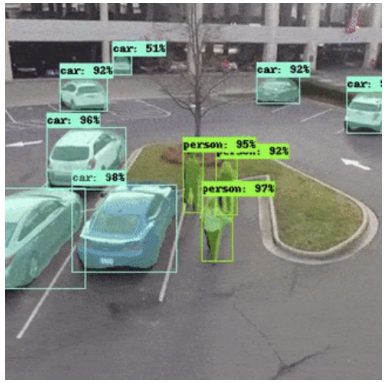
N-Gram Graph: Simple Unsupervised Representation for Graphs, with Applications to Molecules

Mehmet F. Demirel, Shengchao Liu, Siddhant Garg, Yingyu Liang

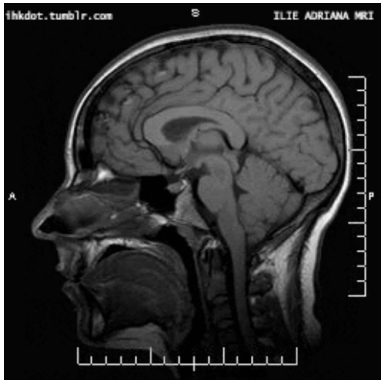
IFDS Ideas Forum

December 21, 2020

Machine Learning Everywhere

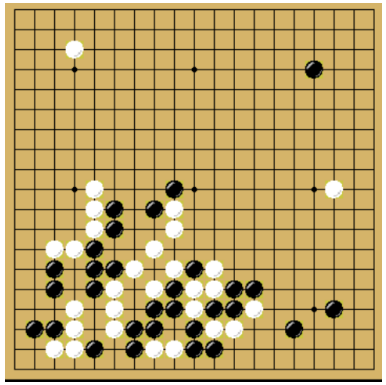


Computer Vision



Medical Imaging

NLP

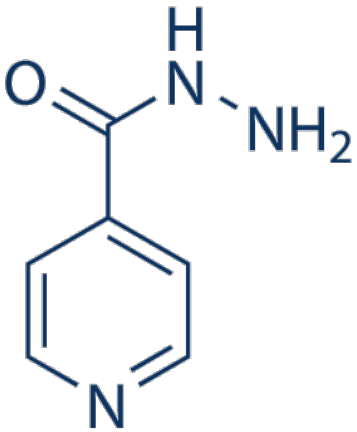


Game Playing

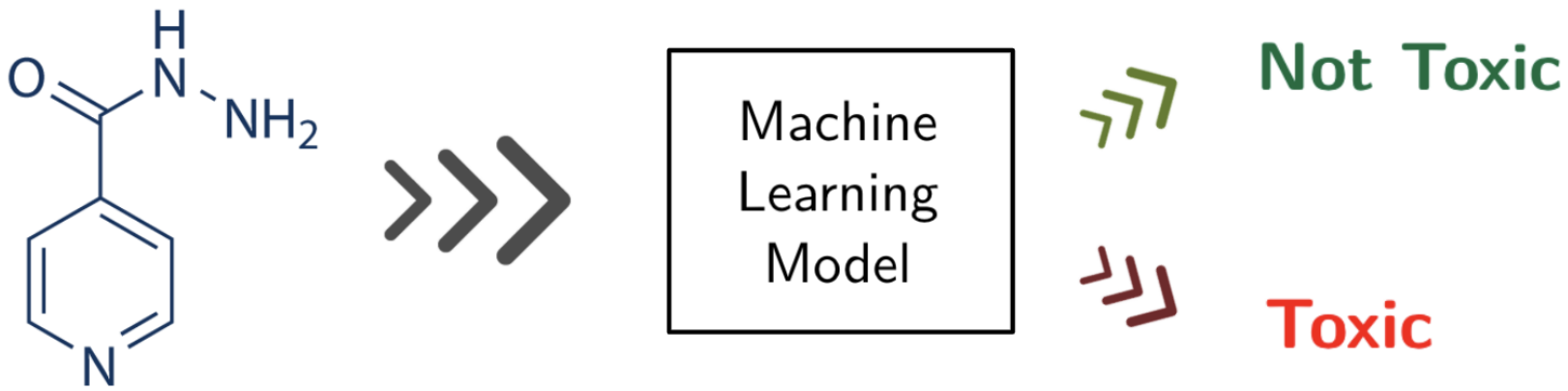
And more!

Molecules?

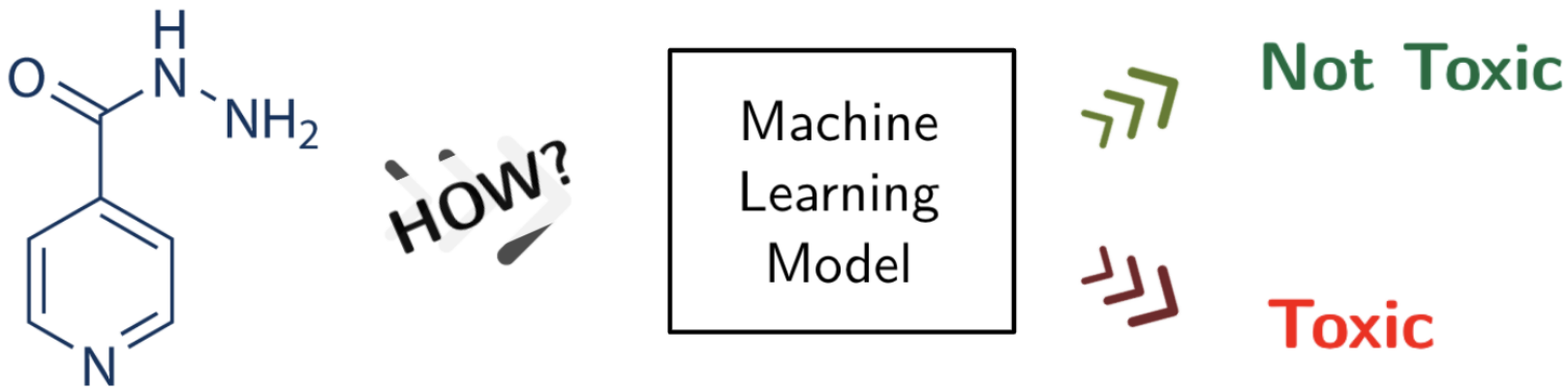
What about molecules?



Molecular Property Prediction



Molecular Property Prediction



Challenge: Representation

Input to traditional machine learning models: **vectors**

How to represent a molecule as a vector?

- Fingerprints *e.g.* Morgan fingerprints
- Graph kernels *e.g.* WL-kernel
- Graph neural networks (GNN): GCN, Weave

Fingerprints/kernels are unsupervised and fast to compute.
GNNs are end-to-end supervised, more expensive; but powerful.

Challenge: Representation

Input to traditional machine learning models: **vectors**

How to represent a molecule as a vector?

- Fingerprints *e.g.* Morgan fingerprints
- Graph kernels *e.g.* WL-kernel
- Graph neural networks (GNN): GCN, Weave

Fingerprints/kernels are unsupervised and fast to compute.
GNNs are end-to-end supervised, more expensive; but powerful.

Challenge: Representation

Input to traditional machine learning models: **vectors**

How to represent a molecule as a vector?

- Fingerprints *e.g.* Morgan fingerprints
- Graph kernels *e.g.* WL-kernel
- Graph neural networks (GNN): GCN, Weave

Fingerprints/kernels are unsupervised and fast to compute.
GNNs are end-to-end supervised, more expensive; but powerful.

(Simple) N-gram Graph

Our previous work [1] is inspired by the **N-gram approach in NLP**.

- Unsupervised
- Fast to compute
- Overall better performance than traditional methods

[1] Liu, Shengchao, Mehmet F. Demirel, and Yingyu Liang. "N-gram graph: Simple unsupervised representation for graphs, with applications to molecules." Advances in Neural Information Processing Systems. 2019.

N-gram approach in NLP

n-gram is a contiguous sequence of n words from a given sentence.

"I love living in Madison"

N-gram approach in NLP

n-gram is a contiguous sequence of n words from a given sentence.

"I love living in Madison"

N-gram approach in NLP

n-gram is a contiguous sequence of n words from a given sentence.

"**I love** living in Madison"

- **2-grams:** "I love"

N-gram approach in NLP

n-gram is a contiguous sequence of n words from a given sentence.

"I **love living** in Madison"

- **2-grams:** "I love", "love living"

N-gram approach in NLP

n-gram is a contiguous sequence of n words from a given sentence.

"I love **living in** Madison"

- **2-grams:** "I love", "love living", "living in"

N-gram approach in NLP

n-gram is a contiguous sequence of n words from a given sentence.

"I love living **in Madison**"

- **2-grams:** "I love", "love living", "living in", "in Madison"

N-gram approach in NLP

n-gram is a contiguous sequence of n words from a given sentence.

"I love living in Madison"

- **1-grams:** "I", "love", "living", "in ", "Madison"
- **2-grams:** "I love", "love living", "living in", "in Madison"
- **3-grams:** "I love living", "love living in", "living in Madison"

⋮

N-gram approach in NLP

n-gram is a contiguous sequence of n words from a given sentence

N-gram count vector $c_{(n)}$ is a numeric representation vector:

- its coordinates correspond to all n-grams
- its coordinate values are the number of times the corresponding n-gram shows up in the sentence

Notice that $c_{(1)}$ is just the histogram of the words in the sentence.

N-gram approach in NLP

n-gram is a contiguous sequence of n words from a given sentence

N-gram count vector $c_{(n)}$ is a numeric representation vector:

- its coordinates correspond to all n-grams
- its coordinate values are the number of times the corresponding n-gram shows up in the sentence

Notice that $c_{(1)}$ is just the histogram of the words in the sentence.

Dimensionality Reduction

Problem: N-gram vector $c_{(n)}$ has high dimensions: $|V|^n$ for vocabulary V.

Solution: Dimensionality reduction by word embeddings: $f_{(1)} = W c_{(1)}$

Dimensionality Reduction

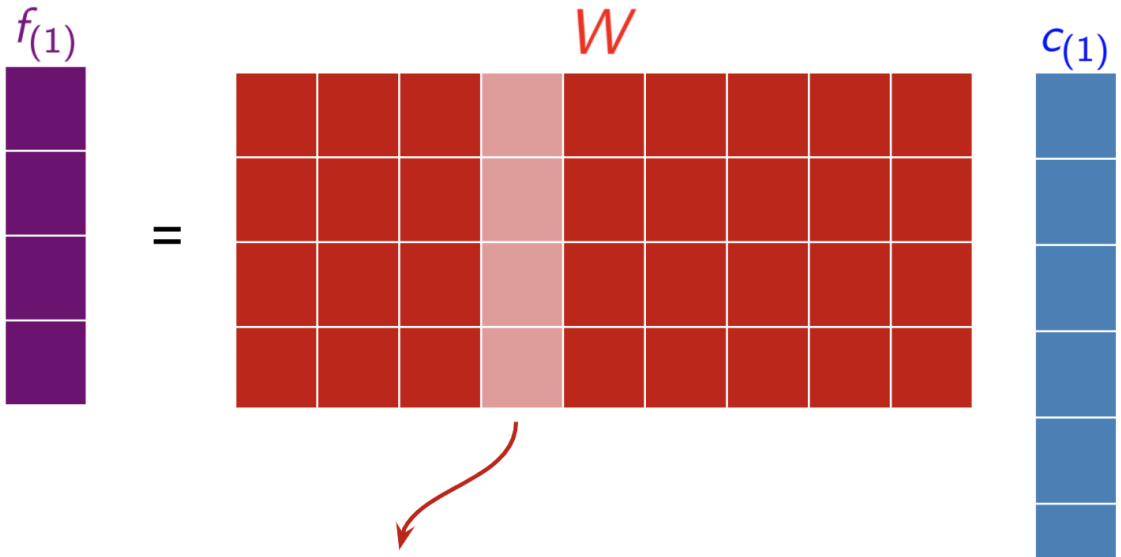
Problem: N-gram vector $c_{(n)}$ has high dimensions: $|V|^n$ for vocabulary V.

Solution: Dimensionality reduction by word embeddings: $f_{(1)} = W c_{(1)}$

Dimensionality Reduction

Problem: N-gram vector $c_{(n)}$ has high dimensions: $|V|^n$ for vocabulary V.

Solution: Dimensionality reduction by word embeddings: $f_{(1)} = W c_{(1)}$

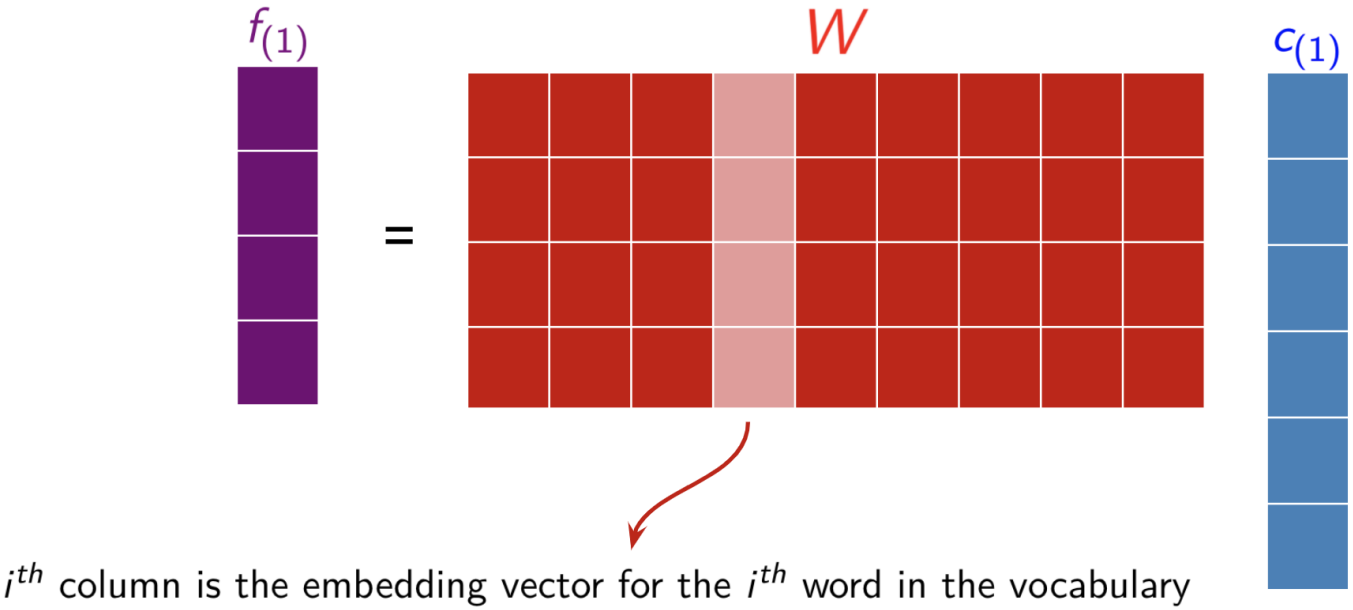


i^{th} column is the embedding vector for the ith word in the vocabulary

Dimensionality Reduction

Problem: N-gram vector $c_{(n)}$ has high dimensions: $|V|^n$ for vocabulary V.

Solution: Dimensionality reduction by word embeddings: $f_{(1)} = W c_{(1)}$



$f_{(1)}$ is just the sum of the word vectors in the sentence!

Dimensionality Reduction

Problem: N-gram vector $c_{(n)}$ has high dimensions: $|V|^n$ for vocabulary V.

Solution: Dimensionality reduction by word embeddings: $f_{(1)} = W c_{(1)}$

For general n:

- Embedding of an n-gram is the entry-wise product of its word vectors.
- $f_{(n)}$ is the sum of the embeddings of the n-grams in the sentence.

N-gram *graphs*?

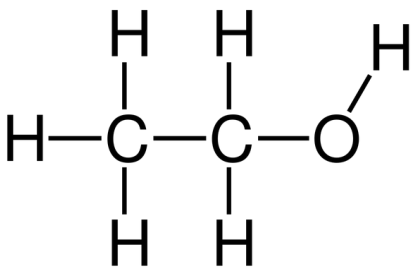
Sentences are linear graphs on words.
Molecules are graphs on atoms with attributes!

N-gram *graphs*?

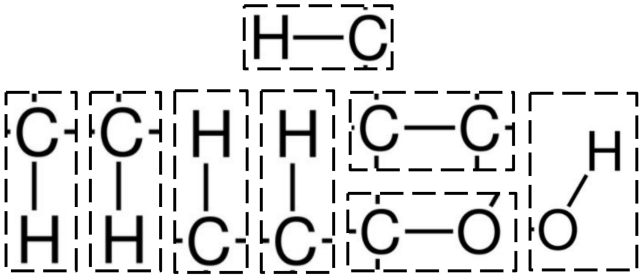
Sentences are linear graphs on words.
Molecules are graphs on atoms with attributes!

We can view:

- atoms with different attributes as different words
- walks of length n as n-grams.



A molecule



Its 2-grams

N-gram Graph algorithm

Sentences are linear graphs on words.
Molecules are graphs on atoms with attributes!

Given the embeddings for the atoms (vertex vectors):

- Enumerate all n-grams (walks of length n)
- Embedding of an n-gram: entry-wise product of its vertex vectors
- $f_{(n)}$: sum of embeddings of all n-grams
- Final N-gram Graph embedding f_G: concatenation of $f_{(1)}, f_{(2)}, \ldots, f_{(T)}$

N-gram Graph as simple GNN

Given vectors f_i for vertices i and graph adjacency matrix $\mathcal{A}$:

$F_{(1)} = F = [f_1, \ldots, f_m], f_{(1)} = F_{(1)}\mathbf{1}$
for each $n \in [2, T]$ **do**
$\quad F_{(n)} = (F_{(n-1)}\mathcal{A}) \odot F$
$\quad f_{(n)} = F_{(n)}\mathbf{1}$
end for
$f_G = [f_{(1)}; \ldots; f_{(T)}]$

Equivalent to a simple GNN without any parameters!

Experimental Results

60 tasks on 10 datasets from MoleculeNet [2].

Methods:

- WL-Kernel + SVM
- Morgan FP + RF or XGB
- Graph CNN (GCNN), Weave Neural Network, Graph Isomorphism Network (GIN)

- N-gram Graph + RF or XGB
- Vertex embedding dimension $r = 100$ and $T = 6$

[2] Wu, Zhenqin, et al. "MoleculeNet: a benchmark for molecular machine learning." Chemical science 9.2 (2018): 513-530

Experimental Results

N-gram Graph + XGB: top-1 in 21 and top-3 in 48 out of 60 tasks
Overall better performance than other methods

Table 2: Performance overview: (# of tasks with top-1 performance, # of tasks with top-3 performance) is listed for each model and each dataset. For cases with no top-3 performance on that dataset are left blank. Some models are not well tuned or too slow and are left in "-".

Dataset	# Task	Eval Metric	WL SVM	Morgan RF	Morgan XGB	GCNN	Weave	GIN	N-Gram RF	N-Gram XGB
Delaney	1	RMSE					1, 1	–	0, 1	0, 1
Malaria	1	RMSE		1, 1				–	0, 1	0, 1
CEP	1	RMSE		1, 1				–	0, 1	0, 1
QM7	1	MAE					0, 1	–	0, 1	1, 1
QM8	12	MAE		1, 4	0, 1	7, 12	2, 6	–	0, 2	2, 11
QM9	12	MAE	–		0, 1	4, 7	1, 8	–	0, 8	7, 12
Tox21	12	ROC-AUC	0, 2	0, 7		0, 2	0, 1		3, 12	9, 12
clintox	2	ROC-AUC	0, 1			1, 2	0, 1			1, 2
MUV	17	PR-AUC	4, 12	5, 11	5, 11			0, 7	2, 4	1, 6
HIV	1	ROC-AUC		1, 1					0, 1	0, 1
Overall	60		4, 15	9, 25	5, 13	12, 23	4, 18	0, 7	5, 31	**21, 48**

Theoretical Analysis

Recall $f_{(1)} = W c_{(1)}$

- W is the vertex embedding matrix.
- $c_{(1)}$ is the count vector.

With sparse $c_{(1)}$ and random W, $c_{(1)}$ can be recovered from $f_{(1)}$.

- Well-known in compressed sensing.

Theoretical Analysis

Recall $f_{(1)} = W c_{(1)}$

- W is the vertex embedding matrix.
- $c_{(1)}$ is the count vector.

With sparse $c_{(1)}$ and random W, $c_{(1)}$ can be recovered from $f_{(1)}$.

- Well-known in compressed sensing.

In general, $f_{(n)} = T_{(n)} c_{(n)}$ for some linear mapping $T_{(n)}$ depending on W.

With sparse $c_{(n)}$ and random W, $c_{(n)}$ can be recovered from $f_{(n)}$.

Theoretical Analysis

Recall $f_{(1)} = W c_{(1)}$

- W is the vertex embedding matrix.
- $c_{(1)}$ is the count vector.

With sparse $c_{(1)}$ and random W, $c_{(1)}$ can be recovered from $f_{(1)}$.

- Well-known in compressed sensing.

In general, $f_{(n)} = T_{(n)} c_{(n)}$ for some linear mapping $T_{(n)}$ depending on W.

With sparse $c_{(n)}$ and random W, $c_{(n)}$ can be recovered from $f_{(n)}$.

Therefore, $f_{(n)}$ preserves information in $c_{(n)}$.

Furthermore, we can prove that regularized linear classifier on $f_{(n)}$ is competitive to the best linear classifier on $c_{(n)}$.

Current work: Parametric N-gram Graph

The (simple) N-gram graph algorithm has no parameter and requires no training. Therefore, it is efficient in computation. However:

- Huge design space for adding trainable parameters.
- Concatenated with a classifier, it becomes end-to-end.

Why parametrize the algorithm, though?

Weighted Vertex Features

Some features are more important, some are dummy features.

- Automatic weighting of vertex features
- Better representation

Given vectors $F = [f_1, \ldots, f_m]$ for m vertices and graph adjacency matrix $\mathcal{A}$:

$F_{(1)} = F$, $f_{(1)} = F_{(1)}\mathbf{1}$
for each $n \in [2, T]$ **do**
$\quad F_{(n)} = (F_{(n-1)}\mathcal{A}) \odot F$
$\quad f_{(n)} = F_{(n)}\mathbf{1}$
end for
$f_G = [f_{(1)}; \ldots; f_{(T)}]$

Weighted Vertex Features

Some features are more important, some are dummy features.

- Automatic weighting of vertex features
- Better representation

Given vectors $F = [f_1, \ldots, f_m]$ for m vertices and graph adjacency matrix $\mathcal{A}$:

$F_{(1)} = \sigma(W_1 F)$, $f_{(1)} = F_{(1)}\mathbf{1}$
for each $n \in [2, T]$ **do**
$F_{(n)} = (F_{(n-1)}\mathcal{A}) \odot F$
$f_{(n)} = F_{(n)}\mathbf{1}$
end for
$f_G = [f_{(1)}; \ldots; f_{(T)}]$

Weighted Vertex Features

Some features are more important, some are dummy features.

- Automatic weighting of vertex features
- Better representation

Given vectors $F = [f_1, \ldots, f_m]$ for m vertices and graph adjacency matrix $\mathcal{A}$:

$F_{(1)} = \sigma(W_1 F)$, $f_{(1)} = F_{(1)}\mathbf{1}$
for each $n \in [2, T]$ **do**
$\quad F_{(n)} = (F_{(n-1)}\mathcal{A}) \odot F$
$\quad f_{(n)} = F_{(n)}\mathbf{1}$
end for
$f_G = [f_{(1)}; \ldots; f_{(T)}]$

Then for a vertex embedding f_i, $W_1 f_i$ will stretch its components along W_1's larger singular vectors while relatively shrink the components along the smaller ones.

Attentive Messages

Some nodes have more impact on their neighbors.

- Weighted sum of latent vectors from neighbors (with attention).

Given vectors $F = [f_1, \ldots, f_m]$ for m vertices and graph adjacency matrix $\mathcal{A}$:

$F_{(1)} = \sigma(W_1 F), f_{(1)} = F_{(1)}\mathbf{1}$
for each $n \in [2, T]$ **do**
  $F_{(n)} = (F_{(n-1)}\mathcal{A}) \odot F$
  $f_{(n)} = F_{(n)}\mathbf{1}$
end for
$f_G = [f_{(1)}; \ldots; f_{(T)}]$

Attentive Messages

Some nodes have more impact on their neighbors.

- Weighted sum of latent vectors from neighbors (with attention).

Given vectors $F = [f_1, \ldots, f_m]$ for m vertices and graph adjacency matrix $\mathcal{A}$:

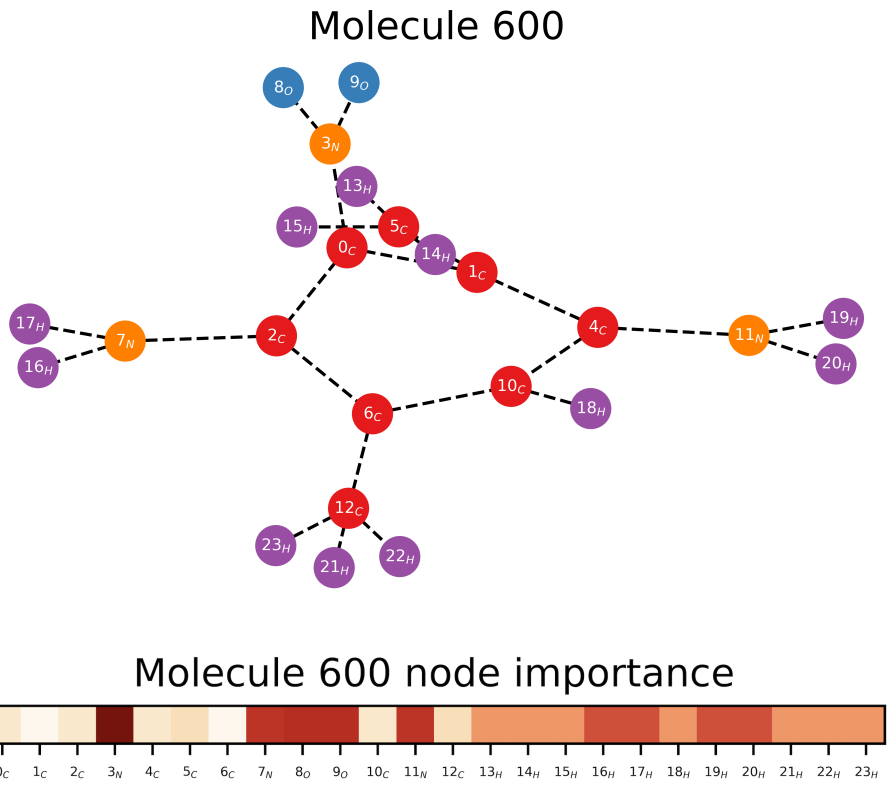
$F_{(1)} = \sigma(W_1 F), f_{(1)} = F_{(1)}\mathbf{1}$
for each $n \in [2, T]$ **do**
$F_{(n)} = (F_{(n-1)}(\mathcal{A} \odot \bar{S})) \odot F_{(1)}$
$f_{(n)} = F_{(n)}\mathbf{1}$
end for
$f_G = [f_{(1)}; \ldots; f_{(T)}]$

$$s_{ji} = [F_{(n-1)}]_j^\top W_2 [F_{(n-1)}]_i$$

$$\bar{S}_{ji} = \frac{\exp(s_{ji})}{\sum_{k \in \text{neighbors of } i} \exp s_{ki}}$$

Attentive Messages: Visualization

Mutagenicity dataset: NO_2 and NH_2 atom groups are known to have a mutagenic effect in a molecule.



Weighted Summarization of Vertices

Similar to the previous case in the vertex feature space, the downstream learning tasks may prefer certain directions in the final embedding space.

Given vectors $F = [f_1, \ldots, f_m]$ for m vertices and graph adjacency matrix $\mathcal{A}$:

$F_{(1)} = \sigma(W_1 F), f_{(1)} = F_{(1)}\mathbf{1}$
for each $n \in [2, T]$ **do**
$\quad F_{(n)} = (F_{(n-1)}\mathcal{A}) \odot F$
$\quad f_{(n)} = F_{(n)}\mathbf{1}$
end for
$f_G = [f_{(1)}; \ldots; f_{(T)}]$

$$s_{ji} = [F_{(n-1)}]_j^\top W_2 [F_{(n-1)}]_i$$

$$\bar{S}_{ji} = \frac{\exp(s_{ji})}{\sum_{k \in \text{neighbors of } i} \exp s_{ki}}$$

Weighted Summarization of Vertices

Similar to the previous case in the vertex feature space, the downstream learning tasks may prefer certain directions in the final embedding space.

Given vectors $F = [f_1, \ldots, f_m]$ for m vertices and graph adjacency matrix $\mathcal{A}$:

$F_{(1)} = \sigma(W_1 F), f_{(1)} = F_{(1)}\mathbf{1}$
for each $n \in [2, T]$ **do**
$\quad F_{(n)} = (F_{(n-1)}(\mathcal{A} \odot \bar{S})) \odot F_{(1)}$
$\quad f_{(n)} = \sigma(W_3 F_{(n)})\mathbf{1}$
end for
$f_G = [f_{(1)}; \ldots; f_{(T)}]$

$$s_{ji} = [F_{(n-1)}]_j^\top W_2 [F_{(n-1)}]_i$$

$$\bar{S}_{ji} = \frac{\exp(s_{ji})}{\sum_{k \in \text{neighbors of } i} \exp s_{ki}}$$

Parametric N-gram: Experiments

Preliminary experimental results on some classification tasks (the best model on each task is **underlined** and the top-3 are **bolded**).

Task	Metric	FP+RF	FP+XGB	WL	GCNN	Weave	GAT	GIN	N-Gram XGB	N-Gram RF	Parametric N-gram
NR-AR	ROC-AUC	0.7633	0.7524	0.7009	0.7615	0.7739	0.7545	0.7586	**0.7765**	**0.7696**	**0.7819**
NR-AR-LBD	ROC-AUC	0.8579	0.8523	0.8609	0.8443	0.8240	0.7995	0.8299	**0.8732**	**0.8629**	**0.8676**
NR-AhR	ROC-AUC	**0.8904**	0.8847	0.8758	**0.8863**	0.8570	0.8227	0.8718	**0.8973**	0.8772	0.8823
NR-Aromatase	ROC-AUC	0.8214	0.7978	0.8185	0.8277	0.8267	0.7436	0.7596	**0.8476**	**0.8523**	**0.8535**
NR-ER	ROC-AUC	0.7257	0.7228	0.7041	0.7371	0.7362	0.7062	0.6828	**0.7536**	**0.7378**	**0.7626**
NR-ER-LBD	ROC-AUC	**0.8383**	0.8062	0.7985	0.8134	0.8090	0.7643	0.7715	**0.8341**	0.8308	**0.8399**
NR-PPAR-gamma	ROC-AUC	0.8400	0.8219	**0.8445**	0.8164	0.8035	0.7585	0.7803	**0.8569**	0.8054	**0.8540**
SR-ARE	ROC-AUC	0.8204	0.7990	0.8007	0.8093	0.7706	0.7349	0.7945	**0.8514**	**0.8385**	**0.8242**
SR-ATAD5	ROC-AUC	**0.8495**	0.8091	0.8143	0.8273	0.7652	0.7543	0.8026	**0.8494**	**0.8526**	0.8409
SR-HSE	ROC-AUC	0.7969	0.7586	**0.8031**	0.7742	0.7488	0.6865	0.7404	**0.8082**	0.8012	**0.8002**
SR-MMP	ROC-AUC	**0.8897**	0.8801	0.8746	0.8771	0.8859	0.8340	0.8721	**0.9045**	0.8842	**0.9038**
SR-p53	ROC-AUC	**0.8445**	0.8255	0.8416	0.8179	0.7866	0.7328	0.8174	**0.8597**	**0.8462**	0.8324
CT_TOX	ROC-AUC	0.7708	0.8133	0.8296	**0.8600**	0.8437	0.8280	**0.8594**	0.8493	0.8277	**0.8930**
FDA_APPROVED	ROC-AUC	0.7753	0.7952	0.8615	0.8664	0.8221	**0.8990**	**0.8834**	0.8518	0.7949	**0.8883**
hiv	ROC-AUC	**0.8558**	**0.8452**	0.8114	0.8131	0.5560	0.7834	0.8290	**0.8429**	0.8262	0.8240
MUV-466	ROC-AUC	**0.7653**	0.7377	0.7079	0.7359	0.6337	**0.7491**	0.7055	0.7244	0.7155	**0.8081**
MUV-548	ROC-AUC	0.9020	**0.9535**	0.9169	**0.9599**	0.8209	0.7638	0.7932	0.9252	0.8097	**0.9786**
MUV-600	ROC-AUC	0.5063	0.5360	0.5360	0.5699	0.5751	0.4367	0.5746	**0.5863**	**0.6746**	**0.6802**
MUV-644	ROC-AUC	**0.8927**	0.8642	**0.9442**	0.8854	0.7865	0.7619	0.7490	0.7995	0.6692	**0.9031**
MUV-652	ROC-AUC	**0.7217**	**0.7247**	0.6530	0.6942	0.7215	0.4930	0.6454	0.6881	0.5693	**0.8268**
MUV-689	ROC-AUC	0.6762	0.5887	**0.7352**	0.6711	0.5759	0.5526	**0.7750**	0.6692	0.6188	**0.7503**
MUV-692	ROC-AUC	**0.6509**	**0.6931**	0.4475	0.5809	0.5448	0.6261	0.6286	0.6060	0.4979	**0.6638**
MUV-712	ROC-AUC	**0.9272**	0.8997	0.8892	**0.9358**	0.8538	0.7599	0.7727	0.7751	0.8121	**0.9315**
MUV-713	ROC-AUC	0.5471	0.5538	**0.7866**	**0.7314**	0.6865	0.5863	0.5666	0.7147	0.6734	**0.7733**
MUV-733	ROC-AUC	**0.7090**	0.6276	0.7075	**0.7507**	0.8195	0.6372	0.5576	0.6962	0.6903	**0.8126**
MUV-737	ROC-AUC	0.7367	0.7910	0.7727	0.7960	0.7842	0.6748	0.7233	**0.8470**	**0.8792**	**0.8932**
MUV-810	ROC-AUC	**0.7816**	**0.7943**	**0.8745**	0.7140	0.5929	0.5881	0.6822	0.6803	0.4921	0.7778
MUV-832	ROC-AUC	**0.9863**	**0.9802**	0.9640	0.9264	0.8436	0.9234	0.9183	0.9692	0.8890	**0.9723**
MUV-846	ROC-AUC	0.8770	0.8708	**0.8837**	**0.9109**	0.8920	0.8625	0.7637	0.7801	0.7814	**0.9593**
MUV-852	ROC-AUC	0.8421	**0.8898**	0.8673	**0.8823**	0.8588	0.7430	0.7346	0.8342	0.8050	**0.9307**
MUV-858	ROC-AUC	0.5291	**0.7011**	0.6774	**0.7051**	0.6545	0.6498	**0.7461**	0.6299	0.4391	0.6720
MUV-859	ROC-AUC	0.4581	0.5295	0.5334	**0.6128**	0.6093	0.4987	0.6073	**0.7236**	0.5776	**0.6524**

Thank you!