Isnad AI at IslamicEval 2025: A Rule-Based System for Identifying Islamic Citation in LLM Outputs

Fatimah Emad Eldin
Department of Computer and Information Sciences
Faculty of Graduate Studies for Statistical Research, Cairo University
12422024441586@pg.cu.edu.eg

Abstract

This paper presents the Isnad AI system developed for the IslamicEval 2025 Shared Task 1A, which focuses on identifying character-level spans of Quranic verses (Ayahs) and Prophetic sayings (Hadiths) within Large Language Model (LLM) outputs. This task is formulated as a token classification problem using a fine-tuned AraBERTv2 model. The primary contribution is a novel rule-based data preprocessing and augmentation pipeline, through which a large-scale, high-quality training corpus is systematically generated from raw religious texts. Through comprehensive ablation studies, it is demonstrated that the controlled synthetic data generation approach significantly outperforms traditional database lookup methods and basic fine-tuning approaches. The system achieved an F1 score of 66.97% in the official test set, demonstrating the effectiveness of principled synthetic data generation for specialized religious text verification tasks. To support reproducibility and future research in Islamic citation detection, all code, generated datasets, and experimental resources are made publicly available on GitHub and Hugging Face.

1 Introduction

The proliferation of Large Language Models (LLMs) has created an urgent need for robust mechanisms to verify factual accuracy (**?**Li et al., 2024), particularly in specialized domains like Islamic studies (Nagoudi et al., 2022; Antoun et al., 2021). The IslamicEval 2025 Shared Task 1A addresses this by requiring systems to detect precise character-level spans of religious citations within Arabic LLM responses (Mubarak et al., 2025), representing a foundational step for subsequent fact-checking systems. The submitted system employs a token classification method using a fine-tuned AraBERTv2 model (Antoun et al., 2020).

Given the absence of large, manually annotated corpora for this task. A rule-based process was developed to programmatically generate clean, contextualized training data, embedding authentic religious texts within varied templates to simulate LLM citation patterns.

Ablation studies revealed that this rule-based data generation methodology outperforms database lookup, and basic fine-tuning. To foster reproducibility and support future research in Islamic religious citation, all experimental code, dataset, and the final fine-tuned model are publicly available on GitHub[1] and Hugging Face[2].

2 Background

This work addresses the IslamicEval 2025 Shared Task 1A, which requires identifying character-level spans of Quranic verses (Ayahs) and Prophetic sayings (Hadiths) within LLM-generated Arabic text (Mubarak et al., 2025). For a given text containing citations, the system must identify the exact start and end character indices. The required submission format is detailed in the Appendix B.3 in Table 4. This task is structured as a token classification problem using the standard BIO schema to label the boundaries of religious citations (Ramshaw and Marcus, 1995; Devlin et al., 2019).

2.1 Related Work

This work is situated within the broader field of adapting language models for specialized religious domains (Nagoudi et al., 2022). While there has been progress in this area, this verification task presents a unique challenge because existing resources are not suitable for precise, character-level span detection. Foundational datasets like the Quranic Arabic Corpus (Dukes and Habash, 2010) provide deep morphological analysis, and there are

[1] https://github.com/astral-fate/IslamicEval
[2] https://huggingface.co/collections/FatimahEmadEldin/isnad-ai-at-islamiceval-68a64677910651f034b9cdf4

Dataset Split	Unique Texts	Ayah Examples	Hadith Examples	Total Generated
Training Set	30,548	20,622	72,477	93,099
Validation Set	13,354	20,313	20,313	40,626
TOTAL	**43,902**	**40,935**	**92,790**	**133,725**

(a) Final generated splits with class breakdown.

Corpus	Original Count
Quranic Verses (Ayahs)	6,236
Hadith Narrations	34,662
Total Unique Texts	**40,898**

(b) Original source data.

Corpus	Preprocessed Count
Total Unique Ayahs	13,456
Total Unique Hadiths	30,446
Total Unique Texts	**43,902**

(c) After preprocessing.

Table 1: Corrected dataset statistics at each stage. Table (a) shows the final splits and total generated examples based on the actual output files. Table (b) shows the initial counts from source files. Table (c) shows the total number of unique texts available for splitting after all processing and augmentation.

models fine-tuned for Islamic question-answering (Ellbendis, 2024; Justdeen, 2024).

However, these resources were not designed for the specific purpose of identifying exact citation boundaries within a larger text.

This creates a significant data scarcity problem for this particular task. To address this gap, the primary contribution of this work is a novel rule-based data generation pipeline. This approach was developed to create a suitable, large-scale training corpus, directly overcoming the lack of annotated data for this specialized verification task (Hedderich et al., 2021).

3 System Overview

3.1 Core Model

The foundation of the system is a fine-tuned implementation of AraBERTv2 (Antoun et al., 2020), a powerful transformer-based model pre-trained on a large corpus of Arabic text. For this task, the model was adapted for token classification and fine-tuned to predict labels according to the standard BIO schema: B-Ayah, I-Ayah, B-Hadith, I-Hadith, or O (Outside). Through this approach, the system can effectively identify the precise boundaries of religious citations at a granular level within LLM-generated text. The model was trained exclusively on the synthetically generated dataset, which is detailed in section 4.

3.2 Training Data Generation

The central methodological contribution is the programmatic generation of a large-scale training corpus. This approach was developed to overcome the lack of manually annotated data by creating high-quality, contextualized examples to simulate how they appear as in-context citations within LLM outputs. The process is detailed in section 4.3.

4 Data and Preprocessing Pipeline

The entire training and validation dataset was synthetically generated from raw Islamic texts using a multi-stage pipeline designed to create diverse and realistic training examples.

4.1 Data Sources

Two foundational datasets of sacred Islamic texts were utilized for this paper. These datasets, provided in a pre-processed format by the task organizers, consist of the following:

- **The Holy Quran** (KFG, 2025): The complete text of the Holy Quran, presented in a JSON file where each entry corresponds to a specific verse (*ayah*) [3].
- **The Hadith**: A collection of prophetic traditions (narrations) from the Six Major Books of Hadith, provided in a JSON file [4].

For model fine-tuning, only Hadith entries containing a non-empty 'Matn' (the core narrative text of the prophetic tradition) were used. The initial

[3] https://github.com/qcri/IslamicEval-2025-Subtask-1/blob/main/Quran/quranic_verses.json

[4] https://github.com/qcri/IslamicEval-2025-Subtask-1/blob/main/Hadith/six_hadith_books.json

distribution of these datasets is summarized in Table 1b.

4.2 Data Preprocessing and Augmentation Pipeline

The preprocessing pipeline systematically transforms raw Islamic texts into a comprehensive training corpus through five interconnected stages (detailed methodology in Appendix D). The process begins with systematic text segmentation and Arabic script normalization, followed by template-based contextual generation that embeds authentic religious texts within realistic citation patterns. The complete pipeline workflow is illustrated in Figure 1, Figure 2 and Figure 3.

1. **Text Splitting**: Quranic verses were analyzed using the AraBERTv2 tokenizer. Any verse exceeding a 25-token length was split into two smaller, more manageable segments. This process increased the total number of Ayah from 6,236 to 6,910.

2. **Normalization and Augmentation**: To improve the model's robustness against variations in Arabic script, a data augmentation technique was applied. For every Ayah (both original and segmented), a duplicate version was created with all diacritics (*Tashkeel*) removed.

3. **Template-Based Generation**: The core of the pipeline involves embedding the processed religious texts into contextual templates. A set of common prefixes and suffixes were manually curated for both Ayahs and Hadiths based on a qualitative analysis of common citation patterns in contemporary Arabic writing. The lists in Table 13 provide the comprehensive examples of suffix and prefix of the rule-based templates.

The data distribution after these preprocessing and augmentation is shown in Table 1c.

4.3 Dataset Splits

The synthetic data generation pipeline produced a corpus from 43,902 unique religious texts. This corpus was split into the internal training and validation sets to fine-tune the AraBERTv2 model. A 70/30 split was employed, allocating 70% of the unique source texts for the training set and 30% for the internal validation set. The template-based generation process was then applied to these partitioned texts, resulting in the final example counts shown in Table 1a. For final evaluation, the official datasets provided by the shared task organizers was used. The model's performance on the development set (referred to as "Dev F1" in Table 2) was measured against the organizers' manually annotated 'dev SubtaskA' files, containing 210 records. The final competition score (referred to as "Test F1") was evaluated on the official blind test set of 190 records. The internal validation set was used exclusively for hyperparameter tuning and to prevent overfitting during the fine-tuning phase.

Methodology	**Dev F1**	**Test F1**
Rule-Based Model	65.08%	**66.97%**
Ablation Baselines:		
Database Lookup	52.00%	34.80%
Basic Fine-Tuning	33.00%	44.70%

Table 2: Comprehensive results across development and test sets compared to ablation baselines.

5 Experimental Setup

5.1 Evaluation Metric

The official evaluation metric for the task is the Macro-Averaged F1 Score computed at the character level (Mubarak et al., 2025). Unlike span-based evaluation, this metric treats each character of the response string as an independent classification unit, assigning it one of three labels: Ayah, Hadith, or Neither. The F1 score is then computed as the harmonic mean of Precision (P) and Recall (R). The macro-averaged F1 score computes the F1 score for each class independently and then averages them, giving equal weight to each class regardless of its frequency. This character-level evaluation ensures the system is assessed on its ability to precisely identify the boundaries of religious texts at the finest granularity, making it more stringent than span-based metrics (Tjong Kim Sang and De Meulder, 2003).

6 Results

6.1 Ablation Study Analysis

Comprehensive ablation studies were conducted to evaluate the proposed rule-based synthetic data generation approach against two baseline methodologies: database lookup and basic fine-tuning without synthetic augmentation. The experimental results demonstrate substantial superiority of

the rule-based model across both evaluation sets. As presented in Table 2, the rule-based approach achieved macro F1 scores of 65.08% on the development set and 66.97% on the official test set. The baseline models performed significantly worse on the development set, with the database lookup method achieving an F1 score of 52% and the basic fine-tuning approach achieving 33%. While both baselines showed limitations, the results validate the effectiveness of principled synthetic data generation, demonstrating a performance improvement of 22.27% over basic fine-tuning on the official test set.

7 Error Analysis

The error analysis was conducted on the development set, detailed in Appendix F, as the ground truth for the final blind test set was not provided by the shared task organizers.

7.1 Impact of Class Imbalance

A significant class imbalance exists, with the 'Neither' class comprising 67.8% of characters, while 'Ayah' and 'Hadith' account for only 20.2% and 12.0%, respectively (see Appendix C). This class imbalance is reflected in the F1-scores: 0.90 for the majority 'Neither' class, 0.67 for 'Ayah', and a significantly lower 0.39 for the 'Hadith' class. The primary weakness is identifying Hadith, a challenge compounded by their narrative style and significant textual variance across different Hadith books, making them harder to distinguish, in comparison to Quranic verses.

7.2 Span-Level Error Patterns

A span-level analysis reveals the model produced more False Negatives (101 missed spans) than True Positives (78 correct spans). Missed spans were comparable in length to correctly identified ones, suggesting the model tends to miss entire citations. Conversely, False Positives were predominantly short fragments, indicating a tendency to misclassify small, unrelated phrases.

8 Discussion

The experimental results highlight the critical role of data quality in training models for specialized verification tasks. Several approaches were evaluated, including a database lookup method and basic fine-tuning. A generative synthetic data approach using AraGPT2 (Antoun et al., 2021) was evaluated; however, the generative synthetic data proved inappropriate. Using the prompt templates shown in Table 14, the model produced significant noise. As detailed in Appendix G and exemplified in Table 15, the outputs included nonsensical fragments and contextual hallucinations, creating misleading training data. These results validate that the structured, rule-based approach to synthetic data generation was the most effective strategy for this task. The system's primary challenge remained in Hadith identification, where performance was hindered by significant textual variation in narrations across the six major books of Hadith. This high degree of narrative variation, unlike the uniformity of Quranic verses, poses a significant modeling challenge.

9 Conclusion

This paper presented the Isnad AI system for identifying religious citations in LLM outputs using fine-tuned AraBERTv2 with a novel rule-based synthetic data generation pipeline. The system achieved 66.97% F1 on the test set, significantly outperforming database lookup (34.80%) and basic fine-tuning (44.70%), validating the effectiveness of principled synthetic data generation for specialized verification tasks. The primary limitation was Hadith identification (F1: 0.39 vs. Quranic verses: 0.67), attributed to the textual variation of the Matn across different narrators. Future work should confine training to a single Hadith book, such as Sahih al-Bukhari (al Bukhari, 1871), explore class-balanced sampling, and develop techniques for detecting corrupted or paraphrased citations. The latter could be achieved by enhancing the lookup baseline with fuzzy matching algorithms or by augmenting the training data with synthetically generated textual variations to improve the deep learning model's robustness.

Acknowledgments

The organizers of IslamicEval 2025 Shared Task are gratefully acknowledged for their efforts in creating this important benchmark for Arabic religious text processing.

References

2025. *Al-Qur'an al-Karim (Mushaf al-Madinah an-Nabawiyyah)*. King Fahd Complex for the Printing of the Holy Qur'an, Medina, Saudi Arabia. Arabic

text based on the Uthmanic (script), in the narration of Hafs from Asim. Corresponds to Hijri year 1446-1447 AH.

Muhammad Ismail al Bukhari. 1871. *Sahih al-Bukhari*. King Fahd National Library - Riyadh.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France. European Language Resource Association.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2021. AraGPT2: Pre-trained transformer for Arabic language generation. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 196–207, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

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

Kais Dukes and Nizar Habash. 2010. Morphological annotation of Quranic Arabic. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Ellbendis. 2024. Qwen3-4b-quran-lora-fine-tuned. https://huggingface.co/Ellbendls/Qwen3-4b-Quran-LoRA-Fine-Tuned. Accessed: 2025-08-16.

Michael A. Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. 2021. A survey on recent approaches for natural language processing in low-resource scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2545–2568, Online. Association for Computational Linguistics.

Justdeen. 2024. Quranplus. https://huggingface.co/justdeen/QuranPlus. Accessed: 2025-08-16.

Siheng Li, Cheng Yang, Taiqiang Wu, Chufan Shi, Yuji Zhang, Xinyu Zhu, Zesen Cheng, Deng Cai, Mo Yu, Lemao Liu, Jie Zhou, Yujiu Yang, Ngai Wong, Xixin Wu, and Wai Lam. 2024. A survey on the honesty of large language models. *arXiv preprint arXiv:2409.18786*.

Hamdy Mubarak, Rana Malhas, Watheq Mansour, Abubakr Mohamed, Mahmoud Fawzi, Majd Hawasly, Tamer Elsayed, Kareem Darwish, and Walid Magdy. 2025. IslamicEval 2025: The First Shared Task of Capturing LLMs Hallucination in Islamic Content. In *Proceedings of the Third Arabic Natural Language Processing Conference (ArabicNLP 2025)*, Suzhou, China. Association for Computational Linguistics. Co-located with EMNLP 2025, November 5–9.

El Moatez Billah Nagoudi, AbdelRahim Elmadany, and Muhammad Abdul-Mageed. 2022. AraT5: Text-to-text transformers for Arabic language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 628–647, Dublin, Ireland. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

A Experimental Configuration

Table 3 provides the hyperparameter settings used for fine-tuning the AraBERTv2 model (Antoun et al., 2020).

Parameter	Value
Model	aubmindlab/bert-base-arabertv2
Max Epochs	10 (with early stopping)
Learning Rate	2×10^{-5}
Batch Size (per device)	4
Gradient Accumulation Steps	4
Effective Batch Size	16
Optimizer	AdamW
Weight Decay	0.01
Warmup Steps	500
Mixed Precision	fp16 enabled
Max Sequence Length	512 tokens
Early Stopping Patience	3 epochs

Table 3: Complete hyperparameter configuration for model training.

Question_ID	Span_Start	Span_End	Span_Type
A-Q001	11	25	Ayah
A-Q001	34	52	Ayah
A-Q001	67	87	Hadith

Table 4: Example submission file format.

B Data Structures and Format Specifications

This appendix provides detailed specifications of the data structures used throughout the system, including source data formats and submission requirements.

B.1 Source Data Structures

Quranic Verses The Quranic data is structured as a JSON array, in which each object corresponds to a single verse.

```
{
  "surah_id": 1,
  "surah_name": "الفاتحة",
  "ayah_id": 1,
  "ayah_text": "بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ"
},
```

Hadith The Hadith data was structured as a JSON array, with each object representing a Hadith.

```
{
  "hadithID": 5,
  "BookID": 1.0,
  "title": "...",
  "hadithTxt": "...",
  "Matn": "إِنَّمَا الأَعْمَالُ بِالنِّيَّاتِ..."
},
```

B.2 Test Data Format

The test data is in XML format, with each <Question> block containing the LLM's response.

```
<Question>
  <ID>A-Q001</ID>
  <Model>Model-6</Model>
  <Text>هل يمكن أن يكون
...الابتلاء</Text>
  <Response>
    نعم، يمكن أن يكون الابتلاء...

  </Response>
</Question>
```

B.3 Submission Format

The required submission is a Tab-Separated Values (TSV) file with the columns: Question_ID, Span_Start, Span_End, and Span_Type. An example is shown in Table 4.

C Development Set: Exploratory Data Analysis

To better understand the composition of the dataset used for evaluation, an exploratory data analysis (EDA) was performed on the development set. This set consists of 50 questions and their corresponding responses, containing a total of 210 manually annotated spans of text. The analysis was conducted at the character level to align with the official scoring methodology. The primary finding is a significant class imbalance within the data, as detailed in Table 5a. The 'Neither' class, representing text that is not part of a religious quotation, constitutes over two-thirds of the total characters. The 'Ayah' class is the most represented quotation type, accounting for 20.2% of the characters, while the 'Hadith' class is the least represented at 12.0%. Further analysis of the annotated spans, summarized in Table 5b, reveals additional insights. There are more distinct 'Ayah' spans (118) than 'Hadith' spans (76). A notable characteristic is the high variance in the length of these spans. For both classes, the standard deviation is nearly as large as the mean, and the lengths range from very short fragments to extensive passages of over 600 characters. This indicates that the model must be capable of identifying quotations of highly variable lengths.

D Comprehensive Data Preprocessing Pipeline

This appendix details the rule-based data preprocessing pipeline designed to transform raw Islamic texts into a high-quality training corpus for token classification. The architecture is composed of five sequential stages: (1) Data Acquisition and Validation, (2) Text Preprocessing and Augmentation, (3) Template-Based Data Generation, (4) Dataset Partitioning, and (5) Tokenization with Label Assignment.

D.1 Stage 1: Data Acquisition and Validation

The initial corpus was established from two primary sources provided by the shared task organizers: the Quranic corpus, containing 6,236 verses (Ayahs), and a Hadith collection of 34,662 prophetic narrations. During acquisition, textual content was extracted from designated fields within the source JSON files: 'ayah text' for the Quran and 'Matn' (the core narrative) for the Hadith. To ensure corpus integrity and manage com-

Class	Count	Pct.
Neither	44,273	67.8%
Ayah	13,173	20.2%
Hadith	7,864	12.0%
Total	**65,310**	**100.0%**

(a) Character-level class distribution.

Class	Spans	Mean	Std.	Min	Max
Ayah	118	111.6	105.1	6	678
Hadith	76	103.5	93.0	8	690

(b) Descriptive statistics for annotated spans.

Table 5: Summary of the development set's class distribution by character count (a) and annotated span statistics (b).

putational resources, two validation measures were implemented:

- **Length Threshold:** A maximum text length of 1,500 characters was imposed to prevent memory overflow, while retaining the vast majority of authentic texts.
- **Encoding Validation:** All texts were validated for proper UTF-8 encoding to ensure correct handling of the Arabic script.

D.2 Stage 2: Text Preprocessing and Augmentation

This stage addresses linguistic and tokenizer-specific challenges through text segmentation and script normalization.

D.2.1 Text Segmentation

To accommodate the processing limitations of the AraBERTv2 tokenizer, texts exceeding a 25-token threshold were systematically segmented. The segmentation algorithm identifies the approximate midpoint of a text and performs a backward search for the nearest word boundary (whitespace). This content-aware strategy prevents splitting words, thus preserving semantic coherence. This process expanded the initial 6,236 Quranic verses into 6,910 processable text segments. While some Quranic verses remained long even after bisection, the split was limited to two parts to minimize the risk of excessive fragmentation and loss of contextual meaning; multi-part splitting strategies are reserved for future work.

D.2.2 Arabic Script Normalization

To enhance model robustness against script variations, a data augmentation strategy involving diacritic removal was applied. For each Ayah (original and segmented), a normalized variant was generated by removing all diacritical marks (Tashkeel) and the Tatweel character, which correspond to the Unicode range [\u064B-\u0652\u0640]. This technique effectively doubled the unique Ayah corpus to 13,456, ensuring the model can recognize verses regardless of their vocalization.

D.3 Stage 3: Template-Based Contextual Generation

This stage is the core of the synthetic data generation pipeline, designed to programmatically create a large-scale, high-quality training corpus. The primary objective is to embed the clean, preprocessed religious texts from the previous stage into varied contextual templates, thereby simulating the patterns commonly observed when Large Language Models (LLMs) cite religious sources. The generation process is algorithmic and designed to produce multiple unique examples from a single source text, significantly augmenting the dataset. For each source text (either a Quranic Ayah or a Hadith), the system executes the following steps, as illustrated in the workflow diagram in Figure 2 3:

1. **Template Component Selection**: Based on the text's label (Ayah or Hadith), the system randomly selects a corresponding prefix and suffix from a manually curated list. These lists, detailed in the paper's Table 13, contain common introductory and concluding phrases used in contemporary Arabic writing to cite religious texts.
2. **Contextual Enrichment**: To enhance the realism of the generated examples, a neutral or transitional sentence is added with a 30% probability. This sentence is randomly selected from a predefined list and is inserted either before or after the religious text to break simplistic patterns and better mimic the flow of natural language.
3. **Concatenation and Normalization**: The selected components are combined in one of the following structures:

- suffix + source_text + prefix
- + source_text + neutral_context + prefix suffix
- + neutral_context + source_text + prefix suffix

The resulting string is then normalized to ensure consistent spacing.

4. **Span Detection and Labeling**: The exact start and end character indices of the original source_text are programmatically located within the final concatenated string (full_text). This step is critical for creating the precise character-level annotations required for training.

This automated process was applied to the partitioned source texts, ultimately generating **93,099** examples for the training set and **40,626** for the validation set. The final output is a structured dataset where each entry contains the original text, the newly generated contextual sentence, and the precise citation boundaries. An example of the final generated data structure is shown in Table 6.

D.4 Stage 4: Dataset Partitioning

A systematic partitioning strategy was applied at the source text level to create distinct training and validation sets. The corpus of unique source texts was split using a 70-30 ratio, allocating 30,548 texts for training and 13,354 for internal validation. This split was performed using stratified sampling to preserve the original distribution of Ayah and Hadith texts in both partitions. A fixed random seed (42) was used to ensure the reproducibility of the splits.

D.5 Stage 5: Tokenization and Label Assignment

The final stage converts the generated examples into a format suitable for model training.

- **BIO Schema:** A five-class BIO (Beginning-Inside-Outside) tagging schema was employed: O (Outside), B-Ayah, I-Ayah, B-Hadith, and I-Hadith. This schema allows the model to learn the precise boundaries of each citation type.
- **Tokenization and Alignment:** Each example was tokenized using the AraBERTv2 tokenizer with a maximum sequence length of 512 tokens. The character-level span indices were mapped to their corresponding token indices. The first token of a span was assigned the 'B' label, subsequent tokens within the span received the 'I' label, and all other tokens were labeled 'O'. To handle subword tokenization, continuation tokens within a word were assigned an ignore index of -100, ensuring that the loss function only considers the primary token of each word.

D.5.1 Validation Framework

A multi-tiered validation approach was used for comprehensive performance assessment.

- **Internal Validation Set:** Created from the 30% partition of the original source texts. This set, containing approximately 40,626 synthetically generated examples, was used exclusively for hyperparameter optimization and model selection during development. To test generalization, the templates used to generate these examples differed from those used for the training set.
- **Official Development Set:** A set of 210 manually annotated examples provided by the task organizers. This set was used to evaluate the model's ability to generalize from synthetic data to authentic LLM-generated content.
- **Official Test Set:** A blind set of 190 examples used for the final competitive evaluation. Performance on this set determined the final reported scores.

D.5.2 Quality Control

Several measures were implemented to ensure the integrity of the generated dataset:

- **Failure Tracking:** Generation failures, such as span detection errors, were tracked, and only successfully generated examples were included in the final corpus.
- **Data Validation:** Routine checks were performed to verify character encodings, label consistency, and the integrity of tokenizer outputs.
- **Statistical Monitoring:** Statistics on the class distribution and the ratio of source texts to generated examples were monitored for transparency.

original_text	full_text	prefix	suffix	char_start	char_end	label_type	target_span	variation_number	dataset_split
وأمددناهم بفاكهة ولحم مما يشتهون	ومن آيات الله: وفي هذا هداية للمؤمنين. "وأمددناهم بفاكهة ولحم مما يشتهون" آية كريمة	ومن آيات الله:	آية كريمة	40	72	Ayah	وأمددناهم بفاكهة ولحم مما يشتهون	1	training
فويل يومئذ للمكذبين	ومن آيات الله: "فويل يومئذ للمكذبين" ولذلك عبرة للمعتبرين	ومن آيات الله:	ولذلك عبرة للمعتبرين	16	35	Ayah	فويل يومئذ للمكذبين	2	training

Table 6: Example of Final Generated Data Structure

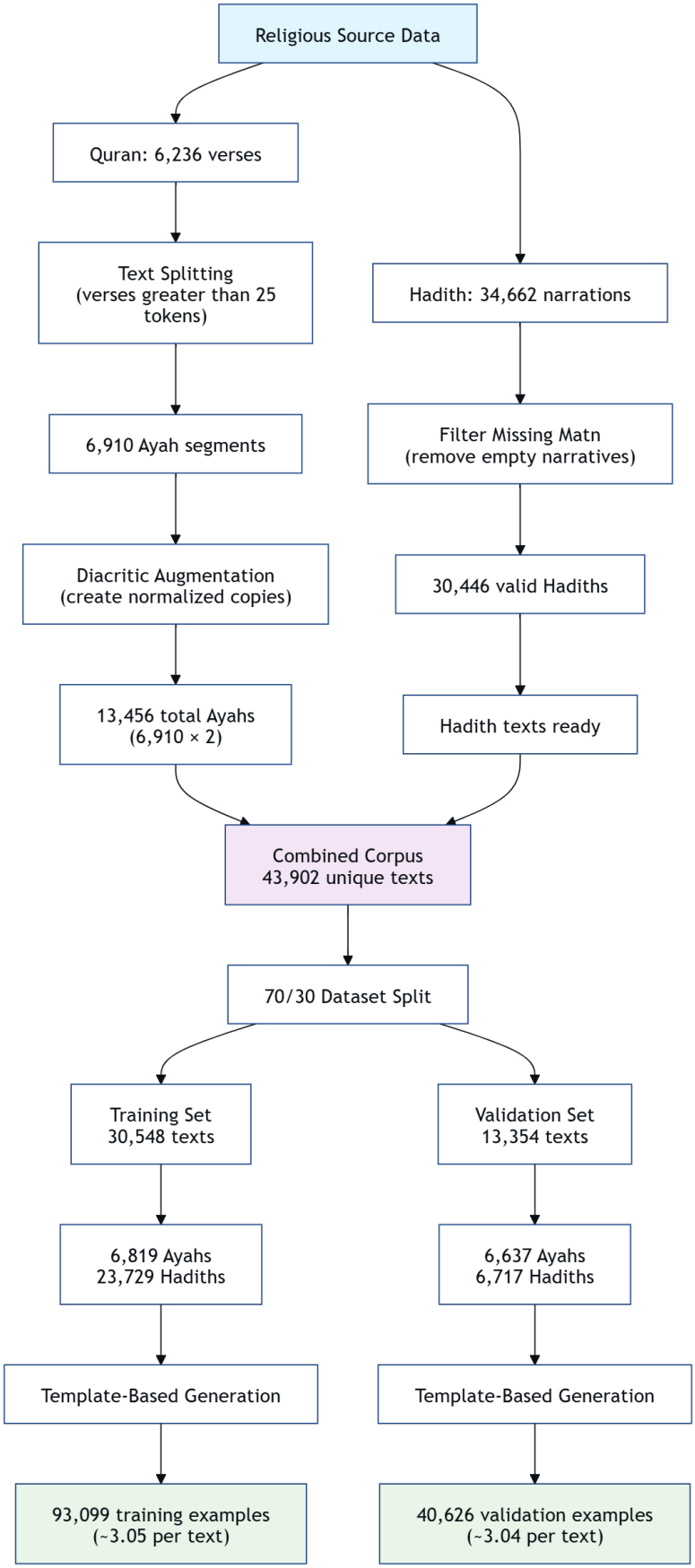


Figure 1: Data preprocessing pipeline transforming 40,898 raw Islamic texts into 133,725 training and validation examples through filtering, augmentation, and template-based generation.

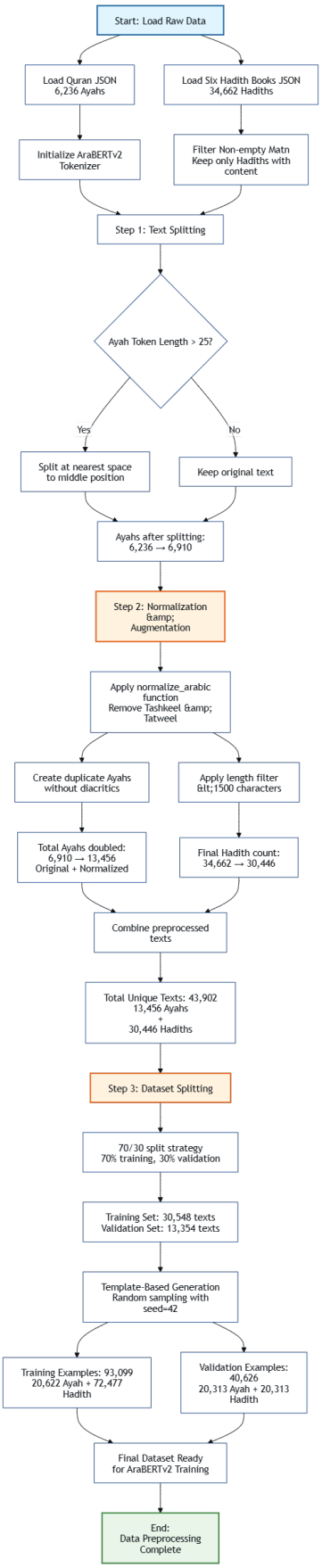


Figure 2: A high-level diagram of the data prepossessing pipeline

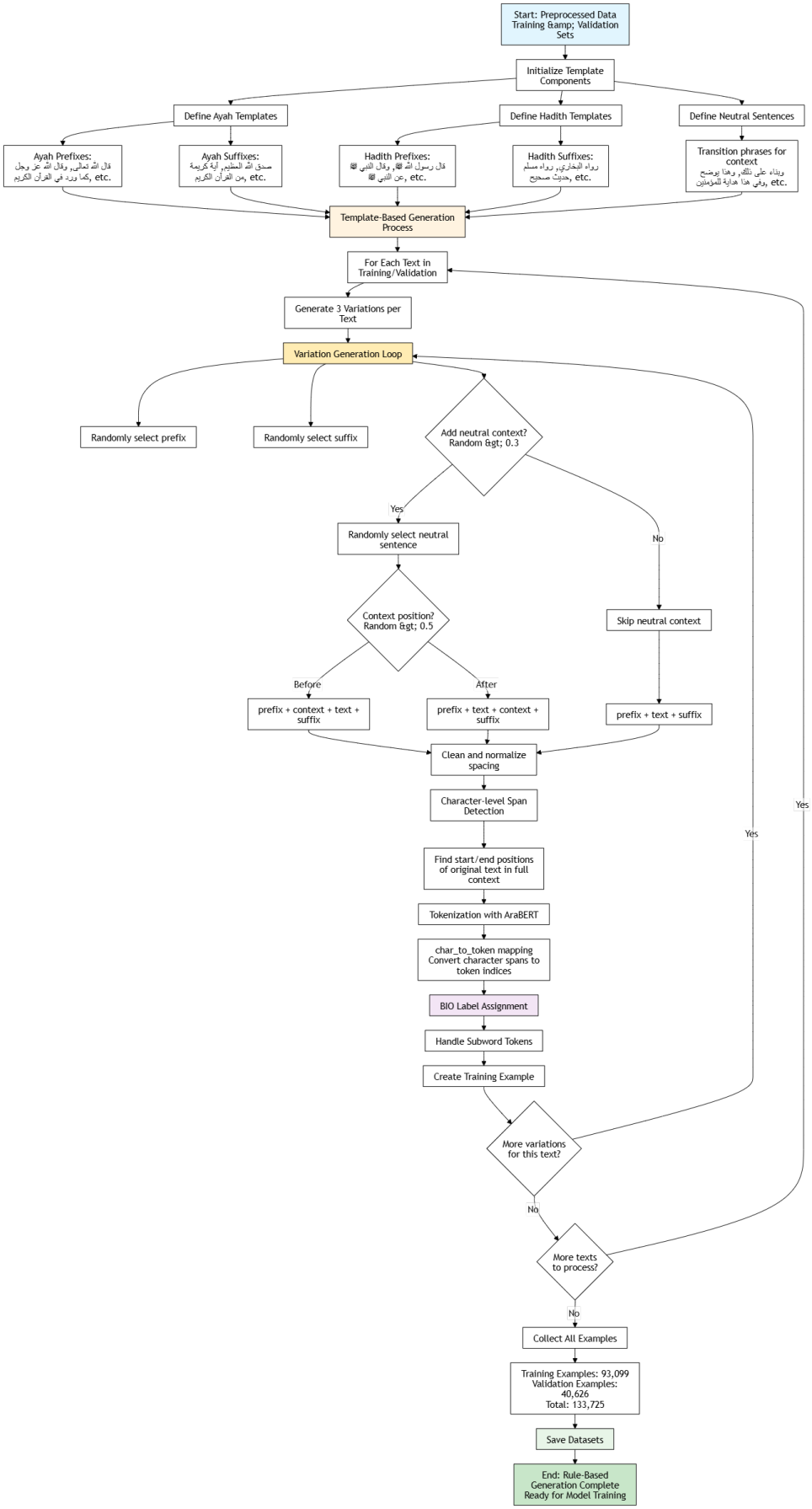
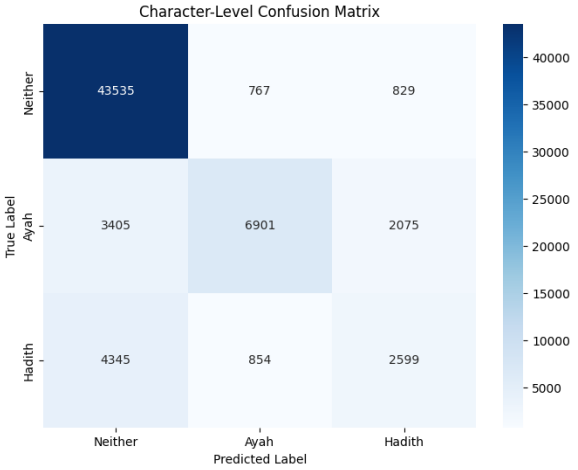
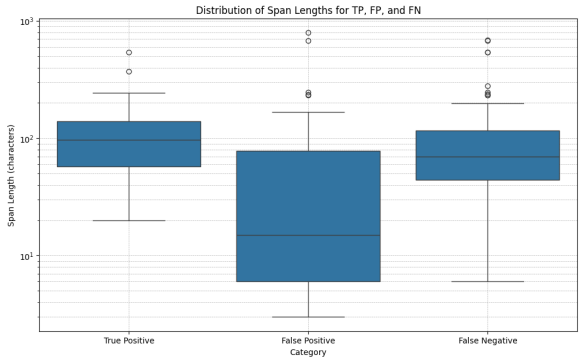


Figure 3: A high-level diagram of the rule-based data generation process.

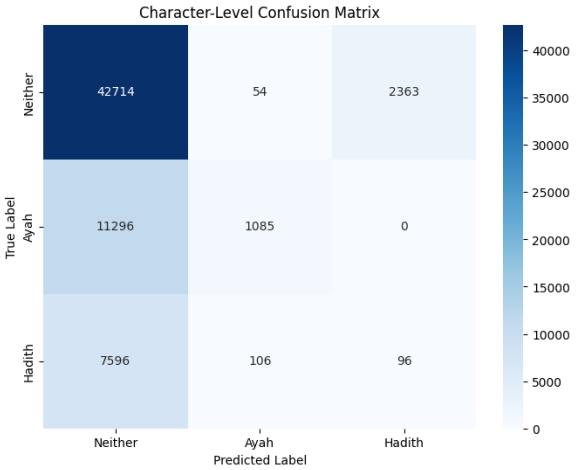


(a) Character-Level Confusion Matrix (Rule-Based Model)

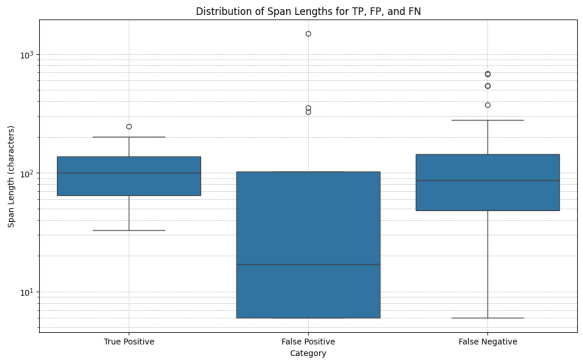


(b) Distribution of Span Lengths for TP, FP, and FN (Rule-Based Model)

Figure 4: Span-Level Error Logging for the Rule-Based Model.

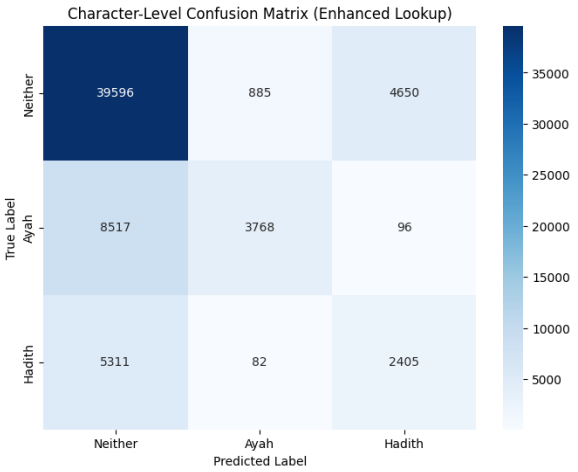


(a) Character-Level Confusion Matrix (Fine-tuned Model)

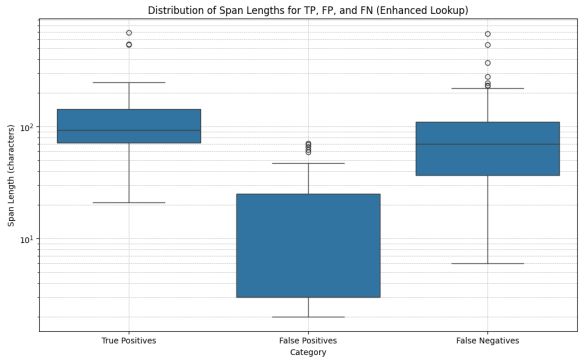


(b) Distribution of Span Lengths for TP, FP, and FN (Fine-Tuned Model)

Figure 5: Performance by Span Length for the Basic Fine-Tuning Model.



(a) Character-Level Confusion Matrix (Lookup Method)



(b) Distribution of Span Lengths for TP, FP, and FN (Lookup Method)

Figure 6: Performance by Span Length for the Lookup Method.

E Database Lookup Methodology

This appendix provides a detailed, step-by-step description of the database lookup method, which was implemented as a key baseline in the ablation study (see Table 2). This method relies on direct string matching against an enhanced knowledge base, serving as a non-neural benchmark to evaluate the performance of the fine-tuned model. The entire process can be broken down into two main stages: (1) Knowledge Base Enhancement and (2) The Span Detection Algorithm.

E.1 Stage 1: Knowledge Base Construction and Enhancement

The effectiveness of a lookup-based approach is highly dependent on the comprehensiveness of its knowledge base. To maximize the chances of finding a match, the raw source texts were significantly augmented through a multi-step enhancement pipeline.

E.1.1 Initial Data Loading

The process begins by loading the complete set of Quranic verses and Hadith narrations from the source JSON files provided by the task organizers (quran.json and six_hadith_books.json). The core textual content is extracted from the ayah_text field for Quranic verses and the Matn field for Hadiths. These texts form the initial, unprocessed knowledge base.

E.1.2 Arabic Script Normalization

To handle variations in Arabic script and vocalization, a normalization function was applied to every text in the knowledge base. This function removes all Arabic diacritics (*Tashkeel*) and the Tatweel character by targeting the Unicode range [\u064B-\u0652\u0640]. This step is crucial because LLM outputs may not include the same diacritics as the canonical source texts, and this normalization makes the matching process robust against such differences.

E.1.3 Text Segmentation for Partial Matching

LLMs often cite partial verses or fragmented Hadiths. To account for this, a text segmentation strategy was implemented. Any text (both original and normalized) is split into smaller, overlapping segments. The algorithm generates segments ranging from a minimum of 5 words to a maximum of 15 words, with a step size of 3 words. This process creates a large set of smaller text chunks. For example, a 20-word Hadith would be broken down into multiple 5-word, 6-word, ..., up to 15-word segments. This significantly increases the likelihood of detecting a partial citation.

E.1.4 Final Knowledge Base Aggregation

The final, enhanced knowledge base is an aggregation of multiple text variations for each original Ayah and Hadith. For each source text, the knowledge base contains:

1. The original, unaltered text.
2. The normalized (diacritic-free) version of the text.
3. All overlapping segments generated from the original text.
4. All overlapping segments generated from the normalized text.

This augmentation process results in a massive increase in the number of potential strings to search for, thereby improving the recall of the lookup method.

E.2 Stage 2: Span Detection Algorithm

With the enhanced knowledge base constructed, the span detection algorithm processes each LLM response to identify matching text.

E.2.1 Prioritization of Longer Matches

To ensure the quality of the matches, all entries in the enhanced Ayah and Hadith knowledge bases are sorted by string length in descending order. The detection algorithm iterates through these sorted lists, meaning it always attempts to match the longest possible text segments first. This is a critical step that prevents a short, partial match (e.g., a 5-word segment) from being identified if it is already part of a larger, more complete match (e.g., the full 30-word Ayah).

E.2.2 Iterative String Matching

For each LLM response, the algorithm iterates through every entry in the sorted knowledge bases (first Ayahs, then Hadiths). It uses a standard substring search to find all occurrences of a given knowledge base entry within the response text.

E.2.3 Overlap Prevention

To avoid redundant or overlapping annotations, the algorithm maintains a character-level boolean array for each response text, which tracks whether a character has already been assigned to a span. When a potential match is found, the algorithm checks this array to see if any character within the candidate span has already been classified. If there is no overlap, the span's start and end indices are recorded, and the corresponding characters in the tracking array are marked as classified. This ensures that once a sequence of text is identified as an Ayah, it cannot also be partially or wholly identified as another Ayah. If, after searching through the entire knowledge base, no spans are found for a given response, a "No_Spans" entry is recorded for that Question ID, as per the task requirements.

F Appendix: Development Set Error Analysis

This entire error analysis is conducted on the official development set provided by the shared task organizers, which consists of 210 manually annotated records.

F.1 Rule-Based Model Development Results

The rule-based model achieved a Macro F1 of **65%** on the development set. The detailed character-level report is shown in Table 7.

Class	Precision	Recall	F1-Score
Neither	0.85	0.96	0.90
Ayah	0.81	0.56	0.66
Hadith	0.47	0.33	0.39
Accuracy			**0.81**
Macro Avg	**0.71**	**0.62**	**0.65**
Weighted Avg	**0.80**	**0.81**	**0.80**

Table 7: Character-Level Classification Report for the Rule-Based Model.

F.1.1 Further Error Analysis

Table 8 provides descriptive statistics for the lengths of true positive, false positive, and false negative spans.

Category	Count	Mean	Min	Max
True Positives	78	108.59	20	541
False Positives	61	69.62	3	795
False Negatives	101	104.78	6	690

Table 8: Span Length Statistics (Rule-Based Model).

F.2 Basic Fine-Tuning Development Results

The basic fine-tuning model achieved a Macro F1 of **33%** on the development set. The detailed report is shown in Table 9.

Class	Precision	Recall	F1-Score
Neither	0.69	0.95	0.80
Ayah	0.87	0.09	0.16
Hadith	0.04	0.01	0.02
Accuracy			**0.67**
Macro Avg	**0.53**	**0.35**	**0.33**
Weighted Avg	**0.65**	**0.67**	**0.59**

Table 9: Development set classification report for the basic fine-tuning approach.

F.2.1 Further Error Analysis

Table 10 presents the descriptive statistics for span lengths.

Category	Count	Mean	Min	Max
True Positives	12	111.33	33	247
False Positives	13	184.46	6	1488
False Negatives	173	110.21	6	690

Table 10: Span Length Statistics (Basic Fine-Tuning).

F.3 Database Lookup Development Results

The database lookup approach achieved a Macro F1 of **52%** on the development set. The detailed classification report is shown in Table 11.

Class	Precision	Recall	F1-Score
Neither	0.74	0.88	0.80
Ayah	0.80	0.30	0.44
Hadith	0.34	0.31	0.32
Accuracy			**0.70**
Macro Avg	**0.62**	**0.50**	**0.52**
Weighted Avg	**0.70**	**0.70**	**0.68**

Table 11: Development set classification report for the database lookup approach.

The database lookup approach shows a significant class imbalance in its performance. While it achieves high recall for the "Neither" class (88%), its ability to identify religious texts is limited. For "Ayah" spans, the model has good precision (80%) but low recall (30%), indicating it is confident when it makes a prediction but misses many actual verses. The performance on "Hadith" spans is poor across all metrics (F1-score of 32%). This model's tendency to over-predict the 'Neither' class high-

lights the inherent difficulty of relying solely on exact-match lookups for this task.

F.3.1 Further Error Analysis

Table 12 provides descriptive statistics for span lengths of the lookup method.

Category	Count	Mean	Min	Max
True Positives	57	130.49	21	690
False Positives	467	12.19	2	71
False Negatives	109	93.11	6	677

Table 12: Span Length Statistics (Lookup Method).

G Generative Data Augmentation Ablation Study

As referenced in the discussion, an ablation study was conducted to evaluate the efficacy of using a generative Large Language Model for synthetic data augmentation. This approach, was compared against the primary rule-based methodology to determine its suitability for creating a training corpus. This appendix details the complete methodology, from data preprocessing to the final generation of contextualized examples.

Methodology

The generative approach utilized the aubmindlab/aragpt2-base model, a transformer-based model for Arabic language generation, accessed via the Hugging Face 'transformers' library. The core strategy was to embed authentic religious texts into open-ended prompt templates and have the model generate a plausible continuation, thereby creating a full, contextualized sentence around the original text.

1. Data Preprocessing

Before being used in prompts, the raw source texts underwent several preprocessing steps to increase data diversity and manage sequence length:

- **Text Loading**: The full set of Quranic verses (*Ayahs*) and Prophetic narrations (*Hadiths*) were loaded from their respective source JSON files.
- **Text Splitting**: Quranic verses exceeding a 25-token limit (as determined by the AraBERTv2 tokenizer) were split into two smaller segments. This was done to prevent truncation and ensure the model could process the entire text.
- **Normalization Augmentation**: To make the model robust to script variations, a duplicate version of each Ayah was created with all diacritics (*Tashkeel*) removed. The final pool of texts for generation included originals, split segments, and their normalized counterparts.

G.1 Template Examples for Data Generation

The core of the generative augmentation strategy involved embedding authentic religious texts within specific prompt templates to simulate natural-language citations. As shown in Table 14, these prompts were designed to frame the religious text as evidence or a quotation within a larger sentence. During the generation process, the text placeholder was dynamically replaced with a Quranic verse or Hadith, which was then used to prompt the AraGPT2 model to generate a contextual continuation.

As referenced in Section 8, an ablation study was conducted to evaluate the efficacy of using a generative Large Language Model for synthetic data augmentation. This approach was compared against the primary rule-based methodology to determine its suitability for creating a training corpus for the verification task. This appendix details the methodology, the prompt templates used, and the analysis of its significant limitations.

G.1.1 Generative Process

For each religious text, the following generative process was executed:

1. A prompt template was selected at random from the list above.
2. The religious text was inserted into the template.
3. The complete prompt was passed to the AraGPT2 text-generation pipeline with specific parameters:
    - **max_new_tokens=30**: To generate a short, contextual continuation rather than a long, potentially divergent paragraph.
    - **no_repeat_ngram_size=2**: To prevent the model from getting stuck in repetitive loops and improve the quality of the generated text.
4. The model's output, a new, longer string containing the original text, was captured as the 'full text'.

5. Finally, the character start and end indices of the original ‘span text‘ were located within the newly generated ‘full text‘ to create the final labeled data point. A fallback mechanism was included to use the prompt itself if the generation process failed.

Table 16 provides examples of the final structured data produced by this pipeline.

G.2 Limitations and Analysis of Generated Data

While the objective was to create diverse training examples, the generative methodology proved inappropriate for this verification task.The outputs were frequently plagued by factual inaccuracies, nonsensical statements, and linguistic artifacts, introducing significant noise into the training data.For a verification task in a sensitive domain like Islamic studies, the integrity of the source text and its context is paramount.The generative model’s tendency to ”hallucinate” or produce illogical continuations is a critical failure that undermines the purpose of the training data, as it creates misleading training signals. Table 15 provides representative examples of these failure modes.

Component	Training Examples	Validation Examples
Ayah Prefixes	قال الله تعالى: وقال الله عز وجل: كما ورد في القرآن الكريم: وفي كتاب الله: ومن آيات الله: يقول سبحانه وتعالى: وفي هذا الشأن يقول الله:	وفي القرآن الكريم نجد: ومن آيات الله: وقد أنزل الله: ويقول الحق تبارك وتعالى: وفي الذكر الحكيم: وفي كتاب الله نقرأ: والدليل على ذلك قوله تعالى:
Ayah Suffixes	صدق الله العظيم آية كريمة من القرآن الكريم كلام الله عز وجل من الذكر الحكيم ولذلك عبرة للمعتبرين وهذا بيان للناس	هذا من كلام الله آية عظيمة من القرآن الكريم كلام رب العالمين من الذكر الحكيم آية كريمة (صدق الله العظيم)
Hadith Prefixes	قال رسول الله صلى الله عليه وسلم: وقال النبي صلى الله عليه وسلم: عن النبي صلى الله عليه وسلم: روى أن النبي صلى الله عليه وسلم قال: وفي الحديث الشريف: وعن أبي هريرة رضي الله عنه قال:	وفي السنة النبوية: ومن هدي النبي صلى الله عليه وسلم: وقد علمنا الرسول صلى الله عليه وسلم: وفي الحديث الشريف نجد: كما جاء في الحديث:
Hadith Suffixes	رواه البخاري رواه مسلم حديث صحيح صلى الله عليه وسلم من السنة النبوية (متفق عليه) أو كما قال صلى الله عليه وسلم	من السنة النبوية حديث نبوي شريف من هدي المصطفى صلى الله عليه وسلم (رواه الترمذي)
Neutral & Transition Sentences	وبناء على ذلك، يمكننا أن نستنتج. وهذا يوضح عظمة التشريع. وفي هذا هداية للمؤمنين. إن في ذلك لآيات لقوم يعقلون. وهذا هو القول الراجح.	ولنتأمل معاً وفي هذا السياق وللتوضيح وإليكم المثال وفي هذا الصدد وهذا يبين لنا أهمية الموضوع.

Table 13: Template for rule-based data generation across training and validation sets.

Component	Arabic Examples	English Translation
Prompt Templates	يُستشهد بالنص التالي: ،{text} وهذا يدل على أن	The following text is cited: {text}, and this indicates that
	من الأدلة الشرعية على هذه المسألة، النص التالي: ،{text} ويُفهم من ذلك أن	Among the legal evidence for this issue is the following text: {text}, and it is understood from this that
	يتناول النص التالي قضية مهمة، وهي: ،{text} حيث أن	The following text addresses an important issue, which is: {text}, as
	يمكن الاستفادة من قوله: ،{text} في واقعنا المعاصر عن طريق	We can benefit from the saying: }text{, in the contemporary reality by

Table 14: Prompt templates used for the generative data augmentation experiment. The {text} placeholder was replaced with an authentic religious text.

Original Text (Ayah)	Full Generated Output	Analysis of Failure
وَلَتَنَازَعْتُمْ فِي الْأَمْرِ وَلَكِنَّ اللَّهَ سَلَّمَ ۗ إِنَّهُ عَلِيمٌ بِذَاتِ الصُّدُورِ	يمكن الاستفادة من قوله: ،{text} في واقعنا المعاصر عن ا صول تائب **التائب طريق المذبون ، جازمونا**	**Nonsensical Artifact**: The generated continuation (in bold) is grammatically incorrect and semantically meaningless. It represents the "illogical artifacts" mentioned in the discussion, creating a confusing and useless training example.
ليأكلوا من ثمره وما	يُستشهد بالنص التالي: ،{text} **هذا النوع وهذا يدل على أن من الفاكهة له فوائد صحية كثيرة ، حيث أنه يحتوي على نسبة عالية من الألياف الغذائية التي تساعد في الوقاية من العديد من الأمراض مثل السرطان وأمراض القلب**	**Contextual Hallumination**: The model correctly identifies the text fragment as relating to food but proceeds to hallucinate a detailed, modern nutritional context (fiber, cancer, heart disease) that is not present in the original Quranic narrative. This creates a factually incorrect and misleading association.

Table 15: Examples of incorrect and nonsensical data produced by the AraGPT2-based generative augmentation method.

Full Generated Text (full_text)	Original Span (span_text)	Start (char_start)	End (char_end)	Label (label_type)
يُستشهد بالنص التالي: اركض برجلك ﷺ ، وهذا يدل على أن الله سبحانه وتعالى هو الذي يراك ، وهو الذي يعلمك ما لا تعلم من أمور الدنيا والآخرة. وإذا تأملنا في القرآن الكريم نجد أنه	اركض برجلك ﷺ	22	34	Ayah
من الأدلة الشرعية على هذه المسألة، النص التالي: اتَّقُوا اللَّهَ وَقُولُوا قَوْلًا سَدِيدًا، ويُفهم من ذلك أن يذكروا ما هو واجب عليهم ، وما هو مستحب لهم ، وأن يستخدموه في طاعة الله تعالى ورسوله صلى الله عليه وسلم.	اتَّقُوا اللَّهَ وَقُولُوا قَوْلًا سَدِيدًا	48	91	Ayah
يتناول النص التالي قضية مهمة، وهي: أَلَّا تَعْبُدُوا إِلَّا اللَّهَ ﷺ ، حيث أن الله تعالى قد أمر عباده المؤمنين بأن لا يعبدوا إلا إياه ، وأن لا يتخذوا من دونه أولياء ولا شفعاء ، وأن	أَلَّا تَعْبُدُوا إِلَّا اللَّهَ ﷺ	34	59	Ayah

Table 16: Examples of the final structured output from the generative data augmentation pipeline using AraGPT2. This table illustrates the format of the generated data, including the full generated text and the identified character spans of the original religious text embedded within it.