ChuLo: Chunk-Level Key Information Representation for Long Document Understanding

Anonymous ACL submission

Abstract

Transformer-based models have achieved remarkable success in various Natural Language Processing (NLP) tasks, yet their ability to handle long documents is constrained by computational limitations. Traditional approaches, such as truncating inputs, sparse self-attention, and chunking, attempt to mitigate these issues, but they often lead to information loss and hinder the model's ability to capture long-range dependencies. In this paper, we introduce ChuLo, a novel chunk representation method for long document understanding that addresses these limitations. Our ChuLo groups input tokens using unsupervised keyphrase extraction, emphasizing semantically important keyphrase based chunks to retain core document content while reducing input length. This approach minimizes information loss and improves the efficiency of Transformer-based models. Preserving all tokens in long document understanding, especially token classification tasks, is important to ensure that fine-grained annotations, which depend on the entire sequence context, are not lost. We evaluate our method on multiple long document classification tasks and long document token classification tasks, demonstrating its effectiveness through comprehensive qualitative and quantitative analysis.

1 Introduction

Transformer-based models (Vaswani et al., 2017), including LLMs (Radford, 2018; Radford et al., 2019; Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023a,b; Chowdhery et al., 2023; Anil et al., 2023; Dubey et al., 2024), have achieved remarkable success across a wide range of Natural Language Processing (NLP) tasks, including Machine Translation, Text Summarization, Text Generation, and Text Classification. A key factor behind their success is the self-attention mechanism, which allows the model to capture long-range dependencies by computing the similarity between any two tokens and aggregating information accordingly. However, this mechanism incurs a quadratic computational cost in terms of both time and space, relative to input length. This computational burden makes it difficult for Transformer-based models to scale to long documents, limiting their application to real-world data with unrestricted document lengths. To address this challenge, several approaches have been proposed for applying Transformer-based models to long documents while managing computational resources. One of them is truncating, where the model discards content exceeding a predefined input length. For instance, BERT (Kenton and Toutanova, 2019) processes up to 512 tokens, and LLaMa (Touvron et al., 2023a) handles up to 2048 tokens, with any additional content being ignored. Another one is sparse self-attention, which reduces computational complexity by restricting each query token to attend only to a subset of key tokens (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Wei et al., 2021; Li et al., 2023a). Lastly, chunking divides long documents into smaller, manageable segments that are processed independently by the model (Zhao et al., 2021; Zhang et al., 2022).

While these methods enable Transformer-based models to process long documents, they have limitations. Truncation risks discarding important information that falls beyond the maximum input length. Although more efficient, Sparse attention reduces each token's receptive field, leading to potential information loss from the neglected tokens. Similarly, chunking breaks the input into isolated segments, which can disrupt long-range dependencies critical for a comprehensive understanding of the document. Preserving all tokens is particularly important in tasks that require fine-grained token-level understanding, such as token classification. In such tasks, dropping tokens can severely impact the accuracy of fine-grained annotations, which often depend on the full context of the document. Therefore, there is a need for methods that can handle long documents efficiently while retaining all key information from the input.

In this paper, we introduce ChuLo, a novel chunk-level key information representation method that addresses these challenges in long document classification and token classification. Our method reduces input length while minimizing information loss by strategically grouping tokens using unsupervised keyphrase extraction. By identifying and emphasizing semantically important tokens, ChuLo ensures that each chunk retains the core content of the document. The resulting chunk representation is used for training Transformer models, with more weight assigned to keyphrases to make them more salient in each chunk. We evaluate ChuLo on various long document classification tasks and long document token classification tasks, demonstrating its effectiveness through competitive results and thorough analysis.

The key contributions of this paper are as follows: **1) Novel Chunk Representation Method**: We introduce ChuLo, a chunk representation method for long document understanding that leverages unsupervised keyphrase extraction to prioritize semantically important information, effectively reducing input length while preserving core content. **2) Enhanced Document and Token Classification**: Our proposed method is designed to handle both document-level and token-level tasks, addressing the limitations of existing models in retaining fine-grained annotations and global context in long documents. **3) Scalable and Efficient Solution**: ChuLo offers a scalable and efficient approach for long document understanding, making it suitable for various NLP tasks where handling long-range dependencies and context preservation is critical.

2 Related Work

2.1 Long Document Understanding

Document understanding involves global understanding (e.g., classification) and token-level tasks (e.g., named entity recognition). Transformer-based models face performance issues with long inputs, addressed through input processing and architecture optimization. Input processing methods include truncating tokens beyond the input limit (Park et al., 2022) and chunking, as seen in Hierarchical Transformer (Pappagari et al., 2019) and RoR (Zhao et al., 2021), though these often neglect full document context. Architecture optimizations improve efficiency using sparse attention (Beltagy et al., 2020; Zaheer et al., 2020; Roy et al., 2021) or approximations (Peng et al., 2021; Wang et al., 2020; Choromanski et al., 2020). Other approaches incorporate RNN concepts, such as cache memory (Dai et al., 2019; Hutchins et al., 2022; Li et al., 2023b). These methods balance performance and efficiency, highlighting the need to reduce input length effectively.

For document NER, text length is less studied, with recent work addressing low-resource languages (Çetindağ et al., 2023; Mengliev et al., 2024), complex domains (Park et al., 2023; Bhattacharya et al., 2023), prompt-based methods (Wang et al., 2023; Dagdelen et al., 2024; Hu et al., 2024), and multimodal data (Yu et al., 2023; Zhang et al., 2023). Our work tackles these challenges[1] with a novel chunk representation that preserves semantic information while reducing input length, enhancing both classification and token-level tasks.

2.2 Unsupervised Keyphrase Extraction

Unsupervised keyphrase extraction identifies representative phrases to summarize content without labelled data (Hasan and Ng, 2014). Methods include statistics-based (e.g., TfIdf (El-Beltagy and Rafea, 2009), co-occurrence (Liu et al., 2009), and context statistics (Campos et al., 2020; Won et al., 2019)), graph-based (e.g., TextRank (Mihalcea and Tarau, 2004) and its variants (Wan and Xiao, 2008; Bougouin et al., 2013; Florescu and Caragea, 2017; Yu and Ng, 2018)), and embedding-based approaches (e.g., EmbedRank (Bennani-Smires et al., 2018), SIFRank (Sun et al., 2020), and PromptRank (Kong et al., 2023)). While effective, these methods prioritize phrase extraction and ranking over improving downstream tasks. Our work integrates keyphrase extraction with chunk representation to enhance long document understanding.

3 ChuLo

We propose ChuLo, a novel chunk representation method that enhances long document understanding by reducing input length while preserving semantic content. Unlike existing approaches like truncation and standard chunking, ChuLo minimizes information loss and maintains contextual dependencies. The method segments documents into non-overlapping chunks, integrates key semantic information using unsupervised keyphrase ex-

[1] The summary of Long Document Understanding related works can be found in Appendix A.1

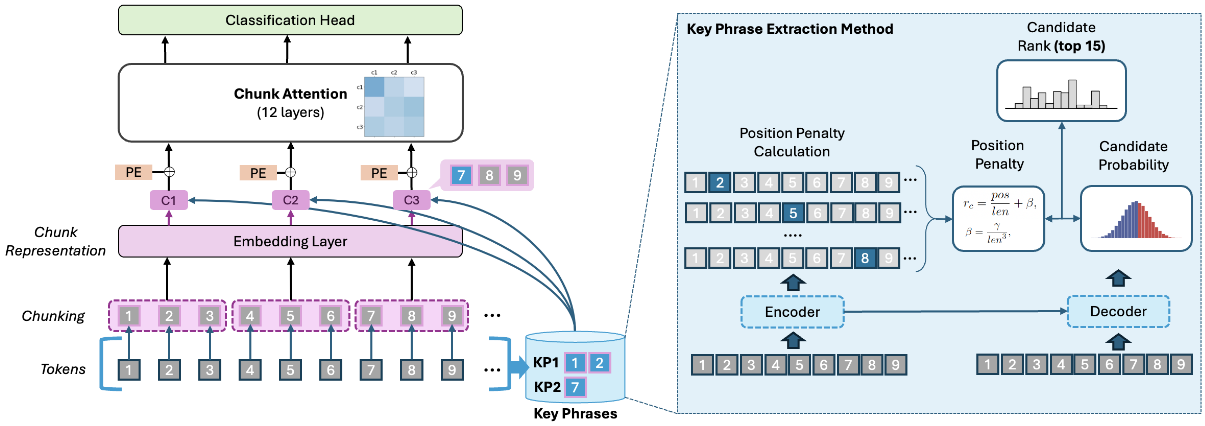


Figure 1: The Overall ChuLo Framework proposed in this paper. Each chunk is surrounded by a pink box. C_1 ... C_n represents the chunk representation.

traction, and assigns higher weights to keyphrase tokens in chunk representations. These enriched chunks train a Transformer-based chunk attention module, enabling efficient processing of long documents while retaining global and local context. Further details are provided in subsequent subsections, with the framework illustrated in Figure 1.

3.1 Document Input Chunking

To effectively manage long document inputs, we employ a chunking strategy that reduces input length while preserving all relevant information. Common approaches to long document processing, such as truncation and sparse attention, either disregard parts of the document(Lewis et al., 2020; Park et al., 2022) or restrict the receptive field of individual tokens (Beltagy et al., 2020; Zaheer et al., 2020; Brown et al., 2020), resulting in potential information loss. Our approach mitigates these issues by segmenting the document into non-overlapping chunks before feeding them into the model. It enables complete self-attention among chunks, ensuring that all information is retained and enabling the model to process larger portions of the document context. Specifically, we first tokenize the document $D = (t_0, t_1, \ldots, t_{l_D - 1})$ and divide it into fixed-length chunks $C_D = (C_0, C_1, \ldots, C_{m-1})$, where l_D is the number of the tokens, each chunk C consists of n tokens, and $m = \lceil \frac{l_D}{n} \rceil$ is the number of chunks. The incomplete chunks will be padded with the [PAD] tokens. The chunk size n is a hyper-parameter controlling the degree of input length reduction. By grouping tokens this way, we maintain the integrity of the input content while alleviating the computational burden associated with processing long sequences.

3.2 Semantic Key Information Extraction

The fundamental reason for extracting keyphrases from the chunks, as defined in the document chunking step, is to maintain the integrity of the document's semantic content while reducing input length. During chunking, the document is divided into smaller segments, which can inadvertently distribute important semantic information unevenly across chunks or even cause it to be diluted. Simply treating each chunk equally may lead to overlooking critical context essential for accurate document classification and token-level understanding. Identifying and highlighting critical phrases within these chunks ensures that the most relevant information is preserved and emphasized, allowing the model to focus on the core content even within a limited input space. This compensates for the information fragmentation caused by chunking and guides the Transformer's attention mechanism to prioritize the most informative parts of the text, enhancing the model's ability to capture the document's overall meaning and relationships. Thus, extracting keyphrases from chunks is crucial for bridging the gap between document segmentation and semantic coherence, ultimately improving the effectiveness of the chunk-based representation for long document understanding. To achieve this, we extract semantically important keyphrases to identify the core content of the entire document. Since document understanding, such as document classification or token classification, inherently involves semantic understanding, it is crucial to highlight the most informative parts of the text to create meaningful chunk representations. By making the extracted keyphrases more salient, we can effectively emphasize the content that contributes most to the document's overall meaning. Hence, we employ unsupervised keyphrase extraction methods, ensuring our approach remains adaptable across diverse domains without requiring annotated data. Building on the principles of PromptRank (Kong et al., 2023), we adapt and enhance its template-based approach to prioritize keyphrases that are contextually significant across the entire document. Our modified strategy, the Semantic Keyphrase Prioritization (SKP) Algorithm, leverages prompts to assess the importance of each candidate keyphrase, ensuring that semantically crucial information is highlighted for downstream document understanding. The details of this process are provided in Appendix A.2. In this way, our method emphasizes keyphrases during chunk representation, making critical semantic information more salient. Consequently, our method bridges the gap between document segmentation and semantic coherence by ensuring that key content is preserved and highlighted within the entire document, despite input length constraints.

3.3 Chunk Representation Production

After extracting the semantically significant keyphrases, we construct a chunk representation that preserves and highlights this key information, ensuring that the chunk retains the core semantic content of the document. While chunking helps reduce the input length, it may also result in an uneven distribution of meaningful content across chunks. Thus, it is crucial to re-emphasize the importance of these keyphrases within the chunk to maintain semantic integrity. Our approach dynamically adjusts the representation of each chunk by assigning greater importance to keyphrase tokens, enabling the model to focus on the most relevant content during downstream processing. To achieve this, we label the tokens corresponding to the extracted keyphrases in the original text as keyphrase tokens T_k, while other tokens are labelled as non-keyphrase tokens T_{nk}. Then, we feed these chunked tokens t into the embedding layer to obtain their embeddings. The chunk embedding $\boldsymbol{c}$ is then computed using a weighted average of these token embeddings, as defined in Formula 1:

$$\begin{cases} w_t = \begin{cases} a, \text{t is } T_k \\ b, \text{t is } T_{nk} \end{cases} \\ \boldsymbol{c} = \frac{\sum w_t * \boldsymbol{t}}{\sum w_t} \end{cases} \tag{1}$$

Here, w_t represents the weight assigned to each token t in the chunk, where a and b are hyperparameters with $a > b$. $\boldsymbol{t}$ denotes the embedding of token t, and $\boldsymbol{c}$ is the resulting chunk embedding that captures the weighted importance of keyphrase and non-keyphrase tokens. By assigning a higher weight a to keyphrase tokens, we ensure that the resulting chunk representation emphasizes the most critical information while maintaining a compact input length. Finally, the chunk embeddings are fed into the Transformer-based model, allowing it to effectively leverage the enhanced chunk representations during long document classification or token-level classification tasks. This method not only preserves the semantic coherence of the document but also allows the model to retain meaningful context and relationships, ultimately enhancing its performance on long document tasks.

3.4 Chunk Representation Training

In this final step, we train a Transformer-based model using our keyphrase-enhanced chunk representations to effectively incorporate the core semantic content of the document. We selected BERT-base according to Table 12. By emphasizing key information in the chunk embeddings, we ensure that the model can focus on the most relevant aspects of the text, thereby improving its ability to handle long document inputs without losing critical context. We leverage a Transformer-based backbone model, which is used to initialize the weights of the chunk attention module, as illustrated in Figure 1. This chunk attention module is designed to capture the intricate contextual relationships among chunks while retaining the influence of keyphrases. By doing so, the module can better understand local and global semantic patterns, enabling the model to perform robustly across various long document tasks. The chunk embeddings are processed through the chunk attention module to produce refined contextual representations, which are then fed into a classification head to generate the final predictions. Our framework, ChuLo, is adaptable to any transformer-based architecture, from pretrained to large language models, making it versatile for tasks involving long document understanding. Through integrating keyphrase-enhanced chunk representations, the model achieves superior performance in both document classification and token-level tasks, highlighting the effectiveness of our approach in leveraging semantic information to tackle the challenges associated with long document processing.

4 Experiments Set-up

We evaluate ChuLo on document and token classification tasks to highlight our motivation. While document classification primarily requires global contextual understanding, token classification tasks test the model's ability to retain and utilize detailed token-level information within long documents. We compare it with BERT (Kenton and Toutanova, 2019) and BERT variants (Park et al., 2022), Longformer (Beltagy et al., 2020), ToBERT (Pappagari et al., 2019), CogLTX (Ding et al., 2020), ChunkBERT (Jaiswal and Milios, 2023), and instructions with LLMs, GPT4o[2] and Gemini1.5pro[3]. **Baselines Details** are listed in Appendix A.3.

Datasets: We conduct experiments using three(3) datasets for long document classification and two(2) for long document token classification. For document classification, we use HP(Kiesel et al., 2019), LUN (Rashkin et al., 2017), and Eurlex57k (Chalkidis et al., 2019). These datasets vary in average document length from 707 to 1147 tokens, enabling us to assess performance on documents of different lengths and complexities. Further details on dataset statistics and splits are available in Appendix A.5. **1) HP** (Hyperpartisan News Detection) evaluates the classification of news articles based on hyperpartisan argumentation (Kiesel et al., 2019). We use the 'byarticle' version, which contains 238 hyperpartisan and 407 non-hyperpartisan articles. The same train-test split as in (Beltagy et al., 2020) is adopted. **2) LUN** uses for unreliable news source classification, this dataset includes 17,250 articles from satire, propaganda, and hoaxes (Rashkin et al., 2017). Our goal is to predict the source type for each article. **3) Eurlex57k** consists of 47,000 English EU legislative documents with 4,271 EUROVOC concepts. We also include a simulated **Inverted-Eurlex57k** version, where the header and recitals are moved to the end, compelling the model to read the entire document for key information. For token classification, we use GUM and CoNLL-2012 for Named Entity Recognition (NER) tasks: **1) GUM** (Georgetown University Multilayer) is a richly annotated collection of 235 documents across various genres such as academic texts, news, fiction, and interviews (Lin and Zeldes, 2021). GUM's various linguistic styles and structures make it an excellent benchmark for assessing token-level understanding in lengthy documents, ensuring that the model captures complex entity relationships over extended contexts. **2) CoNLL**-2012 originally designed for coreference resolution, and is adapted for NER in our work (Pradhan et al., 2013). We convert the data to a document-level format and select the top-k longest documents in each split, emphasizing the model's ability to process extended text sequences for token classification tasks.

Metrics and Implementation: For the HP and LUN datasets, we use **accuracy** as the evaluation metric, while for Eurlex57k, Inverted Eurlex57k, GUM, and CoNLL-2012, we adopt **micro F1**. These metrics are chosen to maintain consistency with prior work and facilitate direct comparison. Regarding implementation, we provide key details here, with the complete setup in Appendix A.6. We use CrossEntropy loss for training on the Hyperpartisan, LUN, CoNLL and GUM datasets, and Binary CrossEntropy loss for the Eurlex57k and Inverted Eurlex57k datasets. All models are optimized using the AdamW optimizer, and training employs early stopping based on the respective validation metric, with a patience threshold set to 10 epochs. A learning rate search is conducted for each experiment to ensure optimal model performance for comparison. Top-n value is set to 15[4].

5 Results

5.1 Document Classification

We evaluate the effectiveness of our ChuLo by comparing it with fine-tuned PLMs and previously published baselines (Park et al., 2022; Jaiswal and Milios, 2023) on several benchmark datasets: HP, LUN, EURLEX57K, and Inverted EURLEX57K. The comparative results are summarized in Table 1, with input configurations provided in Table 2 and detailed descriptions available in Appendix A.4. Our method demonstrates clear superiority on three of the four datasets, achieving a significant improvement of 6.43% accuracy on the LUN dataset compared to the second-best model, BERT. This marked improvement presents ChuLo's ability to capture comprehensive document context through its keyphrase-based chunk representation, despite using only 512 input embeddings. The results suggest that our method effectively mitigates the limitations of traditional truncation and

[2] https://openai.com/index/hello-gpt-4o/

[3] https://deepmind.google/technologies/gemini/pro/

[4] We tested with different n values, but 15 was generally better in most datasets

Model	HP	LUN	EUR	I-EUR
BERT	0.9200	0.5797	0.7309	0.7053
ToBERT	0.8954	0.3697	0.6757	0.6731
CogLTX	0.9477	-	0.7013	0.7080
Longformer	**0.9569**	0.5552	0.5453	0.5647
BERT+TextRank†	0.9115	0.4880	0.7287	0.7130
BERT+Random†	0.8923	0.3015	0.7322	0.7147
ChunkBERT	0.9300	-	0.6494	0.6294
Ours	0.9538	**0.6440**	**0.7332**	**0.7244**

Table 1: Document classification Result. Following previous work, we use accuracy for HP and LUN, and micro F1 for other datasets. Results for LUN are obtained by our own experiment based on provided baseline codes and methods, while baseline results for the other datasets are from previous work(Park et al., 2022; Jaiswal and Milios, 2023). † are the BERT variants proposed by (Park et al., 2022). The best performance for each dataset is bolded while the second best is underscored, and we can see that our final model, a BERT-based backbone, generally outperforms other baselines across all datasets by achieving the best or second-best.

chunking strategies by preserving critical semantic information, which contributes to higher classification accuracy. On the EURLEX57K and Inverted EURLEX57K datasets, ChuLo achieves consistent performance gains over baselines, further validating its capability to handle long documents efficiently. In these datasets, which have hierarchical labels and require understanding complex semantic structures, our model benefits from enhanced chunk representations that emphasize key content. This allows ChuLo to capture document semantics better, even when compared to models that can process larger input lengths. While our model delivers competitive results on the HP dataset, it trails behind Longformer by a slight margin of 0.0031 in accuracy. This marginal difference corresponds to only one additional correctly classified instance out of a total of 65 test samples.

Model	The Usage of Input
BERT	F-512 tokens
ToBERT	All
CogLTX	S-512 tokens
Longformer	F-4096 tokens
BERT+TextRank	F-512 + S-512 tokens
BERT+Random	F-512 + S-512 tokens
ChunkBERT	All
Ours	All (512*Chunk Size)

Table 2: The usage of the input content in the experiments."F-512" and "F-4096" means the first 512 tokens and the first 4096 tokens, "S-512" means the selected 512 tokens.

Interestingly, for the other datasets, Longformer underperforms compared to models like BERT variants or CogLTX, which use the first 512 tokens and focus on selecting key sentences. This observation indicates that unfiltered additional content can introduce noise, negatively impacting classification accuracy. In contrast, ChuLo expands the input content and strategically emphasizes key semantic elements during chunk representation. This approach mitigates noise interference, ensuring that only the most relevant information is retained and highlighted. Overall, the results confirm that ChuLo consistently outperforms standard PLM baselines and existing chunking methods in long document classification tasks. Its ability to retain and emphasize key semantic content, while efficiently handling long inputs, makes it a robust solution for various document classification challenges.

5.2 Longer Document Classification

To further validate the robustness of our model, we evaluate its classification performance across various document length ranges, with a particular focus on longer documents. For this analysis, we consider the documents with more than 1024 tokens and more than 2048 tokens in the test set. We use Longformer and off-the-shelf LLMs, GPT4o and Gemini1.5 pro for comparison. As shown in Table 3, our model consistently outperforms others on longer documents in the LUN dataset. Specifically, for documents exceeding 2,048 tokens, ChuLo maintains a higher accuracy compared to all baselines, demonstrating its capacity to handle lengthy inputs effectively. This performance gain can be attributed to our chunk representation's emphasis on keyphrases, which preserves crucial semantic content even when document length increases. On the HP dataset, ChuLo and Longformer achieve perfect accuracy (1.0) for documents longer than 2,048 tokens. However, for shorter documents (more than 1,024 tokens), ChuLo surpasses Longformer. This improvement is likely due to our chunk representation strategy, which selectively highlights key content rather than averaging information across the entire document. As a result, ChuLo maintains high semantic fidelity, leading to better overall performance even with condensed text inputs.

We also benchmarked against newly released LLMs, GPT-4o and Gemini 1.5 Pro, using longer document inputs for both the LUN and HP datasets. On LUN, GPT-4o achieved an accuracy of 0.7143 and Gemini 1.5 Pro scored 0.6531, both surpassing Longformer. However, ChuLo achieved the highest accuracy of 0.7959, showcasing its superiority in handling long documents with diverse content. On the HP dataset, GPT-4o (0.8889) and Gemini 1.5 Pro (0.7778) performed worse than Longformer

LUN	**All(2250)**	**1024(243)**	**2048(49)**
Longformer	0.5552	0.4062	0.5306
GPT4o	-	-	0.7143
Gemini1.5pro	-	-	0.6531
Ours	**0.6741**	**0.5911**	**0.7959**

(a) LUN dataset

HP	**All(65)**	**1024(28)**	**2048(9)**
Longformer	**0.9538**	0.8929	**1.000**
GPT4o	-	-	0.8889
Gemini1.5pro	-	-	0.7778
Ours	**0.9538**	**0.9286**	**1.000**

(b) HP dataset

Table 3: Document classification results for comparison on documents of different lengths: all documents in the test set, the subset of documents longer than 1024 tokens, and longer than 2048 tokens respectively. Values in brackets indicate the number of documents for each specific document set. The best performance (Accuracy) for each document set is bolded.

and ChuLo, both of which achieved a perfect accuracy of 1.0 on the longer documents. This highlights ChuLo's robustness and consistency in classifying documents with varying length, even compared to advanced language models. The prompt and response samples are in Appendix A.8 and A.9. Overall, these results demonstrate that ChuLo not only outperforms standard PLM baselines and chunking methods on long documents but also remains competitive against the latest large language models. By prioritizing key semantic elements and managing document length, ChuLo maintains stable performance across varying input lengths.

Model	**CoNLL**	**GUM**
Longformer (4096)	0.5560	0.9427
BigBird (4096)	0.5553	0.9418
GPT4o	0.2290	0.3231
Gemini1.5	0.3036	0.3262
Ours (All)	**0.9334**	**0.9555**

Table 4: Results on token classification tasks. The best performance for each dataset is bolded, and our model achieves the best on both datasets.

5.3 Token Classification

To further demonstrate the effectiveness of our chunk representation method, we evaluated it on a token-level classification task—specifically, Named Entity Recognition (NER) using long documents. We compared our model against two state-of-the-art long-document pre-trained models, Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020), as well as newly released large language models, GPT-4o and Gemini 1.5 Pro. As shown in Table 4, our model consistently outperforms Longformer, BigBird and LLM models on the NER tasks, particularly on the CoNLL, where document lengths often exceed the input limitations of these baseline models. To leverage the broader context captured by our chunk representation, we integrate a BERT-decoder module that utilizes the enhanced chunk embeddings to predict token labels more accurately. This configuration allows the model to maintain a global understanding of the document while focusing on the local dependencies necessary for precise token classification. All baselines struggle with these longer inputs due to their limited capacity for handling extensive sequences. In contrast, our method's ability to encode the entire document's context through keyphrase-based chunk representations enables it to achieve higher accuracy in recognizing and classifying named entities. This is particularly evident in cases where long-distance dependencies and contextual nuances play a critical role in determining the correct labels. Overall, the results indicate that our model's chunk representation not only enhances performance on document-level classification tasks but also proves highly effective for token-level tasks such as NER, validating its application in downstream tasks that require a detailed and comprehensive understanding of long document tokens.

CoNLL	**ALL (20)**	**> 2048 (17)**	**> 4096(6)**	**> 8192 (2)**
Longformer	0.5560	0.5268	0.3156	0.3116
BigBird	0.5553	0.5261	0.3145	0.3106
GPT4o	0.2290	0.2217	0.1252	0.0282
Gemini 1.5	0.3036	0.2633	0.1652	0.0584
Ours	**0.9334**	**0.9325**	**0.9287**	**0.9206**

(a) Results on CoNLL dataset.

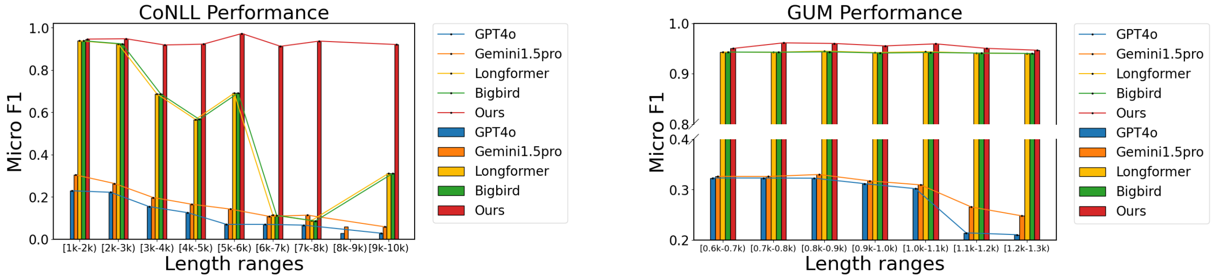
GUM	**ALL (26) - > 512**	**> 1000(8)**	**> 1042(6)**
Longformer	0.9427	0.9427	0.9439
BigBird	0.9418	0.9417	0.9426
GPT4o	0.3231	0.3018	0.2808
Gemini 1.5	0.3262	0.3093	0.3215
Ours	**0.9555**	**0.9558**	**0.9574**

(b) Results on GUM dataset.

Table 5: NER results for comparison on documents of different lengths. >*number* represents the documents longer than the number, with the values in brackets indicating the corresponding counts for the documents. The best performance (Micro F1) is bolded and the second best is underscored, and our model consistently outperforms all the baselines for each document set.

5.4 Token Classification in Longer Documents

We further analyze the NER performance across different document length ranges. As presented in Table 5a and Table 5b, we report the number of documents exceeding specific length thresholds and their corresponding performance metrics. On the CoNLL, as document lengths exceed the maximum input capacities of Longformer and BigBird, both models exhibit significant performance drops to



(a) CoNLL Performance (Range: 1798 to 9778)

(b) GUM Performance (Range: 628 to 1281)

Figure 2: Comparison of performance in different length ranges for CoNLL and GUM datasets. Values of brackets includes the min and max length of each dataset.

31.56% and 31.45%, respectively. In contrast, our model experiences a minimal decrease of 1.28%, showcasing its resilience and effectiveness in handling long sequences. For the GUM, where all document lengths are within the acceptable range for these models, performance remains stable across all models, with our approach consistently achieving the best results. Figures 2a and 2b visualize the performance breakdown across varying length ranges. For the CoNLL, our model maintains high performance in all length intervals, while Longformer and BigBird exhibit comparable performance within the [1k-2k) range but degrade significantly for longer texts, even for documents that do not exceed their maximum input length. This discrepancy suggests that the uneven distribution of document lengths in their pretraining corpora may lead to inconsistent performance on longer sequences. In contrast, our model's ability to compress the entire document into 512-length chunks for the decoder enables it to leverage complete contextual information, resulting in better stability and accuracy even on longer documents. For the GUM, where document lengths are shorter (up to 1.3k tokens), our model consistently outperforms Longformer and BigBird in all intervals. The stable performance of all models on GUM aligns with the results on CoNLL, further confirming that our approach's chunk representation is particularly effective when documents reach lengths that exceed the standard input capacities of the baselines. These results underscore the effectiveness of our chunk representation, which emphasizes keyphrase information, for coarse-grained document classification and fine-grained token-level classification tasks like NER. The ability to maintain performance across varying document lengths highlights the importance of incorporating global contextual information in NER tasks—a largely underexplored aspect. Additionally, off-the-shelf LLMs such as GPT-4o and Gemini 1.5 Pro show suboptimal performance on NER tasks without fine-tuning, and their performance deteriorates further as document length increases. This indicates that, despite their advancements, LLMs still require substantial optimization for effective long document understanding.

5.5 Ablation Studies

We conducted more ablation studies, including 1) keyphrase extraction methods, 2) sentence embedding approaches, and 3)backbone model analysis, shown in Appendix A.7.

5.6 Qualitative Analysis

We performed a qualitative analysis by visualizing each sample document from different datasets, comparing the outputs of Longformer, GPT-4o, Gemini 1.5 Pro, and our ChuLo. ChuLo captures the context and semantic patterns of the document, providing accurate predictions, whereas the other models struggle to maintain coherence and consistency. We have more examples in Appendix A.9.

6 Conclusion

We introduced ChuLo, a novel chunk representation method that enhances the performance of Transformer-based models on long document classification and token-level classification tasks. By utilizing unsupervised keyphrase extraction, ChuLo effectively reduces input length while preserving critical information, addressing the limitations of truncation and sparse attention. Extensive experiments demonstrate that ChuLo outperforms existing methods by maintaining both global context and high accuracy, even for lengthy inputs. Our research results highlight the effectiveness of ChuLo as a robust solution for long document understanding, enabling processing of complex texts in NLP applications.

7 Limitation

There are several opportunities for future work, including extending the chunk representation to generative tasks such as long text generation, where chunk representation may extend the LLM's context range limitation and enhance generation quality. However, the performance of the keyphrase extraction method poses a potential risk, as its quality directly affects the overall effectiveness of the approach. We believe this work offers valuable insights into long text understanding and lays a foundation for advancements in related tasks.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 221–229.

Medha Bhattacharya, Swati Bhat, Sirshasree Tripathy, Anvita Bansal, and Monika Choudhary. 2023. Improving biomedical named entity recognition through transfer learning and asymmetric tri-training. *Procedia Computer Science*, 218:2723–2733.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*, pages 543–551.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Can Çetindağ, Berkay Yazıcıoğlu, and Aykut Koç. 2023. Named-entity recognition in turkish legal texts. *Natural Language Engineering*, 29(3):615–642.

Ilias Chalkidis, Emmanouil Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. Large-scale multi-label text classification on eu legislation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6314–6322.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, David Belanger, Lucy Colwell, et al. 2020. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S Rosen, Gerbrand Ceder, Kristin A Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33:12792–12804.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Samhaa R El-Beltagy and Ahmed Rafea. 2009. Kp-miner: A keyphrase extraction system for english and arabic documents. *Information systems*, 34(1):132–144.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 1105–1115.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), pages 1262–1273, Baltimore, Maryland. Association for Computational Linguistics.

Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng, Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, et al. 2024. Improving large language models for clinical named entity recognition via prompt engineering. *Journal of the American Medical Informatics Association*, page ocad259.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. *Advances in Neural Information Processing Systems*, 35:33248–33261.

Aman Jaiswal and Evangelos Milios. 2023. Breaking the token barrier: Chunking and convolution for efficient long text classification with bert. *arXiv preprint arXiv:2310.20558*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 829–839.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, and Xiaoyan Bai. 2023. PromptRank: Unsupervised keyphrase extraction using prompt. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9788–9801, Toronto, Canada. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Miao Li, Eduard Hovy, and Jey Han Lau. 2023a. Towards summarizing multiple documents with hierarchical relationships. *arXiv preprint arXiv:2305.01498*.

Xianming Li, Zongxi Li, Xiaotian Luo, Haoran Xie, Xing Lee, Yingbin Zhao, Fu Lee Wang, and Qing Li. 2023b. Recurrent attention networks for long-text modeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3006–3019, Toronto, Canada. Association for Computational Linguistics.

Jessica Lin and Amir Zeldes. 2021. WikiGUM: Exhaustive entity linking for wikification in 12 genres. In *Proceedings of The Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop (LAW-DMR 2021)*, pages 170–175, Punta Cana, Dominican Republic.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 257–266.

Davlatyor Mengliev, Vladimir Barakhnin, Nilufar Abdurakhmonova, and Mukhriddin Eshkulov. 2024. Developing named entity recognition algorithms for uzbek: Dataset insights and implementation. *Data in Brief*, 54:110413.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 838–844. IEEE.

Hyunji Park, Yogarshi Vyas, and Kashif Shah. 2022. Efficient classification of long documents using transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 702–709.

Yeon-Ji Park, Min-a Lee, Geun-Je Yang, Soo Jun Park, and Chae-Bong Sohn. 2023. Web interface of ner and re with bert for biomedical text mining. *Applied Sciences*, 13(8):5163.

H Peng, N Pappas, D Yogatama, R Schwartz, N Smith, and L Kong. 2021. Random feature attention. In *International Conference on Learning Representations (ICLR 2021)*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 2931–2937.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Yi Sun, Hangping Qiu, Yu Zheng, Zhongwei Wang, and Chaoran Zhang. 2020. Sifrank: a new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access*, 8:10896–10906.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023. Gpt-ner: Named entity recognition via large language models. *arXiv preprint arXiv:2304.10428*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Miguel Won, Bruno Martins, and Filipa Raimundo. 2019. Automatic extraction of relevant keyphrases for the study of issue competition. In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 648–669. Springer.

Jianfei Yu, Ziyan Li, Jieming Wang, and Rui Xia. 2023. Grounded multimodal named entity recognition on social media. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9141–9154.

Yang Yu and Vincent Ng. 2018. Wikirank: Improving keyphrase extraction based on background knowledge. *arXiv preprint arXiv:1803.09000*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Xin Zhang, Jingling Yuan, Lin Li, and Jianquan Liu. 2023. Reducing the bias of visual objects in multimodal named entity recognition. In *Proceedings of the Sixteenth ACM international conference on web search and data mining*, pages 958–966.

Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed Awadallah, Dragomir Radev, and Rui Zhang. 2022. Summn: A multi-stage summarization framework for long input dialogues and documents: A multi-stage summarization framework for long input dialogues and documents. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1592–1604.

Jing Zhao, Junwei Bao, Yifan Wang, Yongwei Zhou, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2021. Ror: Read-over-read for long document machine reading comprehension. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1862–1872.

A Appendix

A.1 Related Works

As shown in Table 6, most of the previous works addressing the problem of processing long documents cannot fully utilize all the content. Those methods either reduce input length via truncation or focus on local context learning to improve efficiency by applying sparse attention, approximated attention or RNN integration. Such approaches will lead to a certain level of information loss, unlike our chunking approach which can take all the content into consideration. Hierarchical Transformer (Pappagari et al., 2019) splits documents into non-overlapping chunks and computes chunk representations. RoR (Zhao et al., 2021) generates regional answers from chunks, which are combined for the final answer. However, neither considers the entire document context when chunking. In addition, previous works applying the chunking method for processing long document context only focus on a single task, either document classification or token classification, while our framework can be applied to both tasks to guarantee both document-level and token-level understanding.

A.2 Keyphrase Extraction

We employ the Semantic Keyphrase Prioritization (SKP) algorithm to extract keyphrases that encapsulate the key semantic information of the entire document. The detailed are shown in Algorithm 1. While PromptRank uses prompts to rank keyphrases across the first segment of the document determined by its encoder model, our SKP applies this concept at the entire document level to ensure that each chunk can preserve the most informative content for the entire document. After obtaining the sorted phrases set K_s, we select top-n phrases as the keyphrases of the document, which can be regarded as ranked phrases according to their contextual significance within the entire document.

A.3 Baselines

We use BERT (Kenton and Toutanova, 2019) as our backbone model, comparing it with ToBERT (Pappagari et al., 2019), CogLTX (Ding et al., 2020), Longformer (Beltagy et al., 2020), various BERT variants (Park et al., 2022) and ChunkBERT (Jaiswal and Milios, 2023) for the document classification task. For the NER task, we compare against Longformer, BigBird (Zaheer et al., 2020), and two large language models, GPT4o and Gemini1.5pro. Below are brief descriptions of the baseline models:

- **BERT**: A transformer model pre-trained on masked language modeling (MLM) and next-sentence prediction (NSP). We fine-tuned the BERT-base variant on each dataset.
- **ToBERT**: A BERT variant designed for long document classification, utilizing an additional transformer layer to learn inter-chunk relationships.
- **CogLTX**: A framework for applying BERT to long documents by training a key sentence identification model to assist in document understanding
- **Longformer**: Optimized for long sequences using sparse attention, combining dilated sliding window and global attention patterns
- **BigBird**: Utilizes block sparse attention, integrating sliding window, global, and random attention patterns across token blocks.
- **BERT+TextRank** and **BERT+Random**: Proposed to select other tokens randomly or with the help of TextRank(Mihalcea and Tarau, 2004) to feed into the BERT model as the supplementation for long document classification.
- **ChunkBERT**: A BERT variant for long document classification that processes self-attention within document chunks and adds a TextCNN module for classification using the chunk representation.
- **GPT-4o**: A transformer-based multi-modal large language model developed by OpenAI, which leverages large-scale pretraining data to process diverse language tasks via instruction prompts.
- **Gemini 1.5 Pro**: an advanced multi-modal AI model from Google, leveraging a Sparse Mixture-of-Experts (MoE) Transformer architecture, with a context window of up to 2 million tokens. This architecture allows for the efficient handling of long documents.

Algorithm 1 Semantic Keyphrase Prioritization (SKP) Algorithm

```
Input: A tokenized document $D$, an encoder-decoder pretrained model represented by $\mathcal{F}_E$ and $\mathcal{F}_D$, a POS tagger $\mathcal{F}_{POS}$, a regular expression $\mathcal{F}_{REG} = \langle \text{NN.} * |\text{JJ}\rangle * \langle \text{NN.}*\rangle$
Parameter: Experimentally determined $\alpha, \gamma$
Output: Sorted keyphrases set $K_s$

Let $S = \emptyset, K_s = \emptyset, i = 0, j = 0$.
Get the candidate phrases set:
    $K = \mathcal{F}_{REG}(\mathcal{F}_{POS}(D)) = \{k_0, k_1, \ldots, k_{n-1}\}$
Split $D$ into segments $S = \{D_0, D_1, \ldots, D_{m-1}\}$ to meet the input requirement of $\mathcal{F}_E$
for $i < n$ do
    Calculate the position penalty $r_i = \frac{L_c}{l_d} + \frac{\gamma}{(l_d)^3}$
    where $L_c$ is the first occurrence position of $k_i$ in the document, $l_d$ is the length of the document
    Construct the prompt $P$ "The * mainly discusses $k_i$" and tokenize, * is the category of the document.
    for $j < m$ do
        Calculate the probability $p_{ij}$ of the phrase $k_i$:
        $p_{ij} = \frac{1}{(l_P)^\alpha} \sum_{g=h}^{h+l_k-1} \log p(t_g \mid t_{<g})$,
        $p(t_g \mid t_{<g}) = \mathcal{F}_D(\mathcal{F}_E(D_j), t_{<g})$
        where $l_P$ is the length of the tokenized $P$, $h$ is the start index of $k_i$ in the prompt, $l_k$ is the length of $k_i$, $t$ is the token of the prompt.
    end for
    Calculate the final score of $k_i$: $s_i = r_i \times \sum_{j=0}^{j<m} p_{ij}$
end for
return $K_s = Sort(K, s)$
```

Model	Year	Task	Lengthy Document Solution	Core Architecture
Efficient Classification (Park et al., 2022)	2022	D	Truncating	Transformer
Hierarchical transformer (Pappagari et al., 2019)	2019	D	Chunking (Partial, Phrase)	Transformer
RoR (Zhao et al., 2021)	2021	T	Chunking (Partial, Voting)	Transformer
Longformer (Beltagy et al., 2020)	2020	D, T	Sparse Attention	Transformer
BigBird (Zaheer et al., 2020)	2020	D, T	Sparse Attention	Transformer
Routing Transformer (Roy et al., 2021)	2021	D, T, G	Sparse Attention	Transformer
Macformer (Peng et al., 2021)	2021	D, T	Approximated Attention	Transformer
Linformer (Wang et al., 2020)	2020	D, T, G	Approximated Attention	Transformer
Performer (Choromanski et al., 2020)	2020	D, T, G	Approximated Attention	Transformer
Transformer-xl (Dai et al., 2019)	2019	G	RNN Integration	Transformer
Block-Recurrent Transformer (Hutchins et al., 2022)	2022	G	RNN Integration	Transformer
RAN (Li et al., 2023b)	2023	D, T	RNN Integration	Attention
(Çetindağ et al., 2023)	2023	T	N/A	LSTM
(Mengliev et al., 2024)	2024	T	N/A	Neural Network
(Park et al., 2023)	2023	T	N/A	Transformer
(Bhattacharya et al., 2023)	2023	T	N/A	LSTM
Gpt-NER (Wang et al., 2023)	2023	T	N/A	Transformer
(Dagdelen et al., 2024)	2024	T	N/A	Transformer
(Hu et al., 2024)	2024	T	N/A	Transformer
(Yu et al., 2023)	2023	T	N/A	Transformer
(Zhang et al., 2023)	2023	T	N/A	Transformer
Ours	**2024**	**D, T**	**Chunking (Entire)**	**Transformer**

Table 6: Summary of Related Works. D, T, G represent tasks of document classification, token classification, and text generation, respectively.

A.4 Baseline Input

We selected these baseline models because they represent diverse methods for processing long documents. As summarized in Table 7, BERT truncates the input to 512 tokens. Longformer and BigBird utilize sparse attention mechanisms, allowing them to process up to 4096 tokens while conserving computational resources. ToBERT divides the input into 200-token chunks, feeds them to BERT for chunk representations, and uses a transformer layer for downstream tasks. However, it cannot capture dependencies across the entire input sequence. CogLTX selects key sentences to form a 512-token input, limiting its input size to that constraint. BERT variants like BERT+TextRank and BERT+Random select up to 512 tokens using TextRank or random selection. They concatenate the [CLS] representation of the initial 512 tokens with the selected tokens, creating an augmented input for a fully connected classification layer, with a maximum input length of 1024 tokens. ChunkBERT splits the input into chunks, computes self-attention, and feeds chunk representations into a TextCNN module for classification. The original implementation processes up to 4096 tokens per document. It has the same limitation as the ToBERT. For GPT4o and Gemini1.5pro, we input all tokens together with our instruction in the prompt due to the large input size supported by these large language model. In contrast to these baseline models, our approach flexibly segments the entire input into chunks of varying sizes, using semantic keyphrases to minimize information loss. Additionally, we compute chunk-level attention to capture long-range dependencies more effectively.

Model	Input
BERT (Kenton and Toutanova, 2019)	The first 512 tokens
ToBERT (Pappagari et al., 2019)	Segmented all input tokens
CogLTX (Ding et al., 2020)	Selected 512 tokens
Longformer (Beltagy et al., 2020)	The first 4096 tokens
BigBird (Zaheer et al., 2020)	The first 4096 tokens
BERT+TextRank (Park et al., 2022)	The first 512 tokens with the selected 512 tokens
BERT+Random (Park et al., 2022)	The first 512 tokens with the selected 512 tokens
ChunkBERT (Jaiswal and Milios, 2023)	The first 4096 tokens
GPT4o	All input tokens with instruction
Gemini1.5pro	All input tokens with instruction

Table 7: The input of the baseline models

A.5 Details of datasets

Datasets	Train/Dev/Test	#Classes	Avg. Length
HP	516/64/65	2	705
LUN	12003/2992/2250	3	480
EURLEX57k	45000/6000/6000	4271	707
-INVERTED	45000/6000/6000	4271	707
GUM	179/26/26	21	972
CoNLL	120/20/20	37	5065

Table 8: The split and statistics of the datasets, including document classification task (HP, LUN, EURLEX57K, and Inverted EURLEX57K) and token classification task (GUM, CoNLL)

We analyzed the data distribution across the datasets used in this paper. Of these, the CoNLL dataset has the highest average number of tokens per document at 5,065. In contrast, LUN has the shortest average length, with 480 tokens per document. Both HP and EURLEX57K have similar average document lengths, measuring 705 and 707 tokens respectively. GUM presents a relatively higher token count, averaging 972 tokens per document.

Regarding the number of classes, EURLEX57K is the most complex dataset, containing 4,271 unique labels. In comparison, HP and LUN are more limited, with only 2 and 3 classes respectively. GUM and CoNLL are more diverse, with 21 and 37 different classes. Beyond label diversity, EURLEX57K also has the largest sample size, comprising 45,000 training samples, 6,000 development samples, and 6,000 test samples. LUN is the second-largest dataset, with 12,003 training samples, 2,992 development samples, and 2,250 test samples. Due to our selection strategy, CoNLL has the longest average document length, with the fewest samples. It has a total of 160 documents split into 120/20/20 for training, development, and test sets. GUM follows a similar distribution with 179/26/26 samples. The HP dataset includes 516 training samples, 64 development samples, and 65 test samples.

A.6 Implementation details

A.6.1 Experiment hyperparameters

We performed extensive experiments to select the hyperparameters, including chunk size, token weights, learning rates, and warm-up strategies and steps. The optimal hyperparameters for each dataset for our proposed ChuLo model are presented in Table 9.

A.6.2 Hardware Information

Our experiments are run on the Linux platform with an A6000 Nvidia graphic card and an AMD Ryzen Threadripper PRO 5955WX 16-core CPU, and the RAM is 128G.

A.7 Ablation Studies

We performed a few ablation studies on the HP and LUN to assess the impact of various components within our model. First, we analyzed the effectiveness of different keyphrase extraction methods and the effect of using average chunk representations. As shown in Table 10, the PromptRank-based method yields the highest performance across both datasets, outperforming alternatives like YAKE-based. This improvement can be attributed to PromptRank's ability to extract higher-quality keyphrases by considering semantic relationships within the document, whereas YAKE relies primarily on statistical features such as phrase frequency, resulting in less semantically rich keyphrases. Then, we explored the effect of incorporating sentence embeddings into the chunk representations to introduce global sentence-level context. Surprisingly, as shown in Table 11, the results indicate a performance drop when sentence embeddings are included. We hypothesize that adding sentence-level information at the initial representation stage may cause chunk embeddings within the same sentence to become too similar, hindering the model's ability to learn distinctive patterns and reducing overall classification performance. Then, we evaluated the performance of

Hyperparameter	**HP**	**LUN**	**EURLEX57K**	**I-EURLEX57K**	**CoNLL**	**GUM**
Number of top-n phrases	15	15	15	15	15	15
Chunk size n	10	50	5	5	20	50
Weight for T_k	0.8	0.5	0.8	0.8	0.8	0.8
Weight for T_{nk}	0.1	0.1	0.1	0.1	0.1	0.1
Learning Rate	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5
Batch Size	16	32	16	16	2	8
Warm-up Strategy	Linear	Linear	Cosine	Cosine	Linear	Linear
Warm-up Steps	10%	10%	5%	5%	10%	10%
Mex epoch	100	100	100	100	100	100
Stop Patience	10	10	10	10	10	10
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Optimizer Weight Decay	1e-2	1e-2	1e-2	1e-2	1e-2	1e-2
Optimizer Betas	0.9, 0.999	0.9, 0.999	0.9, 0.999	0.9, 0.999	0.9, 0.999	0.9, 0.999

Table 9: The optimal hyperparameters used in our experiments.

different backbone models for the chunk attention module while keeping the keyphrase extraction and chunk representation settings consistent. Table 12 shows that BERT outperforms Longformer as the backbone. This result suggests that, after document chunking, the input sequences become relatively short, making it difficult for Longformer to leverage its long-range attention capabilities fully. Consequently, Longformer may suffer underutilization during training, resulting in suboptimal performance compared to BERT.

Keyphrase method	**HP**	**LUN**
Average	**0.9538**	0.5951
YAKE	0.8769	0.5951
PromptRank	**0.9538**	**0.6440**

Table 10: Effect of keyphrase extraction methods; Average: Average Chunk Representations

Sentence Embedding	**HP**	**LUN**
w/o sentence emb.	**0.9538**	**0.6440**
sentence emb.	0.9076	0.5537

Table 11: Effect of sentence embedding, adding the sentence-level information to the chunk representations.

Backbone	**HP**	**LUN**
BERT (Ours)	**0.9538**	**0.6440**
RoBERTa	0.8615	0.5906
Longformer	0.8923	0.5600

Table 12: Effect of different backbone models for the chunk attention.

A.8 Prompt Method

We employed zero-shot prompting with large language models (LLMs), specifically Gemini 1.5 Pro and GPT4o, in our experiments. The prompts used for each dataset are detailed in Table 13 and 14:

Table 3 shows that LLMs outperform Longformer in the document classification task with zero-shot prompt tuning. However, their performance drops significantly in the NER task in Table 5a and Table 5b. For instance, in Figure 11, both GPT4o and Gemini1.5pro only predicted a single correct label, “O”. Moreover, the models often fail to predict a sufficient number of token labels for longer inputs, or they repeatedly predict all “O” labels or redundant label sequences. These inconsistencies suggest that LLMs struggle to generate outputs matching the input length in token classification, highlighting substantial room for improvement in this area.

Dataset	**Prompt**
LUN	Task Definition: You are provided with a news article. Your task is to classify the article into one of the following categories: "Satire” "Hoax” or "Propaganda” Respond only with the appropriate category. The news is: [{input}].
HP	Task Definition: You are provided with a news article. Your task is to classify whether the article is hyperpartisan. Respond only with "True” if the news is hyperpartisan or "False” if it is not. The news is: [{input}].

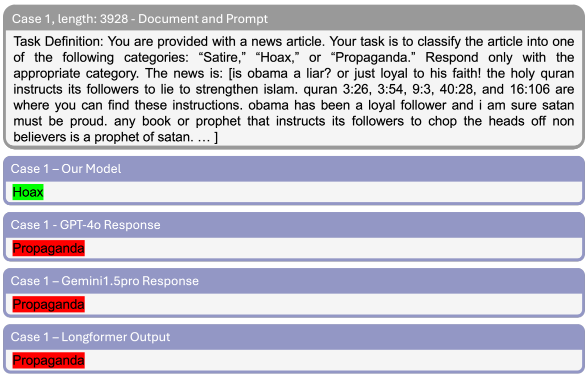
Table 13: The prompt we used for each dataset in our experiments.

A.9 More Case Studies

In this section, we will present several prompt and output samples for the long documents from the LUN (Figures 3) and 4) and Hyperpartisan (Figures 5 and 6) datasets for document classification, as well as GUM (Figures 7, 8 and 9) and CoNLL (Figures 10, 11, 12 and 13) datasets for NER task. Documents with various lengths are randomly selected to see the comparison of our model against GPT-4, Gemini1.5pro and Longformer. While there is always at least one baseline which predicts wrongly for the difficult cases presented for the document classification task, we can observe that our model consistently classifies those documents well. For the token classification task, our model can also correctly classify more tokens than each baseline across the shown cases.

Dataset	Prompt
CoNLL	In the task of Named Entity Recognition, the B-, I-, and O- prefixes are commonly used to annotate slot types, indicating the boundaries and types of slots. These labels typically represent: B- (Begin): Signifies the beginning of a slot, marking the start of a new slot. I- (Inside): Represents the interior of a slot, indicating a continuation of the slot. O (Outside): Denotes parts of the input that are not part of any slot. For instance, in a sentence where we want to label a "date" slot, words containing date information might be tagged as "B-date" (indicating the beginning of a date slot), followed by consecutive words carrying date information tagged as "I-date" (indicating the continuation of the date slot), while words not containing date information would be tagged as "O" (indicating they are outside any slot). Definition: In this task, you are given a conversation, where the words spoken by a person are shown as a list. Your job is to classify the words in the following conversation into one of the 37 different entities. The entities are: "O", "B-PERSON", "I-PERSON", "B-NORP", "I-NORP", "B-FAC", "I-FAC", "B-ORG", "I-ORG", "B-GPE", "I-GPE", "B-LOC", "I-LOC", "B-PRODUCT", "I-PRODUCT", "B-DATE", "I-DATE", "B-TIME", "I-TIME", "B-PERCENT", "I-PERCENT", "B-MONEY", "I-MONEY", "B-QUANTITY", "I-QUANTITY", "B-ORDINAL", "I-ORDINAL", "B-CARDINAL", "I-CARDINAL", "B-EVENT", "I-EVENT", "B-WORK_OF_ART", "I-WORK_OF_ART", "B-LAW", "I-LAW", "B-LANGUAGE", "I-LANGUAGE". Only output entities. And the entity types should be output as a list without any explanation. The input is [{input}].
GUM	In the task of Named Entity Recognition, the B-, I-, and O- prefixes are commonly used to annotate slot types, indicating the boundaries and types of slots. These labels typically represent: B- (Begin): Signifies the beginning of a slot, marking the start of a new slot. I- (Inside): Represents the interior of a slot, indicating a continuation of the slot. O (Outside): Denotes parts of the input that are not part of any slot. For instance, in a sentence where we want to label a "date" slot, words containing date information might be tagged as "B-date" (indicating the beginning of a date slot), followed by consecutive words carrying date information tagged as "I-date" (indicating the continuation of the date slot), while words not containing date information would be tagged as "O" (indicating they are outside any slot). Definition: In this task, you are given a conversation, where the words spoken by a person are shown as a list. Your job is to classify the words in the following conversation into one of the 37 different entities. The entities are: "I-abstract", "B-object", "B-place", "I-substance", "I-time", "I-place", "B-time", "B-abstract", "I-person", "B-plant", "B-substance", "I-animal", "B-organization", "I-event", "B-person", "B-event", "I-plant", "I-organization", "O", "I-object", "B-animal". Only output entities. And the entity types should be output as a list without any explanation. The input is [{input}].

Table 14: The prompt we used for each dataset in our experiments.



Case 1, length: 3928 - Document and Prompt

Task Definition: You are provided with a news article. Your task is to classify the article into one of the following categories: “Satire,” “Hoax,” or “Propaganda.” Respond only with the appropriate category. The news is: [is obama a liar? or just loyal to his faith! the holy quran instructs its followers to lie to strengthen islam. quran 3:26, 3:54, 9:3, 40:28, and 16:106 are where you can find these instructions. obama has been a loyal follower and i am sure satan must be proud. any book or prophet that instructs its followers to chop the heads off non believers is a prophet of satan. …]

Case 1 – Our Model

Hoax

Case 1 - GPT-4o Response

Propaganda

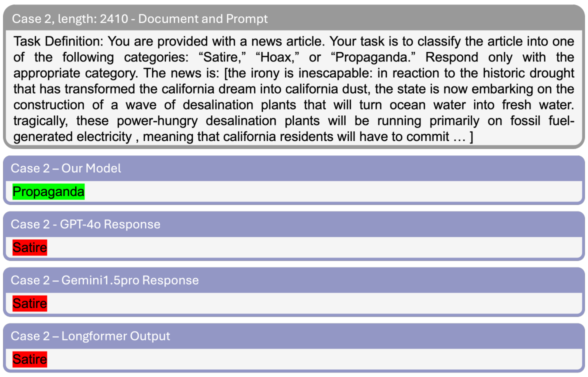
Case 1 – Gemini1.5pro Response

Propaganda

Case 1 – Longformer Output

Propaganda

Figure 3: Prompt and output for a sample document of length 3928 in **LUN** dataset, where the correct prediction is highlighted in green and wrong predictions are highlighted in red. Compared to GPT4o, Gemini1.5pro and Longformer, our model can **correctly** classify the given document as **Hoax**.



Case 2, length: 2410 - Document and Prompt

Task Definition: You are provided with a news article. Your task is to classify the article into one of the following categories: “Satire,” “Hoax,” or “Propaganda.” Respond only with the appropriate category. The news is: [the irony is inescapable: in reaction to the historic drought that has transformed the california dream into california dust, the state is now embarking on the construction of a wave of desalination plants that will turn ocean water into fresh water. tragically, these power-hungry desalination plants will be running primarily on fossil fuel-generated electricity , meaning that california residents will have to commit …]

Case 2 – Our Model

Propaganda

Case 2 - GPT-4o Response

Satire

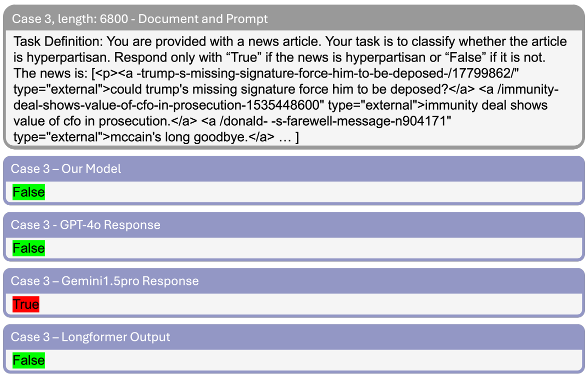
Case 2 – Gemini1.5pro Response

Satire

Case 2 – Longformer Output

Satire

Figure 4: Prompt and output for a sample document of length 2410 in **LUN** dataset, where the correct prediction is highlighted in green and wrong predictions are highlighted in red. Compared to GPT4o, Gemini1.5pro and Longformer, our model can **correctly** classify the given document as **Propaganda**.



Case 3, length: 6800 - Document and Prompt

Task Definition: You are provided with a news article. Your task is to classify whether the article is hyperpartisan. Respond only with “True” if the news is hyperpartisan or “False” if it is not. The news is: [<p><a -trump-s-missing-signature-force-him-to-be-deposed-/17799862/" type="external">could trump's missing signature force him to be deposed?</a> <a /immunity-deal-shows-value-of-cfo-in-prosecution-1535448600" type="external">immunity deal shows value of cfo in prosecution.</a> <a /donald- -s-farewell-message-n904171" type="external">mccain's long goodbye.</a> …]

Case 3 – Our Model

False

Case 3 - GPT-4o Response

False

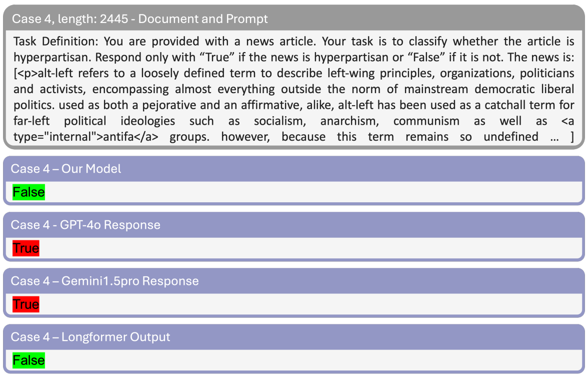
Case 3 – Gemini1.5pro Response

True

Case 3 – Longformer Output

False

Figure 5: Prompt and output for a sample document of length 6800 in **Hyperpartisan** dataset, where correct predictions are highlighted in green and the wrong prediction is highlighted in red. Compared to Gemini1.5pro, our model, GPT4o and Longformer can **correctly** classify the given document as **False**.



Case 4, length: 2445 - Document and Prompt

Task Definition: You are provided with a news article. Your task is to classify whether the article is hyperpartisan. Respond only with “True” if the news is hyperpartisan or “False” if it is not. The news is: [<p>alt-left refers to a loosely defined term to describe left-wing principles, organizations, politicians and activists, encompassing almost everything outside the norm of mainstream democratic liberal politics. used as both a pejorative and an affirmative, alike, alt-left has been used as a catchall term for far-left political ideologies such as socialism, anarchism, communism as well as <a type="internal">antifa</a> groups. however, because this term remains so undefined …]

Case 4 – Our Model

False

Case 4 - GPT-4o Response

True

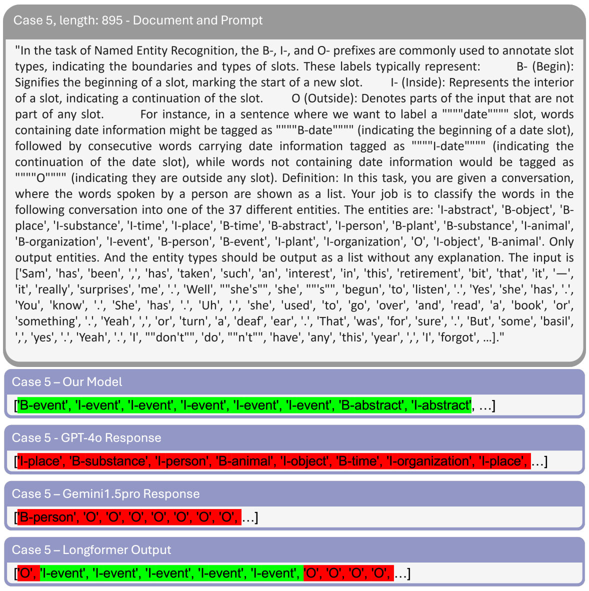
Case 4 – Gemini1.5pro Response

True

Case 4 – Longformer Output

False

Figure 6: Prompt and output for a sample document of length 2445 in **Hyperpartisan** dataset, where correct predictions are highlighted in green and wrong predictions are highlighted in red. Compared to GPT4o and Gemini1.5pro, our model and Longformer can **correctly** classify the given document as **False**.



Case 5, length: 895 - Document and Prompt

"In the task of Named Entity Recognition, the B-, I-, and O- prefixes are commonly used to annotate slot types, indicating the boundaries and types of slots. These labels typically represent: B- (Begin): Signifies the beginning of a slot, marking the start of a new slot. I- (Inside): Represents the interior of a slot, indicating a continuation of the slot. O (Outside): Denotes parts of the input that are not part of any slot. For instance, in a sentence where we want to label a """"date"""" slot, words containing date information might be tagged as """"B-date"""" (indicating the beginning of a date slot), followed by consecutive words carrying date information tagged as """"I-date"""" (indicating the continuation of the date slot), while words not containing date information would be tagged as """"O"""" (indicating they are outside any slot). Definition: In this task, you are given a conversation, where the words spoken by a person are shown as a list. Your job is to classify the words in the following conversation into one of the 37 different entities. The entities are: 'I-abstract', 'B-object', 'B-place', 'I-substance', 'I-time', 'I-place', 'B-time', 'B-abstract', 'I-person', 'B-plant', 'B-substance', 'I-animal', 'B-organization', 'I-event', 'B-person', 'B-event', 'I-plant', 'I-organization', 'O', 'I-object', 'B-animal'. Only output entities. And the entity types should be output as a list without any explanation. The input is ['Sam', 'has', 'been', ',', 'has', 'taken', 'such', 'an', 'interest', 'in', 'this', 'retirement', 'bit', 'that', 'it', '—', 'it', 'really', 'surprises', 'me', '.', 'Well', ""she's"", 'she', ""'s"", 'begun', 'to', 'listen', '.', 'Yes', 'she', 'has', '.', 'You', 'know', '.', 'She', 'has', '.', 'Uh', ',', 'she', 'used', 'to', 'go', 'over', 'and', 'read', 'a', 'book', 'or', 'something', '.', 'Yeah', ',', 'or', 'turn', 'a', 'deaf', 'ear', '.', 'That', 'was', 'for', 'sure', '.', 'But', 'some', 'basil', ',', 'yes', '.', 'Yeah', '.', 'I', ""don't"", 'do', ""n't"", 'have', 'any', 'this', 'year', ',', 'I', 'forgot', ...]."

Case 5 – Our Model

['B-event', 'I-event', 'I-event', 'I-event', 'I-event', 'I-event', 'B-abstract', 'I-abstract', …]

Case 5 - GPT-4o Response

['I-place', 'B-substance', 'I-person', 'B-animal', 'I-object', 'B-time', 'I-organization', 'I-place', …]

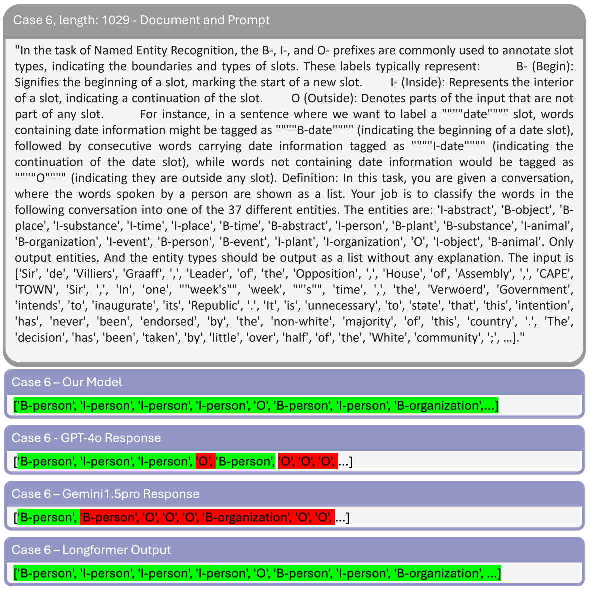
Case 5 – Gemini1.5pro Response

['B-person', 'O', 'O', 'O', 'O', 'O', 'O', 'O', …]

Case 5 – Longformer Output

['O', 'I-event', 'I-event', 'I-event', 'I-event', 'I-event', 'O', 'O', 'O', 'O', …]

Figure 7: Prompt and output for a sample document of length 895 in **GUM** dataset for NER task, where correct predictions are highlighted in green and wrong predictions are highlighted in red.



Case 6, length: 1029 - Document and Prompt

"In the task of Named Entity Recognition, the B-, I-, and O- prefixes are commonly used to annotate slot types, indicating the boundaries and types of slots. These labels typically represent: B- (Begin): Signifies the beginning of a slot, marking the start of a new slot. I- (Inside): Represents the interior of a slot, indicating a continuation of the slot. O (Outside): Denotes parts of the input that are not part of any slot. For instance, in a sentence where we want to label a """"date"""" slot, words containing date information might be tagged as """"B-date"""" (indicating the beginning of a date slot), followed by consecutive words carrying date information tagged as """"I-date"""" (indicating the continuation of the date slot), while words not containing date information would be tagged as """"O"""" (indicating they are outside any slot). Definition: In this task, you are given a conversation, where the words spoken by a person are shown as a list. Your job is to classify the words in the following conversation into one of the 37 different entities. The entities are: 'I-abstract', 'B-object', 'B-place', 'I-substance', 'I-time', 'I-place', 'B-time', 'B-abstract', 'I-person', 'B-plant', 'B-substance', 'I-animal', 'B-organization', 'I-event', 'B-person', 'B-event', 'I-plant', 'I-organization', 'O', 'I-object', 'B-animal'. Only output entities. And the entity types should be output as a list without any explanation. The input is ['Sir', 'de', 'Villiers', 'Graaff', ',', 'Leader', 'of', 'the', 'Opposition', ',', 'House', 'of', 'Assembly', ',', 'CAPE', 'TOWN', 'Sir', ',', 'In', 'one', ""week's"", 'week', ""'s"", 'time', ',', 'the', 'Verwoerd', 'Government', 'intends', 'to', 'inaugurate', 'its', 'Republic', '.', 'It', 'is', 'unnecessary', 'to', 'state', 'that', 'this', 'intention', 'has', 'never', 'been', 'endorsed', 'by', 'the', 'non-white', 'majority', 'of', 'this', 'country', '.', 'The', 'decision', 'has', 'been', 'taken', 'by', 'little', 'over', 'half', 'of', 'the', 'White', 'community', ';', ...]."

Case 6 – Our Model

['B-person', 'I-person', 'I-person', 'I-person', 'O', 'B-person', 'I-person', 'B-organization',...]

Case 6 - GPT-4o Response

['B-person', 'I-person', 'I-person', 'O', 'B-person', 'O', 'O', 'O', ...]

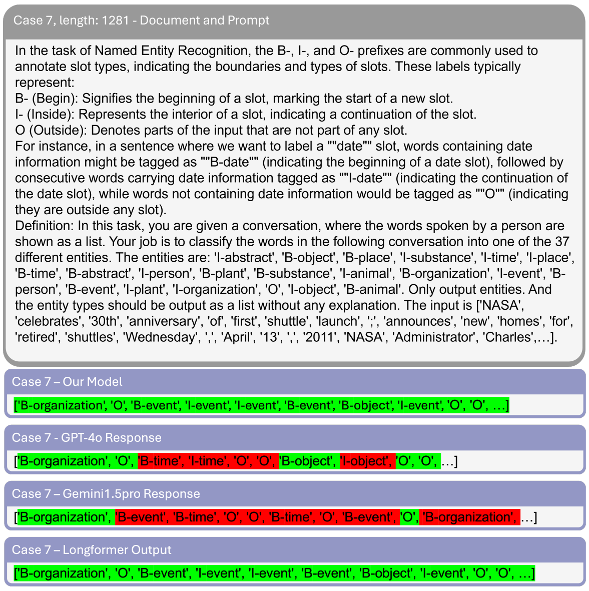
Case 6 – Gemini1.5pro Response

['B-person', 'B-person', 'O', 'O', 'O', 'B-organization', 'O', 'O', ...]

Case 6 – Longformer Output

['B-person', 'I-person', 'I-person', 'I-person', 'O', 'B-person', 'I-person', 'B-organization', ...]

Figure 8: Prompt and output for a sample document of length 1029 in **GUM** dataset for NER task, where correct predictions are highlighted in green and wrong predictions are highlighted in red.



Case 7, length: 1281 - Document and Prompt

In the task of Named Entity Recognition, the B-, I-, and O- prefixes are commonly used to annotate slot types, indicating the boundaries and types of slots. These labels typically represent:
B- (Begin): Signifies the beginning of a slot, marking the start of a new slot.
I- (Inside): Represents the interior of a slot, indicating a continuation of the slot.
O (Outside): Denotes parts of the input that are not part of any slot.
For instance, in a sentence where we want to label a ""date"" slot, words containing date information might be tagged as ""B-date"" (indicating the beginning of a date slot), followed by consecutive words carrying date information tagged as ""I-date"" (indicating the continuation of the date slot), while words not containing date information would be tagged as ""O"" (indicating they are outside any slot).
Definition: In this task, you are given a conversation, where the words spoken by a person are shown as a list. Your job is to classify the words in the following conversation into one of the 37 different entities. The entities are: 'I-abstract', 'B-object', 'B-place', 'I-substance', 'I-time', 'I-place', 'B-time', 'B-abstract', 'I-person', 'B-plant', 'B-substance', 'I-animal', 'B-organization', 'I-event', 'B-person', 'B-event', 'I-plant', 'I-organization', 'O', 'I-object', 'B-animal'. Only output entities. And the entity types should be output as a list without any explanation. The input is ['NASA', 'celebrates', '30th', 'anniversary', 'of', 'first', 'shuttle', 'launch', ';', 'announces', 'new', 'homes', 'for', 'retired', 'shuttles', 'Wednesday', ',', 'April', '13', ',', '2011', 'NASA', 'Administrator', 'Charles',…].

Case 7 – Our Model

['B-organization', 'O', 'B-event', 'I-event', 'I-event', 'B-event', 'B-object', 'I-event', 'O', 'O', …]

Case 7 - GPT-4o Response

['B-organization', 'O', 'B-time', 'I-time', 'O', 'O', 'B-object', 'I-object', 'O', 'O', …]

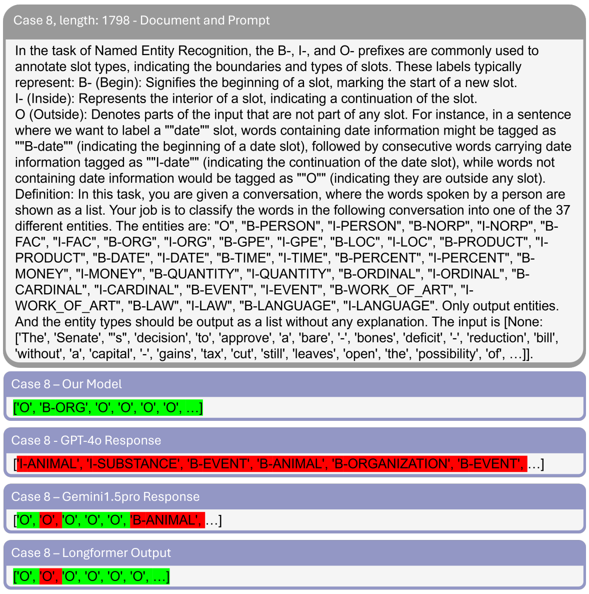
Case 7 – Gemini1.5pro Response

['B-organization', 'B-event', 'B-time', 'O', 'O', 'B-time', 'O', 'B-event', 'O', 'B-organization', …]

Case 7 – Longformer Output

['B-organization', 'O', 'B-event', 'I-event', 'I-event', 'B-event', 'B-object', 'I-event', 'O', 'O', …]

Figure 9: Prompt and output for a sample document of length 1281 in **GUM** dataset for NER task, where correct predictions are highlighted in green and wrong predictions are highlighted in red.



Case 8, length: 1798 - Document and Prompt

In the task of Named Entity Recognition, the B-, I-, and O- prefixes are commonly used to annotate slot types, indicating the boundaries and types of slots. These labels typically represent: B- (Begin): Signifies the beginning of a slot, marking the start of a new slot.
I- (Inside): Represents the interior of a slot, indicating a continuation of the slot.
O (Outside): Denotes parts of the input that are not part of any slot. For instance, in a sentence where we want to label a ""date"" slot, words containing date information might be tagged as ""B-date"" (indicating the beginning of a date slot), followed by consecutive words carrying date information tagged as ""I-date"" (indicating the continuation of the date slot), while words not containing date information would be tagged as ""O"" (indicating they are outside any slot).
Definition: In this task, you are given a conversation, where the words spoken by a person are shown as a list. Your job is to classify the words in the following conversation into one of the 37 different entities. The entities are: "O", "B-PERSON", "I-PERSON", "B-NORP", "I-NORP", "B-FAC", "I-FAC", "B-ORG", "I-ORG", "B-GPE", "I-GPE", "B-LOC", "I-LOC", "B-PRODUCT", "I-PRODUCT", "B-DATE", "I-DATE", "B-TIME", "I-TIME", "B-PERCENT", "I-PERCENT", "B-MONEY", "I-MONEY", "B-QUANTITY", "I-QUANTITY", "B-ORDINAL", "I-ORDINAL", "B-CARDINAL", "I-CARDINAL", "B-EVENT", "I-EVENT", "B-WORK_OF_ART", "I-WORK_OF_ART", "B-LAW", "I-LAW", "B-LANGUAGE", "I-LANGUAGE". Only output entities. And the entity types should be output as a list without any explanation. The input is [None: ['The', 'Senate', "'s", 'decision', 'to', 'approve', 'a', 'bare', '-', 'bones', 'deficit', '-', 'reduction', 'bill', 'without', 'a', 'capital', '-', 'gains', 'tax', 'cut', 'still', 'leaves', 'open', 'the', 'possibility', 'of', …]].

Case 8 – Our Model

['O', 'B-ORG', 'O', 'O', 'O', 'O', …]

Case 8 - GPT-4o Response

['I-ANIMAL', 'I-SUBSTANCE', 'B-EVENT', 'B-ANIMAL', 'B-ORGANIZATION', 'B-EVENT', …]

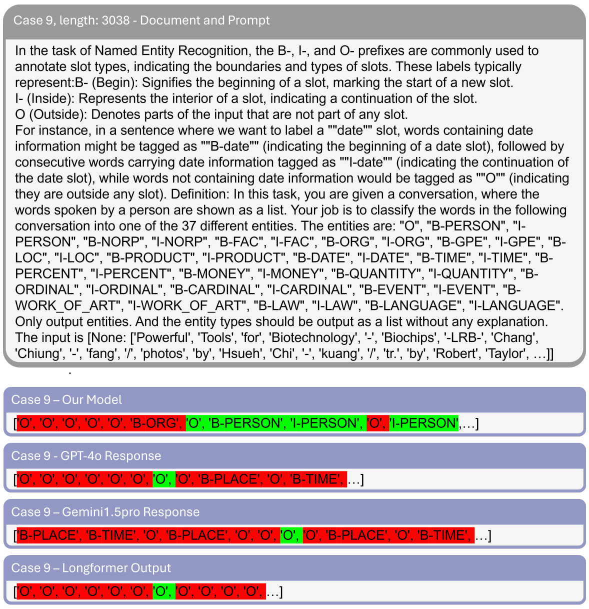
Case 8 – Gemini1.5pro Response

['O', 'O', 'O', 'O', 'O', 'B-ANIMAL', …]

Case 8 – Longformer Output

['O', 'O', 'O', 'O', 'O', 'O', …]

Figure 10: Prompt and output for a sample document of length 1798 in **CoNLL** dataset for NER task, where correct predictions are highlighted in green and wrong predictions are highlighted in red.



Case 9, length: 3038 - Document and Prompt

In the task of Named Entity Recognition, the B-, I-, and O- prefixes are commonly used to annotate slot types, indicating the boundaries and types of slots. These labels typically represent:B- (Begin): Signifies the beginning of a slot, marking the start of a new slot.
I- (Inside): Represents the interior of a slot, indicating a continuation of the slot.
O (Outside): Denotes parts of the input that are not part of any slot.
For instance, in a sentence where we want to label a ""date"" slot, words containing date information might be tagged as ""B-date"" (indicating the beginning of a date slot), followed by consecutive words carrying date information tagged as ""I-date"" (indicating the continuation of the date slot), while words not containing date information would be tagged as ""O"" (indicating they are outside any slot). Definition: In this task, you are given a conversation, where the words spoken by a person are shown as a list. Your job is to classify the words in the following conversation into one of the 37 different entities. The entities are: "O", "B-PERSON", "I-PERSON", "B-NORP", "I-NORP", "B-FAC", "I-FAC", "B-ORG", "I-ORG", "B-GPE", "I-GPE", "B-LOC", "I-LOC", "B-PRODUCT", "I-PRODUCT", "B-DATE", "I-DATE", "B-TIME", "I-TIME", "B-PERCENT", "I-PERCENT", "B-MONEY", "I-MONEY", "B-QUANTITY", "I-QUANTITY", "B-ORDINAL", "I-ORDINAL", "B-CARDINAL", "I-CARDINAL", "B-EVENT", "I-EVENT", "B-WORK_OF_ART", "I-WORK_OF_ART", "B-LAW", "I-LAW", "B-LANGUAGE", "I-LANGUAGE".
Only output entities. And the entity types should be output as a list without any explanation.
The input is [None: ['Powerful', 'Tools', 'for', 'Biotechnology', '-', 'Biochips', '-LRB-', 'Chang', 'Chiung', '-', 'fang', '/', 'photos', 'by', 'Hsueh', 'Chi', '-', 'kuang', '/', 'tr.', 'by', 'Robert', 'Taylor', …]]

Case 9 – Our Model

['O', 'O', 'O', 'O', 'O', 'B-ORG', 'O', 'B-PERSON', 'I-PERSON', 'O', 'I-PERSON',…]

Case 9 - GPT-4o Response

['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-PLACE', 'O', 'B-TIME', …]

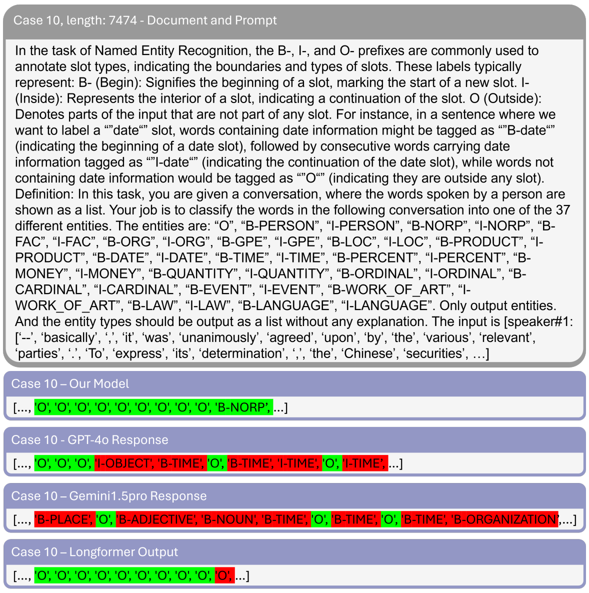
Case 9 – Gemini1.5pro Response

['B-PLACE', 'B-TIME', 'O', 'B-PLACE', 'O', 'O', 'O', 'O', 'B-PLACE', 'O', 'B-TIME', …]

Case 9 – Longformer Output

['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', …]

Figure 11: Prompt and output for a sample document of length 3038 in **CoNLL** dataset for NER task, where correct predictions are highlighted in green and wrong predictions are highlighted in red.



Case 10, length: 7474 - Document and Prompt

In the task of Named Entity Recognition, the B-, I-, and O- prefixes are commonly used to annotate slot types, indicating the boundaries and types of slots. These labels typically represent: B- (Begin): Signifies the beginning of a slot, marking the start of a new slot. I- (Inside): Represents the interior of a slot, indicating a continuation of the slot. O (Outside): Denotes parts of the input that are not part of any slot. For instance, in a sentence where we want to label a “”date“” slot, words containing date information might be tagged as “”B-date“” (indicating the beginning of a date slot), followed by consecutive words carrying date information tagged as “”I-date“” (indicating the continuation of the date slot), while words not containing date information would be tagged as “”O“” (indicating they are outside any slot).
Definition: In this task, you are given a conversation, where the words spoken by a person are shown as a list. Your job is to classify the words in the following conversation into one of the 37 different entities. The entities are: “O”, “B-PERSON”, “I-PERSON”, “B-NORP”, “I-NORP”, “B-FAC”, “I-FAC”, “B-ORG”, “I-ORG”, “B-GPE”, “I-GPE”, “B-LOC”, “I-LOC”, “B-PRODUCT”, “I-PRODUCT”, “B-DATE”, “I-DATE”, “B-TIME”, “I-TIME”, “B-PERCENT”, “I-PERCENT”, “B-MONEY”, “I-MONEY”, “B-QUANTITY”, “I-QUANTITY”, “B-ORDINAL”, “I-ORDINAL”, “B-CARDINAL”, “I-CARDINAL”, “B-EVENT”, “I-EVENT”, “B-WORK_OF_ART”, “I-WORK_OF_ART”, “B-LAW”, “I-LAW”, “B-LANGUAGE”, “I-LANGUAGE”. Only output entities. And the entity types should be output as a list without any explanation. The input is [speaker#1: [‘--’, ‘basically’, ‘,’, ‘it’, ‘was’, ‘unanimously’, ‘agreed’, ‘upon’, ‘by’, ‘the’, ‘various’, ‘relevant’, ‘parties’, ‘.’, ‘To’, ‘express’, ‘its’, ‘determination’, ‘,’, ‘the’, ‘Chinese’, ‘securities’, …]

Case 10 – Our Model

[..., 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-NORP', ...]

Case 10 - GPT-4o Response

[..., 'O', 'O', 'O', 'I-OBJECT', 'B-TIME', 'O', 'B-TIME', 'I-TIME', 'O', 'I-TIME', ...]

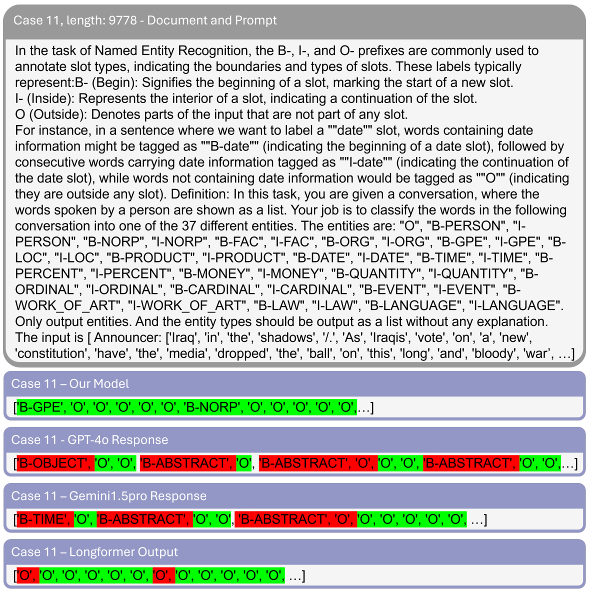
Case 10 – Gemini1.5pro Response

[..., 'B-PLACE', 'O', 'B-ADJECTIVE', 'B-NOUN', 'B-TIME', 'O', 'B-TIME', 'O', 'B-TIME', 'B-ORGANIZATION',...]

Case 10 – Longformer Output

[..., 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', ...]

Figure 12: Prompt and output for a sample document of length 7474 in **CoNLL** dataset for NER task, where correct predictions are highlighted in green and wrong predictions are highlighted in red.



Case 11, length: 9778 - Document and Prompt

In the task of Named Entity Recognition, the B-, I-, and O- prefixes are commonly used to annotate slot types, indicating the boundaries and types of slots. These labels typically represent:B- (Begin): Signifies the beginning of a slot, marking the start of a new slot.
I- (Inside): Represents the interior of a slot, indicating a continuation of the slot.
O (Outside): Denotes parts of the input that are not part of any slot.
For instance, in a sentence where we want to label a ""date"" slot, words containing date information might be tagged as ""B-date"" (indicating the beginning of a date slot), followed by consecutive words carrying date information tagged as ""I-date"" (indicating the continuation of the date slot), while words not containing date information would be tagged as ""O"" (indicating they are outside any slot). Definition: In this task, you are given a conversation, where the words spoken by a person are shown as a list. Your job is to classify the words in the following conversation into one of the 37 different entities. The entities are: "O", "B-PERSON", "I-PERSON", "B-NORP", "I-NORP", "B-FAC", "I-FAC", "B-ORG", "I-ORG", "B-GPE", "I-GPE", "B-LOC", "I-LOC", "B-PRODUCT", "I-PRODUCT", "B-DATE", "I-DATE", "B-TIME", "I-TIME", "B-PERCENT", "I-PERCENT", "B-MONEY", "I-MONEY", "B-QUANTITY", "I-QUANTITY", "B-ORDINAL", "I-ORDINAL", "B-CARDINAL", "I-CARDINAL", "B-EVENT", "I-EVENT", "B-WORK_OF_ART", "I-WORK_OF_ART", "B-LAW", "I-LAW", "B-LANGUAGE", "I-LANGUAGE". Only output entities. And the entity types should be output as a list without any explanation. The input is [Announcer: ['Iraq', 'in', 'the', 'shadows', '/.', 'As', 'Iraqis', 'vote', 'on', 'a', 'new', 'constitution', 'have', 'the', 'media', 'dropped', 'the', 'ball', 'on', 'this', 'long', 'and', 'bloody', 'war', …]

Case 11 – Our Model

['B-GPE', 'O', 'O', 'O', 'O', 'O', 'B-NORP', 'O', 'O', 'O', 'O', 'O',…]

Case 11 - GPT-4o Response

['B-OBJECT', 'O', 'O', 'B-ABSTRACT', 'O', 'B-ABSTRACT', 'O', 'O', 'O', 'B-ABSTRACT', 'O', 'O',…]

Case 11 – Gemini1.5pro Response

['B-TIME', 'O', 'B-ABSTRACT', 'O', 'O', 'B-ABSTRACT', 'O', 'O', 'O', 'O', 'O', 'O', …]

Case 11 – Longformer Output

['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', …]

Figure 13: Prompt and output for a sample document of length 9778 in **CoNLL** dataset for NER task, where correct predictions are highlighted in green and wrong predictions are highlighted in red.