Word Complexity Prediction Through ML-Based Contextual Analysis

Muhammad Uzzam, Amal Htait,
College of Engineering and Physical Sciences, Aston University, Birmingham, UK
uzzam28@gmail.com, a.htait@aston.ac.uk

Abstract: This paper conducted a comparative evaluation two approaches for predicting word complexity using contextual sentence information, a challenge that traditional methods often struggle to address. Two distinct methods were explored in this work. The first approach combines XLNet word embeddings with a Random Forest classifier to processes both sentence and word embeddings to predict complexity levels. The second approach employs a dual Bidirectional Encoder Representations from Transformers (BERT) model, consisting of two separate models: one for sentence-level complexity and another for word-level complexity, with their predictions combined for more context-sensitive result. A diverse dataset covering the domains of religion, biomedical, and parliamentary texts was used, as it is pre-categorised into five complexity levels (Very-easy, Easy, Medium, Hard, Very-hard). To ensure balanced class representation, data augmentation techniques were applied. Evaluation metrics revealed that the XLNet-based model has performed slightly superior to dual-BERT method, achieving macro-average F1-score of 0.79, excelling particularly at identifying highly complex words (F1-score = 0.95). In comparison, dual-BERT achieved a macro-average F1-score equal to 0.78.

Keywords: Natural Language Processing, Complexity Prediction, Machine Learning, XLNet, BERT

1. Introduction

This paper presents a comparative study of two machine learning-based approaches for predicting word complexity within sentence contexts. We evaluate how well these models use contextual information to assess word difficulty. Understanding word complexity in context is a key challenge in Natural Language Processing (NLP), with significant implications for education, accessibility, and language learning. NLP's rapid evolution has increased the need for techniques that interpret words in context. Word complexity depends on several factors, with grammatical structure playing a key role. Word order can influence how difficult a word appears. Additionally, surrounding words greatly affect perceived complexity. A word that seems simple in isolation may become difficult when used with challenging vocabulary, and vice versa.

Word complexity cannot be judged alone, as identical spellings may convey different meanings depending on context. For example, ring can refer to jewellery or a phone sound; bat to an animal or a sports item; and park to a place or an action. Traditional methods often ignore context when assessing word complexity [1], which limits their accuracy. Machine learning addresses this by capturing nuanced word meanings based on usage. By comparing two established approaches, we explore how each handles complexity and the practical advantages they offer. This analysis has educational applications, helping automatically identify and explain complex words to support comprehension for learners and non-native speakers.

The remainder of the paper is structured as follows: Section 2 reviews related work in word complexity prediction and contextual embeddings. Section 3 details the two evaluated models. Section 4 presents the comparative results and evaluation metrics. Section 5 presents future Work to mitigate key limitations and threats to validity and Section 6 provides the conclusion along with an analysis of key achievements.

2. Related Work

The study of word complexity word complexity has undergone significant evolution and has evolved from early rule-based methods to advanced machine learning and deep learning models. Initial approaches used readability formulas based on surface-level linguistic features. For instance, Flesch's Reading Ease Score [1] assessed readability using sentence length and syllable count, assuming longer sentences and multi-syllabic words are harder. The SMOG index [2] estimated the educational level required to understand a text, while the Dale-Chall formula [3] used a list of familiar words, penalizing rare vocabulary. Though intuitive, these methods overlooked semantic context and syntactic structure, both crucial for assessing word complexity.

The Cloze test [4], introduced in the 1950s, marked a shift toward context-sensitive assessment. By evaluating predictability of missing words, it incorporated human judgment and comprehension, offering a more nuanced measure of lexical difficulty and emphasizing contextual cues.

With the rise of Natural Language Processing (NLP), modeling word complexity moved beyond fixed formulas to computational techniques that capture semantic and contextual relationships. Early vector-based methods like Latent Semantic Analysis (LSA) represented word meanings through co-occurrence patterns [5], but generated static representations, failing to handle polysemy.

Word2Vec [6] introduced neural embeddings that encoded semantic relationships in dense vectors, improving contextual understanding. GloVe [7] combined local context with global statistics, enriching word representations. FastText [8] further addressed rare and morphologically complex words by using sub-word information to form embeddings, even for unseen terms.

Alongside better embeddings, machine learning models like Random Forest [9], SVMs [10], and Naïve Bayes [11] were applied to lexical complexity tasks. These relied on features like word length, frequency, and part-of-speech tags. While effective to a point, they lacked deeper linguistic and semantic comprehension.

Deep learning models brought major advances. Recurrent neural networks, especially Bi-LSTMs, captured long-range dependencies and context-aware predictions [12], processing both preceding and succeeding words to better understand contextual word complexity.

Transformer-based models such as BERT [13] and XLNet [14] pushed performance further. Using self-attention and bidirectional (BERT) or permutation-based (XLNet) training, they captured complex word relationships and offered state-of-the-art results in tasks like complexity prediction.

However, many studies still treat sentence-level features and word-specific attributes separately. Most focus on either context or lexical features rather than integrating both in a unified model.

To bridge this gap, we propose a hybrid approach that merges sentence embeddings with target word features, utilizing XLNet for contextual encoding and Random Forest for classification. This approach combines XLNet's deep contextual understanding with the robustness and interpretability of traditional models. We also use data augmentation to address class imbalance, enhancing model generalizability and reliability.

This paper presents a comparative analysis of two methods: one combining sentence- and word-level features via XLNet and Random Forest, and another employing a deep learning architecture using contextual embeddings. Our goal is to evaluate their relative effectiveness in predicting contextual word complexity.

3. Methodology

3.1. Dataset Preprocessing

This work uses the SemEval LCP 2021 dataset [15], which contains texts from diverse domains such as EU Parliament proceedings, the Bible, and biomedical literature. The data was annotated across five complexity levels: Very Easy, Easy, Medium, Difficult, and Very Difficult, based on word familiarity and contextual clarity.

Key preprocessing steps were applied to reduce data noise that could hinder analysis while preserving semantic integrity:

- Removal of punctuation: Eliminated commas, periods, and hyphens to reduce noise and focus on meaningful features.
- Lowercasing words: Converted all text to lowercase for consistency and better model generalization.
- Removal of null rows: Discarded rows with missing values to maintain data quality.
- Synonym replacement: Augmented underrepresented classes by substituting words with synonyms, improving balance and diversity.

3.2. Method 1: XLNet Embeddings with Random Forest

The first method presents a novel combination of XLNet embeddings of the numerical words and sentences representation with the robust capabilities of the classic machine learning model, Random Forest. XLNet embedding captures the contextual relationships between words and sentences, while the Random Forest classifier uses these combined embeddings as independent variables and the complexity level of target words as a dependent variable. This enables the prediction of word complexity within the context of its sentence. By obtaining separate embeddings for words and sentences and merging them into a single variable, this approach improves the understanding of a word's contextual properties within its sentence.

3.2.1 Overview of XLNet Model

XLNet is a pre-trained transformer model trained on large-scale datasets [16]. It provides contextual embeddings by modeling the relationships among all words in a sentence. Unlike traditional models with a left-to-right approach, XLNet captures bidirectional dependencies by considering all possible word orderings, resulting in richer representations of both words and sentences [17]. XLNet has key advantages: it generates embeddings for both individual words and full sentences [18], and its autoregressive architecture captures long-range dependencies, enabling it to produce highly contextualized embeddings [19].

3.2.2 XLNet Model: Implementation Steps

The implementation of the XLNet model involves key steps for extracting contextual embeddings of words and sentences, which we list as follows.

1. Embedding Extraction of Words and Sentences: The embedding extraction step is the initial phase of this approach. It uses the XLNet pretrained transformer model to produce numerical representations of both specific target words within a sentence and the entire sentence itself. These embeddings encode rich semantic and syntactic features [16]. After sentence tokenization (Equation 3.1), two sub-steps follow: first, the model generates the sentence embedding (Equation 3.2), which captures the sentence's global context and the syntactic-semantic influence of surrounding words on the target word; and second, it generates the target word embedding (Equation 3.3), which focuses on the inherent attributes of target words individually. These embeddings preserve characteristics that contribute to a word's morphological complexity.

   Implementation Details: We used the 'xlnet-base-cased' model from HuggingFace Transformers for embedding extraction. Both the full sentence and the target word were tokenized and passed through the model separately. The mean of the last hidden state was computed to form a 768-dimensional embedding for each.

   $$H = \text{XLNet}(\text{XLNetTokenizer}(S)) \tag{3.1}$$

   In Equation 3.1, S represents the input text (sentence or target word), which is tokenized and passed to the XLNet model. The output H is a matrix of hidden states for each token in the input.

   $$S_{emb} = \frac{1}{n}\sum_{i=1}^{n} h_i \tag{3.2}$$

In Equation 3.2, S_{emb} is the final sentence embedding. It is calculated by averaging all n token-level hidden states h_i from the sentence.

$$W_{emb} = \frac{1}{m} \sum_{j=1}^{m} h_j \tag{3.3}$$

In Equation 3.3, W_{emb} is the embedding for the target word. It is similarly obtained by averaging the m token-level hidden states h_j from the tokenized target word.

2. Combining the Two Extracted Embedded Features: Once both sentence-level and target word-level embeddings are extracted, they are concatenated into a single unified feature vector (Equation 3.4). This approach effectively represents both global (sentence) and local (word) contexts. The unified vector provides comprehensive input for the classifier by capturing the global context, which reflects the syntactic and semantic structure of the sentence in which the target word appears, as well as the local context, which encodes intrinsic characteristics of the target word, such as length, frequency, and morphological complexity.

   Implementation Details: Each 768-dimensional embedding vector (sentence and word) was concatenated to form a 1536-dimensional combined feature vector. This was done using NumPy's 'concatenate()' method across all rows of the dataset. The resulting feature matrix was stored and later used as input for training the Random Forest classifier.

$$F = [S_{emb}; W_{emb}] \tag{3.4}$$

   In Equation 3.4, S_{emb} and W_{emb} refer to the 768-dimensional embeddings of the sentence and the target word, respectively. The notation $[S_{emb}; W_{emb}]$ represents the concatenation of these two vectors into a single 1536-dimensional feature vector F, capturing both the sentence-level and word-level contextual information.

3. Classification with Random Forest: After extracting XLNet embeddings for both sentences and target words and combining them into a unified vector containing global and local context, the next step is to classify complexity using this vector as input and complexity level as the output, via a random forest classifier. Random Forest is selected for its strengths: robustness in handling high-dimensional vectors (1536 dimensions) from XLNet [9], resistance to overfitting through ensemble decision trees, minimal need for hyper-parameter tuning compared to deep neural networks [9], and interpretability through feature importance scores, which help analyze the contributions of sentence-level versus word-level features.

   The dataset is split into 80% training and 20% testing (Equation 3.5). XLNet embeddings generate numerical representations of sentences and target words, and due to the high-dimensional input, as mentioned above, Random Forest is a suitable choice. It builds multiple decision trees during training and outputs the mode of their predictions [9].

$$(F_{\text{train}}, y_{\text{train}}), (F_{\text{test}}, y_{\text{test}}) \tag{3.5}$$

   Once trained (Equation 3.6), the model is evaluated on the test set, predicting the complexity level of each target word based on its XLNet-generated feature vector. The classifier learns to recognize patterns corresponding to five complexity labels: Very Easy, Easy, Medium, Hard, and Very Hard. The decision-making process proceeds in three steps: first, the model compares test embeddings with the learned decision boundaries; second, it uses multiple decision trees to vote on the complexity level; and third, it predicts the final label based on the majority vote. This ensemble-based approach ensures stability and robustness in the final prediction.

$$f(F) = \arg\max_{c} \sum_{t=1}^{T} g_t(F) \tag{3.6}$$

   With T trees.

3.3. Method 2: Dual BERT Model

In method 2, two separate BERT models are combined to predict the complexity level of a target word within the context of the sentence. In the first model, sentences are embedded using BERT embedding, and these embedded values are trained with complexity level for prediction. In the second model, target words are embedded using BERT embedding, and these values are trained for complexity level prediction. For the final prediction, a weightage 50% is given to both models. This ensures that the contribution of the sentence-level prediction 50% maintains the global context of a sentence and captures the syntactic and semantic influence of other words on the specific target word. The remaining 50% contribution to the prediction at the target word level is responsible for the individual complexity and representation of the target word, including intrinsic features at the word level and morphological complexity.

3.3.1 Overview of BERT

Bidirectional Encoder Representations from Transformers (BERT) is a deep learning model used to produce context-aware embeddings, which are numerical encodings of words and sentences [13]. BERT produces contextually rich and syntactically aware numerical embeddings because its architecture consists of 12 transformer layers, 768 hidden units, and 12 attention heads. BERT's self-attention mechanism allows it to consider the relationship of every single word in a sentence with other words in a bidirectional manner—analyzing relationships from both left-to-right and right-to-left [13].

3.3.2 Dual-BERT Model: Implementation Steps

The implementation of the Dual-BERT model follows key steps to extract contextual embeddings for both words and sentences, as outlined below.

1. Data Split and Tokenization: The dataset is first split into 80% training and 20% testing sets. Sentences and target words are then tokenized separately using the BERT Tokenizer and converted into numerical embeddings, with complexity levels encoded from 0 to 4 for processing. A batch size of 16 is chosen for efficient evaluation.

   Let S be the input sentence and W the target word. The tokenized inputs are passed into two separate BERT models:

   $$H_s = \text{BERT}_{\text{sentence}}(\text{Tokenizer}(S)), \quad H_w = \text{BERT}_{\text{word}}(\text{Tokenizer}(W)) \tag{3.7}$$

2. Model Training: Two independent BERT models were fine-tuned using the HuggingFace bert-base-uncased architecture. One model was trained on full sentence inputs, while the other was trained on isolated target words, with both mapped to five complexity levels. Input sequences were tokenized and truncated to a maximum length of 128 tokens.

   Each model outputs raw logits—z_s for the sentence model and z_w for the word model—which are converted into class probability distributions using the softmax function, as shown in (Equation 3.8):

   $$P_s = \text{softmax}(z_s), \quad P_w = \text{softmax}(z_w) \tag{3.8}$$

   Model training was conducted over 3 epochs using the AdamW optimizer with a learning rate of 2e-5 and a batch size of 16. A weight decay of 0.01 and a warmup period of 500 steps were applied for regularization. The training objective was to minimize the cross-entropy loss between the predicted and true class distributions, defined in (Equation 3.9):

   $$\mathcal{L} = -\sum_{c=1}^{C} y_c \log P_c \tag{3.9}$$

   Here, $C = 5$ denotes the number of complexity classes, y_c is the one-hot encoded true label, and P_c is the predicted probability for class c. Early stopping was employed based on validation loss to reduce overfitting. All experiments were conducted using the HuggingFace and Trainer API.

3. Prediction and Fusion Strategy: After training, both BERT models were used to independently predict the complexity level of the target word in its sentence context. Each model output raw logits, which were passed through a softmax function to produce probability distributions over the five complexity classes, as defined earlier in (Equation 3.8).

   To integrate the predictions from both models, a fusion strategy was applied. Specifically, the final prediction probability vector was computed by averaging the softmax probabilities from the sentence model (P_s) and the word model (P_w), as defined in (Equation 3.10):

   $$P_{\text{final}} = 0.5 \cdot P_s + 0.5 \cdot P_w \tag{3.10}$$

   The final predicted class $\hat{y}$ was selected as the class with the highest combined probability in P_{final}, as shown in (Equation3.11):

   $$\hat{y} = \arg\max(P_{\text{final}}) \tag{3.11}$$

   This dual prediction mechanism balances global sentence-level context with local word-level information, enabling the model to make more context-aware and interpretable complexity classifications.

4. Results

4.1. Results of Method 1

The first approach utilizes XLNet embeddings for both sentence and word representations, combined with a Random Forest classifier to predict word complexity levels. Table 1 presents the detailed classification metrics.

	Precision	Recall	F1-score
Very Easy	0.84	0.87	0.85
Easy	0.65	0.78	0.71
Medium	0.76	0.53	0.62
Hard	0.84	0.85	0.85
Very Hard	0.91	0.99	0.95
Accuracy			0.80

Table 1. Classification Report Method 1: XLNet word embeddings and Random Forest

As shown in Table 1, this model performs especially well on the Very Hard class (F1 = 0.95), suggesting strong ability to identify highly complex words. Similarly, the model handles the Hard and Very Easy classes effectively, each with F1-scores of 0.85. The Easy class demonstrates moderate performance (F1 = 0.71), while the Medium class yields the lowest F1-score (0.62), indicating a potential challenge in distinguishing this class, possibly due to its semantic proximity to adjacent complexity levels.

4.2. Results of Method 2

The second method uses a dual BERT architecture, where separate models process sentence-level and word-level embeddings. Final predictions are obtained by averaging outputs from both models. Table 2 reports the evaluation metrics.

This model also demonstrates strong performance on the Very Hard (F1 = 0.93) and Hard (F1 = 0.84) categories. The Very Easy class is handled effectively (F1 = 0.83), while performance for the Easy and Medium classes remains more limited, with F1-scores of 0.64 and 0.62 respectively.

	Precision	Recall	F1-score
Very Easy	0.77	0.90	0.83
Easy	0.67	0.62	0.64
Medium	0.70	0.55	0.62
Hard	0.84	0.83	0.84
Very Hard	0.88	0.99	0.93
Accuracy			0.78

Table 2. Classification Report Method 2: Dual BERT

4.3. Comparison and Analysis

A comparison of the two methods shows both perform well in identifying words at the extremes of complexity, especially in the Very Hard and Hard categories. Method 1 (XLNet with Random Forest) achieves F1-scores of 0.95 and 0.85, while Method 2 (Dual BERT) scores 0.93 and 0.84, showing a slight edge for Method 1.

In the Very Easy class, Method 1 again leads with 0.85 versus Method 2's 0.83, indicating better handling of simpler vocabulary and structures through XLNet's combined context embeddings.

For Easy and Medium, Method 1 scores 0.71 and 0.62, while Method 2 scores 0.64 and 0.62, respectively. Method 1 has higher precision in the Medium class, while Method 2 slightly favors recall. This class remains difficult due to ambiguity and overlap with neighboring categories.

Overall, Method 1 shows better consistency, with a macro-average F1-score of 0.7966 versus 0.77 and accuracy of 80% compared to 78%. This suggests that XLNet's context-aware embeddings, combined with Random Forest's ensemble strategy, enhance classification reliability. Method 2 remains a competitive option, offering a simpler, fully deep learning-based architecture.

4.4. Comparison with Baseline Model

As discussed in the Related Work section, Bi-LSTM models have been used in complexity classification due to their strength in modeling sequence dependencies [12]. We implemented a dual-input Bi-LSTM baseline that processes both the sentence and target word separately before merging representations. Results are shown in Table 3.

	Precision	Recall	F1-score
Very Easy	0.83	0.81	0.82
Easy	0.55	0.51	0.53
Medium	0.51	0.54	0.53
Hard	0.78	0.81	0.80
Very Hard	0.97	0.97	0.97
Accuracy			0.73

Table 3. Classification Report Baseline Model: Bi-LSTM (Related Work)

Compared to Method 1 (Table 1) and Method 2 (Table 2), the Bi-LSTM baseline shows clear limitations. While it performs well for the Very Hard class (F1 = 0.97) and reasonably for Very Easy (F1 = 0.82) and Hard (F1 = 0.80), it struggles with the Easy and Medium classes, scoring only 0.53 for both.

Overall, the Bi-LSTM yields 73% accuracy and a macro-average F1 of 0.73—both lower than Method 1 (80.1%, F1 = 0.7966) and Method 2 (78%, F1 = 0.77). These results underscore the effectiveness of transformer-based models, especially Method 1, which combines XLNet embeddings with Random Forest. Their superior context modeling gives them an edge over sequential models like Bi-LSTM.

5. Discussion

Our model currently relies on English-only data from a single dataset, which limits its generalizability. Future work should incorporate multilingual and domain-specific corpora to enhance adaptability. Another promising direction is the use of large language models (LLMs), such as GPT-4, for complexity prediction via prompt engineering. LLMs offer broad language understanding with minimal training [20], but their adoption is hindered by high computational costs and infrastructure demands. Future studies will explore LLMs with tailored prompts, compare them to supervised models, and evaluate efficient alternatives such as prompt tuning, model distillation, or smaller open-source LLMs to strike a balance between accuracy and efficiency.

A key threat to validity was the class imbalance in the dataset: Very Easy (9%), Easy (49%), Medium (28%), Hard (11%), and Very Hard (3%). To address this, synonym-based augmentation was applied to all non-target words in each sentence instance using Python libraries such as NLTK and its WordNet interface. This approach preserved the context of the sentences while generating variations, as the augmented sentences maintained the original meaning. Underrepresented classes were oversampled to match the majority class (Easy); for example, Very Hard entries were augmented up to 14 times. Unique synonyms were generated for each instance to ensure lexical diversity. This strategy reduces bias, improves generalization, and preserves construct validity.

The study revealed some interesting results, showing that the XLNet–Random Forest hybrid slightly outperformed the Dual BERT model, suggesting that hybrid architectures offer improved generalization and robustness. The combination of contextual embeddings with ensemble classifiers proved effective for complexity prediction. Another interesting result was that both models achieved high F1-scores for the Very Hard class, likely due to strong contextual or morphological cues.

6. Conclusion

This paper presents a comparative analysis of two approaches for predicting and classifying the complexity level of a target word within sentence context. The achievements include emphasizing the importance of contextual word complexity, as difficulty can vary with sentence structure and semantics. Rather than relying solely on word-level features, the paper uses contextual embeddings to enhance classification. Two models were implemented: XLNet embeddings with a Random Forest classifier and a Dual BERT model. Both categorize complexity into five levels: Very Easy to Very Hard. XLNet–Random Forest achieved 80% accuracy and a 79% macro F1 score, slightly outperforming Dual BERT's 78% accuracy and 78% F1 score.

References

[1] R. Flesch, "A new readability yardstick," *Journal of Applied Psychology*, no. 3, pp. 221–233, 1948.

[2] G. H. McLaughlin, "Smog grading-a new readability formula," *Journal of Reading*, vol. 12, no. 8, pp. 639–646, 1969.

[3] V. P. E. Reyes, "Exploring the use of the phoneme frequency scale method in determining word difficulty levels and readability scores," *Proceedings of the 7th International Conference on Information and Education Technology*, pp. 284–288, 2019.

[4] J. C. Alderson, "The cloze procedure and proficiency in english as a foreign language," *TESOL Quarterly*, pp. 219–227, 1979.

[5] N. E. Evangelopoulos, "Latent semantic analysis," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 4, no. 6, pp. 683–692, 2013.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[7] J. Pennington, R. Socher, and C. D. Manning, "Global vectors for word representation," *Proceedings of EMNLP*, pp. 1532–1543, 2014.

[8] L. S. Costa, I. L. Oliveira, and R. Fileto, "Text classification using embeddings: a survey," *Knowledge and Information Systems*, vol. 65, no. 7, pp. 2761–2803, 2023.

[9] J. Brooke, T. Baldwin, and A. L. Uitdenbogerd, "Melbourne at semeval 2016 task 11: Classifying type-level word complexity using random forests with corpus and word list features," *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pp. 975–981, 2016.

[10] V. Santucci, F. Rinaldi, and M. Conforti, "Learning to classify text complexity for the italian language using support vector machines," in *Computational Science and Its Applications–ICCSA*, pp. 367–376, 2020.

[11] H. Zhang and D. Li, "Naïve bayes text classifier," *IEEE International Conference on Granular Computing*, p. 708, 2007.

[12] B. Jang, I. Kim, J. W. Kim, and H. Kang, "Bi-lstm model to increase accuracy in text classification: Combining word2vec cnn and attention mechanism," *Applied Sciences*, vol. 10, no. 17, p. 5841, 2020.

[13] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?," *Chinese Computational Linguistics: 18th China National Conference, CCL 2019*, pp. 194–206, 2019.

[14] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *arXiv preprint arXiv:1906.08237*, 2019.

[15] M. Shardlow, R. Evans, G. H. Paetzold, and M. Zampieri, "Semeval-2021 task 1: Lexical complexity prediction," *arXiv preprint arXiv:2106.00473*, 2021.

[16] P. Rajapaksha, R. Farahbakhsh, and N. Crespi, "Bert, xlnet or roberta: The best transfer learning model to detect clickbaits," *IEEE Access*, vol. 9, pp. 154704–154716, 2021.

[17] A. Shahriar, D. Pandit, and M. S. Rahman, "Xlnet-cnn: Combining global context understanding of xlnet with local context capture through convolution for improved multi-label text classification," *Proceedings of the 11th International Conference on Networking, Systems, and Security*, pp. 24–31, December 2024.

[18] A. H. Sweidan, N. El-Bendary, and H. Al-Feel, "Sentence-level aspect-based sentiment analysis for classifying adverse drug reactions (adrs) using hybrid ontology-xlnet transfer learning," *IEEE Access*, vol. 9, pp. 90828–90846, 2021.

[19] H. Ye, Z. Chen, D. H. Wang, and B. Davison, "Pretrained generalized autoregressive model with adaptive probabilistic label clusters for extreme multi-label text classification," *Proceedings of the International Conference on Machine Learning*, pp. 10809–10819, November 2020.

[20] D. Youvan, "Building and running large-scale language models: The infrastructure and techniques behind gpt-4," *Journal of Artificial Intelligence Infrastructure*, 2024.