Syntaxa at BAREC Shared Task 2025: BERTnParse - Fusion of BERT and Dependency Graphs for Readability Prediction

Ahmed Bahloul
School of Computation, Information and Technology, Technical University of Munich
Munich, Germany
ahmed.bahloul@tum.de

Abstract

We describe our system submission to **Task 1 (Sentence-level Readability Assessment)** of the BAREC Shared Task 2025 (Elmadani et al., 2025a), in the **strict track**. Task 1 requires predicting the readability level of an Arabic sentence on a scale from 1 (easiest) to 19 (hardest), reflecting reading difficulty. Our approach integrates contextual and syntactic information by combining pretrained BERT embeddings (Devlin et al., 2019) with a Graph Neural Network (GNN) (Zhou et al., 2021) over dependency parse trees (Kipf and Welling, 2017). Our hypothesis is that readability is influenced not only by word choice but also by syntactic complexity—especially in morphologically rich languages like Arabic (Habash, 2010). To capture both aspects, we represent each sentence as a dependency graph with BERT token embeddings as node features, and use a GNN to model the syntactic structure. Experimental results show that our syntax-aware model improves over a strong BERT baseline, highlighting the value of structural linguistic information for fine-grained readability classification.[1]

1 Introduction

Readability assessment aims to estimate the difficulty of a text for a given audience. For Arabic, this task is particularly challenging due to the language's rich morphology, flexible word order, and cliticization. Sentence-level readability prediction demands models that capture subtle syntactic and semantic cues. While transformer-based models like AraBERTv2 (Antoun et al., 2020) encode deep lexical features, they often underutilize syntactic structure—an important aspect of textual complexity.

We propose a hybrid architecture that integrates *syntactic dependency graphs* with *contextual embeddings* from AraBERTv2 for Arabic sentence-level readability prediction. Dependency trees are parsed into graphs and processed with a Graph Neural Network (GNN), while token embeddings from AraBERTv2 are used to represent semantic content. The resulting model jointly reasons over both syntactic and contextual signals.

While GNNs have been combined with transformers in tasks like QA (Yasunaga et al., 2021), document classification (Zhang et al., 2020), and semantic role labeling (Marcheggiani and Titov, 2017), such architectures have not been applied to **readability assessment**. For Arabic, prior work relies on feature-based or PLM-only methods (Liberato et al., 2024) or on word-level readability tools (Hazim et al., 2022; Al Khalil et al., 2020), rather than sentence-level prediction. Recent multilingual efforts such as ReadMe++ (Naous et al., 2024) evaluate both supervised and prompting-based methods, as well as unsupervised approaches, but all rely solely on pretrained language models without incorporating syntactic structure. This leaves the impact of syntax underexplored—especially in morphologically rich languages. The closest related work is by Ivanov (Ivanov, 2022), who compares syntax-based GNNs using fastText embeddings and BERT-based models for sentence complexity in Russian, but does not integrate syntactic and contextual representations into a unified model.

In addition to our architecture, we propose a novel alignment strategy that merges AraBERT subword embeddings and dependency parse nodes into **word-level units**. This is crucial for Arabic, where clitics and morphology lead to tokenization mismatches. Prior work often sidesteps this mismatch by propagating labels across subwords, but we instead ensure structural and semantic alignment through node merging and embedding pooling, enabling effective message passing in the graph.

Contributions: (1) We propose a syntax-aware model that fuses GNN-based syntactic representa-

[1]Code available at: https://github.com/ahmedehabb/BERTnParse

Proceedings of The Third Arabic Natural Language Processing Conference, pages 253–260

Level	Arabic	Transliteration (HSB)	English
6	هنا يلتقي ماجد كل أسبوع بأصدقائه	hnA yltqy mAjd kl Âsbwς bÂSdqAŷh	Here Majid meets his friends every week.
11	ما هو المقصود دول عدم الانحياز؟	mA hw AlmqSwd dwl ςdm AlAnHyAz?	What is meant by the Non-Aligned Movement?

Table 1: Example BAREC sentences with readability levels, CAMeL Tools HSB transliterations, and English glosses.

tions with AraBERTv2 for Arabic sentence-level readability. (2) We introduce a word-level alignment method addressing tokenization mismatches between BERT and dependency parses. (3) We improve over a strong BERT baseline on the BAREC corpus (Elmadani et al., 2025b), especially for complex sentences.

2 Data

We evaluate our approach on the **Balanced Arabic Readability Evaluation Corpus (BAREC)** (Elmadani et al., 2025b), released as part of the BAREC Shared Task 2025. The dataset comprises Arabic sentences labeled with **19 readability levels** (1 = easiest, 19 = hardest), covering diverse topics and genres.

The dataset is split as follows:

- **Train set**: 54,845 sentences
- **Dev set**: 7,310 sentences
- **Test set**: 7,286 sentences
- **Blind Test set**: 3,420 sentences

Each sentence is annotated with a readability level following detailed linguistic and pedagogical guidelines (Habash et al., 2025). To illustrate the dataset, we provide examples with their readability levels, CAMeL Tools HSB transliterations (Habash et al., 2007), and English glosses.[2] Table 1 shows two representative examples.

Additional Arabic readability resources, such as the SAMER corpus (Alhafni et al., 2024), may be useful for future research. However, in line with the **strict track** guidelines of the BAREC Shared Task—where models must be trained exclusively on the BAREC training set—we restrict our experiments to the BAREC dataset only.

[2]Transliteration via the CAMeL Tools CLI: `camel_transliterate -s ar2hsb < file`

3 Methodology

Our approach combines contextual embeddings from AraBERTv2 (Antoun et al., 2020) with a Graph Neural Network (GNN) applied to the syntactic dependency graph of each input sentence. This design enables the model to jointly capture lexical semantics and syntactic structure, addressing key challenges in fine-grained Arabic readability prediction.

3.1 Input Representation

Given an input Arabic sentence $S = (w_1, w_2, \ldots, w_n)$, we first obtain token-level contextual embeddings $\mathbf{h}_i^{\text{BERT}} \in \mathbb{R}^d$ from AraBERTv2, where d is the embedding dimension. AraBERTv2 parameters are **fine-tuned** during training to adapt to the readability task.

Simultaneously, we parse S using CamelParser2.0 (Elshabrawy et al., 2023) to obtain a dependency graph $G = (V, E)$, where $V = \{w_i\}$ are nodes corresponding to tokens, and edges $E \subseteq V \times V$ represent syntactic relations. Each node $w_i \in V$ is also associated with a part-of-speech (POS) tag, and each edge $e = (w_i \to w_j) \in E$ is labeled with a dependency type (e.g., `OBJ`, `SUBJ`).

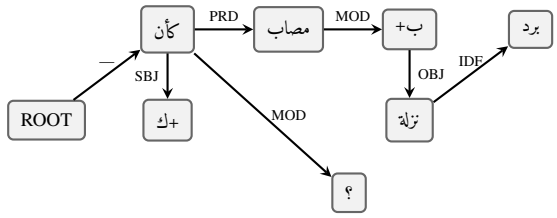
3.2 Token Alignment and Word-Level Processing

A key design decision in our system is to operate at the *word level* rather than on subword units. While AraBERTv2 employs WordPiece tokenization (Wu et al., 2016), which splits words into subword segments, the dependency parser outputs nodes that often correspond to grammatical morphemes or clitics. For example, the Arabic word سأصيدهما (transliteration: sÂSydhmA, translation: "I will catch them") is segmented into the future tense particle +س (s+, PART), the verb stem أصيد (Aṣyd, VRB), and the object pronoun suffix +هما (+hmA, NOM). This linguistically motivated segmentation differs from the subword units generated by BERT,

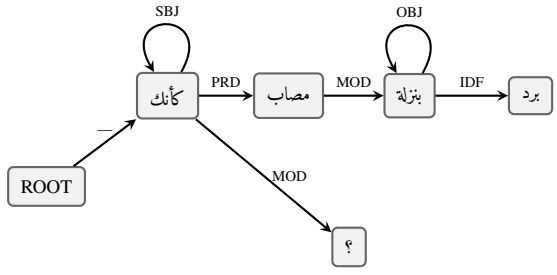
which are learned based on frequency statistics.

To resolve this mismatch, we average the AraBERTv2 subword embeddings into a single word-level vector, following the general idea of leveraging subword information for richer word representations (Bojanowski et al., 2017). On the parsing side, we merge subword nodes (e.g., clitics) and their associated edges into unified word-level graph nodes. This ensures that each word is consistently represented by both a single graph node and a single embedding. Figure 1 illustrates this process by comparing the original token-level dependency graph with the merged word-level version. A detailed description of the merging procedure, along with transliteration and English glosses for the example sentence, is provided in Section 3.3.

This alignment is critical for ensuring consistent and interpretable graph structures. It eliminates discrepancies between the number of embedding vectors and graph nodes, enabling meaningful message passing and feature aggregation in the GNN. To our knowledge, this approach to harmonizing tokenization granularity in Arabic is novel and effectively addresses challenges posed by the language's rich morphology and syntactic structure.



(a) Original token-level dependency graph.



(b) Merged word-level dependency graph.

Figure 1: Comparison between token-level and merged word-level dependency graphs for the Arabic phrase كأنك مصاب بنزلة برد؟

3.3 Subword-to-Word Merging for Dependency Graphs

We parsed the Arabic sentence كأنك مصاب بنزلة برد؟ using CAMeLParser2.0, which outputs token-level dependencies, including affixes and clitics as separate units.

Sentence example with transliteration and translation:

Original: كأنك مصاب بنزلة برد؟
Transliteration: kÂnk mSAb bnzlh brd?
Translation: As if you have a cold?

Token list:

1. كأن — PRT (base)
2. +ك — PRON (enclitic)
3. مصاب — ADJ
4. +ب — ADP (prefix)
5. نزلة — NOUN
6. برد — NOUN
7. ؟ — PUNCT

Original edges (token-level): Token ID 0 refers to the artificial ROOT node.

- (1 → 0) —
- (2 → 1) SBJ
- (3 → 1) PRD
- (4 → 3) MOD
- (5 → 4) OBJ
- (6 → 5) IDF
- (7 → 1) MOD

Merging subword tokens: We merged:

- Tokens [1,2] → كأنك
- Tokens [4,5] → بنزلة

All incoming and outgoing edges of the merged tokens are also combined, so that the resulting word node preserves the original dependency relations for consistent graph construction.

Graph Construction We construct the input graph for the GNN as follows:

- **Nodes and node features**: Each node corresponds to a token w_i and is initialized with the corresponding BERT embedding $\mathbf{h}_i^{\text{BERT}}$. POS tags are encoded as learnable embeddings and concatenated to the token representations. The special `[CLS]` token is used to represent the syntactic root of the sentence and serves as the head node in the graph.
- **Edges and edge features**: Directed edges are constructed based on the dependency parse. Each edge is labeled with a dependency relation, which is encoded as a learnable embedding $\mathbf{e}_{\text{rel}}$ and incorporated via edge-aware message passing.

3.4 GNN Architecture and Training

We employ a multi-layer Graph Neural Network based on `TransformerConv` (Shi et al., 2021) layers to propagate syntactic information. Each TransformerConv layer uses multi-head self-attention (4 heads) over nodes, with attention scores modulated by edge attributes.

At layer l, the hidden state of node i is updated by attending over its neighbors $\mathcal{N}(i)$, conditioning on both node features and edge embeddings. Aggregated edge embeddings for incoming and outgoing edges are computed separately and fused into node representations through a linear projection. Formally, the node representations evolve as:

$$\mathbf{h}_i^{(l+1)} = \texttt{TransformerConv}\big(\mathbf{h}_i^{(l)}, \{\mathbf{h}_j^{(l)} : j \in \mathcal{N}(i)\}, \mathbf{e}_{rel}\big)$$

where $\mathbf{e}_{rel}$ are learned edge embeddings processed by an MLP.

After stacking TransformerConv layers with dropout and layer normalization, node features are aggregated via attentional pooling to produce a graph-level embedding $\mathbf{h}_S$.

This embedding is then fed into two parallel fully connected layers, generating logits for two complementary objectives:

$$\mathbf{z}_{\text{CORAL}} = \mathbf{W}_{\text{c}}\mathbf{h}_S + \mathbf{b}_{\text{c}}, \quad \mathbf{z}_{\text{QWK}} = \mathbf{W}_{\text{q}}\mathbf{h}_S + \mathbf{b}_{\text{q}}.$$

The first layer outputs 18 logits corresponding to ordinal thresholds for the CORAL loss (Cao et al., 2020), while the second produces 19 logits for direct classification used by the Quadratic Weighted Kappa (QWK) loss (de La Torre et al., 2018).

To balance ordinal accuracy and agreement quality, we optimize a combined loss:

$$\mathcal{L} = 0.5 \cdot \mathcal{L}_{\text{CORAL}} + 0.5 \cdot \mathcal{L}_{\text{QWK}},$$

where $\mathcal{L}_{\text{QWK}}$ penalizes larger prediction errors more heavily, enhancing robustness to class imbalance and ordinal inconsistencies.

Figure 2 illustrates the overall model pipeline, highlighting the integration of AraBERTv2 and syntactic parsing through a GNN layer for joint representation learning.

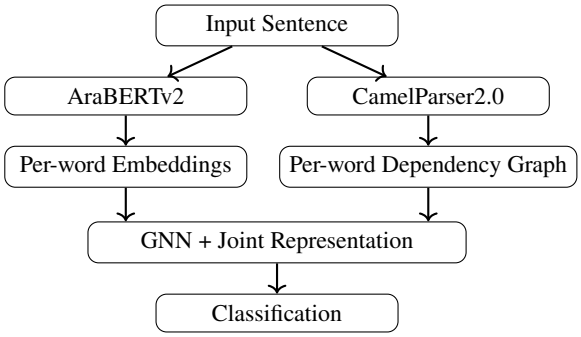


Figure 2: Model architecture integrating lexical and syntactic information for readability prediction.

4 Experimental Setup

4.1 Modeling and Preprocessing

We preprocess each sentence using WordPiece tokenization from AraBERTv2 and dependency parsing via `CamelParser2.0`, which outputs POS tags, syntactic relations, and token-level dependency structures. Following the word-to-subtoken alignment procedure detailed in 3.2, we average subtoken embeddings to form word-level representations. Correspondingly, subtoken-based nodes in the dependency graph are merged into single word-level nodes. The special `[CLS]` token is used to represent the syntactic root of the graph and serves as the anchor node for the sentence-level structure.

Our model integrates these word-level embeddings and dependency graphs through a 4-layer `TransformerConv`-based Graph Neural Network (GNN) with a hidden size of 512. We incorporate learnable embeddings for POS tags and dependency relations. Training is performed with the Adam optimizer using a learning rate of 1×10^{-4}, batch size of 64, dropout rate of 0.2, and early stopping based on validation loss.

Test Set	Model	QWK	Accuracy	Acc ±1	Dist	Acc 7	Acc 5	Acc 3
Internal	Baseline	80.2	**55.9%**	70.0%	1.1	**65.1%**	**69.4%**	**75.2%**
	Our Model	**83.7**	50.5%	**71.9%**	**1.0**	63.1%	68.0%	74.2%
Official Blind	Baseline	81.5	**58.1%**	**72.0%**	**1.0**	**67.7%**	**71.4%**	**76.5%**
	Our Model	**84.3**	51.0%	**72.0 %**	**1.0**	64.4%	68.7%	75.4%

Table 2: Performance on internal and official blind test sets for sentence-level readability prediction.

4.2 Evaluation Metrics

We treat readability assessment as an ordinal classification task. Our primary metric is **Quadratic Weighted Kappa (QWK)**, which penalizes larger prediction errors quadratically. We also report Exact Match Accuracy (Acc19) on the 19-level scale, along with adjacent accuracy (±1), coarser-grained accuracies (Acc7, Acc5, Acc3), and average prediction distance measured by mean absolute error.

5 Results

5.1 Comparison of Model Variants

We evaluate two model variants to assess the contribution of syntactic and structural information:

- **AraBERTv2 baseline**: Fine-tuned on the BAREC-Corpus-v1.0 Word input using cross-entropy loss (Elmadani et al., 2025b).
- **AraBERTv2 + GNN (ours)**: Our proposed approach integrates syntactic dependency parsing using a TransformerConv-based Graph Neural Network over word-level BERT embeddings. Each word node is enriched with POS and syntactic edge features, and the special `[CLS]` token anchors the graph as the syntactic root.

5.2 Performance on Internal and Official Test Sets

Table 2 shows our model achieves superior Quadratic Weighted Kappa (QWK) scores on both internal (83.7 vs. 80.2) and official blind test sets (84.3 vs. 81.5) compared to the baseline, indicating stronger ordinal agreement.

Our method also yields lower average prediction distances (1.0 vs. 1.1 internally) and competitive adjacent accuracy (±1), suggesting more calibrated and consistent predictions. While the baseline slightly outperforms in strict exact match accuracy, our model's improvements in ordinal metrics underscore the benefits of integrating syntactic structure.

6 Conclusion and Future Work

In this work, we explored the integration of contextual semantic features from AraBERTv2 with syntactic structure captured via dependency parsing graphs for the task of Arabic sentence-level readability assessment. Our model incorporates a TransformerConv-based GNN over a dependency graph constructed at the word level, resolving alignment inconsistencies between WordPiece tokenization and morphological segmentation. We demonstrated that augmenting AraBERTv2 with structural information significantly improves performance over a strong BERT-only baseline. Our findings highlight the value of syntactic context in modeling Arabic linguistic complexity and offer a promising direction for fine-grained readability prediction in morphologically rich languages.

For future work, we plan to investigate the use of multilingual pretrained models to leverage cross-lingual knowledge and improve generalization across different Arabic dialects and related languages. Additionally, exploring alternative architectures beyond encoder-only models, such as encoder-decoder or graph transformers, may further enhance the integration of syntactic and semantic information for readability prediction.

References

Muhamed Al Khalil, Nizar Habash, and Zhengyang Jiang. 2020. A large-scale leveled readability lexicon for Standard Arabic. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3053–3062, Marseille, France. European Language Resources Association.

Bashar Alhafni, Reem Hazim, Juan David Pineros Liberato, Muhamed Al Khalil, and Nizar Habash. 2024. The SAMER Arabic text simplification corpus. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16079–16093, Torino, Italia. ELRA and ICCL.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020.

AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France. European Language Resource Association.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. 2020. Rank consistent ordinal regression for neural networks with application to age estimation. *Pattern Recognition Letters*, 140:325–331.

Jordi de La Torre, Domenec Puig, and Aida Valls. 2018. Weighted kappa loss function for multi-class classification of ordinal data in deep learning. *Pattern Recognition Letters*, 105:144–154.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Khalid N. Elmadani, Bashar Alhafni, Hanada Taha, and Nizar Habash. 2025a. BAREC shared task 2025 on Arabic readability assessment. In *Proceedings of the Third Arabic Natural Language Processing Conference*, Suzhou, China. Association for Computational Linguistics.

Khalid N. Elmadani, Nizar Habash, and Hanada Taha-Thomure. 2025b. A large and balanced corpus for fine-grained Arabic readability assessment. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16376–16400, Vienna, Austria. Association for Computational Linguistics.

Ahmed Elshabrawy, Muhammed AbuOdeh, Go Inoue, and Nizar Habash. 2023. CamelParser2.0: A state-of-the-art dependency parser for Arabic. In *Proceedings of ArabicNLP 2023*, pages 170–180, Singapore (Hybrid). Association for Computational Linguistics.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Nizar Habash, Abdelhadi Soudi, and Timothy Buckwalter. 2007. *On Arabic Transliteration*, pages 15–22. Springer Netherlands, Dordrecht.

Nizar Habash, Hanada Taha-Thomure, Khalid N. Elmadani, Zeina Zeino, and Abdallah Abushmaes. 2025. Guidelines for fine-grained sentence-level Arabic readability annotation. In *Proceedings of the 19th Linguistic Annotation Workshop (LAW-XIX-2025)*, pages 359–376, Vienna, Austria. Association for Computational Linguistics.

Reem Hazim, Hind Saddiki, Bashar Alhafni, Muhamed Al Khalil, and Nizar Habash. 2022. Arabic word-level readability visualization for assisted text simplification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 242–249, Abu Dhabi, UAE. Association for Computational Linguistics.

Vladimir Vladimirovich Ivanov. 2022. Sentence-level complexity in russian: An evaluation of bert and graph neural networks. *Frontiers in Artificial Intelligence*, 5:1008411.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *Preprint*, arXiv:1609.02907.

Juan Liberato, Bashar Alhafni, Muhamed Khalil, and Nizar Habash. 2024. Strategies for Arabic readability modeling. In *Proceedings of the Second Arabic Natural Language Processing Conference*, pages 55–66, Bangkok, Thailand. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Tarek Naous, Michael J Ryan, Anton Lavrouk, Mohit Chandra, and Wei Xu. 2024. ReadMe++: Benchmarking multilingual language models for multi-domain readability assessment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12230–12266, Miami, Florida, USA. Association for Computational Linguistics.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. CAMeL tools: An open source python toolkit for Arabic natural language processing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France. European Language Resources Association.

Xintong Shi, Wenzhi Cao, and Sebastian Raschka. 2023. Deep neural networks for rank-consistent ordinal regression based on conditional probabilities. *Pattern Analysis and Applications*, 26(3):941–955.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2021. Masked label prediction: Unified message passing model for semi-supervised classification. *Preprint*, arXiv:2009.03509.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff

Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, and 12 others. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Preprint*, arXiv:1609.08144.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020. Every document owns its structure: Inductive text classification via graph neural networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 334–339, Online. Association for Computational Linguistics.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2021. Graph neural networks: A review of methods and applications. *Preprint*, arXiv:1812.08434.

A Appendix

A.1 Reproducibility Details

A.1.1 Data Preprocessing

- Sentences are cleaned and tokenized using CAMeL Tools.
- We additionally remove *tatweel* characters (ـ) from the text, as many were found to be broken or incorrectly inserted in the middle of words, which can negatively affect tokenization and model performance. For example, the word مدرّسة might appear as مدرّسـة, which we normalize by removing the tatweel and any extra spaces to restore the correct word form.
- POS tag IDs and dependency relation IDs are mapped using predefined dictionaries (`pos2id`, `dep2id`), which we constructed from the train set to cover all observed tags and relations.
- The graph is stored using PyTorch Geometric's `Data` objects with fields: `x`, `edge_index`, `edge_attr`, `pos_tag_ids`, and custom fields like `sentence`.

A.1.2 Model Components

1. AraBERTv2 (Encoder)

- Pretrained weights loaded from `aubmindlab/bert-base-arabertv02`.
- WordPiece tokenization applied via HuggingFace tokenizer.
- Hidden size: 768.
- For each token, embeddings are obtained by mean-pooling over all subtokens aligned via `encoding.word_ids()`, using the mean of the last 4 hidden layers' outputs.
- The first 8 layers of AraBERTv2 are frozen during training, and only the last 4 layers are fine-tuned.

2. Graph Construction

- Sentences are tokenized using CAMeL Tools' (Obeid et al., 2020) morphological segmenter.
- Dependency parses are extracted via the CamelParser2.0.
- For each sentence:
  - Nodes represent surface-level word tokens (segmented, not subword).
  - Directed edges represent syntactic dependencies (head → dependent).
  - Each edge is labeled by the dependency relation (e.g., `SBJ`, `OBJ`).
  - Part-of-speech (POS) tags are extracted per token.
  - The token labeled as `ROOT` by the parser is treated as the syntactic head of the sentence and serves as the root of the dependency tree.

3. Graph Neural Network Architecture Details

- **Node input:** Concatenation of AraBERTv2 embedding and averaged POS tag embedding (32-dimensional).
- **Edge input:** Relation type embedding (hidden size / 2), passed through a feedforward projection.
- **Convolution:** 4-layer `TransformerConv` (with 4 heads), using edge features in attention.

- **Edge Aggregation:** Mean aggregation of outgoing and incoming edge features per node.
- **Normalization:** LayerNorm applied after each GNN layer.
- **Pooling:** `AttentionalAggregation` over graph-level node embeddings.
- **Classifier heads:** One linear layer with $C - 1$ units for CORAL ordinal regression, and a separate linear head with C units for optimizing the Quadratic Weighted Kappa (QWK) loss.

A.2 Ablation Studies and Design Choices

During model development, we conducted extensive ablation studies to identify the most effective architectural components for our task. We evaluated various graph convolutional layers from the `torch_geometric.nn` library, including `NNConv`, `GCNConv`, `GATv2Conv`, and `GraphConv`. Among these, the `TransformerConv` layer consistently achieved the best performance, likely due to its ability to incorporate edge features directly into the attention mechanism and its use of multi-head attention, which captures complex relational patterns between nodes more effectively.

In terms of loss functions, we experimented with a range of objectives, including regression losses, cross-entropy loss, CORN loss (Shi et al., 2023), and direct optimization of the quadratic weighted kappa (QWK) metric. Our final setup combines CORAL loss (Cao et al., 2020) with the weighted QWK loss (de La Torre et al., 2018), yielding improved convergence and performance. This hybrid objective leverages the ordinal-aware structure of CORAL while directly aligning training with the evaluation metric through QWK.

These empirical findings guided our final model design. We recommend using the `TransformerConv` layer in conjunction with a CORAL + QWK loss for tasks involving graph-based ordinal classification.