Graph Neural Networks for NLP

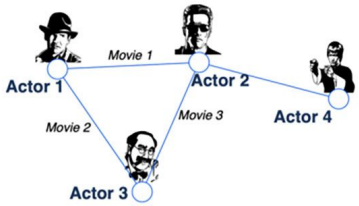
Pranik Chainani
December 17, 2021
CPSC 677

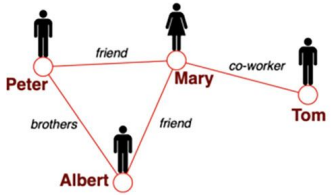
Layout

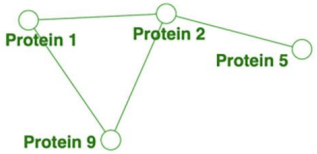
- Graph-induced Data
- Mathematics behind Graph Convolution Operators
- Methods and Overview of GNNs
- Implementing GCNs
- Applications
- CFGNN: Cross Flow Graph Neural Networks for Question Answering on Complex Tables
- GRAND: Graph Neural Diffusion
- Message Passing Attention Networks for Document Understanding
- Learning to Represent Image and Text with Denotation Graph

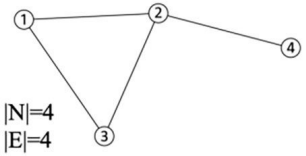
Goals

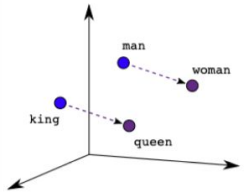
- Develop a solid understanding of spectral theory and GCN operators
- Understand an intuitive and robust appreciation for Deep GNNs
    - Why they are essential in numerous NLP applications
- Gain exposure to novel applications of GNNs in state-of-the-art NLP model architecture



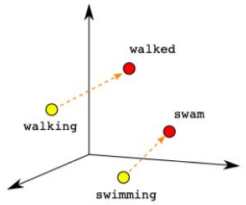




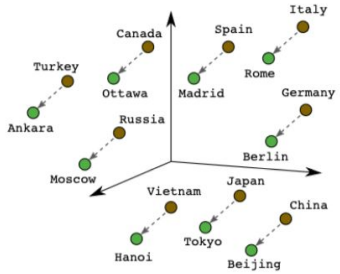




Male-Female

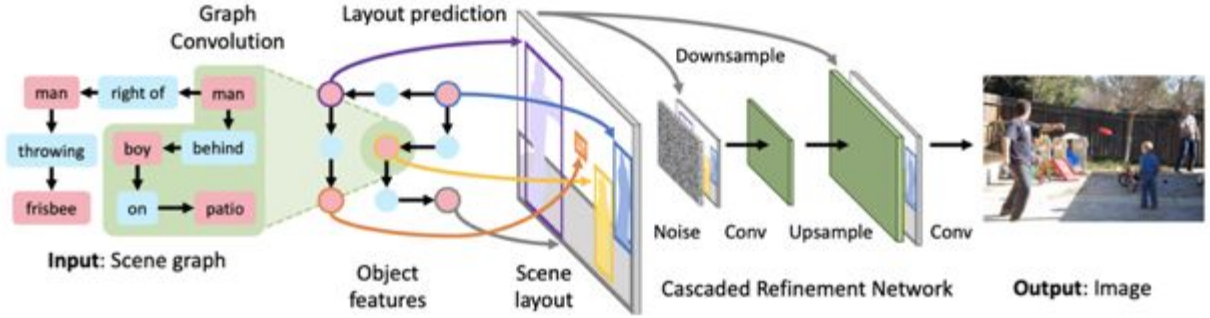


Verb Tense



Country-Capital

Image Source: (Embeddings: Translating to a Lower-Dimensional Space) by Google.



GNNs are powerful and NATURAL

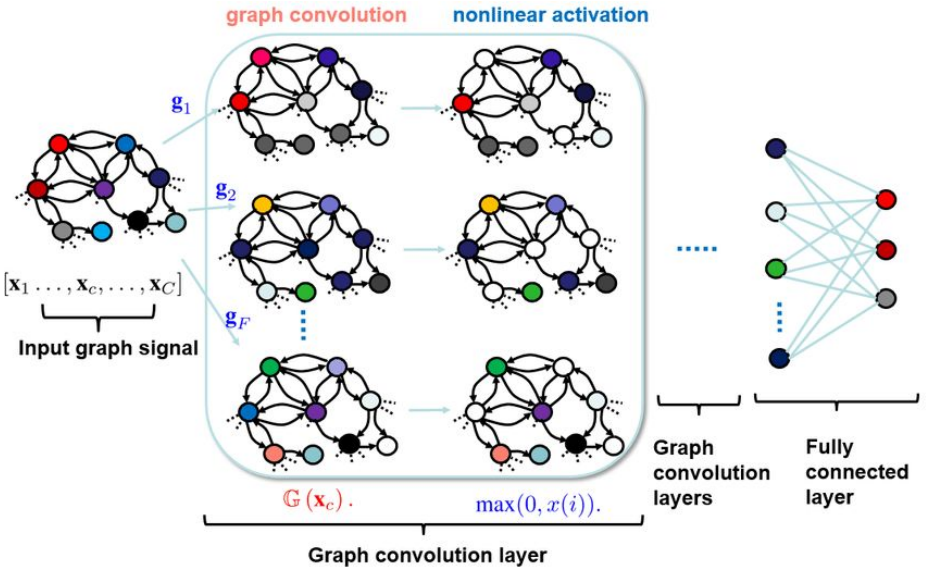
- GNNs learn embeddings (nodes, graphs etc)
- GNNs can handle relations beyond similarity
- Implicit regularization
- GNNs tend to be more effective

GCNs are intrinsic spectral operators

- Graph Convolution operators to learn optimal low pass filtering
  - Harmonic Analysis interpretation
- Chebyshev Interpolation
  - Chebyshev nodes
  - Parameter orthogonality
  - Recursive formulation for fast filtering
  - Coefficients of Chebyshev polynomials K=2
-

Takeaways from Convolutional Neural Networks

- Feature Extraction
  - GNNs respect graph invariant structure
- Translation Invariance
- Localized filters in space

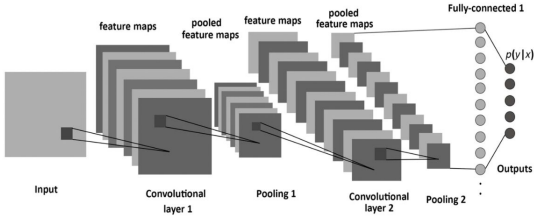


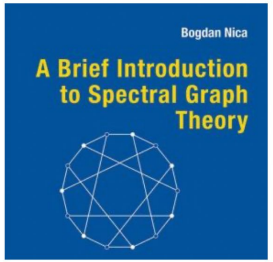
GCN Formulation (natural in multiple learning settings)

Nodes == Words ⟶ Word2vec

Embeddings Nodes == Authors ⟶ 0/1 value indicating frequently used keywords

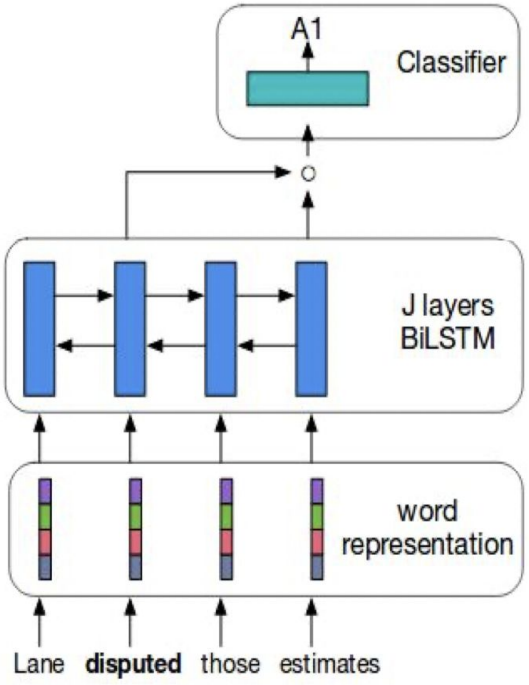
If No features ⟶ One-hot vector (length = #Nodes)

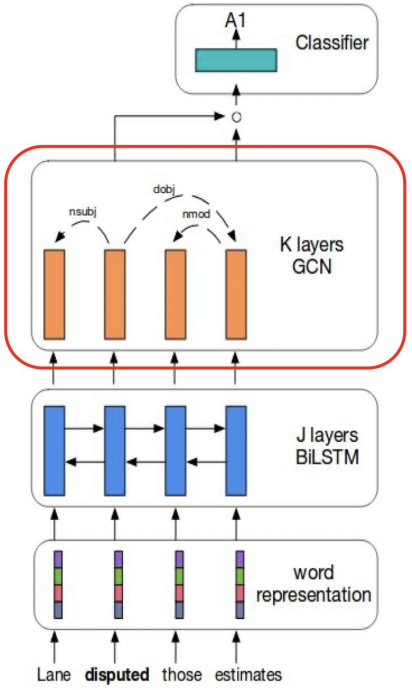




$$h_v = f\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} W x_u + b\right), \quad \forall v \in \mathcal{V}.$$

Standard Deep Learning Architecture for NLP Problems

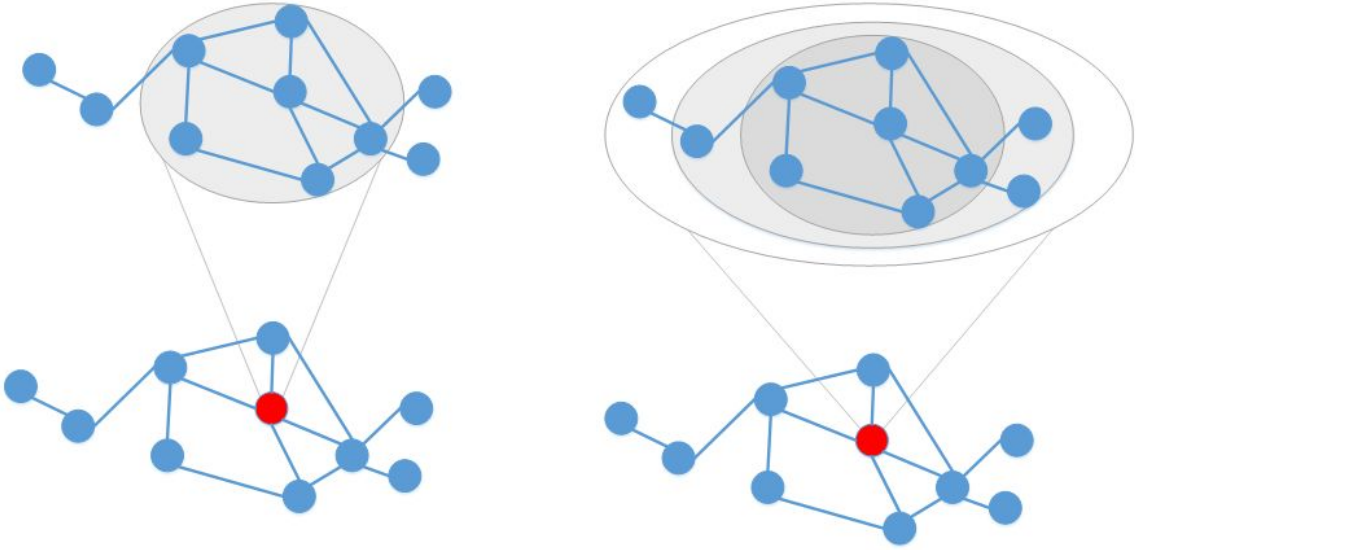




Model with GCN as part of the network

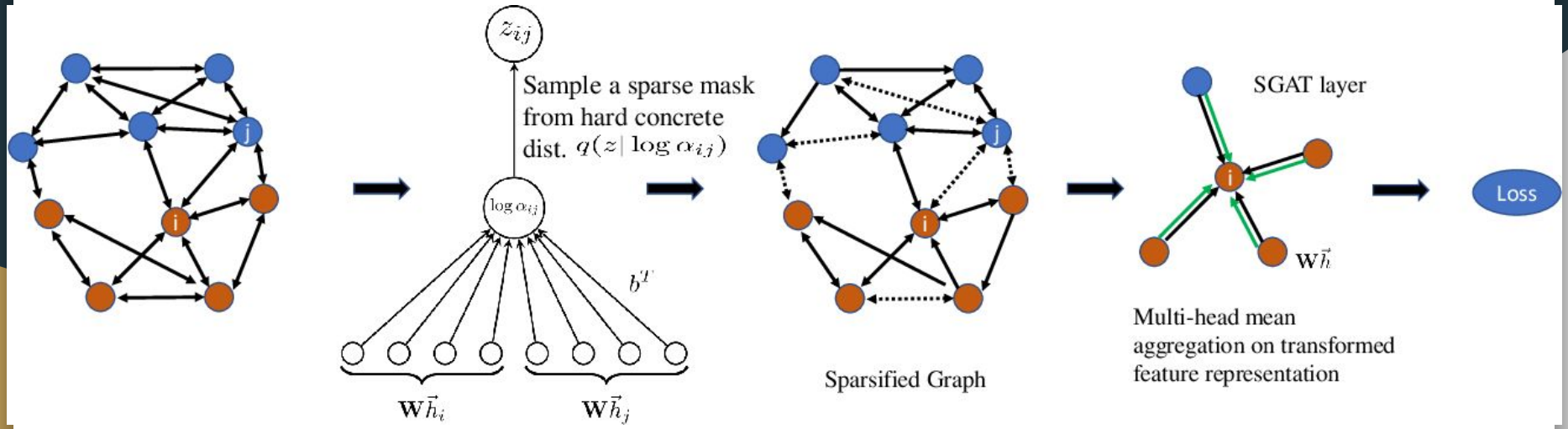
GCN weights are trained based on the final objective

Multi-Hop Aggregation



（a）Original GCN　　（b）Our proposed higher-order GCN

A Sparse scheme for Graph Attention Networks



CFGNN: Cross Flow Graph Neural Networks for Question Answering on Complex Tables

By Xuanyu Zhang, April 2020

https://www.researchgate.net/publication/342239318_CFGNN_Cross_Flow_Graph_Neural_Networks_for_Question_Answering_on_Complex_Tables

Proceedings of the AAAI Conference on Artificial Intelligence

CFGNN Overview

Scheme for Question answering on complex tables

Most of GNNs ignore the relationship of sibling nodes and use summation as aggregation function to model the relationship of parent-child nodes.

Information flow mechanism of parent-child and sibling nodes cross with history states between different layers.

Utilizes hierarchical encoding layer to obtain contextualized representation in tables.

Latent vector field perspective on RNN flow embeddings

Child-to-child flow with RNNs

as aggreggation

Generate cross flows with attention

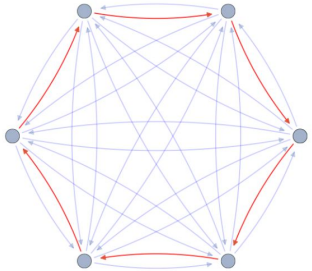
mechanism

$V =$

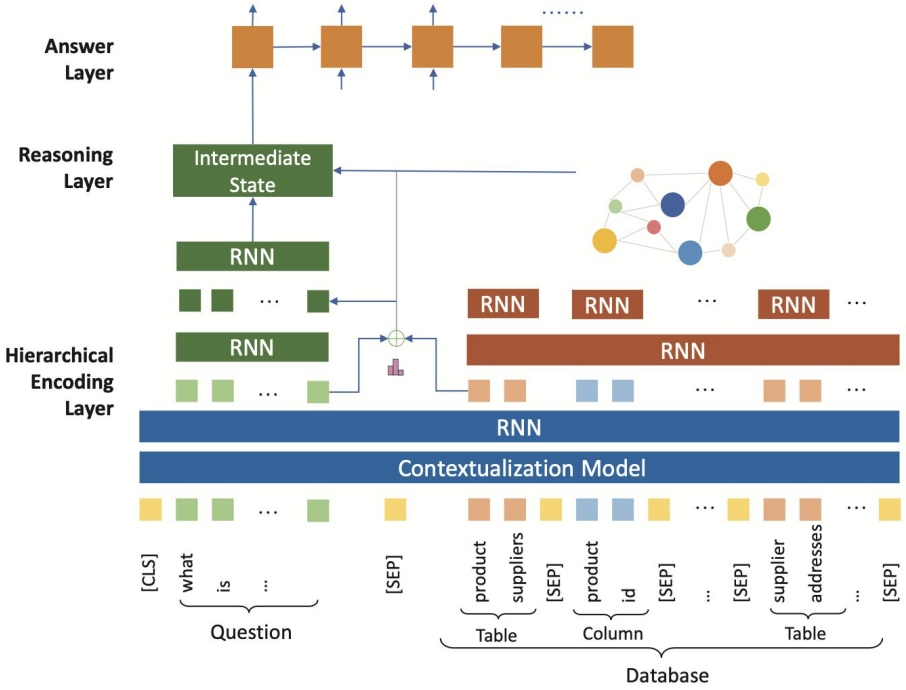
0.0	1.0	-0.25	-0.25	-0.25	-0.25
-0.25	0.0	1.0	-0.25	-0.25	-0.25
-0.25	-0.25	0.0	1.0	-0.25	-0.25
-0.25	-0.25	-0.25	0.0	1.0	-0.25
-0.25	-0.25	-0.25	-0.25	0.0	1.0
1.0	-0.25	-0.25	-0.25	-0.25	0.0

$\mathcal{D}_V =$

0.00	-1.25	0.00	0.00	0.00	1.25
1.25	0.00	-1.25	0.00	0.00	0.00
0.00	1.25	0.00	-1.25	0.00	0.00
0.00	0.00	1.25	0.00	-1.25	0.00
0.00	0.00	0.00	1.25	0.00	-1.25
-1.25	0.00	0.00	0.00	1.25	0.00



Complete Model Architecture



Applications

Develop and understand interactive/complex relationships in SQL format

Code generation

Results and Ablation Study

Question:
What is the id and type code for the template used by the most documents?
Return the id and type code of the template that is used for the greatest number of documents.

Schema:
templates: template_id, template_type_code ...
documents: document_id, template_id, document_name ...
paragraphs: paragraph_id, document_id, paragraph_text ...

Gold Answer / CFGNN:
SELECT documents.template_id, templates.template_type_code FROM documents JOIN templates ON documents.template_id = templates.template_id GROUP BY documents.template_id ORDER BY count (*) DESC LIMIT 1

Schema GNN:
SELECT templates.template_id, templates.template_type_code FROM templates GROUP BY templates.template_id ORDER BY count (*) DESC LIMIT 1

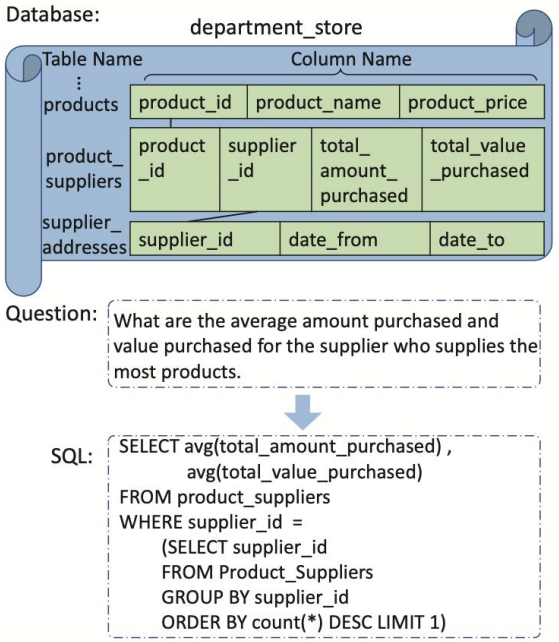


Figure 1: An example from the Spider dataset.

Model	Test					Dev
	Easy	Medium	Hard	Extra Hard	All	All
Seq2Seq[1]	22.0%	7.8%	5.5%	1.3%	9.4%	1.9%
Seq2Seq+Attention	32.3%	15.6%	10.3%	2.3%	15.9%	1.8%
Seq2Seq+Copying	29.3%	13.1%	8.8%	3.0%	14.1%	4.1%
SQLNet[2]	34.1%	19.6%	11.7%	3.3%	18.3%	10.9%
TypeSQL[3]	47.5%	38.4%	24.1%	14.4%	33.0%	8.0%
SyntaxSQLNet[4]	48.0%	27.0%	24.3%	4.6%	27.2%	24.8%
RCSQL[5]	-	-	-	-	24.3%	28.8%
Schema GNN[6]	61.8%	44.7%	26.5%	14.6%	39.4%	40.7%
CFGNN (ours)	**67.0%**	**47.8%**	**33.8%**	**19.1%**	**44.1%**	**48.7%**

Table 1: Accuracy of Exact Matching on SQL with different hardness levels. These models are: Dong and Lapata (2016) [1], Yu et al. (2018b) [2], Yu et al. (2018a) [3], Yu et al. (2018b) [4], Lee (2019) [5], Bogin, Berant, and Gardner (2019) [6]

Method	SELECT	WHERE	GROUP BY	ORDER BY	KEYWORDS
Seq2Seq	13.0%	1.5%	3.3%	5.3%	8.7%
Seq2Seq+Attention	13.6%	3.1%	3.6%	9.9%	9.9%
Seq2Seq+Copying	12.0%	3.1%	5.3%	5.8%	7.3%
SQLNet	44.5%	19.8%	29.5%	48.8%	64.0%
TypeSQL	36.4%	16.0%	17.2%	47.7%	66.2%
SyntaxSQLNet	62.5%	34.8%	55.6%	60.9%	69.6%
RCSQL	68.7%	39.0%	63.1%	63.5%	76.5%
Schema GNN	80.9%	40.7%	67.4%	70.1%	77.0%
CFGNN (ours)	**81.1%**	**42.9%**	**73.0%**	**73.7%**	**80.6%**

Table 2: F1 scores (test set) of Component Matching on SQL.

Questions

GRAND: Graph Neural Diffusion

By Benjamin Paul Chamberlain, James Rowbottom, Maria Gorinova, Stefan Webb, Emanuele Rossi, Michael M. Bronstein, Jun 2021

https://arxiv.org/abs/2106.10934

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

Graph Neural Diffusion

Generalized approach to deep learning on graphs as a continuous diffusion process and

GNNs are viewed as discretizations of an underlying PDE.

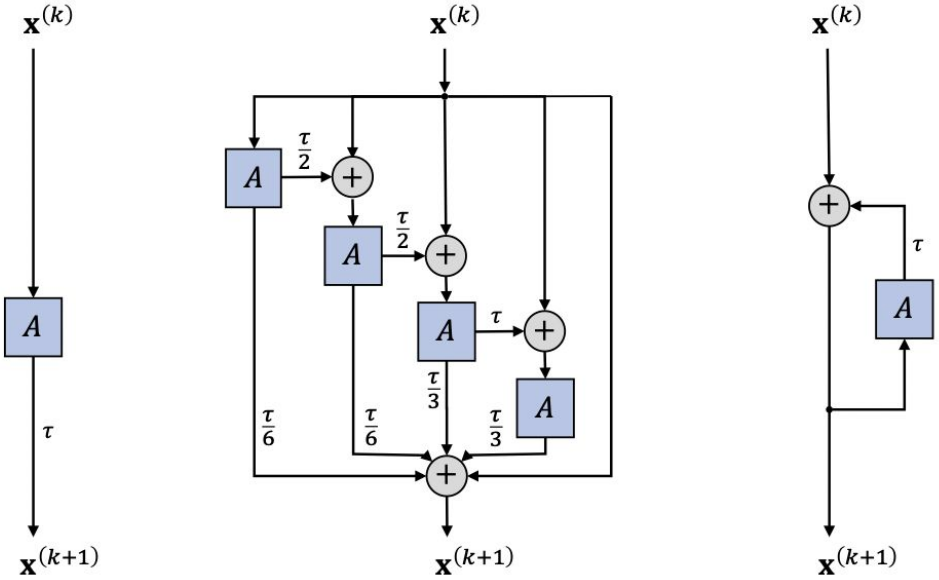
- layer structure and topology correspond to the discretisation choices of temporal and spatial operators.
- Scheme to address depth, oversmoothing, and bottlenecks.

Demonstration of robust model stability with respect to perturbations in data

Implicit Schemes for Diffusion Operators (PDEs)

- Euler
- Implicit Euler
- 4th order Runge-Kutta

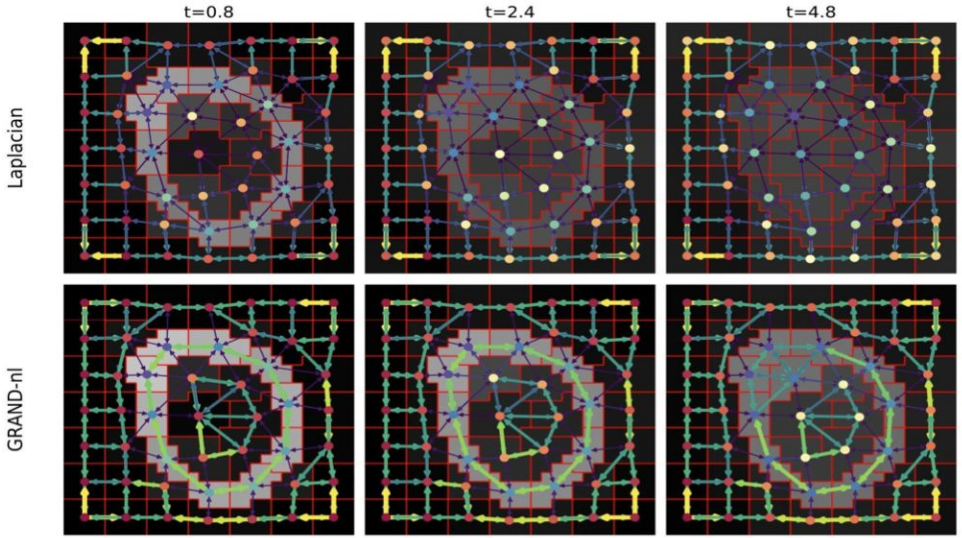
The heat diffusion equation is a parabolic PDE, with temporal and spatial components in which the diffusion operator inherits the natural structure of adjacency of the input graph.



Multi-scale linear diffusion

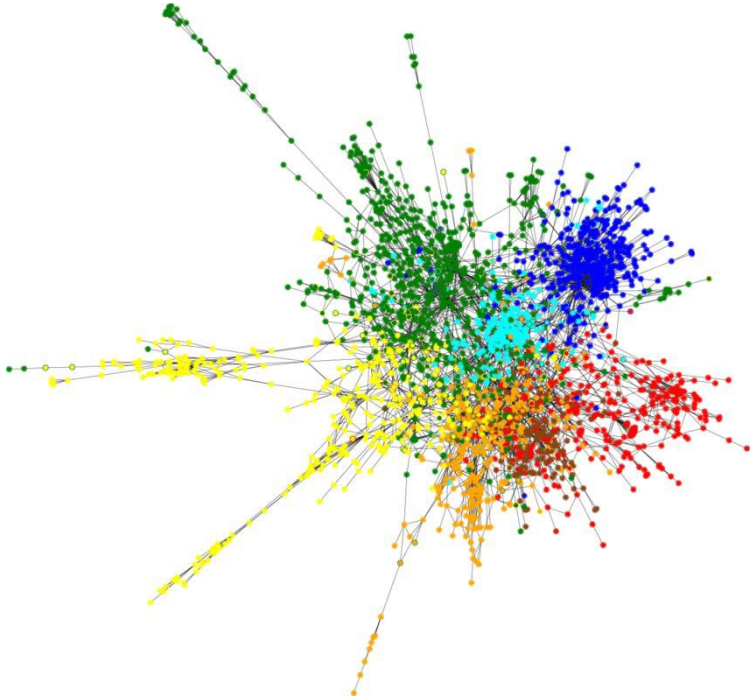
Flow scheme that demonstrators evolution of “heat” dissipation

The diffusivity is modelled with an attention function a(., .)



Citation graph framework dataset

Research citations exhibit a natural embedding on a smooth Riemannian manifold



Questions

In traditional GNNs there is a linear relationship between the number of parameters and depth. Conversely, GRAND *shares parameters across layers*

Planetoid splits	**CORA**	**CiteSeer**	**PubMed**
GCN	81.9 ± 0.8	69.5 ± 0.9	79.0 ± 0.5
GAT	82.8 ± 0.5	71.0 ± 0.6	77.0 ± 1.3
MoNet	82.2 ± 0.7	70.0 ± 0.6	77.7 ± 0.6
GS-maxpool	77.4 ± 1.0	67.0 ± 1.0	76.6 ± 0.8
Lanczos	79.5±1.8	66.2±1.9	78.3±0.3
AdaLanczos	80.4±1.1	68.7±1.0	78.1±0.4
CGNN†	81.7 ± 0.7	68.1 ± 1.2	**80.2 ± 0.3**
GDE*	**83.8 ± 0.5**	**72.5 ± 0.5**	79.9 ± 0.3
GODE*	83.3 ± 0.3	72.4 ± 0.6	80.1 ± 0.3
GRAND-l (ours)	**84.7 ± 0.6**	**73.3 ± 0.4**	**80.4 ± 0.4**
GRAND-nl (ours)	**83.6 ± 0.5**	70.8 ± 1.1	79.7 ± 0.3
GRAND-nl-rw (ours)	82.9 ± 0.7	**73.6 ± 0.3**	**81.0 ± 0.4**

Message Passing Attention Networks for Document Understanding

By Giannis Nikolentzos, Antoine J.-P. Tixier, Michalis Vazirgiannis, Aug 2019

https://arxiv.org/abs/1908.06267

Accepted at AAAI'20

Learning to Represent Image and Text with Denotation Graphs

Utilization of denotation graphs to represent how specific concepts (such as sentences describing images) can be linked to abstract and generic concepts (such as short phrases) that are also visually grounded.

Motivation: Demonstrate that state-of-the-art multimodal learning models can be further improved by leveraging automatically harvested structural relations seen in graphs. These representations lead to stronger empirical results on downstream tasks of:

- cross-modal image retrieval
- referring expression
- compositional attribute-object recognition.

Simple Example of Learning with Denotation Graphs

A denotation graph G simply is a tree where a node v_i in the graph corresponds to a pair of a linguistic expression y_i and a set of images $X_i = \{x_1, x_2, \cdots, x_{n_i}\}$.

A directed edge e_{ij} from a node v_i to its child v_j represents a **subsumption** relation between y_i and y_j.

Semantically, y_i is more abstract (generic) than y_j, and the tokens in y_i can be a subset of y_j's.

For example, TWO DOGS describes all the images which TWO DOGS ARE RUNNING describes, though less specifically.

Concatenation of Image and Text

A given image x and the respective text y are represented by (by a set of) vectors $\varphi(x)$ and $\psi(y)$ respectively.

The author's utilize for $\varphi(\cdot)$ is the last layer of a convolutional neural network and for $\psi(\cdot)$ the contextualized word embeddings from a Transformer network (Vaswani et al., 2017).

The embedding at the end ultimately is the *multimodal* pair as a direct product over $\varphi(x)$ and $\psi(y)$.

Easy to intuitively describe contrastive learning with graphs

Visually mismatched pair - the idea is to randomly sample an image $x_- \in / X_i$ to pair with text y_i, *i.e.*, (x_-, y_i). Note that there is an automatic exclusion of images from v_i's children.

Semantically mismatched pair - the behind this is to randomly sample a text this time $y_{j\sim} \neq y_i$ to form the pair (x_{ik}, y_j). Interestingly, the authors in the paper constrain y_j not to include concepts that could be more abstract than y_i as the more abstract can certainly be used to describe the specific images x_{ik}. Subsumption!!!

Semantically hard pair - the idea behind this is to randomly sample a text y_j that corresponds to an image x_j that is *visually* similar to x_{ik} to form (x_{ik}, y_j).

DG Hard Negatives - the idea is to fully utilize random samples from *sibing* (but not cousin) node v_j to v_i such that $x_{ik} \in / X_j$ to form (x_{ik}, y_j)

Improve image quality by predicting edges

Specifically, for a pair of nodes v_i and v_j, we want to predict whether there is an edge from v_i to v_j , based on each node's corresponding embedding of a pair of image and text. Concretely, this is achieved by minimizing a specific log likelihood.

$$\ell_{\text{EDGE}} = -\sum_{e_{ij}} \sum_{k,k'} \log p(e_{ij} = 1 | \boldsymbol{v}(\boldsymbol{x}_{ik}, \boldsymbol{y}_i), \boldsymbol{v}(\boldsymbol{x}_{jk'}, \boldsymbol{y}_j))$$

Learning Loss

$$l_{DG} = l_{MATCH} + \lambda_1 \cdot l_{SPEC} + \lambda_2 \cdot l_{EDGE}$$

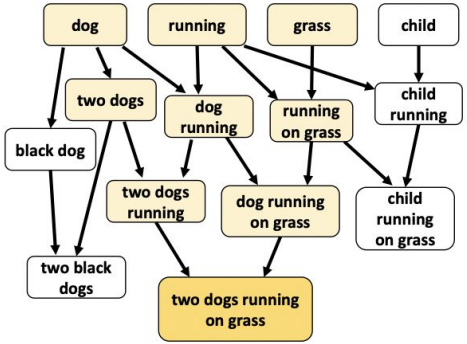
Method	R@1	R@5	R@10	RSUM
FLICKR30K				
ViLBERT	59.1	85.7	92.0	236.7
ViLBERT + DG	63.8	87.3	92.2	243.3
UNITER	62.9	87.2	92.7	242.8
UNITER + DG	66.4	88.2	92.2	246.8
COCO 1K Test Split				
ViLBERT	62.3	89.5	95.0	246.8
ViLBERT + DG	65.9	91.4	95.5	252.7
UNITER	60.7	88.0	93.8	242.5
UNITER + DG	62.7	88.8	94.4	245.9
COCO 5K Test Split				
ViLBERT	38.6	68.2	79.0	185.7
ViLBERT + DG	41.8	71.5	81.5	194.8
UNITER	37.8	67.3	78.0	183.1
UNITER + DG	39.1	68.0	78.3	185.4

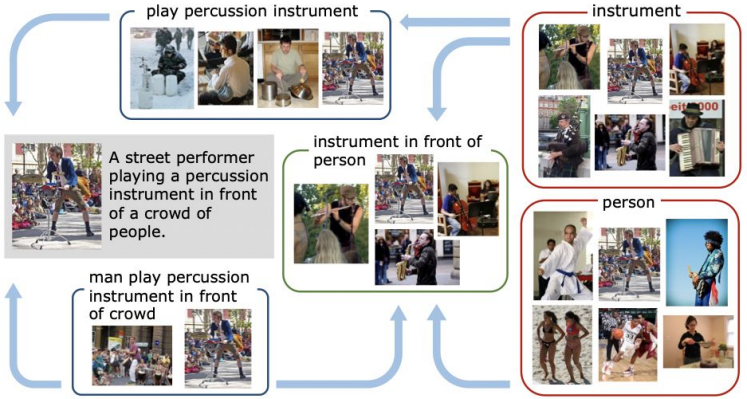
Method	R@1	R@5	R@10	RSUM
FLICKR30K				
ViLBERT	76.8	93.7	97.6	268.1
ViLBERT + DG	77.0	93.0	95.0	265.0
UNITER	78.3	93.3	96.5	268.1
UNITER + DG	78.2	93.0	95.9	267.1
COCO 1K Test Split				
ViLBERT	77.0	94.1	97.2	268.3
ViLBERT + DG	79.0	96.2	98.6	273.8
UNITER	74.4	93.9	97.1	265.4
UNITER + DG	77.7	95.0	97.5	270.2
COCO 5K Test Split				
ViLBERT	53.5	79.7	87.9	221.1
ViLBERT + DG	57.5	84.0	90.1	232.2
UNITER	52.8	79.7	87.8	220.3
UNITER + DG	51.4	78.7	87.0	217.1

Interestingly, Flicker provides a new benchmark for localization of textual entity mentions in an image.

detectors for common objects, a color classifier, and a bias towards selecting larger objects.

Denotation Graphs





Denotation Graph Scheme

- Matching Texts with Images
- Negative Sampling
  - Visually mismatched pairs
  - Semantically mismatches pairs

Results and Ablation Studies

- Zero/Few shot and Transfer learning
  - Transfer across datasets
- Compositional Attribute-Object Recognition
- Image Retrieval from Abstract Concepts

Questions

Learning to Represent Image and Text with Denotation Graph

By Bowen Zhang, Hexian Hu, Vihan Jain, Eugene le, Fei Sha, Nov 2020

https://aclanthology.org/2020.emnlp-main.60/

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)

Message Passing Attention Networks for Document Understanding

Most graph neural networks can be described in terms of:

- message passing,
- vertex update
- readout functions.

Represent documents as word co-occurrence networks.

A document is viewed as a statistical word co-occurrence network with a sliding window of size 2 overspanning sentences.

Each unique word in a given preprocessed document is represented by a node in graph G, and an edge is added between two nodes if they are found together in at least one instantiation of a predefined window.

Message Passing (MP) Framework

The MP framework is based on the core idea of recursive neighborhood aggregation.

At every iteration, the representation of each vertex is updated based on messages received from its neighbors.

The majority of the spectral GNNs can be described in terms of the MP framework.

Idea: Each node exchanges messages with its neighbors and updates its representations based on these messages

The message passing scheme runs for T time steps and updates the representation of each vertex h_v^t based on its previous representation and the representations of its neighbors:

$$m_v^{t+1} = \sum_{u \in \mathcal{N}(v)} M_t(h_v^t, h_u^t, e_{vu})$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

where $\mathcal{N}(v)$ is the set of neighbors of v and M_t, U_t are message functions and vertex update functions respectively

An example of a Message Passing Scheme

$$\mathbf{h}_1^{t+1} = \mathbf{W}_0^t \mathbf{h}_1^t + \mathbf{W}_1^t \mathbf{h}_2^t + \mathbf{W}_1^t \mathbf{h}_3^t$$

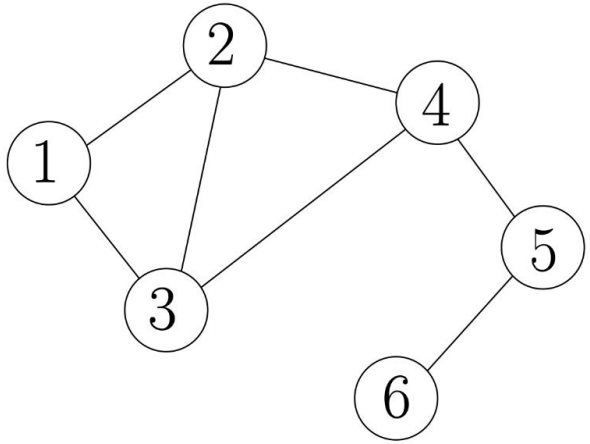
$$\mathbf{h}_2^{t+1} = \mathbf{W}_0^t \mathbf{h}_2^t + \mathbf{W}_1^t \mathbf{h}_1^t + \mathbf{W}_1^t \mathbf{h}_3^t + \mathbf{W}_1^t \mathbf{h}_4^t$$

$$\mathbf{h}_3^{t+1} = \mathbf{W}_0^t \mathbf{h}_3^t + \mathbf{W}_1^t \mathbf{h}_1^t + \mathbf{W}_1^t \mathbf{h}_2^t + \mathbf{W}_1^t \mathbf{h}_4^t$$

$$\mathbf{h}_4^{t+1} = \mathbf{W}_0^t \mathbf{h}_4^t + \mathbf{W}_1^t \mathbf{h}_2^t + \mathbf{W}_1^t \mathbf{h}_3^t + \mathbf{W}_1^t \mathbf{h}_5^t$$

$$\mathbf{h}_5^{t+1} = \mathbf{W}_0^t \mathbf{h}_5^t + \mathbf{W}_1^t \mathbf{h}_4^t + \mathbf{W}_1^t \mathbf{h}_6^t$$

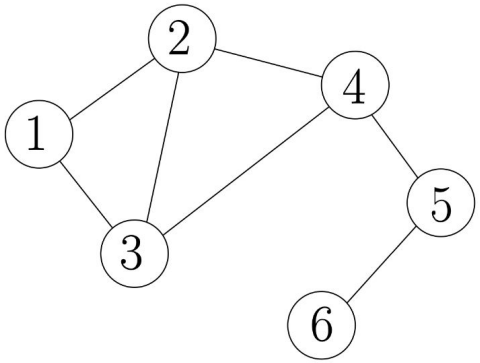
$$\mathbf{h}_6^{t+1} = \mathbf{W}_0^t \mathbf{h}_6^t + \mathbf{W}_1^t \mathbf{h}_5^t$$



Example of Simple Readout scheme

Output of message passing phase:

$$\{\mathbf{h}_1^{T_{max}}, \mathbf{h}_2^{T_{max}}, \mathbf{h}_3^{T_{max}}, \mathbf{h}_4^{T_{max}}, \mathbf{h}_5^{T_{max}}, \mathbf{h}_6^{T_{max}}\}$$



Graph representation:

$$\mathbf{z}_G = \frac{1}{6}\left(\mathbf{h}_1^{T_{max}} + \mathbf{h}_2^{T_{max}} + \mathbf{h}_3^{T_{max}} + \mathbf{h}_4^{T_{max}} + \mathbf{h}_5^{T_{max}} + \mathbf{h}_6^{T_{max}}\right)$$

Document Summarization

Master node: Generated networks also contain a special document node, linked to all other nodes

- can encode a summary of the document

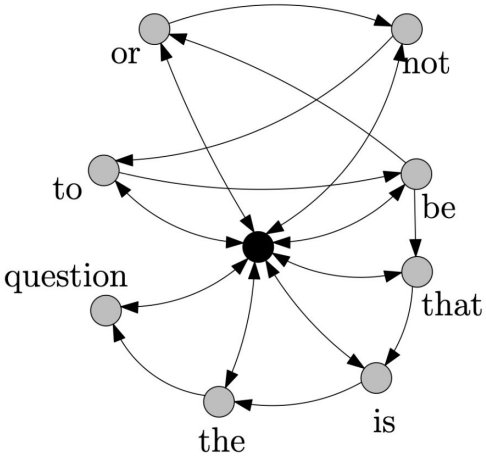


Figure: Graph representation of the document: "to be or not to be: that is the question". The black node corresponds to the **master node**

Generate Message Passing

MPAD utilizes the following $\mathrm{AGGREGATE}$ function:

$$\begin{aligned} \mathbf{X}^{t+1} &= \mathrm{MLP}^{t+1}(\mathbf{H}^t) \\ \mathbf{M}^{t+1} &= \mathbf{D}^{-1}\,\mathbf{A}\,\mathbf{X}^{t+1} \end{aligned} \tag{1}$$

- $\mathbf{A} \Rightarrow$ adjacency matrix of word co-occurence network
- $\mathbf{D} \Rightarrow$ a diagonal matrix such that $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$
- $\mathbf{H}^t \in \mathbb{R}^{n \times d} \Rightarrow$ contains node features ($\mathbf{H}^0$ contains word (node) embeddings)
- Renormalization $\Rightarrow$ matrix product $\mathbf{D}^{-1}\,\mathbf{A}\,\mathbf{X}^{t+1}$ computes average of neighbors' features
  $\hookrightarrow$ avoids numerical instabilities

The COMBINE function corresponds to a GRU:

$$
\begin{aligned}
\mathbf{H}^{t+1} &= GRU(H^t, M^{t+1}) \\
\mathbf{R}^{t+1} &= \sigma(\mathbf{W}_R^{t+1}\mathbf{M}^{t+1} + \mathbf{U}_R^{t+1}\mathbf{X}^{t+1}) \\
\mathbf{Z}^{t+1} &= \sigma(\mathbf{W}_Z^{t+1}\mathbf{M}^{t+1} + \mathbf{U}_Z^{t+1}\mathbf{X}^{t+1}) \\
\tilde{\mathbf{H}}^{t+1} &= \tanh(\mathbf{W}^{t+1}\mathbf{M}^{t+1} + \mathbf{U}^{t+1}(\mathbf{R}^{t+1} \odot \mathbf{X}^{t+1})) \\
\mathbf{H}^{t+1} &= (1 - \mathbf{Z}^{t+1}) \odot \mathbf{X}^{t+1} + \mathbf{Z}^{t+1} \odot \tilde{\mathbf{H}}^{t+1}
\end{aligned}
\tag{2}
$$

where $\mathbf{W}, \mathbf{U}$ are trainable weight matrices

- $\mathbf{R} \Rightarrow$ reset gate controls amount of information from the previous time step that should propagate to the candidate representations $\tilde{\mathbf{H}}^{t+1}$:
- $\mathbf{Z} \Rightarrow$ update gate

After performing updates for T iterations, we obtain a matrix $\mathbf{H}^T \in \mathbb{R}^{n \times d}$ containing the final vertex representations

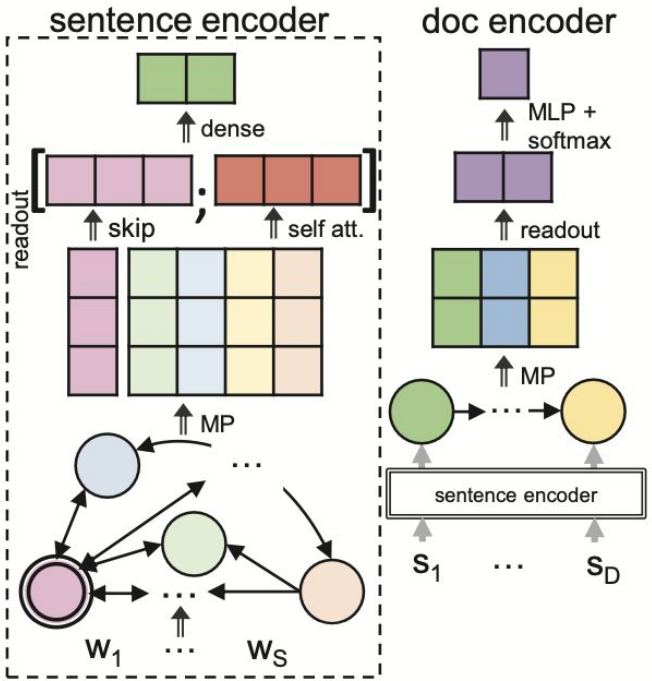
Let $\hat{\mathbf{H}}^T \in \mathbb{R}^{(n-1)\times d}$ be the representation matrix without the row of the **master node**. The $\mathrm{READOUT}$ function applies self-attention to $\hat{\mathbf{H}}^T$:

$$
\begin{aligned}
\mathbf{Y}^T &= \tanh(\hat{\mathbf{H}}^T \mathbf{W}_A^T) \\
\boldsymbol{\alpha}_i^T &= \frac{\exp(\mathbf{Y_i}^T \cdot \mathbf{v}^T)}{\sum_{j=1}^{n-1} \exp(\mathbf{Y_j}^T \cdot \mathbf{v}^T)} \\
\mathbf{u}^T &= \sum_{i=1}^{n-1} \boldsymbol{\alpha}_i^T \hat{\mathbf{H}}_i^T
\end{aligned}
\tag{3}
$$

Then, $\mathbf{u}^T$ is concatenated with the **master node** representation

MPAD Hierarchical Structure

- Master node skip connection
- Readout
  - Multi-readout scheme
    - apply readout to all time steps and concatenate the results, finally obtaining $h_G \in R^{T \times 2d}$
- MPAD sentence attention
- MPAD clique
- documents modeled as graphs where nodes represent sentences
- Generalized MPAD path



Results and Ablation Study

Experiments conducted on 10 standard text classification datasets show that our architectures are competitive with the state-of-the-art:

- Reuters
- BBCSport
- Polarity
- MPQA
- IMDB
- TREC
- SST-1
- SST-2
- Yelp2013

Ablation study here is used to examine impact of hyperparameters on performance

- Number of MP Iterations
- Undirected Edges
- NO master node
- NO renormalization
- Neighbors-only
- NO master node skip connection

Model	Reut.	BBC	Pol.	Subj.	MPQA	IMDB	TREC	SST-1	SST-2	Yelp'13
doc2vec	95.34	98.64	67.30	88.27	82.57	**92.5**	70.80	48.7	87.8	57.7
CNN	97.21	98.37	**81.5**	93.4	89.5	90.28	93.6	48.0	87.2	64.89
DAN	94.79	94.30	80.3	92.44	88.91	89.4	89.60	47.7	86.3	61.55
Tree-LSTM	-	-	-	-	-	-	-	51.0	88.0	-
DRNN	-	-	-	-	-	-	-	49.8	86.6	-
LSTMN	-	-	-	-	-	-	-	47.9	87.0	-
C-LSTM	-	-	-	-	-	-	94.6	49.2	87.8	-
SPGK	96.39	94.97	77.89	91.48	85.78	OOM	90.69	OOM	OOM	OOM
WMD	96.5	98.71	66.42	86.04	83.95	OOM	73.40	OOM	OOM	OOM
DiSAN	97.35	96.05	80.38	**94.2**	**90.1**	83.25	94.2	**51.72**	86.76	60.51
LSTM-GRNN	96.16	95.52	79.98	92.38	89.08	89.98	89.40	48.09	86.38	65.1
HN-ATT	97.25	96.73	80.78	92.92	89.08	90.06	90.80	49.00	86.71	**68.2**
MPAD	97.07	98.37	80.24	93.46*	90.02	91.30	**95.60***	49.09	87.80	66.16
MPAD-sentence-att	96.89	99.32	80.44	93.02	**90.12***	91.70	**95.60***	49.95*	**88.30***	66.47
MPAD-clique	**97.57***	**99.72***	81.17*	92.82	89.96	91.87*	95.20	48.86	87.91	66.60
MPAD-path	97.44	99.59	80.46	93.31	89.81	91.84	93.80	49.68	87.75	66.80*

Questions

THANK YOU

Interesting Libraries to Consider

Neural Structured Learning: Training with Structured Signals

https://www.tensorflow.org/neural_structured_learning

Graph4NLP

https://pythonrepo.com/repo/graph4ai-graph4nlp-python-natural-language-processing

Citations

Images citations/references:
https://arxiv.org/pdf/1604.08120.pdf
https://medium.com/@BorisAKnyazev/tutorial-on-graph-neural-networks-for-computer-vision-and-beyond-part-1-3d9fada3b80d
https://shikhar-vashishth.github.io/assets/pdf/emnlp19_tutorial.pdf
https://arxiv.org/abs/1611.08097 => excellent paper on geometric deep learning that helped me develop good intuition
https://www.semanticscholar.org/paper/Sparse-Graph-Attention-Networks-Ye-Ji/264d4b803e93d933eaa6691836609522b38ba03e
https://www.semanticscholar.org/paper/Learning-to-Represent-Image-and-Text-with-Graphs-Zhang-Hu/73068d13d6e53876c374ebd4c862ec01351c9f39
http://zhijing-jin.com/files/papers/GNN4NLP_Survey_2020.pdf
https://github.com/giannisnik/mpad