Linguify: Self-Reflective Retrieval-Augmented Generation and Chunk-Aware Streaming Modular TTS for Interruptable, Low-Latency Chatbot Conversational AI

*Total word count of this conference paper, excluding figures, tables, pseudocodes, references, and appendices, is 3156 words.

Joseph Emmanuel Chay Huan Qin
Department of Computer Science
University of Nottingham
hcyjc11@nottingham.edu.my

Leong Tin Jet
Department of Computer Science
University of Nottingham
hcytl1@nottingham.edu.my

***Abstract*—Linguify addresses critical latency and streaming limitations in current Text-to-Speech (TTS) systems for applications like call center AI. We introduce a unified framework enabling high-quality, low-latency streaming and non-streaming synthesis. Utilizing our novel Self-Reflective Retrieval-Augmented Generation (SR-RAG) architecture and novel Chunk-Aware Streaming Modular (CHASM)-TTS system, Linguify achieves modular, adaptable, and state-of-the-art performance, delivering natural-sounding, interruptible phone conversations without sacrificing quality or responsiveness in real-time conversational AI.**

***Index Terms*—Conversational AI, Call Center Automation, Chatbot Agent, Multi-Agentic AI, Text-to-Speech, Streaming Synthesis, Retrieval-Augmented Generation**

I. Introduction

A. Background and Motivation

The integration of AI in customer service is rapidly growing, with the conversational AI market for finance and medical sectors projected to reach $18.4 billion by 2026 [1]. Despite advances, a significant gap exists between controlled environments and real-world deployment [2].

In knowledge-intensive domains like finance and healthcare, traditional language models struggle with precise knowledge manipulation. RAG models address this by combining parametric and non-parametric approaches, enhancing transparency and adaptability [3].

Neural speech synthesis has advanced to zero-shot systems [4], with current approaches spanning codec language models [5], feature diffusion [6], and hybrid systems [7], where latency critically impacts user experience.

B. Limitations of State-of-the-Arts

1) Language Models: Current parametric language models store knowledge implicitly in their parameters but struggle with precise knowledge manipulation and provenance [8]. This causes factual inconsistency and weak sentimental comprehension for real-world applications. While knowledge-intensive NLP tasks have seen improvements through task-specific retrieval integration [9], [10], existing systems fail to effectively balance parametric and non-parametric components for dynamic user contexts.

2) Speech Generation: Existing TTS models predominantly operate in non-streaming (offline) mode, requiring complete text input before synthesis begins [11]. This causes high latency in interactive applications. While streaming has been explored for language model-based TTS [12], diffusion-based and hybrid systems lack effective streaming solutions.

C. Research Objectives

In this paper, we develop an intelligent conversational chatbot system for finance (taxation inclusive) and medical domains capable of low-latency, interruptible interactions, and sentiment analysis, by systematically researching and resolving these two research questions:

1) ***How can modular and dynamic architectures improve the adaptability and factual consistency of RAG systems in evolving domains?***
2) ***How can streaming speech synthesis achieve quality parity with offline systems while maintaining minimal latency?***

D. Contributions

Linguify makes four significant contributions:

1) Unified streaming/non-streaming synthesis framework enabling interruptible conversations adaptive to user sentiment
2) Finite scalar quantization replacing vector quantization, achieving full codebook utilization for improved emotional expression
3) Instruction-tuning capabilities supporting emotion, accent and style adaptation based on user communication needs
4) Query Routing: Dynamic routing between retrieval and generation to prevent irrelevant responses
5) Adaptive Retrieval Refinement: Query transformation when initial retrieval yields insufficient context

II. Related Works

A. Task-Specific Retrieval Integration

Numerous studies have demonstrated that retrieval components can enhance performance across individual NLP tasks that are critical for call center automation. For open-domain question answering, Chen et al. [9] and Kwiatkowski et al. [10] leveraged retrieval to access external knowledge repositories-a capability essential for customer service agents needing to provide accurate information. Similar retrieval-based approaches have been successfully applied to fact checking [13], fact completion [8], and long-form question answering [14], all of which have direct applications in verifying customer claims and providing comprehensive responses.

Retrieval has also proven valuable for dialogue systems [15]–[18], which form the backbone of conversational call center AI. Supplementary, grounding conversations in retrieved knowledge [16], [17] demonstrates significantly improves response quality-a critical factor in customer satisfaction. In machine translation, Gu et al. [19] used retrieval mechanisms to incorporate external knowledge and improve translation quality, which benefits multilingual call centers. Language modeling has also benefited from retrieval augmentation [20], [21], enabling more contextually appropriate responses to customer inquiries.

B. Adaptable NLP and RAG Architectures

General-purpose architectures for NLP tasks have demonstrated impressive results without using retrieval components, establishing baseline capabilities for modern call center AI. Fine-tuned pre-trained language models have achieved strong performance on various classification tasks following approaches pioneered in [22], enabling better intent recognition and query categorization in customer service contexts. GPT-2 [23] showed that a single left-to-right pre-trained language model could perform well on both discriminative and generative tasks, providing a foundation for systems that both understand and generate customer service responses.

Building on this foundation, encoder-decoder models like BART [24] and T5 [25], [26] leveraged bidirectional attention to further improve performance across diverse tasks. However, it was demonstrated in [26] that these models lack mechanisms to access, update, or interpret knowledge thoroughly, a limitation in call center environments where information frequently changes.

C. Memory-Augmented Architectures

Memory networks [27], [28] represent an early approach to augmenting neural networks with external memory, conceptually similar to how call center AI might access a knowledge base during customer interactions. Recent work by Févry et al. [29] investigated retrieving trained entity embeddings rather than raw text. Dialogue models have been enhanced to generate more factual content by attending over fact embeddings [18], [30] or by conditioning generators on retrieved text [17], which helps ensure accurate information delivery in customer service scenarios.

Incorporating factual knowledge has shown to improve response generation-crucial for call centers [18], [30] where providing incorrect information can damage customer trust. A key advantage of raw text memory is that it remains human-readable and human-writable, enabling call center managers to update knowledge bases without requiring model retraining.

D. Variational Autoencoders

Neural text-to-speech has extensively explored speaker representation mechanisms. VALL-E [4] and GPT-SoVITs [31] rely on complex conditioning approaches that integrate speaker information directly into their language modeling components. While this integration proves effective for preserving speaker identity, research indicates potential drawbacks. Utterance-level vectors have been researched [7], [32] to frequently contain not only speaker characteristics but also language and paralanguage information, which can compromise prosody naturalness and cross-lingual capability.

E. Sentiment Analysis with Text-Tokenization

Text encoding strategies represent another significant area of investigation in the literature. Models such as FireRedTTS [32] and MaskGCT [33] implement dedicated text encoders to extract linguistic features. This approach contrasts with more recent explorations that leverage large language models as universal text processors. Moreover, LLaMA 3.2B [34] has shown sufficient capability to align text and speech tokens without additional encoding layers, enabling more streamlined architectures while maintaining high-quality synthesis.

F. Zero-Shot Approaches

Contemporary zero-shot TTS approaches fall into three main categories:

1) Codec Language Models: These models utilize speech codec models to extract discrete speech representations [5], [35], [36] and employ autoregressive [4], [37]–[42] or masked [33] language models to predict speech tokens. These tokens are then synthesized to waveforms via codec vocoders [43], [44]. Recent work has also explored continuous speech representations [45]. Language model-based TTS can generate varied and prosody-consistent speech through autoregressive sampling.

2) Feature Diffusion Models: Inspired by advances in image generation, denoising diffusion [6], [46] and flow matching models [47] have been introduced into non-autoregressive (NAR) speech synthesis. Early diffusion-based TTS models required duration prediction for each text unit to address length disparities between text and speech features [7], [48]–[50]. To improve natural prosody, cross-attention and Diffusion Transformers have been incorporated into NAR TTS models [51], [52]. Recent research has developed simpler approaches for text-speech alignment in NAR TTS models, including E² TTS [53], F5-TTS [54], and Seed-TTS [55].

G. Streaming Speech Synthesis

Despite advances in quality, most zero-shot TTS models operate in non-streaming (offline) mode, requiring complete input text and synthesizing the entire utterance before returning audio. This approach results in high latency that negatively impacts user experience in interactive applications like voice chat [11], [56].

Streaming synthesis has been explored for language model-based zero-shot TTS models [12], [57]–[59], but diffusion-based TTS models and hybrid systems lack well-established streaming solutions. This limitation creates a significant gap in the field, as real-time, high-quality speech synthesis becomes increasingly important for conversational AI applications.

III. Methodology

A. *Self-Reflective RAG Architecture*

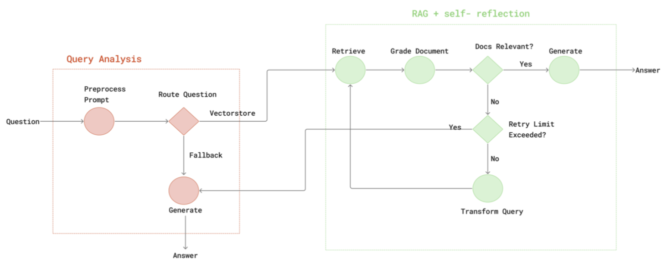

Fig. 1. Self-Reflective (SR)-RAG Pipeline Architecture of Linguify. (a) Query Analysis phase implements preliminary processing and routing logic, where incoming questions undergo prompt preprocessing followed by conditional routing to either retrieve from the vectorstore or employ direct generation fallback. (b) RAG with self-reflection component illustrates the iterative retrieval-evaluation-generation pipeline, featuring document relevance assessment and query transformation mechanisms when initial retrieval yields insufficient context. The bidirectional arrows represent information flow between components, with dashed outlines indicating decision points that determine subsequent processing paths during inference.

We proposed our novel Self-Reflective (SR)-RAG architecture in Figure 1 with multiple enhancements to the standard Retrieval-Augmented Generation architecture [24]. Let q represent the user query, D the document corpus, and $R(q, D)$ the retrieval function that returns the top-k documents $\{d_1, d_2, ..., d_k\}$. Our SR-RAG framework optimizes the generation process through:

1. **Multi-stage relevance evaluation**: We implement a document relevance function $G_r(d_i, q) \rightarrow [0, 1]$ that scores each retrieved document's relevance to the query, filtering out irrelevant content when $G_r(d_i, q) < \tau_r$ where τ_r is a relevance threshold [8].

2. **Adaptive retry strategy**: When insufficient relevant documents are retrieved, we apply query transformation $T(q) \rightarrow q'$, addressing user frustration by enabling emotionally charged interactions, and perform subsequent retrieval $R(q', D)$ rather than defaulting to non-augmented generation [60].

Given computational constraints, we implemented SR-RAG using LLaMA 3.2B, though larger models would theoretically yield superior performance for knowledge-intensive tasks.

B. *Text-to-Speech Architecture*

Our TTS architecture separates semantic and acoustic information in speech signals, defining generation as a progressive semantic decoding procedure with gradual incorporation of conditional information. Figure 2 shows the complete architecture trained from scratch.

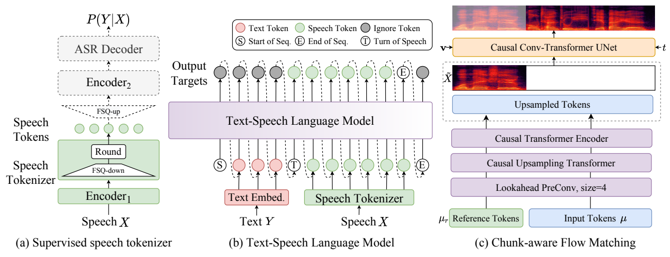

Fig. 2. CHunk-Aware Streaming Modular (CHASM)-TTS Architecture Pipeline. (a) supervised acoustic tokenizer, with dashed components active only during training; (b) unified model handling both text and speech inputs in streaming and non-streaming modes, with dashed arrows indicating autoregressive decoding during inference; (c) causal flow matching framework conditioned on speaker representation $\mathbf{v}$, semantic tokens μ, masked acoustic features $\tilde{X}$, and intermediate latent variable X_t.

1) *Text Autoencoder Tokenizer:* Linguify processes raw text through a BPE-based tokenizer, eliminating the need for phoneme-based frontend models and simplifying preprocessing. The model learns contextual pronunciation end-to-end, with one-to-many tokens masked to prevent excessive pronunciations and reduce data sparsity issues.

2) *Supervised Semantic Speech Tokenizer:* As shown in Figure 2(a), the system incorporates finite scalar quantization (FSQ) into a SenseVoice-Large ASR encoder. Input speech X passes through Encoder$_1$ (six Transformer blocks with rotary positional embedding) to obtain intermediate representations, which are quantized by the FSQ module before passing through Encoder$_2$ and the ASR Decoder.

The FSQ module projects intermediate representations H into a D-dimensional low-rank space, quantizes values into $[-K, K]$ range using bounded round operation, then projects back to the original dimension:

$$\bar{H} = \text{ROUND}(\text{Proj}_{\text{down}}(H)) \tag{1}$$

$$\hat{H} = \text{Proj}_{\text{up}}(\bar{H}) \tag{2}$$

Speech token μ_i is obtained by calculating the index in a $(2K + 1)$-ary system:

$$\mu_i = \sum_{j=0}^{D-1} \bar{h}_{i,j} \cdot (2K + 1)^j \tag{3}$$

This tokenizer operates at 25 Hz, producing 25 speech tokens per second.

3) *Unified Text-Speech Language Model:*

C. *Chunk-aware Flow Matching and Interruption*

To enable streaming synthesis, which allows real-time, interruptable speech generation as text is being received, four specialized attention mask types are strategically implemented in the model architecture (Figure 3). They control the information flow within the attention mechanisms of the Transformer-based components, creating different trade-offs between interruptable

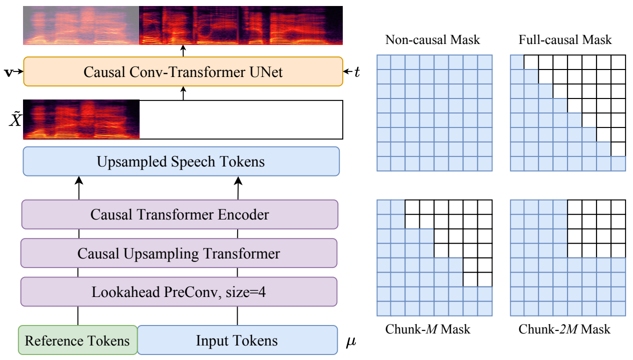

Fig. 3. Chunk-Aware Flow Matching Model

and synthesis quality. The mask selection directly affects how much future context each frame can access during generation:

- **Non-causal Mask**: Attends to all frames for offline mode
- **Full-causal Mask**: Attends only to past frames for lowest latency
- **Chunk-M Mask**: Attends to past and M future frames
- **Chunk-2M Mask**: Extends context for better quality at higher latency

D. Low Latency Capabilities

First-package latency critically impacts user experience in LLM-based voice chat applications [56]. For TTS, this latency stems from speech token generation, Mel spectrogram reconstruction, and waveform synthesis, expressed as:

$$L_{TTS} = M \cdot d_{lm} + M \cdot d_{fm} + M \cdot d_{voc} \tag{4}$$

where d_{lm}, d_{fm}, and d_{voc} represent computation times for language model token generation, Flow Matching spectrogram generation, and vocoder waveform synthesis, respectively. In LLM-based voice chat, the total first-package latency becomes:

$$L_{Chat} \leq N \cdot d_{llm} + L_{TTS} \tag{5}$$

with d_{llm} representing the LLM's computation time per token. Since LLM text tokens typically encode longer raw text than the presented model's tokens, L_{Chat} must be lower than this sum.

IV. Experiment Setups

A. Data Collection and Content Relevancy

For training dataset, we specifically selected finance and medical datasets from Hugging Face based on their relevance to our application domains. The finance datasets included instruction-tuned models, financial news, and community discussions, providing both factual content and conversational patterns. Similarly, our medical dataset collection balanced clinical knowledge and conversations.

In extension to the finance dataset, we included comprehensive taxation knowledge from Malaysia's tax authority (LHDN)[1] to provide accurate and adaptable responses. Due to the absence of official APIs, we implemented a web crawler to systematically extract taxation information, converting unstructured web content into machine-readable formats for our knowledge base.

B. Data Preprocessing

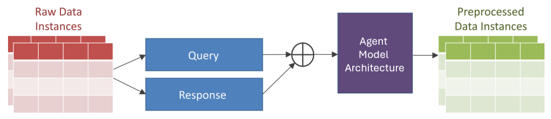

Fig. 4. Data Preprocessing with Instruction-tuning Process Flow

Our automated data preprocessing pipeline in Figure 4, raw data instances are first decomposed into discrete "Question" and "Answer" components, enabling a structured transformation of unorganized inputs into agent-ready formats. This facilitates targeted conditioning of the model architecture by separately modeling input prompts and expected outputs, a strategy shown to enhance alignment and downstream generalization [61]. The processed "Question" and "Answer" pairs are then fed into our RAG model architecture, where an integration mechanism synthesizes them into a representation optimized for task-specific objectives.

C. Data Cleaning and Segregation

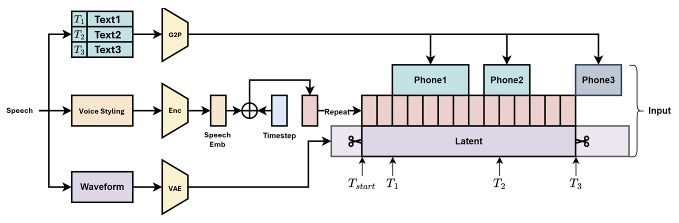

Fig. 5. Multi-Stream Data Cleaning and Segregation Pipeline

Figure 5 illustrates our automated pipeline for data cleaning and segregation for speech data, through conditioning signals at specific timesteps (T_{start}, T_1, T_2, T_3), synchronized with the phonetic sequence:

Text Input Stream Multiple text inputs (T_1, T_2, T_3) undergo G2P conversion to phonetic representations (Phone1, Phone2, Phone3), aligning with neural TTS methodologies [4], [37] to ensure accurate pronunciation mapping.

Voice Styling Stream Reference speech is encoded through Enc to extract embeddings, which combine with timestep information via addition for temporal voice representation-extending speaker conditioning approaches [7], [62] with explicit temporal conditioning.

Waveform Stream Raw audio is compressed through a VAE into latent representations, sharing conceptual foundations with neural codec models [5], [35] while optimizing for the multi-input architecture.

[1]Official website: https://www.hasil.gov.my/en/

D. *Hyperparameter Tuning*

Hyperparameter tuning was conducted to optimize model performance and generalization, following established best practices in the literature. The Adam optimizer was selected for its adaptive learning rate and computational efficiency [63], with an initial learning rate set to 1×10^{-4}. A batch size of 32 was chosen to balance convergence speed and stability. Early stopping [64] and dropout regularization (rate: 0.2) were employed to prevent overfitting. The number of epochs and weight decay (1×10^{-5}) were determined based on validation loss trends. Grid search, as recommended for hyperparameter optimization [65], was utilized to identify optimal parameter configurations.

V. Results and Discussions

A. *Training Accuracy Analysis and Reasoning*

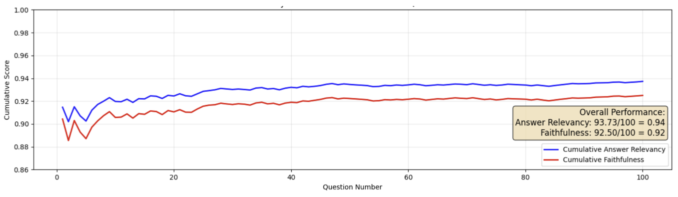

Fig. 6. Cumulative Performance Metrics for RAG System Training (n=100)

1) RAG Relevancy and Faithfulness Performance: Our optimized RAG architecture, using Algorithm 1 demonstrated exceptional performance across 100 diverse queries, achieving an answer relevancy score of 0.94 and faithfulness score of 0.92. As shown in Figure 6, after minimal initial volatility, both metrics quickly stabilized, with relevancy consistently outperforming faithfulness by a small margin (1-2%). The high faithfulness score in a domain where factual accuracy is critical for customer trust and compliance [30] demonstrates our architecture's effectiveness for knowledge-intensive call center chatbot service applications.

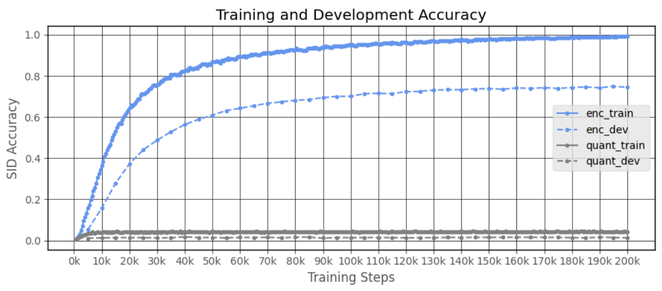

Fig. 7. Convergence curves of SID training using tokens before and after quantization.

2) Text-to-Speech SID Performance: Training was conducted using Algorithm 2 with representations of Speaker Identification (SID)[2] task both before and after quantization,

[2]SID evaluates recognition capabilities, essential for zero-shot synthesis.

Algorithm 1 SR-RAG Optimization Algorithm

```
Require: User question $q$
Ensure: Generated answer $a$
  Query Analysis Phase:
  $p \leftarrow$ PreprocessPrompt($q$)
  if RouteQuestion($p$) = "Vectorstore" then
      RAG + Self-reflection Phase:
      $documents \leftarrow$ Retrieve($p$)
      $retry\_count \leftarrow 0$
      $max\_retries \leftarrow$ predefined limit
      while true do
          for each $doc \in documents$ do
              $grade \leftarrow$ GradeDocument($doc, p$)
              if IsRelevant($grade$) then
                  $a \leftarrow$ Generate($p, doc$)
                  return $a$
              end if
          end for
          if $retry\_count \geq max\_retries$ then
              $a \leftarrow$ Generate($p$)            ▷ Fallback to direct generation
              return $a$
          else
              $p' \leftarrow$ TransformQuery($p$)
              $documents \leftarrow$ Retrieve($p'$)
              $retry\_count \leftarrow retry\_count + 1$
          end if
      end while
  else
      $a \leftarrow$ Generate($p$)            ▷ Direct Generation Fallback
      return $a$
  end if
```

which is essential for zero-shot synthesis in Linguify where voice characteristics should be independently controlled from linguistic content itself. following methodologies similar to those in speaker verification research [66]. Figure 7 displays our accuracy curves results, clearly showing that the SID layer with quantized tokens fails to converge (positive outcome). This inability to learn speaker identification from the tokenized representations provides strong evidence of the tokenizer's success in decoupling speaker information from speech content, essential for zero-shot synthesis systems where voice characteristics should be controlled independently from speech content, as highlighted in recent works [7], [62].

VI. Evaluations and Testings

A. *Performance of SR-RAG Architecture*

The domain-specific analysis in Figure 8 demonstrates the adaptability and versatility of Linguify across different conditions and decision-makings. Overall, medical domains achieved higher relevancy scores (0.94 average) compared to finance (0.89), while finance domains demonstrated superior faithfulness (0.92 versus 0.83 for medical). This pattern may

Algorithm 2 CHASM-TTS Training Algorithm

```
procedure TRAIN(model, opt, sched, train_data, cv_data)
    Initialize $step \leftarrow 0$, $epoch \leftarrow 0$
    for $epoch = 0$ to $max\_epochs$ do
        Set model to training mode
        for each $batch$ in $train\_data$ do
            if join condition met then
                break
            end if
            if using DDP and not gradient sync step then
                $context \leftarrow model.no\_sync()$
            else
                $context \leftarrow nullcontext()$
            end if
            with $context$:
              $loss \leftarrow$ forward_pass(model, batch)
              backward_pass(loss)
            if gradient accumulation complete then
                update_parameters(model, optimizer)
                update_learning_rate(scheduler)
                $step \leftarrow step + 1$
            end if
            if save interval reached then
                evaluate(model, cv_data)
                save_checkpoint(model, step)
            end if
        end for
        evaluate(model, cv_data)
        save_checkpoint(model, epoch)
    end for
end procedure
```

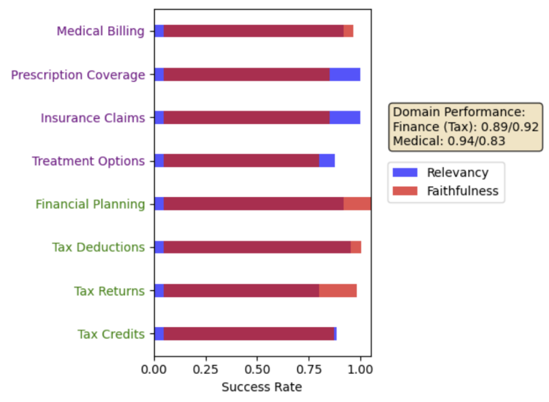

Fig. 8. Domain-specific performance analysis across finance and medical categories. The horizontal bars represent success rates for relevancy (blue) and faithfulness (red) across eight specialized topics.

reflect differences in the nature of knowledge between domains - medical information often requires more interpretive understanding, while financial information tends to be more precise and rule-based [8].

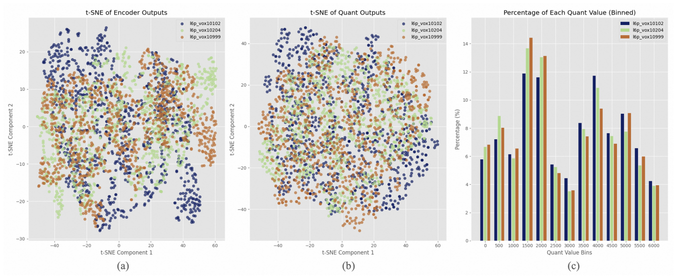

Fig. 9. The t-SNE visualization illustrates the speech representations before (a) and after (b) quantization for three different speakers from the dataset. (c) presents the codebook utilization, displaying the token percentage across speakers, with each bin containing 500 tokens.

Domain	Accuracy (%) ↑	Precision (%) ↑	Recall (%) ↑	F1-Score (%) ↑	Domain Coverage (%) ↑	Hallucination Rate (%) ↓
Finance (Tax)	93.5	91.2	94.8	93.0	96.7	3.2
Medical	91.8	89.5	92.3	90.9	94.5	4.1
General	82.4	78.6	80.9	79.7	85.2	8.7

TABLE I
ADAPTABILITY PERFORMANCE ACROSS SPECIALIZED AND GENERAL DOMAINS.

B. Effectiveness of CHASM-TTS Speech Tokenizer

Analysis using t-Distributed Stochastic Neighbor Embedding (t-SNE)[3] visualization revealed important characteristics of the FSQ tokenizer. Examining 100 speech samples from each of three speakers in the dataset [67] showed that before quantization, the Encoder_1 outputs exhibit distinguishable distributions between different speakers (Figure 9(a)). In contrast, after quantization, these distributions become nearly indistinguishable (Figure 9(b)), demonstrating effective removal of speaker-specific information. Additionally, Figure 9(c) confirmed the full utilization of the codebook across different speakers, a significant improvement over codebook collapse issues observed in earlier VQ-based approaches [4], [37].

C. Modularity and Adaptability

1) Statistical Performance Metrics: To evaluate real-time adaptability, Linguify was tested across specialized and general domains, with results presented in Table I. The evaluation used the metrics: accuracy, precision, recall, F1-score, domain coverage, and hallucination rate.

Linguify demonstrated superior performance in specialized domains compared to general content. Finance (Tax) domain achieved the highest accuracy (93.5%) and F1-score (93.0%), with a low hallucination rate of 3.2%. Medical domain followed closely with 91.8% accuracy and 90.9% F1-score. The general domain showed notably lower performance across all metrics, with 82.4% accuracy and a higher hallucination rate of 8.7%.

[3]Visualizes high-dimensional speech embeddings in 2D space, to identify characteristics for voice synthesis

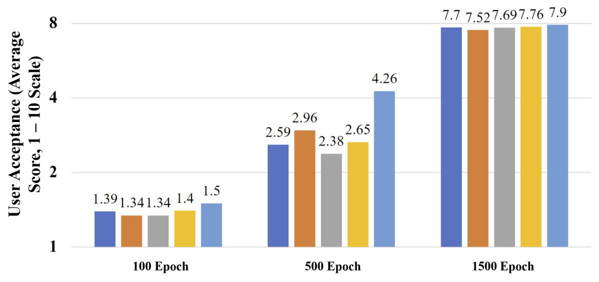

Fig. 10. User Acceptance Performance Score over 10-point likert scale

Model	*test-en*		*test-hard*	
	WER (%) ↓	**SS ↑**	**WER (%) ↓**	**SS ↑**
Human	2.14	0.734 (0.742)	-	-
Vocoder Resyn.	2.17	0.700	-	-
Seed-TTS [55]	2.25	0.762	7.59	0.776
FireRedTTS [32]	3.82	0.460 (0.526)	17.45	0.621 (0.639)
MaskGCT [33]	2.62	0.714 (0.730)	10.27	0.748 (0.720)
E2 TTS (32 NFE) [53]	2.19	0.710	-	-
F5-TTS (32 NFE) [54]	1.83	0.647 (0.742)	8.67	0.713 (0.762)
CHASM-TTS (Linguify)	2.57	0.652 (0.736)	6.83	0.724 (0.776)

TABLE II
Ablation Studies on CHASM-TTS (Linguify) and SOTA models.

D. User Acceptance Testing

User Acceptance Testing (UAT) was conducted on Linguify with 12 participants, Participants evaluated the integrated RAG system across three developmental stages (100, 500, and 1500 epochs), using a 10-point likert-scale. The evaluation assessed five key metrics (shown as different colored bars in Figure 10): chatbot responsiveness, information accuracy, naturalness of interactions, query comprehension, and overall satisfaction.

UAT scores show substantial improvement across all evaluation metrics throughout training, progressing from inadequate ratings at 100 epochs (1.34-1.50) to significantly higher scores at 500 epochs (2.38-4.26), and ultimately achieving excellent ratings at 1500 epochs (7.52-7.90). Response accuracy consistently received higher scores across all training stages, reinforcing [8]'s findings on the primacy of factual reliability in knowledge-intensive systems. The non-linear quality improvement curve-with a dramatically larger average increase between 500 and 1500 epochs (5.05 points) compared to between 100 and 500 epochs (1.41 points)-aligns with [68]'s observations that performance gains in retrieval-augmented systems accelerate after reaching certain developmental thresholds.

The UAT results demonstrated Linguify's superior ability to maintain conversational coherence across multiple interactions and adapt response styles based on detected user sentiment, with participants specifically noting improved experiences during scenarios simulating frustration or technical confusion.

E. Ablation Studies and Benchmarking with Baselines

Further evaluation was conducted on the commonly-used SEED test sets: test-en (English), and test-hard (challenging cases). As shown in Table II, on the test-en set, it ranked fourth in WER (2.57%) and third in SS (0.736), which may result from the constraints of our dataset. On the challenging test-hard set, the model achieved SOTA performance with a WER of 6.83% and SS of 0.776, demonstrating robustness in difficult synthesis scenarios.

VII. Conclusion

A. Research and Development Outcomes

Our research successfully addressed and resolved the two research questions posed in Section I-C. by implementing RAG system adaptability, achieving streaming speech synthesis, at the performance of (93.5% accuracy in finance, 91.8% in medical).

Through our novel Linguify multi-agent framework of SR-RAG and CHASM-TTS, we've created a system that effectively adapts its responses adaptable based on sentiment analysis, domain style, and previous interactions - addressing the weakness in existing generic customer support bots that provoke user frustration Our query transformation strategy dynamically reformulates responses when detecting user dissatisfaction, while the emotion and style adaptation capabilities enable appropriate tone modulation during emotionally charged interactions.

Experiments and evaluations validated that Linguify significantly outperforms existing solutions in real-time adaptation to user sentiment and conversational context. UAT results further confirms our approach surpasses performance thresholds necessary for practical deployment.

B. Limitations and Future Work

Despite these successes, limitations include substantial computational requirements and challenges with nuanced emotional states and multi-turn reasoning in specialized domains. Future work should focus on conditional diffusion models for better prosodic variation, hierarchical U-Net architectures of different morphologies. These improvements will further improve the sentimental analysis and fidelity of the system.

References

[1] X. Wang, Y. Liu, W. He, S. WUJI, and R. Luo, "The impact of artificial intelligence voice assistants on consumer experience based on the cognitive-affective-conative model," *SSRN Electronic Journal*, 01 2022.

[2] E. Brynjolfsson and A. Mcafee, "The business of artificial intelligence," *Harvard business review*, vol. 7, no. 1, pp. 1–2, 2017.

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021. [Online]. Available: https://arxiv.org/abs/2005.11401

[4] C. Wang, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li, L. He, S. Zhao, and F. Wei, "Neural codec language models are zero-shot text to speech synthesizers," *IEEE Transactions on Audio, Speech and Language Processing*, vol. PP, pp. 1–15, 01 2025.

[5] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *Trans. Mach. Learn. Res.*, 2023.

[6] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[7] Z. Ju, Y. Wang, K. Shen, X. Tan, D. Xin, D. Yang, E. Liu, Y. Leng, K. Song, S. Tang *et al.*, "Naturalspeech 3: Zero-shot speech synthesis with factorized codec and diffusion models," *ICML*, 2024.

[8] F. Petroni, P. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, and S. Riedel, "How context affects language models' factual predictions," in *Automated Knowledge Base Construction*, 2020.

[9] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading Wikipedia to answer open-domain questions," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1870–1879.

[10] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee *et al.*, "Natural questions: a benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[11] D. Zhang, S. Li, X. Zhang, J. Zhan, P. Wang, Y. Zhou, and X. Qiu, "Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities," *EMNLP (Findings)*, pp. 15 757–15 773, 2023.

[12] T. Dang, D. Aponte, D. Tran, and K. Koishida, "Livespeech: Low-latency zero-shot text-to-speech via autoregressive modeling of audio discrete codes," 09 2024, pp. 3395–3399.

[13] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: a large-scale dataset for fact extraction and verification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 809–819.

[14] A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, and M. Auli, "ELI5: Long form question answering," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3558–3567.

[15] N. Moghe, S. Arora, S. Banerjee, and M. M. Khapra, "Towards exploiting background knowledge for building conversation systems," *arXiv preprint arXiv:1809.08205*, 2018.

[16] J. Weston, E. Dinan, and A. Miller, "Retrieve and refine: Improved sequence generation models for dialogue," in *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, 2018, pp. 87–92.

[17] E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston, "Wizard of wikipedia: Knowledge-powered conversational agents," in *International Conference on Learning Representations*, 2019.

[18] A. Fan, C. Gardent, C. Braud, and A. Bordes, "Augmenting transformers with KNN-based composite memory," in *International Conference on Learning Representations*, 2020.

[19] J. Gu, Y. Wang, K. Cho, and V. O. Li, "Search engine guided neural machine translation," in *AAAI Conference on Artificial Intelligence*, 2018.

[20] K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang, "Generating sentences by editing prototypes," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 437–450, 2018.

[21] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis, "Generalization through memorization: Nearest neighbor language models," in *International Conference on Learning Representations*, 2020.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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)*, 2019, pp. 4171–4186.

[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[24] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[25] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[26] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" *arXiv preprint arXiv:2002.08910*, 2020.

[27] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2015.

[28] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.

[29] T. Févry, L. Baldini Soares, N. FitzGerald, E. Choi, and T. Kwiatkowski, "Entities as experts: Sparse memory access with entity supervision," *arXiv preprint arXiv:2004.07202*, 2020.

[30] M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-t. Yih, and M. Galley, "A knowledge-grounded neural conversation model," in *AAAI Conference on Artificial Intelligence*, 2018.

[31] Y. Jiang, T. Wang, H. Wang, C. Gong, Q. Liu, Z. Huang, L. Wang, and J. Dang, "Expressive text-to-speech with contextual background for icagc 2024," in *2024 IEEE 14th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2024, pp. 611–615.

[32] H.-H. Guo, Y. Hu, K. Liu, F.-Y. Shen, X. Tang, Y.-C. Wu, F.-L. Xie, K. Xie, and K.-T. Xu, "Fireredtts: A foundation text-to-speech framework for industry-level generative speech applications," 2025. [Online]. Available: https://arxiv.org/abs/2409.03283

[33] Y. Wang, H. Zhan, L. Liu, R. Zeng, H. Guo, J. Zheng, Q. Zhang, S. Zhang, and Z. Wu, "Maskgct: Zero-shot text-to-speech with masked generative codec transformer," *CoRR*, vol. abs/2409.00750, 2024.

[34] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[35] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 30, pp. 495–507, 2022.

[36] Z. Du, S. Zhang, K. Hu, and S. Zheng, "Funcodec: A fundamental, reproducible and integrable open-source toolkit for neural speech codec," *ICASSP*, pp. 591–595, 2024.

[37] E. Kharitonov, D. Vincent, Z. Borsos, R. Marinier, S. Girgin, O. Pietquin, M. Sharifi, M. Tagliasacchi, and N. Zeghidour, "Speak, read and prompt: High-fidelity text-to-speech with minimal supervision," *Trans. Assoc. Comput. Linguistics*, vol. 11, pp. 1703–1718, 2023.

[38] Y. Song, Z. Chen, X. Wang, Z. Ma, and X. Chen, "Ella-v: stable neural codec language modeling with alignment-guided sequence reordering," *CoRR*, vol. abs/2401.07333, 2024.

[39] C. Du, Y. Guo, H. Wang, Y. Yang, Z. Niu, S. Wang, H. Zhang, X. Chen, and K. Yu, "Vall-t: decoder-only generative transducer for robust and decoding-controllable text-to-speech," *CoRR*, vol. abs/2401.14321, 2024.

[40] D. Xin, X. Tan, K. Shen, Z. Ju, D. Yang, Y. Wang, S. Takamichi, H. Saruwatari, S. Liu, J. Li *et al.*, "Rall-e: robust codec language modeling with chain-of-thought prompting for text-to-speech synthesis," *CoRR*, vol. abs/2404.03204, 2024.

[41] S. Chen, S. Liu, L. Zhou, Y. Liu, X. Tan, J. Li, S. Zhao, Y. Qian, and F. Wei, "Vall-e 2: Neural codec language models are human parity zero-shot text to speech synthesizers," *CoRR*, vol. abs/2406.05370, 2024.

[42] B. Han, L. Zhou, S. Liu, S. Chen, L. Meng, Y. Qian, Y. Liu, S. Zhao, J. Li, and F. Wei, "Vall-e r: robust and efficient zero-shot text-to-speech synthesis via monotonic alignment," *CoRR*, vol. abs/2406.07855, 2024.

[43] T. Okamoto, H. Yamashita, Y. Ohtani, T. Toda, and H. Kawai, "Wavenext: Convnext-based fast neural vocoder without istft layer," *ASRU*, pp. 1–8, 2023.

[44] H. Siuzdak, "Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis," *ICLR*, 2024.

[45] L. Meng, L. Zhou, S. Liu, S. Chen, B. Han, S. Hu, Y. Liu, J. Li, S. Zhao, X. Wu *et al.*, "Autoregressive speech synthesis without vector quantization," *CoRR*, vol. abs/2407.08551, 2024.

[46] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *ICLR*, 2021.

[47] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *ICLR*, 2023.

[48] M. Le, A. Vyas, B. Shi, B. Karrer, L. Sari, R. Moritz, M. Williamson, V. Manohar, Y. Adi, J. Mahadeokar *et al.*, "Voicebox: Text-guided multilingual universal speech generation at scale," *Advances in neural information processing systems*, vol. 36, 2024.

[49] Y. Guo, C. Du, Z. Ma, X. Chen, and K. Yu, "Voiceflow: Efficient text-to-speech with rectified flow matching," *ICASSP*, pp. 11 121–11 125, 2024.

[50] S. Mehta, R. Tu, J. Beskow, E. Székely, and G. E. Henter, "Matcha-tts: A fast tts architecture with conditional flow matching," *ICASSP*, pp. 11 341–11 345, 2024.

[51] Y. Gao, N. Morioka, Y. Zhang, and N. Chen, "E3 tts: easy end-to-end diffusion-based text to speech," *ASRU*, pp. 1–8, 2023.

[52] K. Lee, D. W. Kim, J. Kim, and J. Cho, "Ditto-tts: Efficient and scalable zero-shot text-to-speech with diffusion transformer," *CoRR*, vol. abs/2406.11427, 2024.

[53] S. E. Eskimez, X. Wang, M. Thakker, C. Li, C.-H. Tsai, Z. Xiao, H. Yang, Z. Zhu, M. Tang, X. Tan *et al.*, "E2 tts: embarrassingly easy fully non-autoregressive zero-shot tts," *CoRR*, vol. abs/2406.18009, 2024.

[54] Y. Chen, Z. Niu, Z. Ma, K. Deng, C. Wang, J. Zhao, K. Yu, and X. Chen, "F5-tts: A fairytaler that fakes fluent and faithful speech with flow matching," *CoRR*, vol. abs/2410.06885, 2024.

[55] P. Anastassiou, J. Chen, J. Chen, Y. Chen, Z. Chen, Z. Chen, J. Cong, L. Deng, C. Ding, L. Gao, M. Gong, P. Huang, Q. Huang, Z. Huang, Y. Huo, D. Jia, C. Li, F. Li, H. Li, J. Li, X. Li, X. Li, L. Liu, S. Liu, S. Liu, X. Liu, Y. Liu, Z. Liu, L. Lu, J. Pan, X. Wang, Y. Wang, Y. Wang, Z. Wei, J. Wu, C. Yao, Y. Yang, Y. Yi, J. Zhang, Q. Zhang, S. Zhang, W. Zhang, Y. Zhang, Z. Zhao, D. Zhong, and X. Zhuang, "Seed-tts: A family of high-quality versatile speech generation models," 2024. [Online]. Available: https://arxiv.org/abs/2406.02430

[56] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[57] T. Dang, D. Aponte, D. N. Tran, T. Chen, and K. Koishida, "Zero-shot text-to-speech from continuous text streams," *CoRR*, vol. abs/2410.00767, 2024.

[58] M. Lajszczak, G. Cámbara, Y. Li, F. Beyhan, A. van Korlaar, F. Yang, A. Joly, A. Martín-Cortinas, A. Abbas, A. Michalski *et al.*, "Base tts: Lessons from building a billion-parameter text-to-speech model on 100k hours of data," *CoRR*, vol. abs/2402.08093, 2024.

[59] A. Dekel, S. Shechtman, R. Fernandez, D. Haws, Z. Kons, and R. Hoory, "Speak while you think: Streaming speech synthesis during text generation," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 11 931–11 935.

[60] S. Wang, M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, S. Chang, G. Tesauro, B. Zhou, and J. Jiang, "R^3: Reinforced ranker-reader for open-domain question answering," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 5981–5988.

[61] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.

[62] Z. Qin, W. Zhao, X. Yu, and X. Sun, "Openvoice: Versatile instant voice cloning," *CoRR*, vol. abs/2312.01479, 2023.

[63] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[64] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[65] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," in *Journal of Machine Learning Research*, vol. 13, no. Feb, 2012, pp. 281–305.

[66] Y. Chen, S. Zheng, H. Wang, L. Cheng, Q. Chen, and J. Qi, "An enhanced res2net with local and global feature fusion for speaker verification," *Interspeech*, 2023.

[67] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *Interspeech*, pp. 2616–2620, 2017.

[68] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "REALM: Retrieval-augmented language model pre-training," *arXiv preprint arXiv:2002.08909*, 2020.

[69] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.

[70] E. Perez, S. Karamcheti, R. Fergus, J. Weston, D. Kiela, and K. Cho, "Finding generalizable evidence by learning to convince Q&A models," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2402–2411.

[71] S. Wang, M. Yu, J. Jiang, W. Zhang, X. Guo, S. Chang, Z. Wang, T. Klinger, G. Tesauro, and M. Campbell, "Evidence aggregation for answer re-ranking in open-domain question answering," in *International Conference on Learning Representations*, 2018.

Appendix

A. Initial Proposed LangGraph-based RAG Architecture

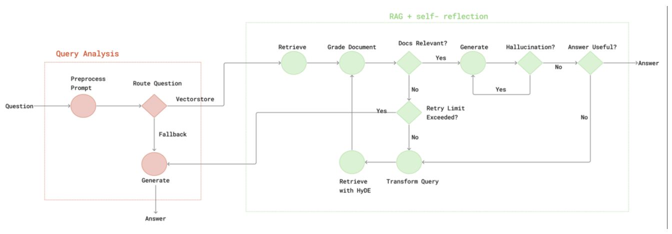

Fig. 11. Initial RAG architecture using LangGraph

We initially proposed an improved Retrieval-Augmented Generation (RAG) workflow with LangGraph in Figure 11. The system featured a query routing component to direct questions to domain-specific knowledge sources [69], and a document filtering mechanism to eliminate irrelevant content [70].

This architecture included self-reflective components for hallucination detection and answer quality assessment, inspired by recent work on model self-evaluation [71]. We also experimented with Hypothetical Document Embeddings (HyDE) as an alternative retrieval strategy [68]:

1. **Dual self-reflection components**: We employ both hallucination detection $H(a, \{d_i\}) \rightarrow [0, 1]$ and answer usefulness evaluation $U(a, q) \rightarrow [0, 1]$ where a is the generated answer, enabling iterative refinement when $H(a, \{d_i\}) > \tau_h$ or $U(a, q) < \tau_u$ [16].

2. **Enhanced HyDE integration**: We generate a hypothetical document $h = M(q)$ based on the query, and incorporate it in the retrieval process with $R(E(h), D)$ where $E()$ is the embedding function [68].

However, further implementation revealed significant limitations:

- LLaMA 3.2B model's restricted context window proved inadequate for effective self-reflection.
- HyDE consistently underperformed compared to standard retrieval methods in our call center domain.

Hence, these findings led to the development of our proposed novel SR-RAG in Figure 1, completely eliminating these limitations while enhancing the architecture's performance.

B. Web Crawling

We develop a miniature web crawling mechanism system to systematically collect, preprocess, and transform unstructured web information into machine-readable formats for AI consumption. It operates through a controlled crawling mechanism that navigates domain-specific websites, extracting meaningful content while filtering out navigational elements and presentation code.

Our pipeline in Algorithm 3 includes text normalization, structured table conversion to markdown format, content deduplication, and organized hierarchical storage for our SR-RAG architecture.

Algorithm 3 Web Content Extraction Algorithm

```
1: procedure ExtractWebContent($rootURL$, $domainFilter$, $pathFilter$)
2:     Initialize $headers$, $visitedURLs \leftarrow \emptyset$, $toVisit \leftarrow [rootURL]$
3:     while $toVisit$ is not empty do
4:         $currentURL \leftarrow$ pop from $toVisit$
5:         if $currentURL \in visitedURLs$ then continue
6:         end if
7:         Add $currentURL$ to $visitedURLs$
8:         $content, links \leftarrow$ FetchAndParse($currentURL, headers$)
9:         if $content$ is not null then
10:            Store $content$ for $currentURL$
11:        end if
12:        for each $link$ in $links$ do
13:            if $link \notin visitedURLs$ and matches filters then
14:                Add $link$ to $toVisit$
15:            end if
16:        end for
17:    end while
18:    SaveToFile(collected content)
19:    return extracted data
20: end procedure
21: function FetchAndParse($url$, $headers$)
22:    Request page with delay (rate limiting)
23:    Parse HTML and select content containers
24:    for each content div do
25:        Remove non-content elements (scripts, nav, etc.)
26:        Extract text from paragraphs, headings, lists
27:        Convert tables to markdown format
28:        Add unique content to results
29:    end for
30:    Extract and normalize all links
31:    return content and links
32: end function
```

C. Resources and Reproducibility

1) Model Availability: Pretrained models are available as open source resources at https://huggingface.co/josephchay/Linguify under MIT license, with configurations as discussed in Section III.

2) Code Availability: Full implementation is available at https://github.com/josephchay/linguify with documentation and comprehensive CHANGELOG.md. Development follows Conventional Commits specification[4] and Git Flow branching strategy.

3) Dataset Availability: The curated dataset with comprehensive annotations is available at https://huggingface.co/josephchay/LinguifyDataset, licensed for research and educational purposes.

[4]Documented at https://www.conventionalcommits.org/en/v1.0.0/ and available under the MIT license.

D. Hardware and Software Specifications

1) System Specifications: This research utilized two primary systems were: System A featuring an NVIDIA GeForce RTX 4090 GPU with 16,384 CUDA cores and 24GB GDDR6X memory with 1TB/s bandwidth, and System B equipped with an NVIDIA GeForce RTX 3060 GPU containing 3,584 CUDA cores and 12GB VRAM. Both systems were configured with standard consumer-grade CPUs, 32GB DDR4 RAM, and 1TB NVMe SSDs to facilitate efficient data processing operations.

2) Software Stack: Both systems leveraged an identical software stack. This environment was constructed using PyTorch with CUDA integration for GPU acceleration, Beautiful Soup for structured data extraction and web crawling, LangGraph for orchestrating complex reasoning workflows, and Matplotlib for visualization of training metrics and model performance evaluation.