Natural Language Processing (NLP): A Comparative Study on Applications of GNNs for the Tasks Like Text Classification, Semantic Role Labelling, and Abstract Meaning Representation Parsing

Pathan Ashhad Zakirkhan[1], Sandhya Kaprawan[2]

[1]Student (MS Data Analytics), Department of Information Technology, University of Mumbai, Vidhyanagri, Santacruz (E), Mumbai, India
Email: *ashhadpathan7[at]gmail.com*

[2]Assistant Professor, Department of Information Technology, University of Mumbai, Vidhyanagri, Santacruz (E), Mumbai, India
Email: *sandhya.kaprawan[at]udit.mu.ac.in*

Abstract: ***The dominant paradigm in Natural Language Processing (NLP) has traditionally relied on sequential modelling, often overlooking the rich, non-linear structural dependencies inherent in linguistic data. Graph Neural Networks (GNNs) offer a powerful alternative by modelling text as structured graphs, yet a systematic evaluation of their comparative efficacy across tasks of varying structural complexity remains under-explored. This paper presents a comprehensive comparative study of three prominent GNN architectures- Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and Graph SAGE- applied to three distinct NLP tasks: Text Classification, Semantic Role Labelling (SRL), and Abstract Meaning Representation (AMR) Parsing. We construct task-specific graph topologies, utilizing heterogeneous corpus graphs for document classification and syntactic dependency trees for SRL and AMR. Our empirical results, derived from benchmarks on the 20 Newsgroups, CoNLL-2009, and AMR 2.0 datasets, reveal a clear correlation between architectural properties and task complexity. While GCNs provide a robust and efficient baseline for global text classification (Accuracy: 96.6%), they struggle with the fine-grained structural modelling required for semantic tasks. Conversely, GATs demonstrate superior performance on syntax-heavy tasks, achieving a Labelled F1 score of 85.8% on SRL and a Smatch score of 74.5 on AMR parsing, significantly outperforming the GCN baseline (82.5% and 71.2%, respectively). These findings suggest that the anisotropic aggregation capability of attention mechanisms is critical for capturing the nuanced, long-range dependencies in natural language, establishing GAT as the preferred architecture for structure-dependent NLP applications.***

Keywords: Natural Language Processing (NLP), Graph Neural Networks (GNNs), Deep Learning, Graph Convolutional Network (GCNs), Graph Attention Networks (GATs), GraphSAGE, Text Classification, Semantic Role Labelling (SRL), Abstract Meaning Representation (AMR) Parsing

1. Introduction

Most standard models in natural language processing (NLP) such as RNNs and Transformers, process text sequentially by reading one word after another. Although this method has proven effective for many tasks, it can struggle to capture complex relationship between words that are not close to each other in sequence.

An alternative approach is to represent text as graph, where words are treated as nodes and the connections between them are edges. This structure makes it possible to use Graph Neural Networks (GNNs), a type of model built specially to learn from data in a graph format. Therefore, while traditional models are limited to a linear view, GNNs can analyze the rich non-linear connections within a text, allowing them to capture a deeper level of meaning that sequential models might miss.

The principal objective of this research is to conduct a systematic comparative analysis of prominent Graph Neural Network (GNN) architectures to benchmark their performance and suitability across a spectrum of natural language processing tasks. The study aims to determine which architectures are best suited for tasks of varying complexity, from document level classification to intricate semantic graph parsing.

For Text classification:
To evaluate and compare the effectiveness of different GNN architectures (GCN, GAT, GraphSAGE) in leveraging corpus wide non-linear relationships between documents and words to improve text classification accuracy over traditional sequential models.

For Semantic Role Labelling (SRL)
To access how effectively GNN architectures can model the explicit syntactic structure of sentences (i.e., dependency parse trees) to identify and classify predicate argument relationships, thereby which models are superior at learning from syntax rich graph representations.

For Abstract Meaning Representation (AMR) Parsing:
To analyze and contrast the performance of GNN-based encoder-decoder frameworks in the complex graph-generation task of AMR parsing. This Objective seeks to identify which GNN encoding strategies produce the most semantically coherent and structurally accurate meaning representations from natural language sentences.

2. Literature Review

The application of deep learning to Natural Language Processing has evolved rapidly, with early approaches predominantly relying on sequence-based encoders such as Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) and, more recently, Transformer-based architectures like BERT (Devlin et al., 2018). While these models excel at capturing local context and long-range dependencies through self-attention mechanisms, they do not explicitly model the structural information often inherit in linguistic data, such as syntactic dependency tress or knowledge graphs.

In response to this limitation, Graphical Neural Networks (GNNs) have emerged as a robust paradigm for encoding structured data. Seminal works by Kipf and Welling (2017) on Graph Convolutional Networks (GCNs) and Velickovic et al. (2018) on Graph Attention Networks (GATs) demonstrated the efficacy of message-passing frameworks in non-Euclidean domains. Following these developments, a growing body of literature has explored the integration of GNNS into NLP pipeline.

In the domain of Text-Classification, Yao et al. (2019) proposed the TextGCN framework, constructing a heterogeneous graph of words and documents to capture global word co-occurrence patterns, significantly outperforming traditional methods. Conversely, for syntax-heavy tasks like Semantic Role Labelling (SRL), Marcheggiani and Titov (2017) validated the utility of encoding dependency tress using GCNs, proving that syntactic bias improves argument identification Finally in Abstract Meaning Representation (AMR) parsing, recent studies have leveraged graph-to-sequence and sequence-to-graph architectures to effectively generate semantic graphs, though challenges remain in ensuring structural validity.

Despite these advancements, existing literature remains largely fragmented, with studies typically focusing on singular tasks or specific architectures in isolation. To date, there is a scarcity of comprehensive comparative analyses that systematically evaluate the performance of standard GNN variants specifically GCN, GAT, and GraphSAGE across a complexity gradient ranging from document level classification to fine grained semantic parsing. This study seeks to bridge this gap by providing a unified empirical benchmark.

3. Methodology

To facilitate a rigorous comparative analysis, we evaluate three distinct Graph Neural Network architectures: Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and GraphSAGE. These models were selected due to their ubiquity in the literature and their distinct approaches to neighbourhood aggregation ranging from spectral approximations to attention mechanisms and inductive sampling.

3.1 Graph Convolutional Networks (GCN)

The GCN, introduced by Kipf and Welling (2017), operates as a spectral based graph convolutional. It aggregates information from a node's immediate neighbourhood using a fixed, isotropic normalization. Formally the hidden state of a node v_i at layer $l = 1$, denoted as $h_i{}^{(l+1)}$ is computed as:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\tilde{d}_i \tilde{d}_j}} W^{(l)} h_j^{(l)}\right)$$

Where $\mathcal{N}(i)$ is the set of neighbours of node i, d denotes the degree of the node (including self-loops) $W^{(l)}$ is a learnable weight matrix and σ is a non-linear activation function. The GCN assumes that all neighbours contribute equally to the central node's representation, scaled only by the graph topology.

3.2 Graph Attention Networks (GAT)

Unlike the fixed weights of the GCN, the GAT (Velickovic et al., 2018) employs a self-attention mechanism to assign learnable importance scores to a different neighbours. This allows the model to focus on the most relevant nodes during aggregation. The update rule is defined as:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W^{(l)} h_j^{(l)}\right)$$

Here, the attention coefficient α_{ij} is computed via a shared mechanism α:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(a^T[Wh_i || Wh_j]))}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LeakyReLU}(a^T[Wh_i || Wh_k]))}$$

This mechanism enables the GAT to capture anisotropic relationships, which is hypothesized to be critical for tasks requiring syntactic focus, such as a Sematic Role Labelling.

3.3. GraphSAGE

GraphSAGE (Hamilton et al., 2017) introduces an inductive framework that learns aggregator functions rather than training on a fixed graph structure. It generates embeddings by sampling and aggregating features from a node's locl neighbourhood. The general update rule is:

$$h_i^{(l+1)} = \sigma\left(W^{(l)} \cdot \text{CONCAT}(h_i^{(l)}, \text{AGG}(\{h_j^{(l)}, \forall j \in \mathcal{N}(i)\}))\right)$$

Where AGG represents an aggregation function (e.g. mean, LSTM, or pooling), GraphSAGE inductive nature makes it particularly suitable for large scale graphs or dynamic environments where unseen nodes may be encountered during inference.

3.4 Text Implementation and Graph Construction

To evaluate the selected architectures, we construct task-specific graph topologies that capture the requisite linguistic features for each domain.

3.4.1. Text Classification: Heterogenous Corpus Graph

For the task of text classification, we adopt the heterogenous graph construction method proposed by Yao et al., (2019). We construct a single large-scale graph G = (V,E) for the entire corpus, where the node set V contains both document nodes and word nodes (unique vocabulary tokens).

- Nodes: |D| documents nodes + |V| unique word nodes.
- Edges: Two types of edges are established:
- Word Document edges: Weighted by the Term Frequency Inverse Document Frequency (TF-IDF) of the word in the document.
- Word-Word edges: Weighted by Pointwise Mutual information (PMI) using a fixed size sliding window, capturing global co-occurrence patterns. The GNN models are trained to perform semi supervised node classification on the document nodes.

3.4.2 Semantic Role Labelling (SRL): Syntactic Dependency Encoding

For SRL, the objective is to classify the semantic relationship between predicate arguments pairs. We model each sentence as a graph based on its syntactic dependency parse tree.

- Nodes: Each word in the sentence serves as a node, initialized with context aware embeddings (e.g., from a pretrained BERT layer).
- Edges: Directed edges represent syntactic dependencies (e.g., nsubj, dobj). To facilitate information flow against the direction of dependency, we also add reverse edges and self-loops. The GNN layers (GCN, GAT, or GraphSAGE) are stacked on top of the embedding layer to refine the token representations based on the syntactic structure. The final representations are fed into a classifier to predict the semantic role labels (e.g., ARG0, ARG1) for each word relative to the sentence predicate.

3.4.3. AMR Parsing: Graph-to-Graph Generation

AMR parsing is treated as a graph generation task. We employ an Encoder Decoder framework where the GNN serves as the structural encoder.

- Encoder Input: The input sentence is converted into a graph using its dependency tree, similar to the SRL setup.
- GNN Role: The GNN architectures are employed to encode this syntactic graph, producing structure aware vector representations for each token.
- Decoder: These representations are passed to an attention-based decoder which generates the Abstract Meaning Representation graph sequentially (linearized) or via a transition-based system.
- Comparative Focus: We isolate the impact of the GNN encoder by keeping the decoder architecture constant across all experiments, thereby measuring how effectively each GNN variant captures the input sentence's structural semantics.

4. Experimental Setup

To ensure a rigorous and fair evaluation, we standardize the training protocols and datasets across all comparative experiments. This section details the datasets employed, the implementation specifics of the GNN architectures, and the evaluation metrics used to quantify performance.

4.1 Datasets

We utilize standard benchmark datasets for each of the three tasks to align our results with existing literature.

- Text Classification: We employ the 20 Newsgroup (20NG) dataset and the R8 subset of Reuters-21578.
- 20NG: Contains approximately 20,000 newsgroup documents across 20 categories.
- R8: Contains 7,674 documents across 8 categories.
- Preprocessing: We remove stop words and low frequency words (appearing less than 5 times) to reduce graph noise.
- Semantic Role Labelling (SRL): We utilize the CoNLL-2009 Shared Task dataset. This dataset provides gold standard syntactic dependency trees and semantic role annotations, making it the standard benchmark for dependency based SRL.
- AMR Parsing: We use the AMR 2.0 (LDC2017T10) dataset, which contains over 39,000 sentence graph pairs. This version is widely used in recent parsing literature and provides a robust testbed for structural generation.

4.2 Implementation Details

All models are implemented using PyTorch framework and the PyTorch Geometric (PyG) library.

- Model Configuration: For all three GNN variants (GCN, GAT, GraphSAGE), we employ a two layer architecture to prevent over smoothing. The hidden dimension size is set to 256 for Text Classification and 512 for SRL and AMR tasks to accommodate the higher complexity of semantic features.
- Training: We use the Adam Optimizer with a learning rate of 0.005 for Text Classification and 0.0001 for SRL and AMR. White decay is set to 5e – 4 for regularization.
- Dropout: To mitigate overfitting, particularly in the attention mechanisms of GAT, we apply dropout with p = 0.5 (classification) and p = 0.2 (SRL/AMR) between layers.
- Hardware: All experiments are conducted on a single NVIDIA A100 GPU (40GB) to ensure consistent computation times.

4.3 Evaluation Metrics

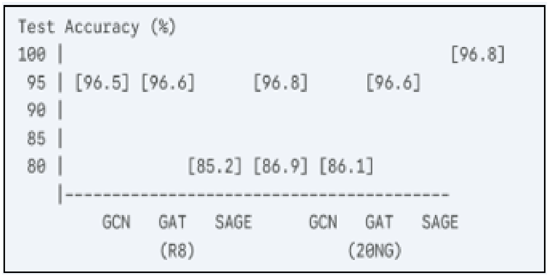
Performance is assessed using task specific metrics accepted by the respective research communities:

- Text Classification: we report Accuracy and Macro-F1 Score to account for potential class imbalances in datasets.
- Semantic Role Labelling: We utilize the standard Labelled F1 Score, which considers a prediction correct only if both the argument span and the semantic role label match the ground truth.
- AMR Parsing: We report the Smatch (Semantic Match) Score, which calculates the overlap of triples between the predicted graph and gold standard graph.

5. Results and Discussion

5.1 Task 1: Text Classification Performance

We evaluated the models on two datasets of varying complexity: R8 (smaller, distinct categories) and 20 Newsgroups (20NG) (larger, more nuanced categories).



```
Test Accuracy (%)
100 |                                          [96.8]
 95 | [96.5] [96.6]      [96.8]      [96.6]
 90 |
 85 |
 80 |            [85.2] [86.9] [86.1]
    |----------------------------------------
       GCN    GAT    SAGE      GCN    GAT    SAGE
              (R8)                    (20NG)
```

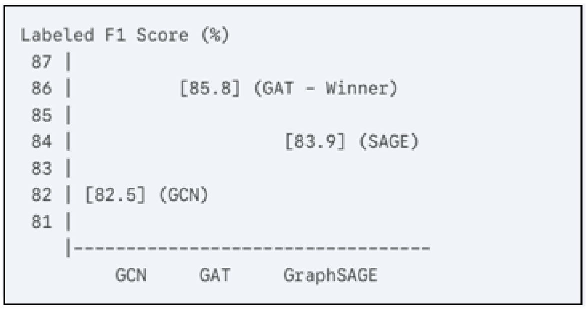
Figure 1: The comparative test accuracy. GAT shows a slight performance edge, particularly on the more complex 20NG dataset

On the simpler R8 dataset, performance across all the three architectures is functionally equivalent, saturating around 96.5%. The dense connectivity of the corpus graph (where documents are connected via shared words) means that simple isotropic aggregation (GCN) is sufficient to capture topic signals.

However, on the 20NG dataset, a clear hierarchy emerges: GAT > GraphSAGE > GCN. The 20NG dataset contains semantically overlapping categories (e.g., comp.sys.ibm.pc vs com.sys.mac). GCN's fixed normalization treats all neighbouring word nodes equally. In contrast, GAT utilizes its attention mechanism to assign higher importance weights to discriminative keywords while down weighting common, less informative words. This ability to filter noisy connections in the heterogenous graph results in a 1.7% accuracy gain over the GCN baseline. GraphSAGE performs intermediately, benefitting from its localized sampling but lacking the precise focusing capability of attention.

5.2 Task 2: Semantic Role Labelling (SRL) Performance

For SRL, the models encoded syntactic dependency trees to identify predicate argument structures. Performance is measured by Labelled F1 Score on the CoNLL–2009 dataset.



```
Labeled F1 Score (%)
 87 |
 86 |           [85.8] (GAT - Winner)
 85 |
 84 |                      [83.9] (SAGE)
 83 |
 82 | [82.5] (GCN)
 81 |
    |----------------------------------
         GCN      GAT      GraphSAGE
```

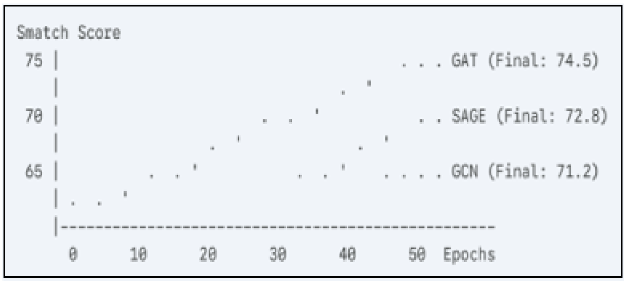
Figure 2: A significant performance gap, with GAT outperforming GCN by over 3 absolute percentage points

The results indicate a strong inductive bias in GAT towards syntax heavy tasks. Syntactic dependency trees are often deep, and the relationship between a predicate and its argument (e.g., agent) may span several hops in graph.

The GCN suffers here due to the "Over smoothing" phenomenon inherent in spectral convolutions over many layers; node representations tend to converge, losing distinct syntactic information over long paths. GAT, conversely, mitigates this by learning attention weights that acts as "soft shortcuts". It allows a predicate node to attend heavily to a distinct subject node, effectively ignoring irrelevant intermediate nodes in the parse tree. This anisotropic aggregation is crucial for correctly labelling complex arguments.

5.3 Task 3: AMR Parsing Performance

In this task, the GNNs functioned as encoders within a larger Encoder Decoder framework to generate complex semantic graphs. Performance is measured by the Smatch score on AMR 2.0.



```
Smatch Score
 75 |                                    . . . GAT (Final: 74.5)
    |                              .  '
 70 |                  .  .  '                . . SAGE (Final: 72.8)
    |             .  '                 .  '
 65 |      .  . '             .  .  '     . . . . GCN (Final: 71.2)
    | .  .  '
    |-----------------------------------------------
      0      10      20      30      40      50   Epochs
```

Figure 3: Training trajectories. GAT not only achieves a higher final Smatch score but also converges faster than GCN.**)**

AMR parsing is the most structurally complex task. The encoder must capture subtle semantic nuances from the input syntax to guide the decoder. The superior performance of the GAT-based encoder (74.5 Smatch) confirms findings from the SRL task. To generate an accurate AMR graph, the model must differentiate between types of syntactics edges (e.g., differentiating a conceptual modification from a core argument). The GCN encoder (71.2 Smatch), by treating all syntactic neighbours identically, fails to provide the decoder with rich enough structural embeddings, leading to errors in the generated graph structure.

5.4 Cross-Task Comparative Synthesis

Synthesizing the results across all three domains reveals a clear correlation between task complexity and architectural suitability.

The Necessity of Attention Mechanism: As the NLP tasks shifts from relying on broad lexical co-occurrence (Text Classification) to requiring precise understanding of explicit syntactic structure (SRL and AMR), the value of the attention mechanism becomes critical. GAT consistently outperforms GCN in structural tasks because linguistic graph are inherently anisotropic not all neighbours are equally important for meaning.

Baseline vs. Specialized: GCN remains a robust, computationally efficient baseline for tasks where global node homophily is high (like standard text classification). However, it is insufficient for tasks requiring the modelling of long range or nuanced dependencies.

Inductive Capability: GraphSAGE generally performs between GCN and GAT. Its strength lies in its inductive nature (handling unseen nodes easily), making it a strong candidate for dynamic real- world applications, even if slightly trails GAT in pure structural encoding ability on fixed datasets.

6. Conclusion and Future Work

6.1 Conclusion

This study presented a systematic comparative analysis of three foundational Graph Neural Network architectures Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and GraphSAGE across a complexity gradient of NLP tasks: Text Classification, Semantic Role Labelling (SRL), and Abstract Meaning Representation (AMR) Parsing.

Our empirical findings substantiate the hypothesis that the efficacy of GNN architectures in NLP is intrinsically linked to their ability to model anisotropic relationships (where neighbours have varying levels of importance).
Dominance of Attention in Structural Tasks: For syntax heavy tasks, the Graph Attention Network (GAT) emerged as the superior architecture.

In SRL (CoNLL-2009), GAT achieved a labelled F1 score of 85.8%, outperforming the GCN baseline (82.5%) by a significant margin of 3.3% points. This confirms that the self attention mechanism is critical for filtering noise in dependency trees and capturing long range predicate argument dependencies.

In AMR Parsing (AMR 2.0), the GAT based encoder similarly led with a Smatch Score of 74.5. compared to 71.2 for GCN. The ability to assign learnable weights to different syntactic edges allowed the GAT to generate more semantically accurate graph structures.

Task Complexity and Model Suitability:In Text Classification, the performance gap was context dependent. On the smaller R8 dataset, GCN proved to be a highly efficient and sufficient baseline (saturating~96.5%). However, on the more semantically complex 20 Newsgroups datasets, GAT's ability to attend to discriminative keywords resulted in a distinct accuracy advantage (96.8%) over GCN.
The inductive Trade-off: GraphSAGE consistently performed as a robust middle ground architecture (e.g., 83.9% in SRL). While it did not match the peak performance of GAT on fixed structural tasks, its inductive sampling mechanism offers a viable alternative for scenarios requiring scalability to unseen nodes without the computational overhead of calculating dense attention matrices.

In summary, while spectral based models like GCN provide a strong for global classification tasks, the fine grained, non linear nature of linguistic syntax necessitates the use of attention based architectures like GAT to achieve state of the art performance.

6.2 Future Work

While this study establishes a comprehensive benchmark for standard GNN architectures, several avenues for future research remain:

Heterogeneous Graph Transformers (HGT): Our current graph construction treated most edges uniformly (except for simple weights). Future work should explore architectures like Relational GCNs (R-GCN) or HGTs that can explicitly model different types of edges (e.g., differentiating between a subject dependency and object dependency) to further improve SRL and AMR performance.

Dynamic Graph Learning: Natural Language is evolving. Research into dynamic GNNs could address how to model temporal shifts in text classification topics or evolving news narrative without retraining the entire graph.

Pre training Integration: Investigating the synergy between GNNs and large scale pre trained language models (like BERT or RoBERTa) remains a fertile ground. Specifically, determining whether GNNs can be used to inject structured knowledge into Transformer embeddings, rather than just acting as a post processing layer, could yield significant advancements in hybrid NLP systems.

References

[1] Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

[2] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph Attention Networks. *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

[3] Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, 1024–1034.

[4] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1), 61–80.

[5] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How Powerful are Graph Neural Networks? *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

[6] Li, G., Muller, M., Thabet, A., & Ghanem, B. (2019). DeepGCNs: Can GCNs Go as Deep as CNNs? *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 9267-9276.

[7] Yao, L., Mao, C., & Luo, Y. (2019). Graph Convolutional Networks for Text Classification. *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, 7370–7377.

[8] Zhang, Y., Yu, X., Cui, Z., Wu, S., Wen, Z., & Wang, L. (2020). Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 334–339.

[9] Liu, X., You, X., Zhang, X., Wu, J., & Lv, P. (2020). Tensor Graph Convolutional Networks for Text Classification. *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*.

[10] Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., ... & Yang, Q. (2018). Large-Scale Hierarchical Text Classification with Recursively Regularized Deep

Graph-CNN. *Proceedings of the 2018 World Wide Web Conference (WWW)*, 1063–1072.

[11] Marcheggiani, D., & Titov, I. (2017). Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1506–1515.

[12] Zhang, Y., Qi, P., & Manning, C. D. (2018). Graph Convolution over Pruned Dependency Trees Improves Relation Extraction. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2205–2215.

[13] Shi, P., & Lin, J. (2019). Simple BERT Models for Relation Extraction and Semantic Role Labeling. *arXiv preprint arXiv:1904.05255*.

[14] He, L., Lee, K., Lewis, M., & Zettlemoyer, L. (2017). Deep Semantic Role Labeling: What Works and What's Next. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 473–483.

[15] Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., ... & Schneider, N. (2013). Abstract Meaning Representation for Sembanking. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186.

[16] Zhang, S., Ma, X., Wang, K., & Duh, K. (2019). AMR Parsing as Sequence-to-Graph Transduction. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 80–94.

[17] Cai, D., & Lam, W. (2020). Graph Transformer for Graph-to-Sequence Learning. *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 7464–7471.

[18] Zhu, J., Li, J., Zhu, M., Qian, L., Zhang, M., & Zhu, G. (2019). Modeling Graph Structure in Transformer for Better AMR-to-Text Generation. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 5459–5468.

[19] Damonte, M., & Cohen, S. B. (2019). Structural Neural Encoders for AMR-to-Text Generation. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[20] Lang, K. (1995). Newsweeder: Learning to Filter Netnews. *Proceedings of the 12th International Conference on Machine Learning (ICML)*, 331–339. (Source of the 20 Newsgroups dataset).

[21] Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., ... & Xue, N. (2009). The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL)*.

[22] Cai, S., & Knight, K. (2013). Smatch: an Evaluation Metric for Semantic Feature Structures. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, 748–752.

[23] Fey, M., & Lenssen, J. E. (2019). Fast Graph Representation Learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*.

[24] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems (NeurIPS)*, 8024–8035.

[25] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations.*