Relational Inductive Bias in Natural Language Processing

Lecture at the Institute of Logic, Language and Computation, University of Amsterdam

Pushkar Mishra
AI Researcher, Meta AI

Meta AI

Motivation

Language modeling is predominantly seen through the lens of time series analysis.

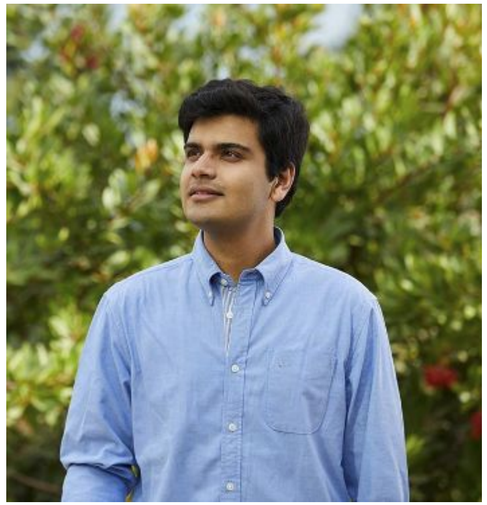
Words are sequences of letters, sentences are sequences of words, documents are sequences of sentences, and so on.

Motivation

Most NLP tasks are also viewed through the same lens.

We are going to see how language modeling can benefit greatly from relational reasoning.

About me



Researcher, Meta AI, United Kingdom

Sessions

13:00 PM

Session 1

Inductive Bias, Invariants, GNNs

13:50 PM

Session 2

Break + Q/A

14:00 PM

Session 3

Graph Learning in NLP

14:40 PM

Session 4

Q/A

Session 1: Agenda

01 Inductive bias & Invariants

02 Invariants in Deep Learning

03 The Unique Case of Graphs

04 Transductive vs. Inductive Learning

05 Graph Neural Networks

01 Inductive Bias & Invariants

Inductive Bias

- The set of assumptions that govern the choice of hypothesis
  - For example, choosing a linear classifier automatically imposes the constraint that the hypothesis must be linear

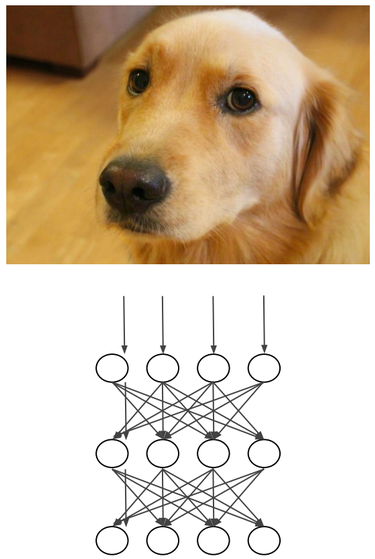
- Essentially, inductive biases make the selection of certain hypotheses preferred over others

- ***Relational inductive bias* — biases that constrain the hypothesis space in terms of interactions amongst units of the input**

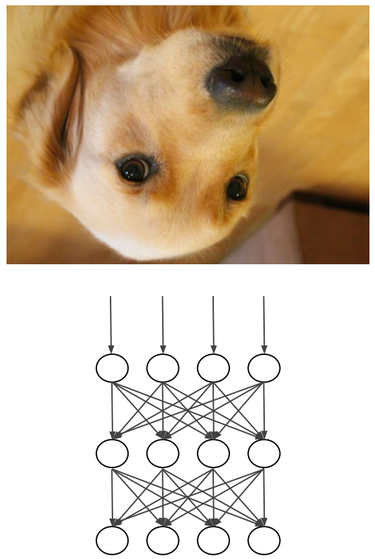
Invariants

- Invariance — the property whereby a certain transformation in input does not affect the output

- Introducing invariants into a model inherently introduces inductive biases



Dog



Cat? Still a dog!

02 Invariants in Deep Learning

Deep Learning

- Deep learning reduces the need for feature engineering
  - Successive layers of the network extract features at different levels of abstractions
  - Modeling is done end-to-end, from raw data to output
  - Many different architectures available, MLPs, CNNs, RNNs

- We will gather important insights into learning over different structures
  - Regular grids, such as images
  - Sequences, such as text and time series

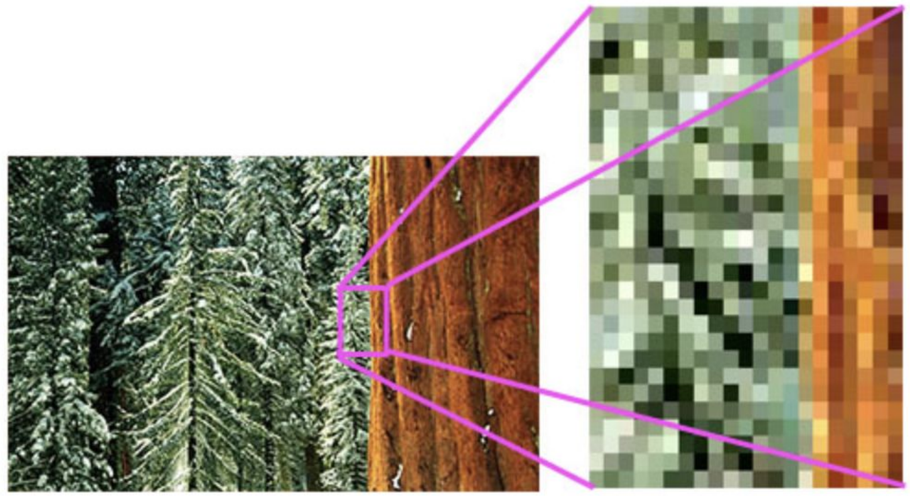
MLP: The “Jack of All” Architecture

- A simple MLP is a universal approximator
  - Can model a wide range of continuous functions over data with different structures
  - Already proven for finite width networks with non-linearities

- MLPs are the jacks of all, but masters of none
  - They perform decently across several tasks, but rarely achieve the state of the art
  - Very few assumptions about the underlying problem => low task-specific inductive bias
  - We did not discuss how hard it is to learn the parameters or how wide the MLPs must be

Deep Learning on Regular Grids

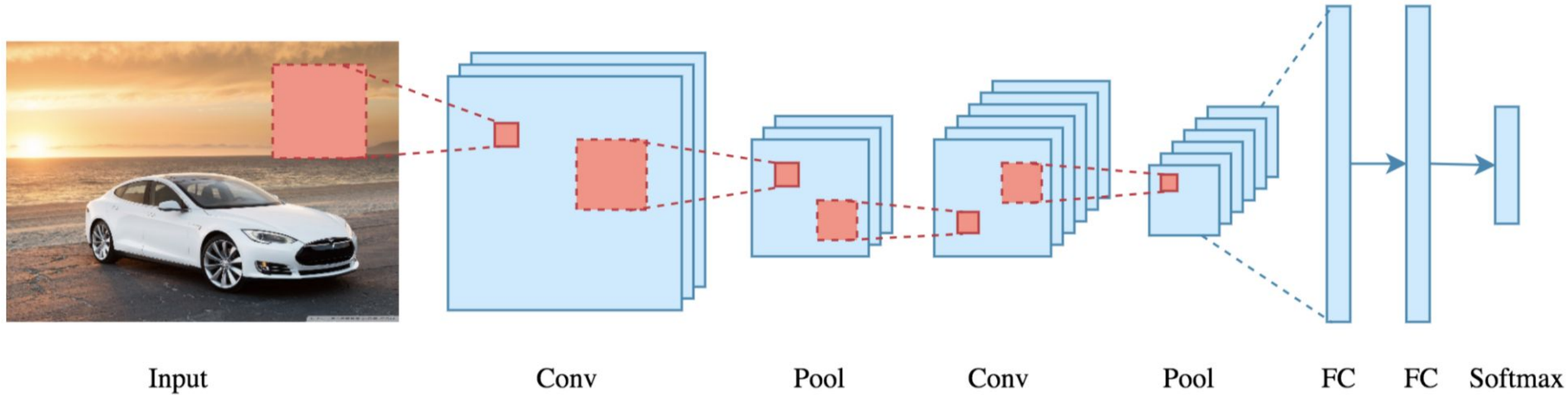
- Images are regular grids of pixels
  - Pixel is the basic unit of information
  - They are all of some fixed dimension
  - Position invariant in terms of properties



- Regular grids can be in 3D space too
  - Basic unit of information is “voxel”
  - Modeling of objects, actions, etc.

CNNs on Regular Grids

- CNNs learn on regular grids using three main components
  - Convolution
  - Pooling
  - Task specific fully-connected layers



Some Representational Details

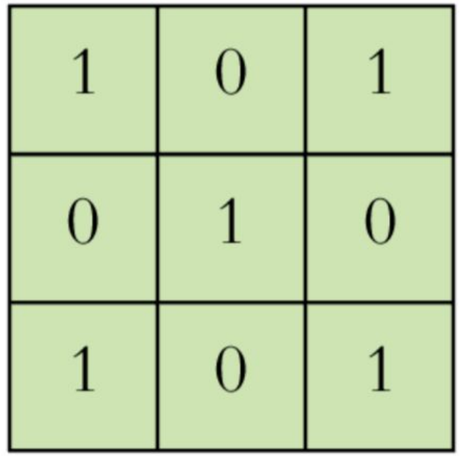
- We will represent images as grids of numbers

- The numbers represent pixel values within the image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolution

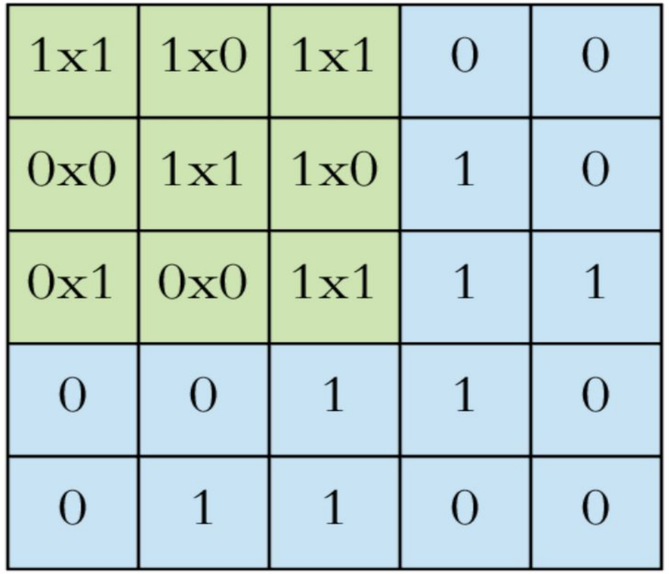
- Convolution is a powerful operation for feature extraction
- Linear element-wise multiplication of a filter with the image
- Filter is a grid of trainable weights



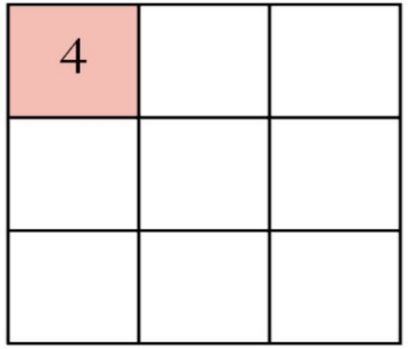
Convolution Operation

The filter slides over the image sequentially

The area that the filter covers is its "*receptive field*"



Input ✶ Filter



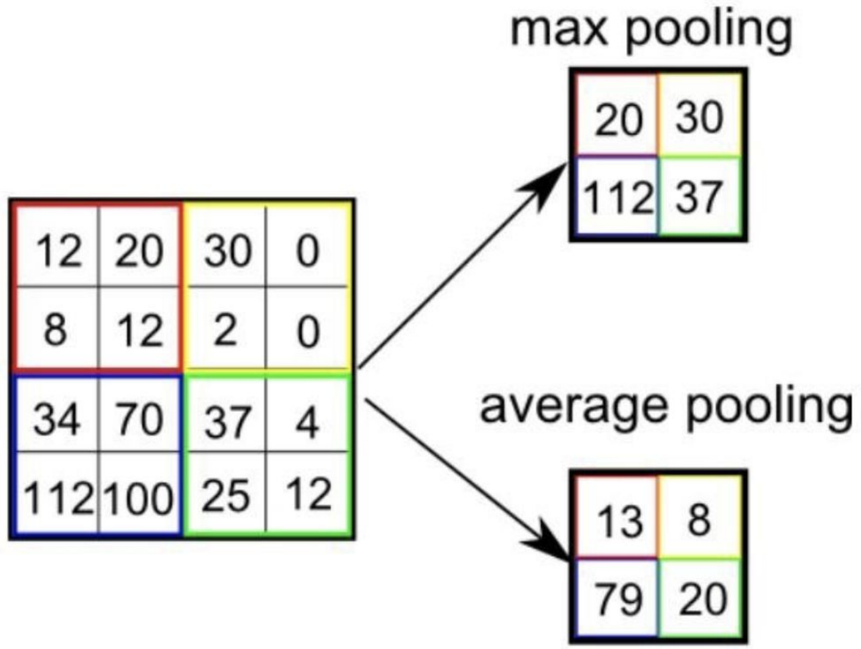
Feature map

We can apply some non-linearity to the feature map

Pooling

- Pooling is essentially a down-sampling operation
  - We do not want to overfit on the feature maps we get
  - Down-sampling helps coarsen or summarize the feature maps

- Different patch operations for pooling
  - Max
  - Sum
  - Average



Important Takeaways

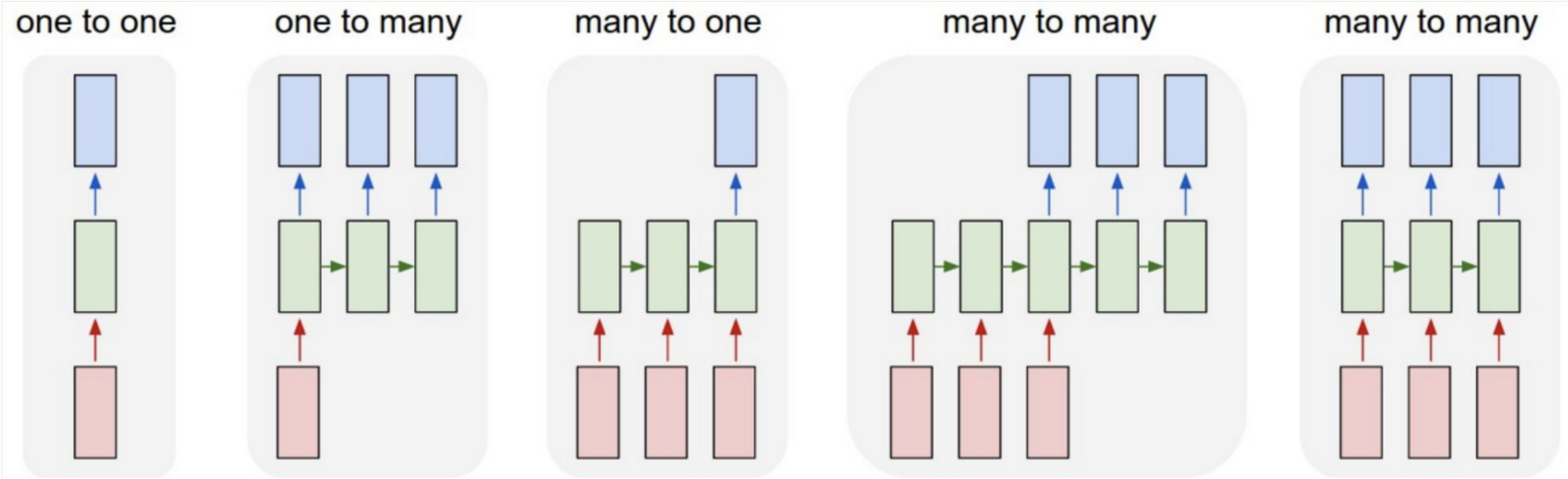
- Translation equivariance and invariance
  - Convolution provides translation equivariance
  - Pooling ensures translation invariance over small regions

- Filters capture spatial locality in input patterns

- Successive convolution-pooling pairs build abstractions over features

Deep Learning over Sequences

- Chain or series like data structure
  - Units indexed in some order, e.g., words in a sentence, time steps in a time series, etc.



Different tasks on sequences

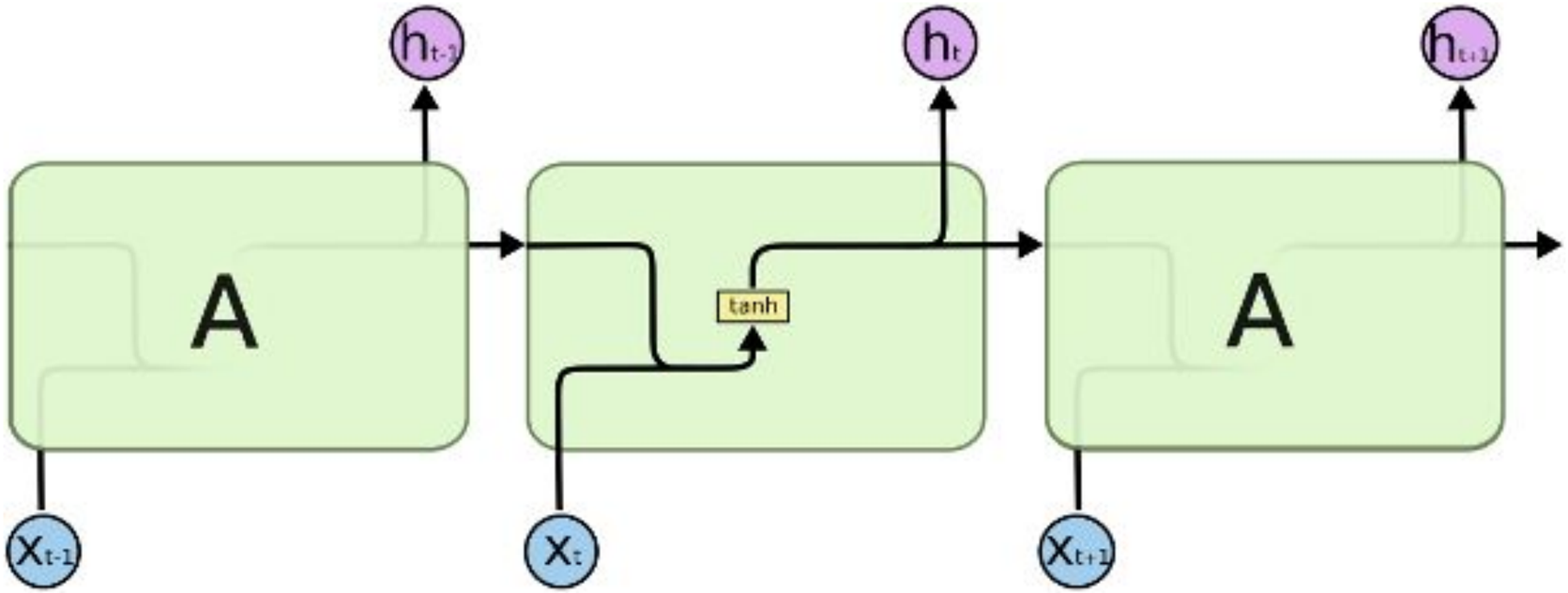
RNNs on Sequences

- Architectures up till now were all acyclic
  - Unidirectional flow of information from input to output layer, i.e., feed-forward
  - No notion of memory or internal state

- RNNs are cyclic
  - They use their own output as part of subsequent input
  - Maintain running hidden state (memory) to inform the next step
  - At each step, the output from before and the input are used to produce a new output

Inside an RNN

RNN cells are repeatedly applied to units of the input (Temporal Invariance)



The last output or outputs from all steps used downstream

Sequential input: $x_1, x_2, \ldots, x_t$ are units of the input

Important Takeaways

- RNNs sequentially process inputs of arbitrary lengths
  - Inherently imposing a direction on the units of the input
- In theory, RNNs have can have infinite Credit Assignment Path lengths
- They can model dependencies amongst the units of the input

Relational Inductive Bias in CNNs and RNNS

- Both CNNs and RNNs inherently express some relational inductive bias

- CNNs impose constraints on spatial locality and translation of units of inputs by design
  - Achieved via filters that have fixed receptive fields

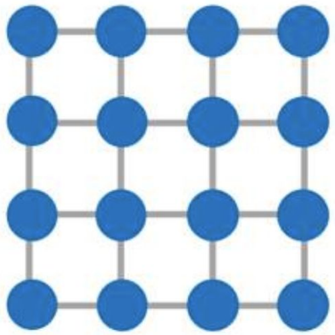
- RNNs impose constraints on temporal variance and sequentiality of units of inputs
  - Temporal invariance achieved by repeatedly applying the same model at different steps
  - Sequential dependency achieved by maintaining a running hidden state

03 The Unique Case of Graphs

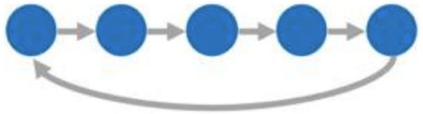
Graphs are irregular structures

Regular structures

Images

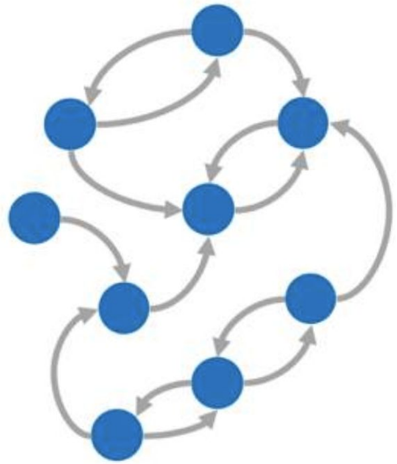


Time Series



Irregular structures

Social Networks
World Wide Web
Telecom Networks
Supply Chains
Biological Systems
Semantic Lexicons
Chemical Models
State Machines
Call Graphs
⋮



Some Notational Details

- A graph is denoted by the tuple (V, E) where V is the set of vertices and E is edges

- Each node is represented by a d-dimensional feature vector xu $\in \mathfrak{R}$d
  - Captures intrinsic properties of the node
  - At the simplest, xu can be a one-hot encoding, i.e., d = |V|
  - Edges can have features too but we can omit them for simplicity

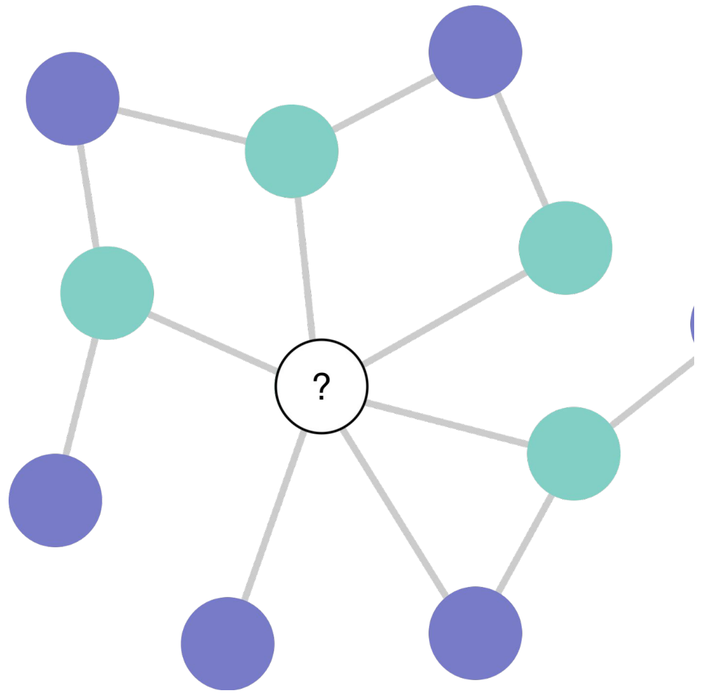
- The adjacency matrix of the graph is denoted by A

Value of Learning over Graphs

- A lot of data is inherently in the form of graphs
- Graphs allow for convenient depiction of various relationships
- They are the richest resource for relational inductive biases in the data

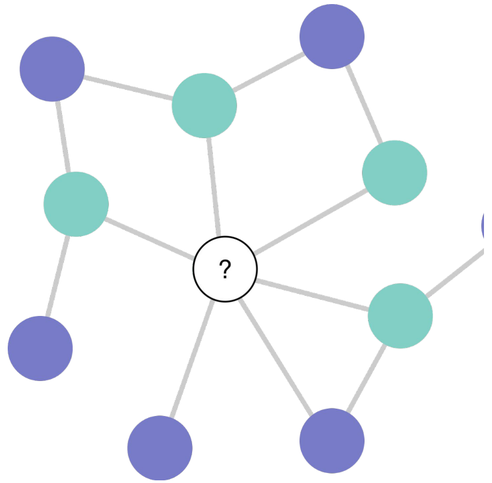
Deep Learning on Graphs

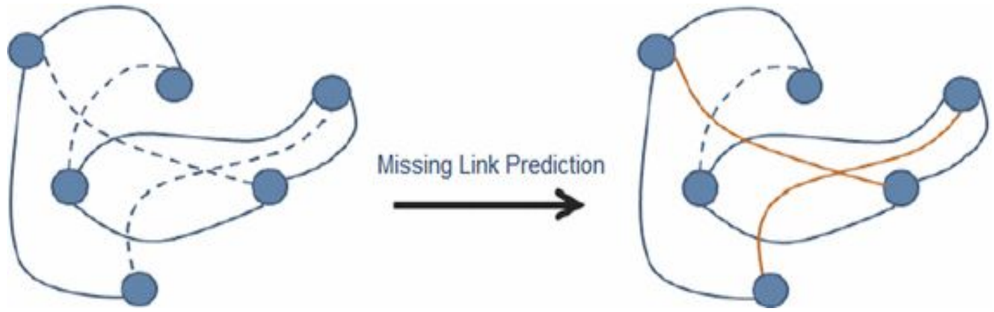
- We want to leverage the relationships that exist in the input to inform our hypothesis
- Here, deep learning allows us to model relational aspects end-to-end
- Reduces the need for feature-engineering as discussed before



A Variety of Tasks

- Classification/Regression of nodes or edges
- Classification/Regression on whole graphs
- Node/Edge embeddings
- Link Prediction





04 Transductive vs. Inductive

Transductive vs. Inductive Learning

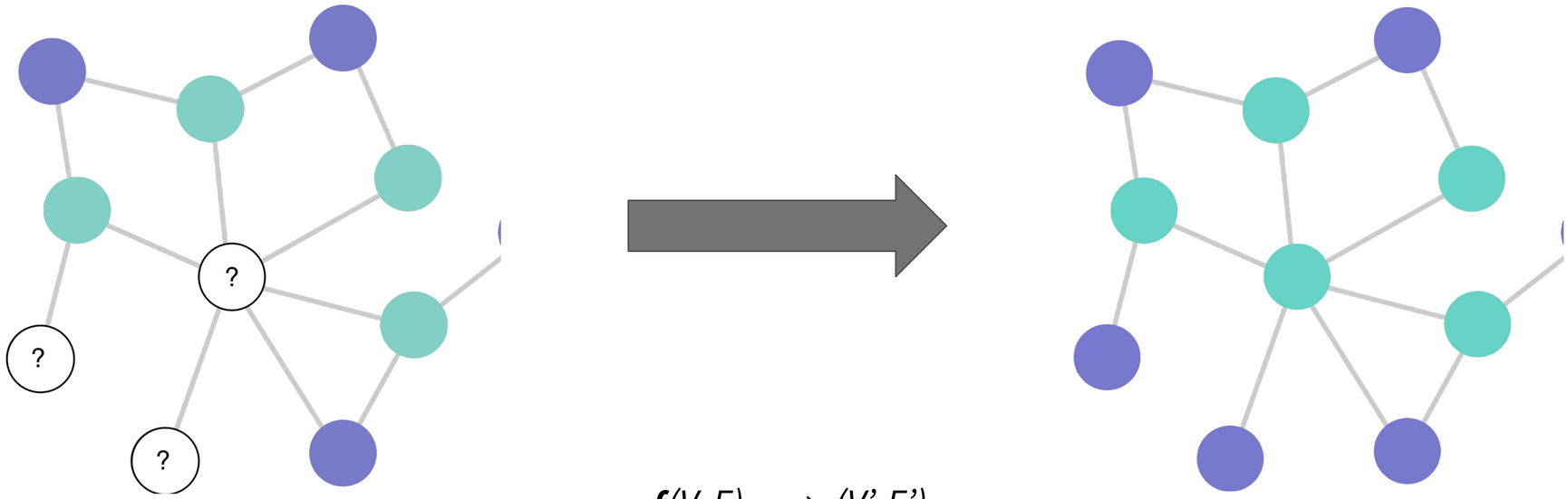
- Transductive learning assumes both train and test sets known beforehand
  - Learning algorithm predicts on test set while taking both the sets into account
  - If new data points come in, the learning process needs to be repeated in entirety

- Inductive learning uses the train set to generate a hypothesis
  - This can then be applied to unseen data

- Transductive learning does not create a model that can generalize to new data

Transduction on Graphs

The learning algorithm ***f*** transforms **the given graph** into representations that exhibit desired properties, e.g., structure of the graph or label information

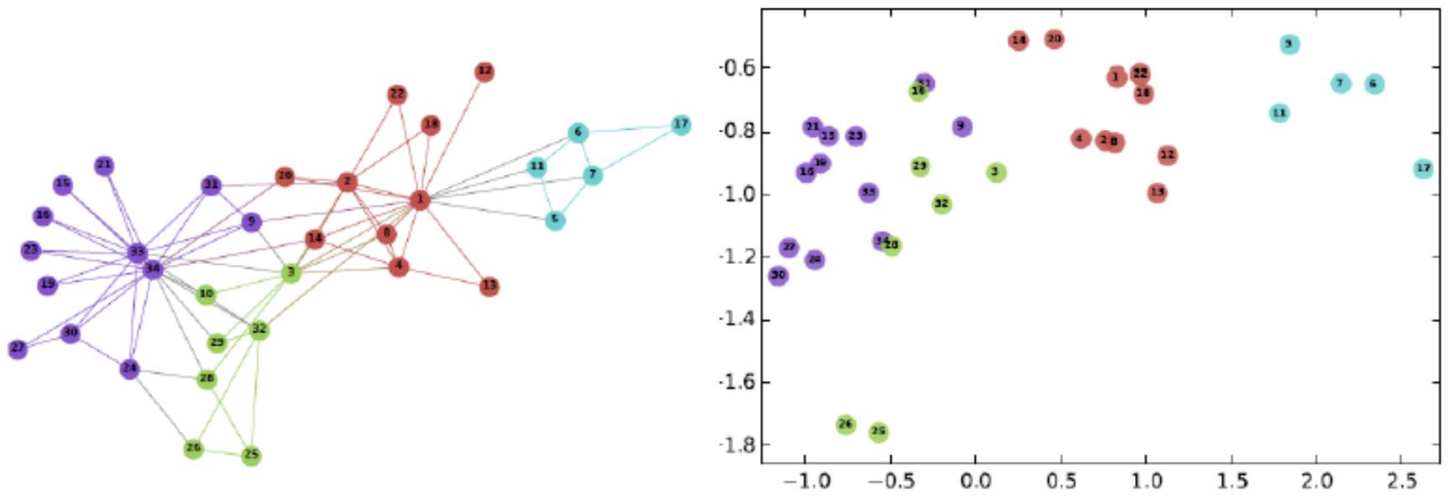


f*(V, E)* ----→ *(V', E')*

Examples of Transductive Learning

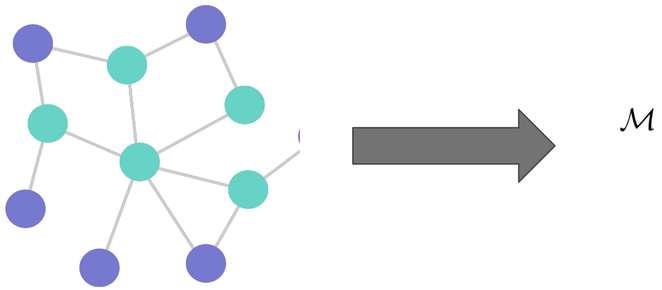
- Node embedding algorithms
  - DeepWalk
  - Node2vec

- No elegant way to embed new nodes apart from full retraining



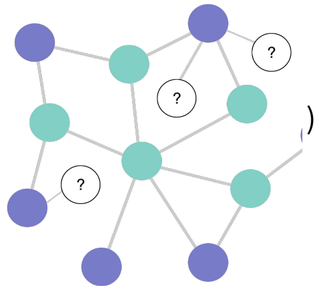
Induction on Graphs

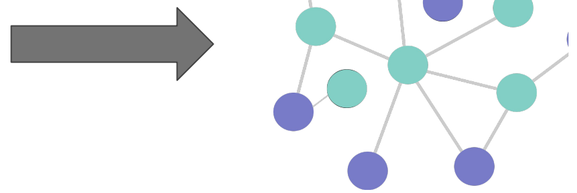
The aim is to learn a model $\mathcal{M}$ from the graph(s) in the train set and then apply it to new nodes or edges that appear in the graph(s) or on completely new graphs



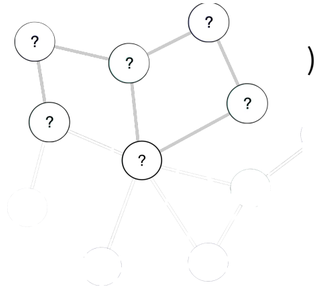
$\mathcal{M}$

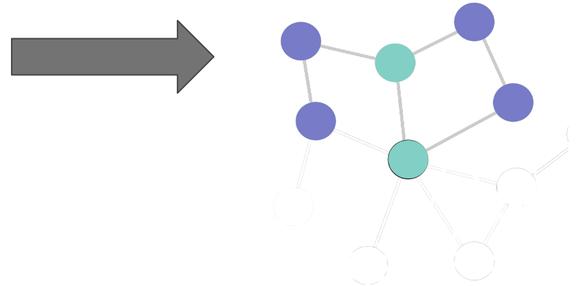
$\mathcal{M}($ $)$





$\mathcal{M}($ $)$





05 Graph Neural Networks

Convolutions on Graphs

- Convolution is a powerful operation for feature extraction
  - But it is not straightforward to extend it to irregular structures

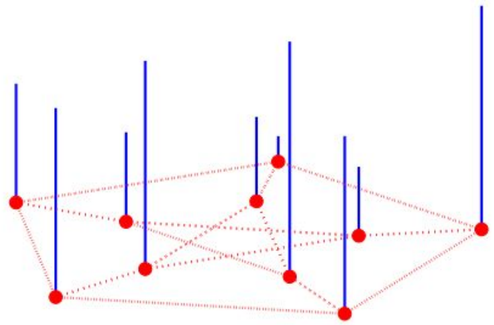
- Approaches to generalizing convolutions to graphs have followed two paths

Spectral
Defines convolution as application of a filter in the spectral domain

Spatial
Treats convolution as a patch operator applied to neighborhoods of nodes

Spectral Graph Convolutions

- View features on nodes as multi-channel signals
  - For each feature variable x, we can define a function
  - Sx: u —> r from nodes to reals



- Spectral convolutions are rooted in the theory of spectral analysis on graphs
  - Analysis of graph signals via spectral properties of the adjacency (or related) matrix of the graph

Defining Spectral Convolution

- Spectral convolution — application of filters to graph signals in the Fourier basis

- The Fourier basis for a graph is defined by the eigenvectors U of the Laplacian (Deg - A)
  - U = $[u_0, u_1, u_2, \ldots, u_{n-1}]$ form an orthonormal basis
  - Graph Fourier transform of a signal x is defined as $U^T x$

- Hence, the convolution operation of a signal x with a filter W is given by

$$x \star W = U(U^T x \,.\, U^T W)$$

Different Spectral GNNs

- Many different flavors exist
  - Graph Convolutional Networks (Kipf & Welling)
  - ChebNet (Defferrard et al.)
  - CayleyNet (Levie et al.)

- They mainly differ in how they approximate the filters
  - ChebNet uses Chebyshev polynomials to approximate localized and shared filters
  - GCN further simplifies the filters to make them localized to 1-hop

Limitations of the Spectral Formulation

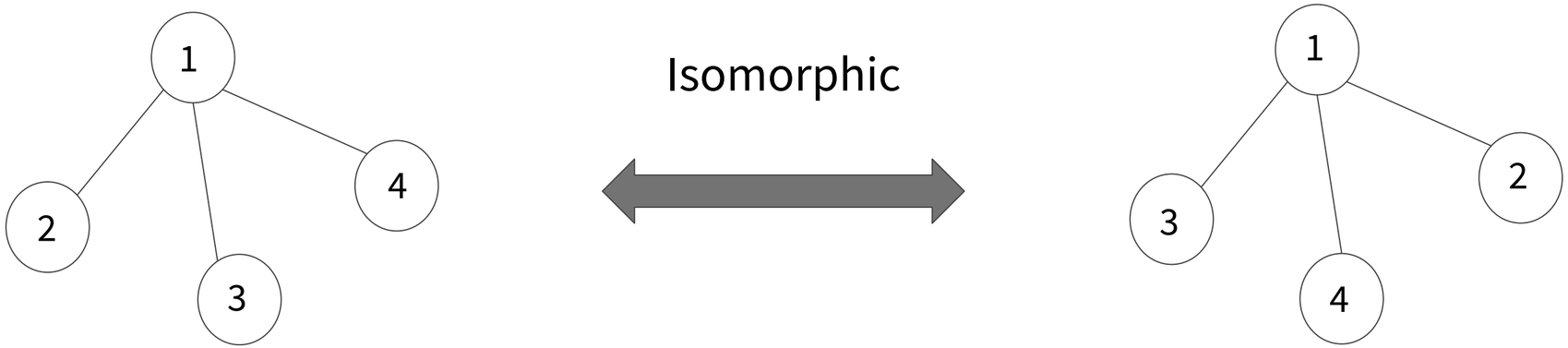
- The filters learnt in the process will act on the entire graph
  - Do not act on localized parts of the graph

- The learnt filters are domain dependent
  - Any perturbation to the graph results in change in eigenbasis

- Eigen-decomposition of the Laplacian is an expensive operation

Spatial Reasoning on Irregular Structures

- We need permutation invariance for spatial modeling on irregular structures
  - For example, there rarely exists a natural ordering of nodes in a graph
  - In such cases, the model should be agnostic to the ordering of nodes in the input



- Additionally, we would also want our model to be structure invariant, i.e., act equivalently on differing structures within the graph

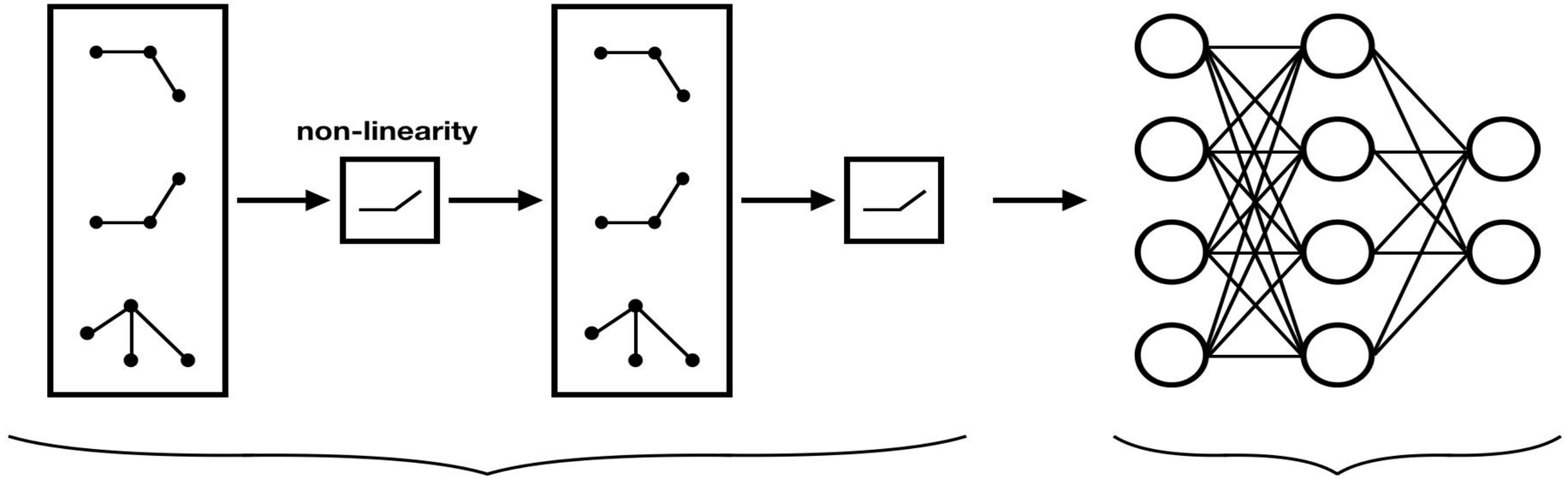
Sets: A Suitable Data Structure

- A set is an unordered collection of unique elements
  - Since there is no ordering, functions defined on sets are permutation invariant

- Pooling functions like max, sum, average are well-defined on all countable sets

- The neighborhood of a node u is a set N_u containing the vectors of it neighbors
  - A model operating on such sets is inherently permutation invariant
  - Can maintain structural invariance by having pooling operations

Spatial Graph Neural Network



Each layer computes for every node *v*:

$$f(x_v \,.\, g(N_v))$$

f : some function modeled with an MLP

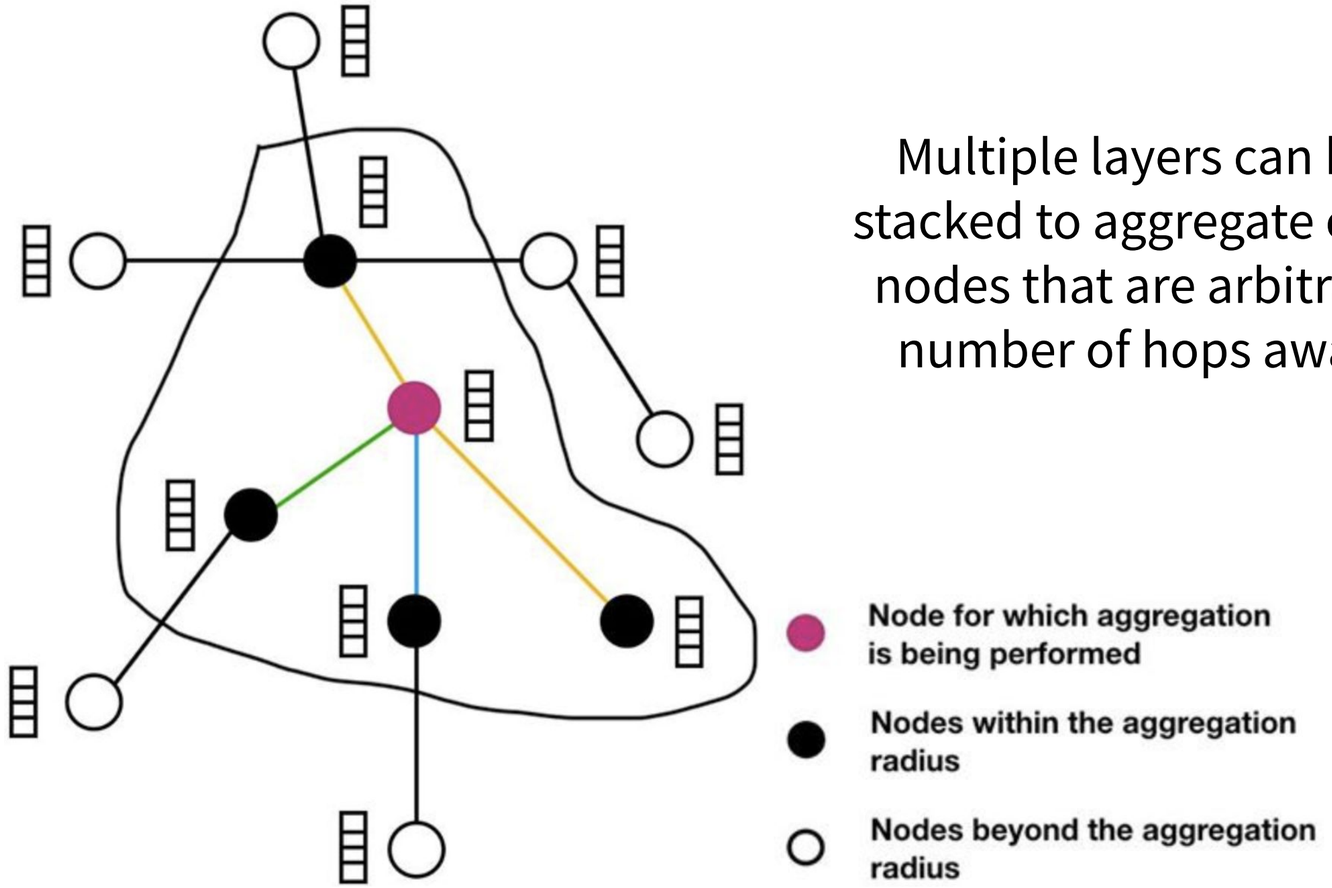
g : function for pooling neighbors' features

. : represents concatenation

These layers compute some task specific function over the generated embeddings

What Spatial GNNs Learn

Each layer of a spatial GNN learns some discriminative aggregator function that maps neighborhoods of nodes to vector representations



Multiple layers can be stacked to aggregate over nodes that are arbitrary number of hops away

Recapping Spectral vs. Spatial

- Spectral GNNs learn spectral filters
- These filters can potentially cover global context of the graph
- Transductive since the learnt filters are task and domain specific
- Spatial GNNs learn aggregator functions
- The functions are localized in space, and can be stacked
- Inductive in nature since they are not domain dependent

Summary

- GNNs are powerful structures for end-to-end deep learning on graphs
- From encoding nodes/edges to task-specific objectives, entirely in one architecture
- They can perform both inductive and transductive learning on graphs
- Impose strong relational inductive bias based on the structure of the underlying graph

Session 2:
Break + Q/A

Session 3: Agenda

01 Abusive Language & Misinformation detection

02 Question Answering and Reasoning

03 Dialogue Systems

04 Core NLP tasks

05 Takeaways and Summary

01 Abusive Language & Misinformation Detection

Traditional Text Classification Approaches

- Classify a given piece of text as abusive language or misinformation

- Featurize the text and feed to a classification model
  - Bag of words, TF-IDF
  - Embeddings-based approaches
  - Pretrained Language Models

What are we missing?

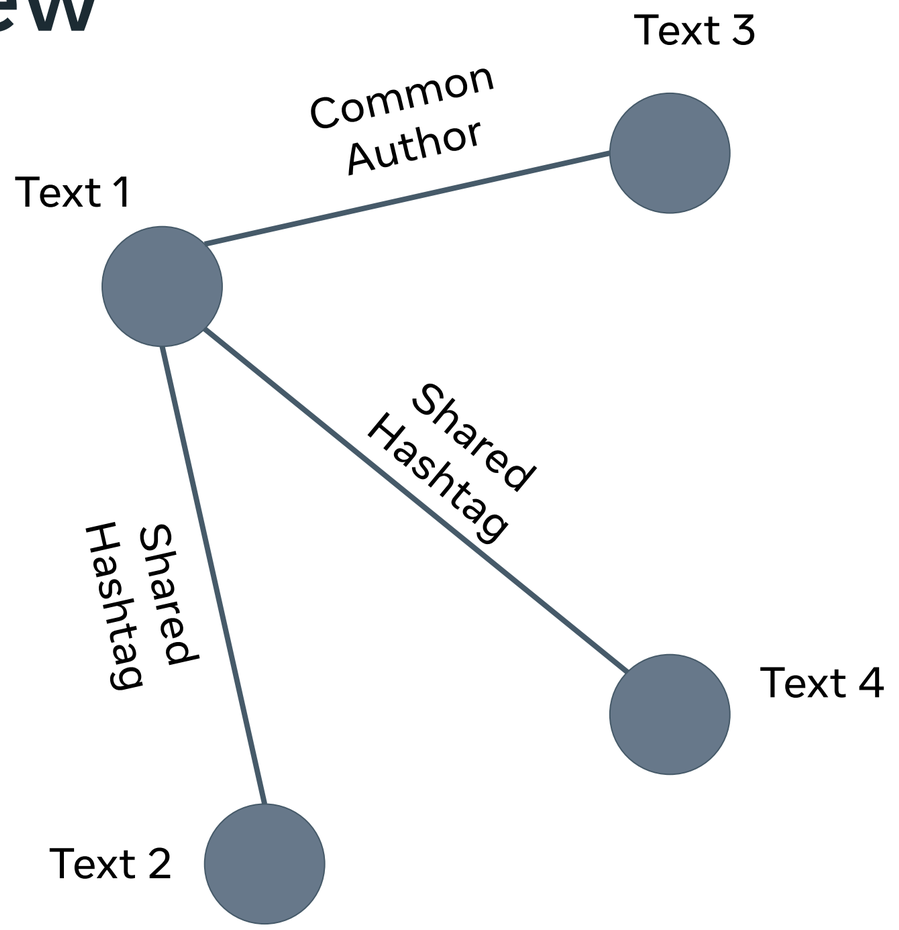
- Unable to capture important phenomena like *Linguistic Homophily*
  - The tendency of people in social space to forge ties with others based on traits
  - Linguistic homophily refers to ties based on similar linguistic traits

- Unable to model *Linguistic Variations*
  - A word may start being used as a novel slang or slur
  - Phrases may have different meanings depending on social context

Bringing in the relational view

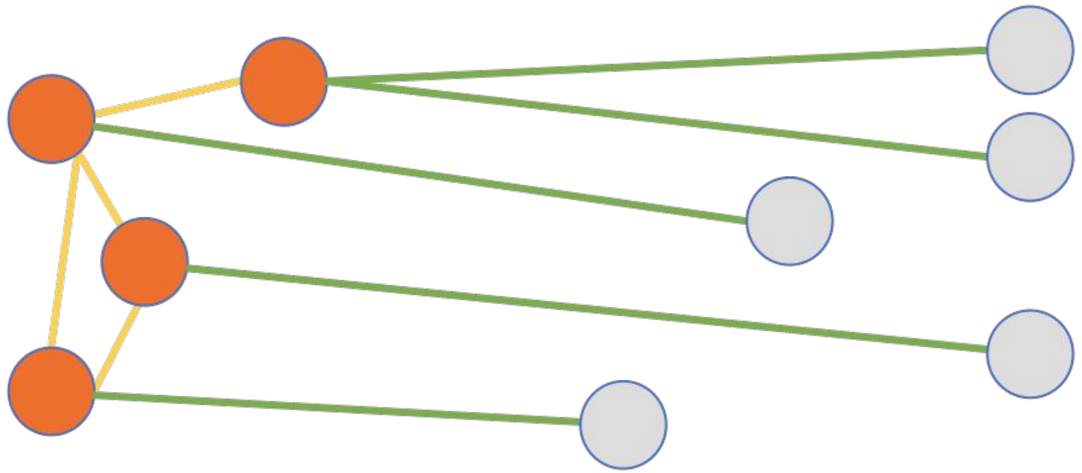
- Instead of classifying pieces of text in isolation, we leverage the relationships amongst them

- Many different ways to form a graph
  - Overlapping terms, e.g., hashtags
  - Shared authorship
  - A combination of different relations



An Example of Author-Tweet Graph

- Mishra et al. (NAACL 2019) created a graph of authors and tweets with two relationships:
  - Follower-following relationship between authors
  - Authorship of tweet



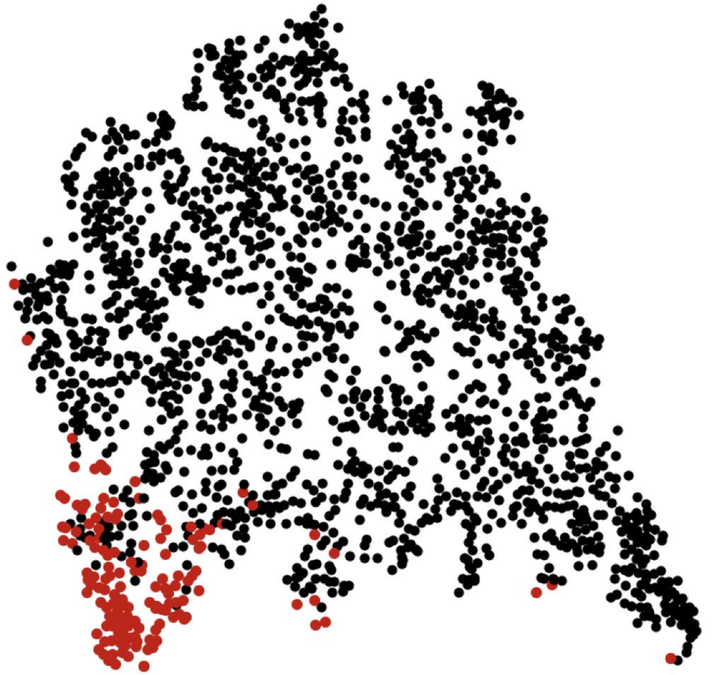
Tweet nodes can have one of the three labels: *sexist*, *racist*, or *none*

Tweet nodes

Author nodes

What we achieve

- Using a GCN model, we created embeddings (profiles) for authors
- These capture the linguistic behavior of authors and their respective communities
- We could classify abuse better based on social context from these profiles, e.g., sexism "*@Mich_McConnell Just "her body" right?"*



02 Question Answering and Reasoning

Traditional Approaches

- Information retrieval (IR) based methodology
  - The question or prompt is processed into a query
  - Documents retrieved for this query, either via embeddings or text match
  - Retrieved documents are processed to return the answer

- Multiple ways to process query and documents for matching
  - Bag of words, TF-IDF
  - Pretrained Language Models like BERT

What are we missing?

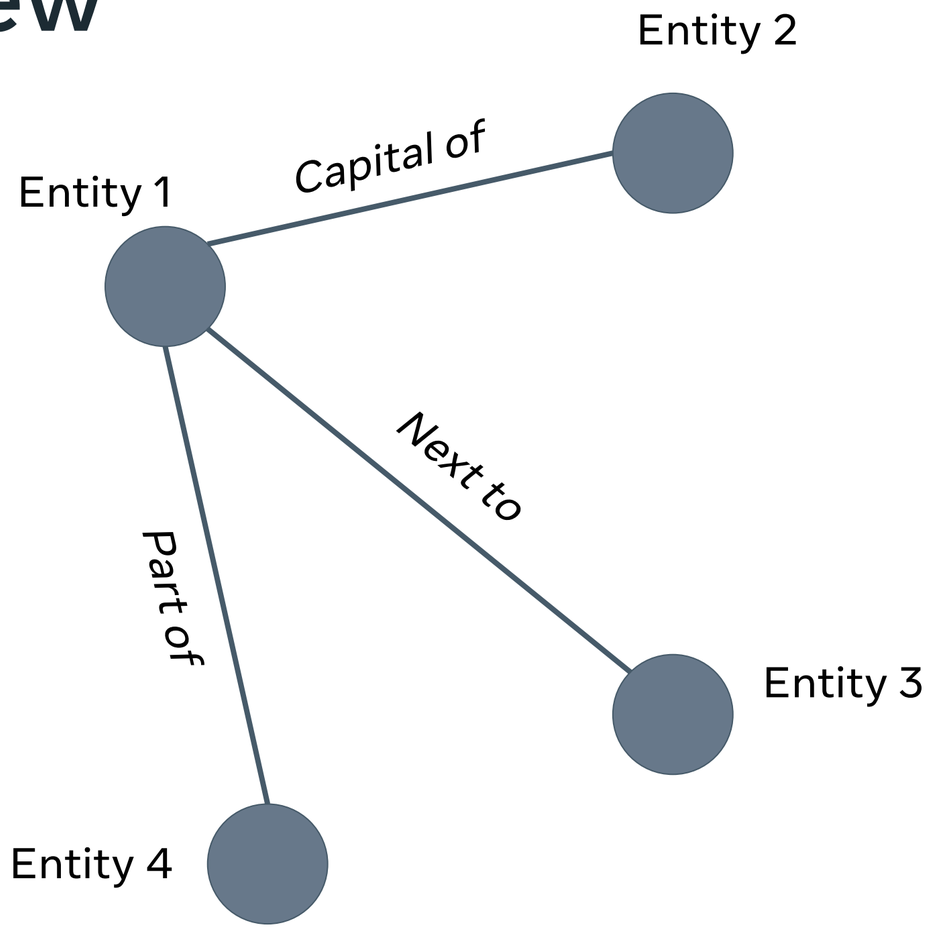
- Unable to perform multi-hop reasoning
  - *Is Amsterdam in Europe?*
  - Maybe we don’t have a document that states the fact
  - But maybe we have documents that state
    - The Netherlands is in Europe
    - Amsterdam is the capital of the Netherlands

- Unable to leverage known facts about entities in the question or prompt
  - *Is Jill Biden the first lady of the US?*
  - Knowing that Jill Biden is the wife of Joe Biden makes the question simpler to answer

Bringing in the relational view

- Instead of relying solely on IR, we can utilize Knowledge Bases (KBs) to leverage factual relationships amongst entities.

- Many different ways to form the KB, e.g.,
  - Homogenous graph of KB of entities only
  - Heterogenous graph of KB of entities and unstructured documents



What we achieve

- GNN models are used to create representations for entities that capture:
  - The particular entity itself
  - The neighborhood of that entity
  - Relevant information from the unstructured documents about the entity

- The question or prompt is transformed to a semantic graph of entities
  - This graph is embedded and scored against candidate answer subgraphs
  - The answer subgraphs are chosen based on entities in the question/prompt

03 Dialogue Systems

Traditional Approaches

- Vanilla methods for dialogue systems rely on sequence-to-sequence modeling

- Several different deep neural architectures have been tried
  - CNNs are employed to extract features from sliding window of dialogues
  - RNNs and their hierarchical variants are employed to do sequence modeling
  - Recently, transformers have been used with word and sentence-level attentions

Bringing in the relational view

- Dialog systems often rely on multiple different knowledge bases
  - Different views over the entities
  - Multiple KBs from different sources

- Entity alignment is an important task within dialogue systems
  - Essential to know two entities within different KBs refer to the same thing

What we achieve

- GNN models are used to create representations for entities that capture:
  - The particular entity itself
  - The neighborhood of that entity

- Entity alignment is done via metric learning
  - Minimize the distance between embeddings from different KBs

04 Core NLP Tasks

Some examples

- Syntactic Parsing
  - Aim is to parse a given piece of text to label dependency relationships

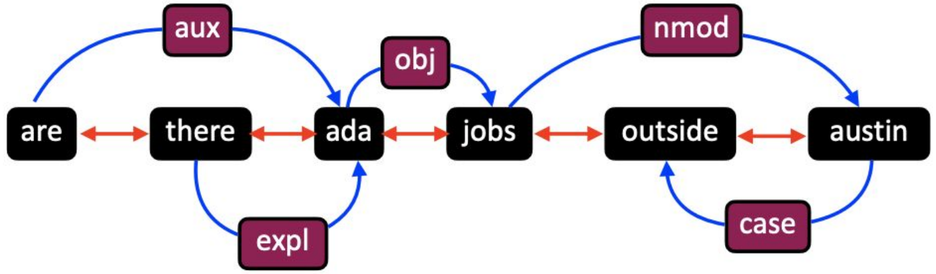


Image: Lingfei Wu, Yu Chen, et al.

- Semantic Parsing
  - Aim is to parse natural language to machine understandable grammar, e.g., free form queries to SQL

- Semantic Role labeling

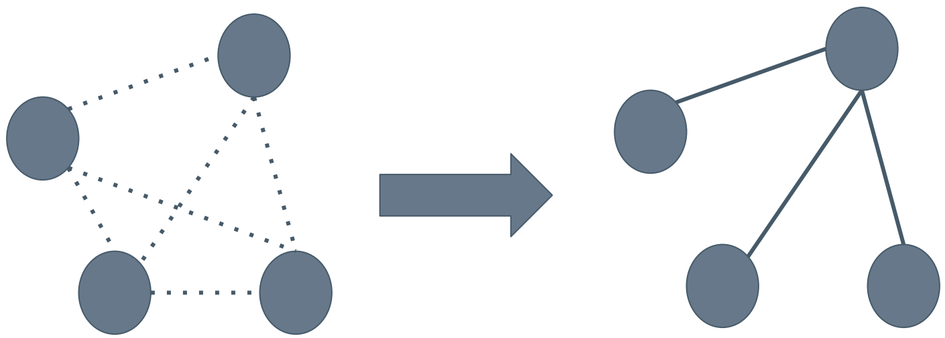
Traditional Approaches

- These tasks are traditionally solved using sequential models or parsers
- Typical deep learning architectures like BiLSTMs are employed
- Such approaches have limitations
  - Cannot incorporate higher order information
  - Long-range relationships are harder to model
  - Both syntax and semantics are not leveraged at the same time

Graph Learning for Parsing

- Given a piece of text, we can start with a fully connected graph
  - Imagine all input units have relationships with each other
  - For e.g., every pair of words in a sentence has a dependency relationship
  - Use models like GNNs in encoder-decoder framework to decode the final structure

- These tasks are examples of graph-to-graph modeling



Graph Learning for Semantic Role Labeling

- Strong interplay between synax and semantics in text

- GNNs can be used to inject information from syntax structures
  - For e.g., embeddings from dependency tree is used alongside word embeddings

- Allows for both syntax and semantics to be taken into account

05 Takeaways and Challenges

Takeaways

- NLP can benefit greatly from relational reasoning
  - Graph structures can be exploited across various NLP applications
  - Core language modeling tasks can also benefit from graph-based approaches

- When approaching a problem
  - Worth thinking about the relational inductive bias we can exploit
  - *What parts of the linguistic input share a relationship?*
  - *What are the latent relationships that could be exploited? Co-occurrence? Synonymy?*

Challenges

- Multi-relationality is crucial
  - Real-world NLP requires heterogenous graphs
  - Large number of different edge/node types can lead to very large models

- Heirarchical graph modeling
  - NLP problems may required heirarchical structures
  - Requires architectures that can model them end-to-end

- Dynamic graph construction for problems that need real-time state keeping

Session 4:
Q/A


Meta AI