
headquartered_in
headquartered_in
places_lived
nationality
Yahoo!
San Jose
places_lived
Seattle
headquartered_in
located_in
Steve Wozniak
Mountain View
located_in
founded
place_founded
Sunnyvale
NeXT
located_in
founded
located_in
Washington
place_founded
Apple Inc.
place_founded
located_in
headquartered_in
California
Cupertino
located_in

How to build a Knowledge Graph

- Extract entities, a.k.a. Named Entity Recognition (NER), which are going to be the nodes of the knowledge graph.

- Extract relations between the entities, a.k.a. Relation Classification (RC), which are going to be the edges of the knowledge graph.

How to build a Knowledge Graph

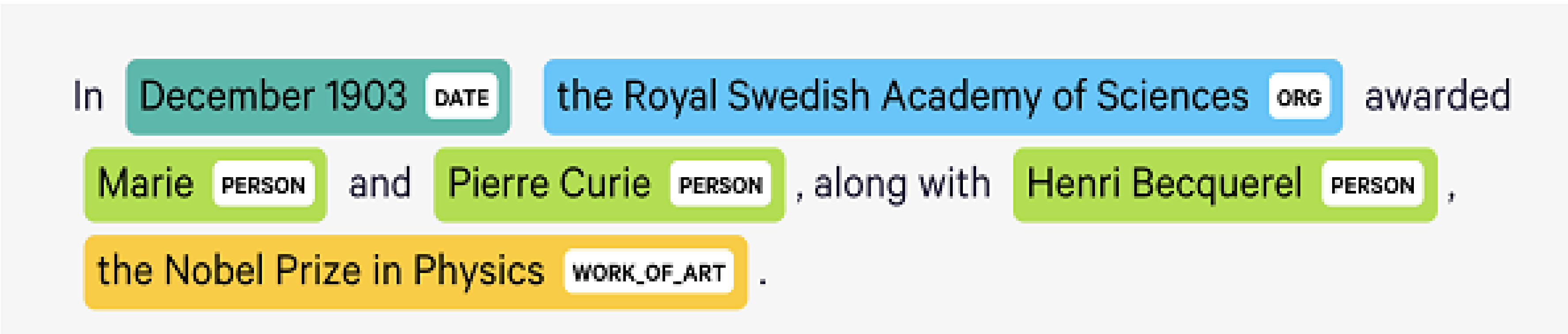


Named Entity Recognition

- Named Entity Recognition (NER) is a sub-task of information extraction in Natural Language Processing (NLP) that classifies **named entities** into predefined categories such as person names, organizations, locations,, time expressions, quantities, monetary values, and more.

- Named Entity Recognition (NER) serves as a bridge between unstructured text and structured data,

Named Entity Recognition



Purpose of Named Entity Recognition

- NER's primary objective is to comb through unstructured text and identify specific chunks as named entities, subsequently classifying them into **predefined categories**.

- This conversion of raw text into structured information makes data more actionable, facilitating tasks like data analysis, information retrieval, and knowledge graph construction.

How NER works?

- **Tokenization:** Before identifying entities, the text is split into tokens, which can be words, phrases, or even sentences. For instance, "Steve Jobs co-founded Apple" would be split into tokens like "Steve", "Jobs", "co-founded", "Apple".

- **Entity identification:** Using various linguistic rules or statistical methods, potential named entities are detected. This involves recognizing patterns, such as capitalization in names ("Steve Jobs") or specific formats (like dates).

How NER works?

- **Entity classification:** Once entities are identified, they are categorized into predefined classes such as "Person", "Organization", or "Location". This is often achieved using machine learning models trained on labeled datasets. For our example, "Steve Jobs" would be classified as a "Person" and "Apple" as an "Organization".

- **Contextual analysis:** NER systems often consider the surrounding context to improve accuracy. For instance, in the sentence "Apple released a new iPhone", the context helps the system recognize "Apple" as an organization rather than a fruit.

How NER works?

- **Post-processing:** After initial recognition and classification, post-processing might be applied to refine results. This could involve resolving ambiguities, merging multi-token entities, or using knowledge bases to enhance entity data.

NER Methods

- **Lexicon/Dictionary-based NER:** The most straightforward approach to NER is based on using a lexicon or dictionary as a vocabulary reference.
- The dictionary includes a limited set of entities that can be identified in the given text using basic string-matching algorithms.
- A dictionary’s finite vocabulary collection limits the dictionary-based approach’s efficiency. This technique works well as long the dictionary is appropriately updated and maintained.

NER Methods

- **Rule-based methods** are grounded in manually crafted rules.

- They identify and classify named entities based on linguistic patterns, regular expressions, context of a word etc.

- While they shine in specific domains where entities are well-defined, such as extracting standard medical terms from clinical notes, their scalability is limited.

- They might struggle with large or diverse datasets due to the rigidity of predefined rules.

NER Methods

- The **machine learning-based approach** to NER involves using statistical models to recognize entities in a given text document.
- Machine learning models observe textual data to create feature-based representations of entities. These representations allow NER systems to detect existing entities even if they are slightly misspelled.
- ML-based NER requires model training on annotated textual data. The trained model is then used for annotating new files. In this way, ML-based NER systems self-improve by automatically evolving their entity knowledge base.

NER Models

- The best Named Entity Recognition (NER) model for general usage depends on various factors such as the specific domain or language for which it will be used, the size of the dataset available for training, and the specific requirements of the NER task.
- Some popular NER models include **Stanford NER, spaCy**, and the **BERT** (Bidirectional Encoder Representations from Transformers) model from Google. Each of these models has its own strengths and weaknesses, so it's important to consider the specific needs of your project when choosing the best NER model.

Relation Classification

- Relation Classification is the task of identifying the **semantic relation** holding between two nominal entities in text.

- Ontology formally represents concepts and relationships within a domain, emphasizing a shared understanding of semantics.

- A knowledge graph may use an ontology as a foundational structure for defining the types of entities and relationships within a domain.

Relation Classification

Relation	Example
Cause-Effect	laugh (cause) wrinkles (effect)
Instrument-Agency	laser (instrument) printer (agency)
Product-Producer	honey (product) bee (producer)
Origin-Entity	message (entity) from outer-space (origin)
Theme-Tool	news (theme) conference(tool)
Part-Whole	the door (part) of the car (whole)
Content-Container	the apples (content) in the basket (container)

Relation Classification Methods

- **Linear Pattern:** one way the cause-effect relation can be expressed in text:
- [effect] is the result of [cause]

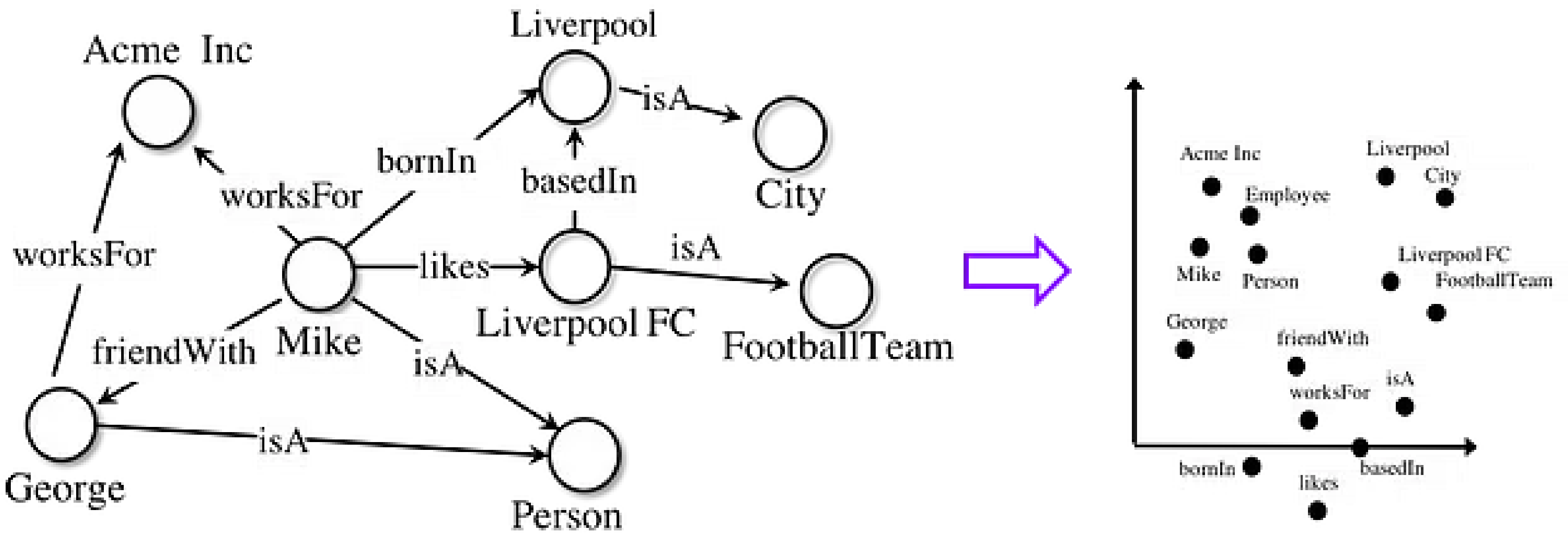
- The fatal accident is the result of drunken driving.

Knowledge Graph Embeddings

- A Knowledge graph embedding (KGE) is a representation of a KG element into a continuous vector space.

- The objective of learning those embeddings is to ease the manipulation of graph elements (entities, relations) for prediction tasks such as entity classification, link prediction or recommender systems.

Knowledge Graph Embeddings



Summary

- Graph Embeddings
- Knowledge Graph