Journal of Computer Science

Research Article

Optimized WFST-Based ASR for Arabic: Balancing Memory, Speed, and Accuracy

Wael A. Sultan[1], Mourad S. Semary[1] and Sherif M. Abdou[2]

[1]*Faculty of Engineering, Benha University, Benha, Egypt*
[2]*Faculty of Artificial Intelligence, Cairo University, Cairo, Egypt*

Article history
Received: 22-07-2025
Revised: 27-08-2025
Accepted: 17-09-2025

Corresponding Author:
Wael A. Sultan
Faculty of Engineering, Benha University, Benha, Egypt
Email: wael.ali@bhit.bu.edu.eg

Abstract: Weighted Finite-State Transducers (WFSTs) have revolutionized Automatic Speech Recognition (ASR) by enabling significantly faster decoding speeds compared to traditional systems that build the search space progressively. However, applying WFSTs to morphology-rich languages such as Arabic presents challenges due to the large vocabulary, resulting in extensive networks that exceed the memory capacity of standard CPUs. This study introduces various strategies to reduce the size of large vocabulary Arabic WFSTs with minimal impact on accuracy. We employed a star architecture for the network topology, which effectively reduced the network size and improved the decoding speed. Additionally, a two-pass decoding approach was adopted: the first pass used a smaller network with a short history language model, and the second pass rescored the produced lattice with a longer history language model. We explored several tuning parameters to find the optimal balance between network size and accuracy. Our results show that by using an optimized search graph built with a 2-gram language model instead of a 3-gram model, we achieve a 45% reduction in the graph's memory footprint with a negligible accuracy loss of less than 0.2% MR-WER. On the MGB3 benchmark, our method achieved 40x real-time Arabic ASR data processing with an accuracy of 83.67%, compared to the 85.82% accuracy of state-of-the-art systems, which only achieve 8x real-time performance on standard CPUs.

Keywords: FST-Decoding, ASR, Arabic Language, Model Footprint, WFST

Introduction

Modular or hybrid systems have been dominant in speech recognition since the beginning. They consist of many components, all of which have been built and engineered individually over decades of accumulated experience (Hannun, 2021). This involves modules for acoustics, pronunciation, and language, which operate together in inference but are trained independently (Wang et al., 2019). Recently, "end-to-end" (E2E) or Monolithic models have been adopted for speech recognition, where, for example, encoder-decoder transformer-based models have been used to represent all system components in one giant network with a large number of parameters (Li, 2022). This shift toward E2E models has been driven by several factors. First, E2E simplifies the system architecture by directly mapping the acoustic features to the textual outputs without relying on intermediate representations, e.g., linguistic units. Additionally, E2E models have shown superior performance, especially with the availability of a large quantity of training data, as they can capture complex patterns and dependencies in the data. Moreover, E2E models enabled end-to-end optimization, allowing for more efficient training procedures. Overall, over the past few years, both worlds have been competing, and each of them has its benefits and limitations (Wang et al., 2019):

- In traditional software, modular approaches have good usability because modular systems have reusable components; particularly in speech recognition, we can reuse the language model (LM) in different acoustic contexts without retraining. Additionally, we can reuse the acoustic model (AM) in different language contexts without retraining. In contrast, E2E models need to be retrained or fine-tuned when moving from one domain to another (Mohamed et al., 2012)
- The downside of modular systems is that they are not robust to errors. They consist of several components that are trained independently; they are not trained to collaborate well or function as a cohesive unit, which means that the error caused by one unit can propagate to all other units

SCIENCE Publications

- Moreover, E2E models learn from data, but it is difficult to encode domain knowledge into these models

Weighted Finite-State Transducers (WFSTs), a concept foundational to speech processing, have seen a resurgence in modern ASR through their integration into end-to-end differentiable frameworks like K2 (Povey et al., 2021). WFSTs provide a good compromise between E2E and modular approaches because they are reusable, adaptable, and robust to errors. But building such models for the Arabic language is challenging because one of the key weaknesses is the limitation of the system vocabulary, especially with respect to dialects (Hussein et al., 2022). For morphologically rich languages like Arabic, Out-of-Vocabulary (OOV) words remain a primary source of errors in large vocabulary systems. (Hussein et al., 2022) have highlighted the necessity of very large vocabularies (over 1 million words) to achieve high coverage, underscoring the severity of the OOV problem. Requiring such a large vocabulary imposes several challenges for the system, including the following (Brants et al., 2007):

- The sparse LM: Building a trigram LM with a dictionary size on the order of a million words would require a very large text corpus beyond available resources; even if this model is built, it would be very large, on the order of hundreds of GigaBytes. To fit in computer memory would require considerable processing time. This problem is more serious for higher-order LMs
- The exploded search space of the decoder: In the FST system, a synchronous Viterbi decoder is used. The decoder expands its search space dynamically during its processing of the input. At each word end hypothesis, the search space is expanded with the number of paths equal to the vocabulary size. This means that for a one million-word dictionary, the search space is expanded with an extra million search paths every time a new word end hypothesis is reached. Using aggressive pruning thresholds to reduce the size of the search space decreases system accuracy
- Decoder speed: With this exploded search space, the processing time becomes much slower, and the system loses its real-time property

In this proposal, we intend to develop and optimize Finite State Transducer (FST)-based Automatic Speech Recognition (ASR) models designed for the Arabic language. Leveraging the K2 framework (Povey et al., 2021), our research will concentrate on Connectionist Temporal Classification (CTC) topology, search graph construction, and refining decoding parameters. By experimenting with various CTC topologies and exploring different methods for constructing search graphs, our goal is to improve the performance and efficiency of ASR systems in processing Arabic speech.

Comprehensive Theoretical Basis

A weighted finite-state transducer is a type of finite-state machine. It is a directed graph consisting of states connected by labeled transitions. Each transition is associated with an input symbol, an output symbol and a weight, which is typically a numerical value that represents the cost or probability of traversing that transition (Mohri, 1997). In a WFST, the input symbols and output symbols can be different, allowing for the modeling of phoneme-to-grapheme mappings, language translations, speech recognition, and other natural language processing tasks.

E2E WFST Speech Recognition

The introduction of differentiable Weighted Finite-State Transducers (WFSTs) by recent tools such as K2 and GTN (Hannun et al., 2020) has significantly advanced the integration of WFSTs with end-to-end (E2E) models. This development has enabled the use of WFSTs in both the training and decoding phases of speech recognition. In E2E speech recognition sequence models, the challenge often lies in computing the conditional probability of an output sequence (text) given an input sequence (audio frames) where the two sequences differ in length. Various criteria, such as automatic segmentation criterion (ASG) (Collobert et al., 2016), Connectionist Temporal Classification (CTC) (Graves et al., 2006) and maximum mutual information loss (MMI)(Povey & Woodland, 2002), have been used to address this issue. These criteria are effectively represented through lattice operations.

Recently, neural transformer-based models have gained prominence in Acoustic Modeling (AM) for Automatic Speech Recognition (ASR), mainly due to the efficacy of the CTC criterion (Hannun et al., 2014). CTC facilitates the computation of conditional probabilities by mapping a sequence of input speech frames to a sequence of output tokens. This is achieved by marginalizing all possible alignments between the input sequence $X = [x_0, \ldots, x_{N-1}]$ and the output $y = [y_0, \ldots, y_{M-1}]$ where $M \leq N$. In the context of WFSTs, this alignment problem can be solved through the intersection of the emission graph, which represents the acoustic outputs of the neural acoustic model, and the supervision graph, which represents the reference token sequences. Conceptually, the graph-based CTC loss can be expressed as:

$$\log P(y|X) = Score(\gamma \circ \epsilon) \tag{1}$$

Here, γ denotes the supervision graph, ϵ represents the emissions graph, $\circ$ signifies the intersection operation, and $Score(.)$ is forward score computed via the "log-sum-exp" (LSE) over all paths for the graph obtained by the intersection. Let's explore a simple example to clarify the concept. Suppose we have an AM with a vocabulary of just three tokens: a, b, and c. We aim to calculate the CTC loss between the reference transcript "ab" and three audio frames, with their respective log probabilities provided in Table 1:

Table 1: Simple example for log probabilities over a three-time frames for a, b, c and $< blk >$ as a blank symbol

	$< blk >$	a	b	c
Frame 0	-1.64	-1.39	-1.17	-1.38
Frame 1	-1.13	-1.78	-1.16	-1.61
Frame 2	-1.42	-1.29	-1.26	-1.58

The steps to find the CTC loss are as follows:

- First, the weighted finite-state acceptor (WFSA), shown in Figure 1, is constructed to represent the above logs, which is referred to as an emission graph ϵ. In this graph, the states correspond to the different time frames of the input audio, while the transitions between states represent the possible output symbols (a, b, c and the blank $< blk >$), In this graph, the states correspond to the different time frames of the input audio, while the transitions between states represent the possible output symbols ϵ thus captures the likelihood of transitioning from one time frame to another while emitting a particular symbol, providing a structured way to map the acoustic input to potential sequences of tokens.

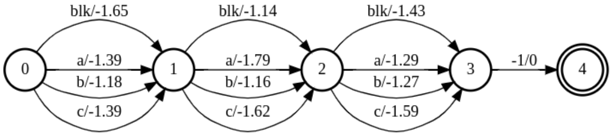

Fig. 1: WFSA representing acoustic emission probabilities

- Next, we convert the reference transcriptions "ab" into the linear WFSA, as shown in Figure 2, we then construct the CTC topology as shown in Figure 3. This topology is represented by a directed complete graph with self-loops. For a vocabulary of size N, including the blank symbol $< blk >$, the graph consists of N states and N^2 arcs.

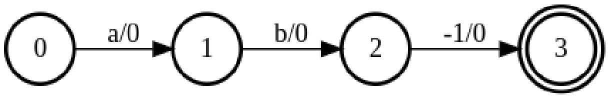

Fig. 2: WFSA representing the reference transcript "ab"

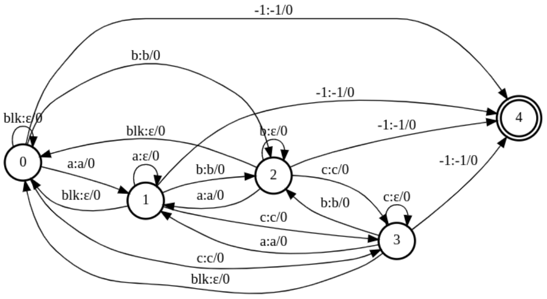

Fig. 3: WFST representing a regular CTC topology

- The alignment or decoding graph γ is generated by composing the linear WFSA (representing the transcript) with the CTC topology, as shown in Figure 4.

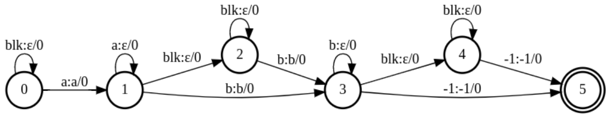

Fig. 4: Composition between a reference transcript and the CTC topology

- Finally, we intersect the last decoding graph γ with the emission graph ϵ, to build the resulting lattice, displayed in Figure 5, we can then compute the forward score on this lattice to determine CTC loss value.

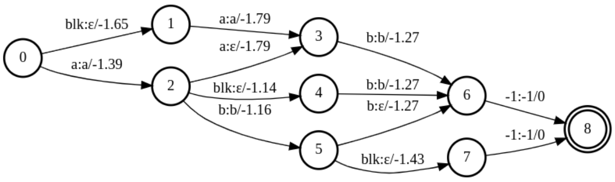

Fig. 5: Searching graph generated from the composition between the decoding and emission graphs

It's worth mentioning that in the second step of the previous procedure to calculate CTC loss, we can use an alternative CTC topology called EESEN (Miao et al., 2015), represented in the graph shown in Figure 6. Unlike the regular CTC topology, EESEN does not require a mandatory blank token $< blk >$ between two consecutive identical tokens. This results in a graph with $N + 2$ states and $3N + 1$ arcs for a vocabulary of size N (Laptev et al., 2021). When applied to the earlier example, this topology produces the decoding graph shown in Figure 7. As with the previous approach, the intersection of the decoding graph with the emission graph yields the search graph shown in Figure 8.

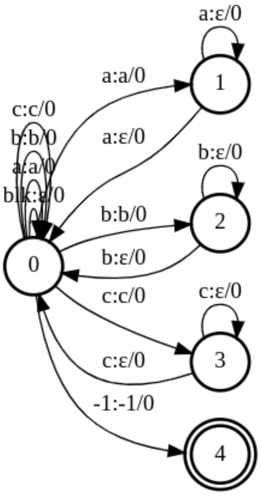

Fig. 6: WFST representing the EESEN-modified CTC topology

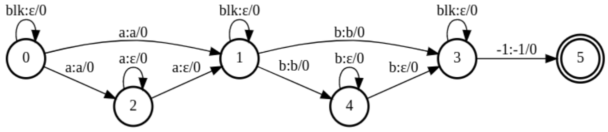

Fig. 7: Composition between a reference transcript and the modified EESEN CTC-topology

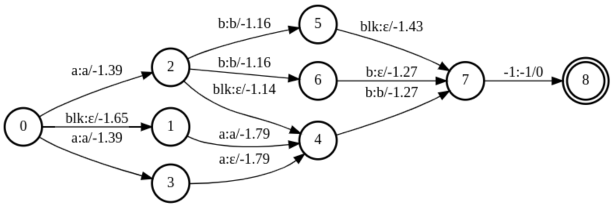

Fig. 8: Modified searching graph generated from the composition between the decoding and emission graphs

Multipass Decoding

In ASR systems, multipass decoding is a widely used strategy to balance computational efficiency with recognition accuracy. This approach involves multiple stages of decoding, where each pass progressively refines the search for the best transcription. In our system, we implement a two-pass decoding strategy. The first pass uses an LM with a short history to minimize computational complexity, which is particularly important for large vocabulary languages such as Arabic. In the second pass, a language model with a longer history is employed to conduct a more thorough search, optimizing the path selection within the lattice generated in the first pass. The following sections provide a detailed explanation of this approach.

First Pass

To enhance the outputs of the AM with domain knowledge, we need to create a search graph that combines information from both sources. This process begins by compiling a transition-labeled graph (TLG), which is formed from three key components represented as WFSTs (Xiang & Ou, 2019):

- **Lexicon**: This component acts as a dictionary mapping each word to its corresponding subunits or tokens. For instance, a word such as "هاتلي"" might be mapped to subword units such as هات, and لي, which helps connect the acoustic information to potential words. The lexicon is built and compiled using a large corpus of text and is denoted as L.
- **Grammar**: This component represents the relationships between words, as n-grams (sequences of words). It captures the typical word order and usage patterns within a specific domain (e.g., "عليكم" often follows "السلام", which is a popular Arabic greeting). This injects domain knowledge, which is denoted as G.
- **Topology**: This refers to the "CTC topology" described earlier. It essentially defines the allowed sequence of sound units based on the capabilities of the AM and denoted as T.

The construction of the TLG involves a series of standard WFST operations. First, the Lexicon (L) and Grammar (G) FSTs are composed. The resulting LG graph is then optimized using determinization and minimization to create a more compact and efficient structure. Finally, this optimized graph is composed with the CTC Topology (T) graph. This entire workflow is concisely represented by the equation:

$$TLG = T \circ min(det(L \circ G)) \quad (2)$$

where min is a minimization operation, det is a determinization operation, and $\circ$ donotes composition operation.

During inference, the TLG graph guides the search for the most likely word sequence as follows:

- First, the AM analyzes the speech utterance U, which provides probabilities for different sound sequences, also called emissions (represented by ϵ).
- Second, a search graph is constructed by combining both ϵ and TLG, where the combined L & G components in the TLG suggest word sequences based on the definitions of the lexicon and the domain-specific grammar. Therefore, the decoder considers both factors (acoustics and domain knowledge) when navigating the TLG graph to find the most likely word sequence that matches the speech and adheres to the domain's language patterns. This process is mathematically described as:

$$S = \epsilon \circ TLG \quad (3)$$

Here, ϵ is the WFST that represents the output of the neural model. Beam search or Viterbi/n-best search is performed efficiently on the lattice with dynamic programming algorithms and sparse representations of the lattice like ragged tensors. In contrast to expanding the search space on the fly during the decoding process, the search space is precomplied in a single FST. Usually, search graph TLG consumes gigabytes (Laptev et al., 2021) for a large vocabulary system. Therefore, we do not perform a full search but instead apply beam pruning. The pruned search graph is represented as:

$$S = prune(\epsilon \circ TLG) \quad (4)$$

This pruning process is controlled by parameters such as beam size and active states, which help to limit the search space during decoding. This prevents the system from considering an overwhelming number of possible word sequences, making the decoding process more efficient while maintaining accuracy. We will explore the impact of these parameters on system performance in our experiments later in this paper. Figure 9 summarizes the first pass decoding process, where the audio input processed by the AM, producing neural network (NNet) outputs that represent probabilities for different sound sequences. These outputs are then combined with the TLG to form a search graph (lattice). This lattice represents all possible paths the decoder might take based on both the AM outputs and the domain-specific knowledge encoded in the TLG. Finally, the most likely word sequence is selected from the lattice, resulting in the final transcript.

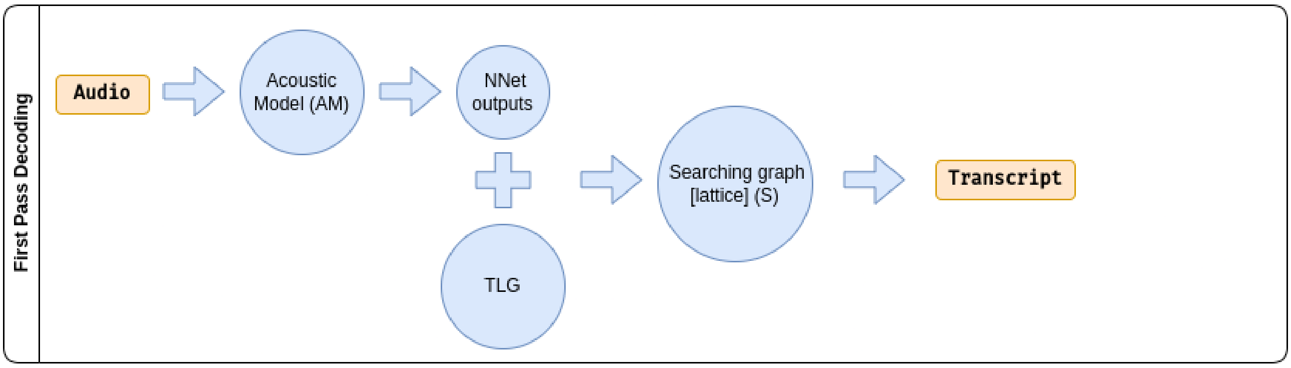

Fig. 9: First pass decoding

Second Pass

In the final stage of the ASR decoding process, a technique known as lattice rescoring is commonly employed to refine the results obtained from the initial decoding pass. This involves using a powerful LM to reassess and improve the quality of the generated lattice. In our experiments, we utilized the following decoding approaches adapted from the Icefall framework:

- **N-best Oracle Decoding**: This technique involves extracting multiple potential transcription paths -denoted as n paths- from the search graph (lattice) generated during decoding. Each of these paths represents a different possible transcription of the input speech, offering a range of candidate sequences. The primary objective of using this method is to evaluate the effectiveness and quality of the compiled TLG. By comparing these candidate paths against a reference transcript, we can identify the path with the minimum edit distance (i.e., the fewest differences) from the reference. This path is then selected as the final decoding output. The rationale behind this approach is that if the TLG is well-constructed, the correct transcription should be among the top n paths, and this method helps confirm the accuracy of the TLG by selecting the most accurate path from the options provided.
- **Whole-Lattice Rescoring**: As illustrated in Figure 10, technique involves reassessing the lattice generated during the first pass decoding using an n-gram LM. The n-gram LM evaluates different sequences of words based on the statistical likelihood of their co-occurrence within the language. By assigning scores to these word sequences, the LM enhances the lattice by prioritizing more probable word combinations. In this process, the scores generated by the n-gram LM are combined with the CTC scores produced during the initial decoding pass. This fusion of scores yields a new set of weighted paths within the lattice, effectively refining the potential transcriptions. The lattice is then rescored, resulting in a "rescored lattice" that reflects these improved probabilities. From this rescored lattice, the path with the highest combined score -representing the most likely transcription- is selected as the final output. This approach is especially useful for correcting errors from the first pass decoding, as it incorporates both the acoustic model's predictions and more sophisticated linguistic knowledge from the n-gram LM, leading to a more accurate transcription. Figure 10 illustrates the entire process, starting with the initial audio input, which is processed by the AM to generate NNet outputs. These outputs are then combined with the TLG to form the first pass search graph (lattice). In the second pass, the search graph is further refined by incorporating a more powerful grammar model (Better G), resulting in a rescored lattice. The final transcript is then derived from this improved lattice, ensuring a higher quality and more accurate output.
- **Attention Decoder**: This approach employs a multistep process similar to the previously described whole-lattice rescoring but focuses specifically on the n-best paths within the search graph. Rather than rescoring the entire lattice, it selectively evaluates the top n most likely hypotheses. The rescoring process combines an enhanced n-gram LM with attention mechanisms derived from the acoustic model's decoder. This attention mechanism allows the model to focus on the most relevant portions of the input, refining the transcription by considering both the acoustic and linguistic contexts. Figure 11 illustrates this process, beginning with the audio input processed by the AM to produce NNet outputs and in this case also the decoder attentions. These outputs are then combined with the TLG to generate the initial search graph (lattice). In the second pass, the lattice undergoes n-

best rescoring with a more advanced grammar model (Better G) and attention mechanisms from the AM decoder. This results in a rescored lattice, from which the most accurate path is selected to produce the final transcript.

While these decoding methods offer significant control over the decoding process through a number of hyperparameters such as beam width, active states, and the number of paths (n), one notable limitation is their high memory consumption. This constraint poses challenges in deploying ASR systems in production environments. Therefore, one of our primary objectives is to investigate and implement optimizations to mitigate this issue, ensuring efficient and scalable ASR performance in real-world applications.

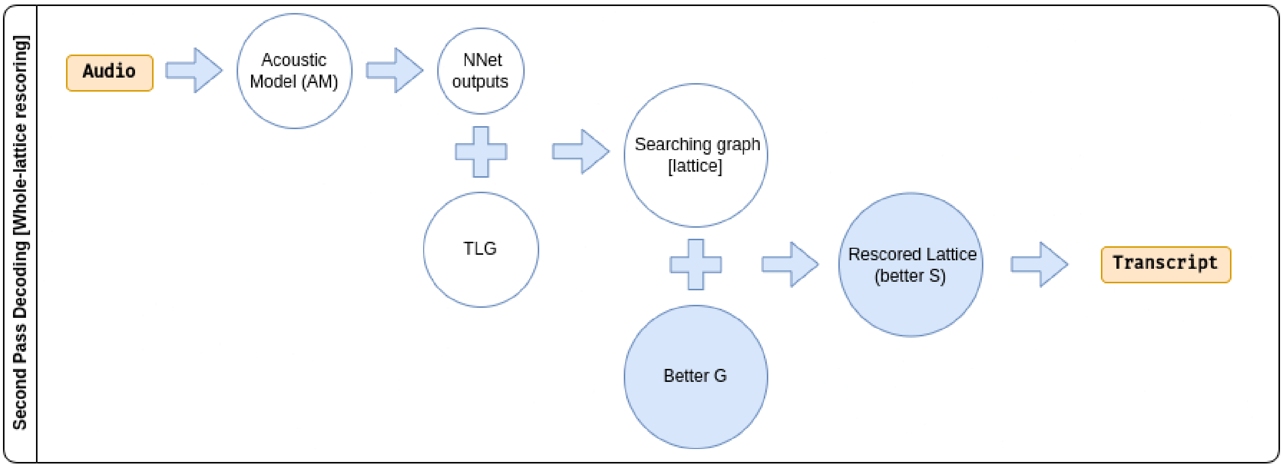
Second Pass Decoding [Whole-lattice rescoring]
Audio
Acoustic Model (AM)
NNet outputs
TLG
Searching graph [lattice]
Better G
Rescored Lattice (better S)
Transcript

Fig. 10: Whole-lattice rescoring

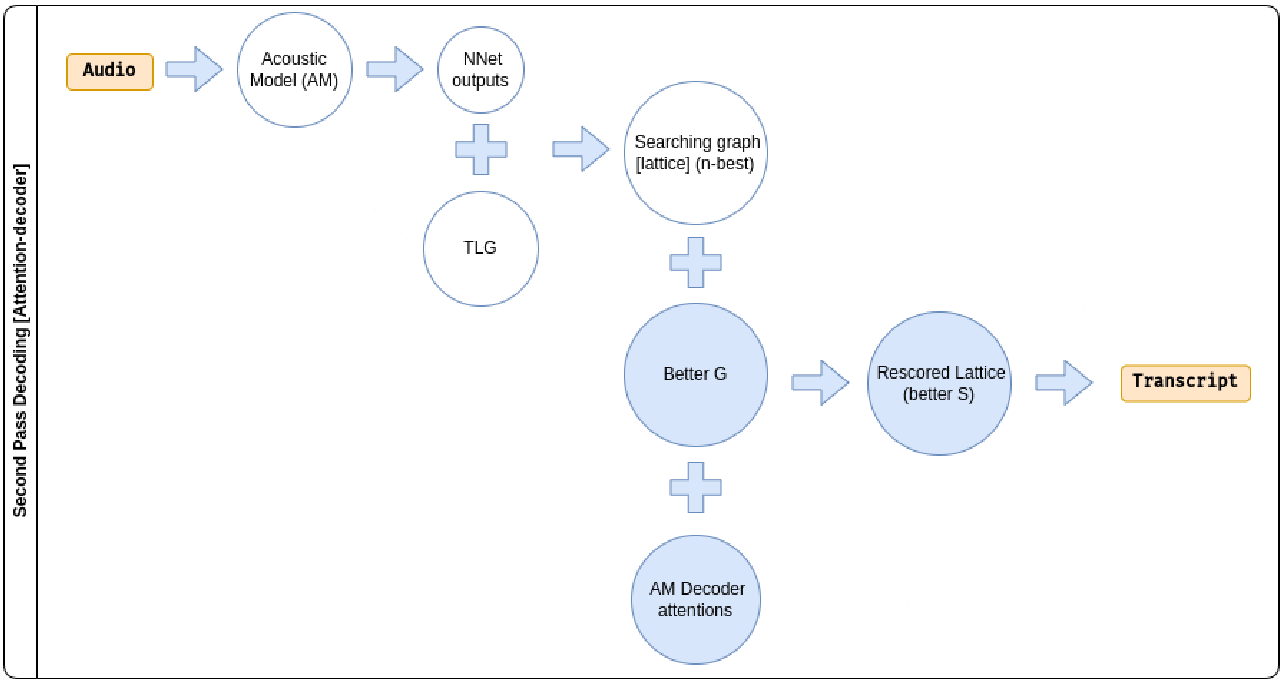
Second Pass Decoding [Attention-decoder]
Audio
Acoustic Model (AM)
NNet outputs
TLG
Searching graph [lattice] (n-best)
Better G
AM Decoder attentions
Rescored Lattice (better S)
Transcript

Fig. 11: Attention Decoder

Methods

As discussed earlier, the development of a multipass decoding ASR system involves a series of complex steps and critical decisions. Figure 12 provides a visual overview of our comprehensive approach to building such a system, designed to accurately recognize Modern Standard Arabic (MSA) and Egyptian Colloquial Arabic (ECA). This section details our methodologies, focusing on optimizing the quality of the TLG, enhancing search graph accuracy, refining decoding strategies, and fine-tuning hyperparameters. The approach is structured into four primary steps:

1. Textual Data Acquisition and TLG Construction: In this step, we systematically explore all available textual data resources, including large corpora for both MSA and ECA. We conduct various cleaning

and normalization scenarios to ensure data quality and consistency. The focus is on building an optimized TLG that accurately represents the linguistic nuances of both MSA and ECA. This involves carefully selecting and weighting the lexicon, grammar, and topology components of the TLG to reflect the diverse language patterns encountered in the target dialects.

2. Audio Data Acquisition and Acoustic Model Training: Here, we gather a diverse set of audio data that covers both MSA and ECA, ensuring a broad representation of speakers, accents, and environments. The collected data is preprocessed and aligned with the corresponding textual data to train a robust AM. The AM is designed to capture the acoustic characteristics of the Arabic language, with particular attention to the phonetic variations between MSA and ECA.
3. Exploring Decoding Strategies: This step involves investigating the previously mentioned decoding strategies to determine the most effective approach for our multipass system. We experiment with different configurations of beam search, lattice rescoring, and n-best list generation. Additionally, we explore the integration of advanced language models and attention mechanisms to further refine the decoding process, ensuring that the system can effectively balance acoustic and linguistic information during transcription.
4. Benchmarking Against Current End-to-End Models: Finally, we benchmark our system against existing state-of-the-art end-to-end ASR models to evaluate its performance. This benchmarking process includes a detailed comparison of accuracy, speed, and resource efficiency.

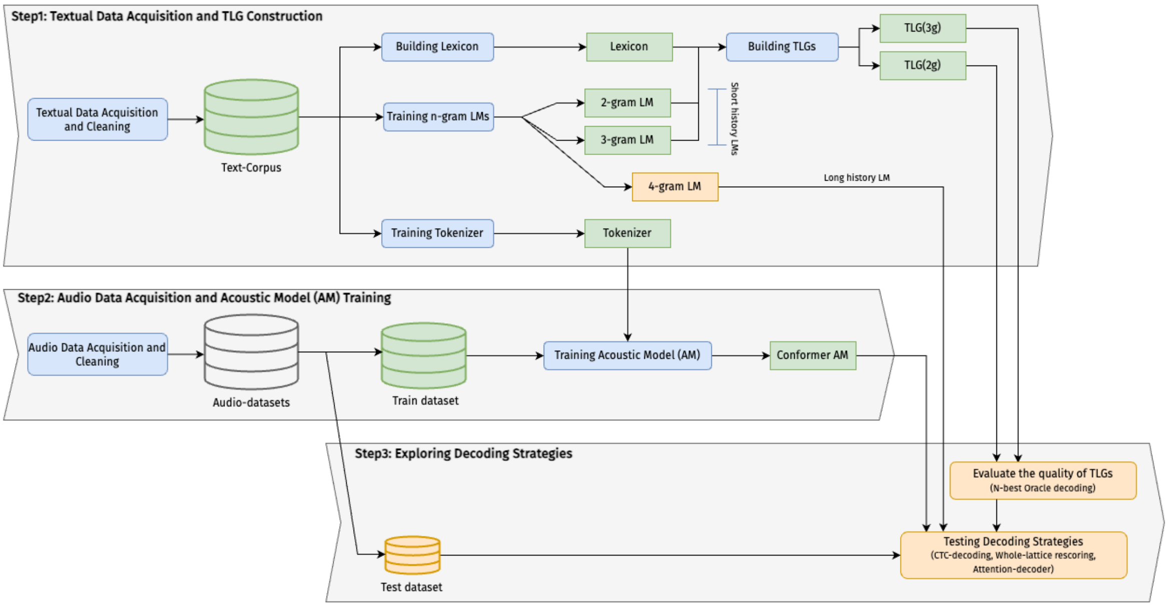
Step1: Textual Data Acquisition and TLG Construction
Textual Data Acquisition and Cleaning
Text-Corpus
Building Lexicon
Training n-gram LMs
Training Tokenizer
Lexicon
2-gram LM
3-gram LM
4-gram LM
Tokenizer
Short history LMs
Long history LM
Building TLGs
TLG(3g)
TLG(2g)
Step2: Audio Data Acquisition and Acoustic Model (AM) Training
Audio Data Acquisition and Cleaning
Audio-datasets
Train dataset
Training Acoustic Model (AM)
Conformer AM
Step3: Exploring Decoding Strategies
Test dataset
Evaluate the quality of TLGs
(N-best Oracle decoding)
Testing Decoding Strategies
(CTC-decoding, Whole-lattice rescoring, Attention-decoder)

Fig. 12: Workflow for developing a multipass decoding ASR system

To provide a detailed understanding of our approach, the following subsections of this section will first introduce the datasets used for audio and textual data acquisition, highlighting the diversity and richness of the sources utilized. Next, we will discuss the tokenizer, including our rationale for selecting the number of tokens. Finally, we will explore the process of constructing the TLGs (Tokenization, Lexicon, and Grammar models), detailing the methodologies employed to build these essential components.

Datasets

The workflow begins with the crucial step of acquiring both speech and text data. Our objective is to develop an ASR system capable of handling both MSA and ECA. This task is particularly challenging due to the scarcity of clean, high-quality data resources for MSA and even more so for its dialects. To address this, we have carefully gathered and prepared a comprehensive set of datasets to support the training and evaluation of our ASR system.

Speech Dataset

Table 2 presents a summary of the speech datasets utilized to create a combined training set for the acoustic modeling, totaling 840 hours of audio data. Our selection criteria for these datasets focused on two key aspects: dialectal coverage and annotation quality. The MGB3-dev (Ali et al., 2017) dataset was specifically chosen because it is a standard benchmark for ECA and, crucially, provides multiple reference annotations. This feature is essential for a more robust evaluation using the multireference word error rate (MR-WER) metric (Ali et al., 2015), as annotating non-MSA speech can involve different methods. For this reason, the MGB3-dev dataset is treated as a test set. To augment our training data, the private AIC-MSA and AIC-EG datasets were incorporated to provide diverse, real-world examples of both MSA and ECA. These datasets cover a wider range of domains and acoustic conditions than many publicly available corpora, which is critical for training a robust acoustic model.

Table 2: Speech dataset statistics

Dataset	# Utterances	Size (hours)	Utterance duration (seconds)			
			Mean	Min	Max	99%
AIC-MSA	290,101	443	5.5	0.6	13	11.5
MGB2 (dev/test) (Ali et al., 2016)	9,135	16	6.3	1.5	20.3	10.1
AIC-EG	136,862	368	9.7	0.5	20	17.5
MGB3 (adapt)(x4) (Ali et al., 2017)	1,013	2	7.4	2.5	12.2	10.2

The AIC-MSA and AIC-EG are from YouTube with the following outlined process:

- The process begins with a thorough manual search and compilation of a comprehensive list of URLs for YouTube videos and channels. This search emphasizes diversity, aiming to cover a broad spectrum of domains, such as education, sports, arts, news, talk shows, and politics
- Special attention is given to filtering clips to avoid repetitive speaker occurrences and to ensure a preference for clean, conversational audio without significant cross-talk
- The selected audio is automatically downloaded and segmented using the Voice Activity Detection (VAD) model
- Finally, each audio segment is manually reviewed and corresponding transcripts are carefully annotated

After summing the datasets, speed perturbations (0.9x, 1x, and 1.1x) and augmentations with the MUSAN (Snyder et al., 2015) dataset are applied, resulting in a total of 2,520 hours of acoustic data ready for AM training.

Text Corpus

Building a robust text corpus for Egyptian Colloquial Arabic involved multiple rounds of collection and processing. As summarized in Table 3, the text data was sourced from various platforms, including Reddit, where Egyptian Arabic discussions take place, and popular Egyptian YouTube channels, which provided a wealth of user-generated comments. Additionally, Twitter was a significant resource, offering a large volume of tweets posted from Egypt. We also incorporated transcripts from the speech datasets previously discussed, ensuring that the corpus was representative and diverse.

The collected data underwent a rigorous cleaning process, which included removing emojis, repetitive sequences, tags, and foreign characters. This process resulted in a corpus of 39 million sentences of colloquial text, encompassing 1.5 million unique words. This cleaned text corpus was then utilized to develop the following components:

- Lexicon: a list of unique words used to construct the lexicon.
- Language Models: 2-gram, pruned 3-gram, and pruned 4-gram language models
- Tokenizer: A 5000-subword byte-pair tokenizer model, tailored to optimize tokenization efficiency for our ASR system

Table 3: Text corpus statistics

Source	# Sentences
Egyptian Arabic data from Reddit	661,377
Egyptian Arabic comments from the top 100 popular Egyptian YouTube Channels	5,057,774
Egyptian Arabic tweets posted from Egypt on Twitter before the 21st of May 2022	35,692,842
All ASR transcriptions in the speech dataset	1,472,174

Tokens

Characters, phones, character-based subwords, phone-based subwords, and words have been used as tokens for ASR models; however, character-based subwords lie between characters and words in terms of complexity and add the advantages of open vocabulary and simplicity to the model (Wang et al., 2020). For a rich morphological language such as the Arabic language, experience shows that choosing 1K tokens or more in a subword-based model is better (Hussein et al., 2022). For our model, the vocabulary size of 5,000 subword tokens was determined empirically. Following common practices in modern ASR, we explored several sizes and found that 5K provided a good balance between model complexity and the ability to represent the intricate morphology of Arabic without excessive fragmentation of words. To generate these units from our text corpus, we employed SentencePiece (Kudo & Richardson, 2018), a widely-used subword tokenization method. It is important to note, however, that a large token set can result in a very large and computationally expensive search graph when using a standard CTC topology. This observation motivated our experiments with a modified, more efficient CTC topology, the results of which are demonstrated later in the paper.

Building TLGs

As highlighted in the previous section on multipass decoding, constructing a TLG (Tokenization, Lexicon, and Grammar) involves a series of intricate steps and the integration of several key components. Our workflow for constructing the TLGs, illustrated in Figure 13, is adapted from the established recipe available in the Icefall toolkit for the LibriSpeech corpus. Our novel contribution is not in the creation of this workflow itself, but in its specific adaptation and optimization for the Arabic language. This involved extensive experimentation with different n-gram orders for the grammar component and a detailed analysis of the resulting memory-accuracy trade-offs, which are not explored in the original recipe. The process begins with the acquisition of the text corpus, as detailed in the earlier subsections. This corpus serves as the foundation for training the subword tokenizer and short-history n-gram LMs. The Lexicon (L) is then generated, encompassing a list of all unique words in the corpus along with their corresponding subword tokens, facilitated by the trained tokenizer. Subsequently, each trained short-history n-gram LM is converted into a Finite State Transducer (FST) and then combined with the Lexicon to produce the (LG) FST. This LG FST is further determinized and minimized to reduce its size and enhance the model's efficiency. Following this, the Topology (T) FST is generated based on the number of tokens, and it is compiled with the LG to create the TLG. This TLG is pruned and optimized, resulting in the final TLG used in the decoding process. Following this workflow, we generated the following FSTs for our experiments:

- TLG(2g): constructed with a bigram G
- TLG(3g): constructed with a trigram G

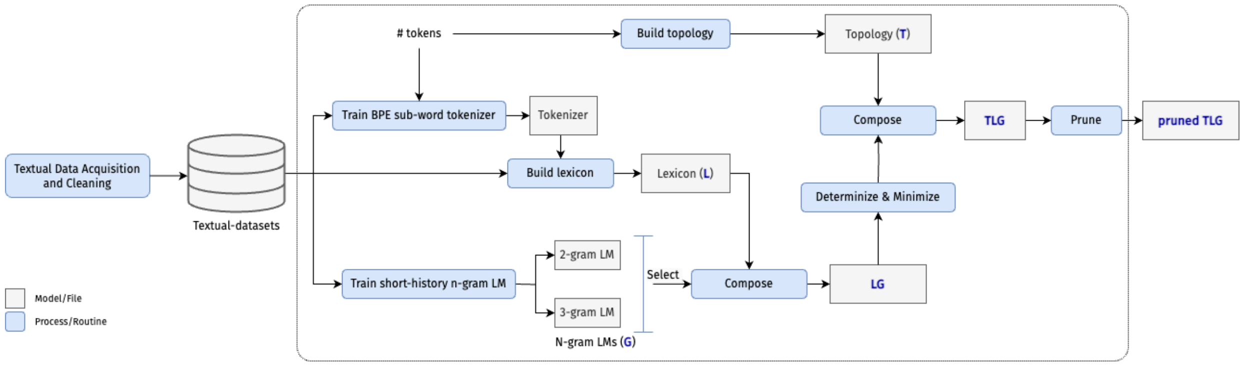

Fig. 13: Workflow for constructing TLGs

Results and Discussion

In this section, we present a through analysis of our ASR system's performance, focusing on the evaluation of various decoding techniques and the influence of different system hyperparameters. We start by comparing the effectiveness of different decoding stratigies, such as CTC decoding, lattic rescoring, and attention-based methods. Additionally, we assess how adjustments of some key hyperparameters, including the n-gram orders, and beam widths, impact the system's overall performance. This analysis aims to identify the optimal configurations that achieve the best balance between recognition accuracy and computational efficiency, enhancing our ASR system.

CTC-Decoding

The experiments began with the training of a Conformer-large model utilizing 5,000 subword tokens, following the LibriSpeech Icefull Conformer recipe. To evaluate the models' performance, we measured the multireference word error rate (MR-WER) (Ali et al., 2015) using CTC-decoding, where the CTC graph served as a search graph. The regular CTC topology yield an MR-WER of 18.89%, while a modified CTC topology (EESEN) resulted in an MR-WER of 20.98%. The latter outcome is attributed to the fact that a regular CTC topology was employed for loss computations during the training process. Despite this, the modified topology demostrated notable memory efficiency. Although the MR-WER for the EESEN topology was approximately 2% higher, its memory footprint was over 1,500 times smaller (0.3 MB vs. 500 MB) and its decoding was over 500 times faster on a CPU, as detailed in Table 4. This demonstrates a significant trade-off, making the EESEN topology a highly compelling choice for deployment in memory- and latency-constrained environments where a marginal loss in accuracy is acceptable for substantial gains in efficiency.

Nbest-Oracle

To evaluate the effectiveness of using complied TLGs in decoding instead of CTC graphs, we first assessed the knowledge embedded within the TLGs. We utilized the `nbest_oracle` method, which empolys reference annotations to retrieve the most accurate path from the search graph. The results for the TLG(3 g) graph, as summarized in Table 5, were obtained using parameters set to `num_paths=10,000`, `nbest_scale=0.6`, `beam_width=20`, and `max_active_states=10,000`. These results suggest that the TLG serves as an effective search graph, as evidenced by the range of WER metrics across different reference annotations.

We further explored the impact of varying the number of paths on the MR-WER, as shown in Table 6. This analysis revealed that increasing the number of paths generally leads to a reduction in MR-WER, indicating improved performance. Additionally, we tested the effect of doubling the beam width on MR-WER, as presented in Table 7. The results indicate that increasing the beam width did not significantly impact the MR-WER, suggesting that the original beam width settings were already optimal for this task.

Finally, to determine the optimal `nbest-scale` for our test set, we experimented with a range of `nbest-scale` values using the compiled TLGs. The resulting oracle MR-WERs for TLG(2 g) and TLG(3 g) with 1,000 and 10,000 paths are shown in Figure 14. The results indicate that an `nbest-scale` between 0.08 and 0.22 consistently yields the lowest MR-WER, making it a reliable choice across all TLGs. Based on this consistent performance, we have selected an `nbest-scale` of 0.2 for further experiments. This selection ensures that our decoding process is optimized for accuracy while balancing computational efficiency and model effectiveness.

Table 4: Comparison between regular CTC and modified CTC topologies, testing a 6-sec-duration sample on an I7-10700K CPU machine

Model	Searching Graph Footprint (MegaBytes)	Decoding Speed (Secs)
Regular CTC-topology	~ 500	29.9
Modified (EESEN) CTC-topology	~ 0.3	0.054

Table 5: WERs for MGB3-dev set with `nbest_oracle` at `num_paths=10000`, with `nbest_scale=0.6`, `beam_width=20`, and `max_active_states=10000`

Reference	MR-WER%	AV-WER%	Alaa-WER%	Ali-WER%	Omar-WER%	Mohmad-WER%
Alaa	14.76	24.9	23.27	25.93	25.32	25.09
Ali	14.56	25.35	26.62	22.81	26.41	25.57
Omar	14.54	24.36	25.14	25.63	23.16	23.53
Mohmad	14.35	24.4	25.56	25.44	24.21	22.39

Table 6: WERs for MGB3-dev set with `nbest_oracle` at different `num_paths` with `nbest_scale=0.6`, `beam_width=20`, and `max_active_states=10,000`

# Paths	50	100	1,000	2,000	4,000	10,000	20,000	60,000	100,000
MR-WER% (Alaa)	19.38	18.36	16.15	15.78	15.52	14.42	14.11	14.07	13.78

Table 7: WERs for the MGB3-dev set with `nbest_oracle` at `num_paths=1,000` and `max_active_states=10000`

	MR-WER%
`beam_width=20`, `output_beam=8`	16.15
`beam_width=40`, `output_beam=16`	16.1

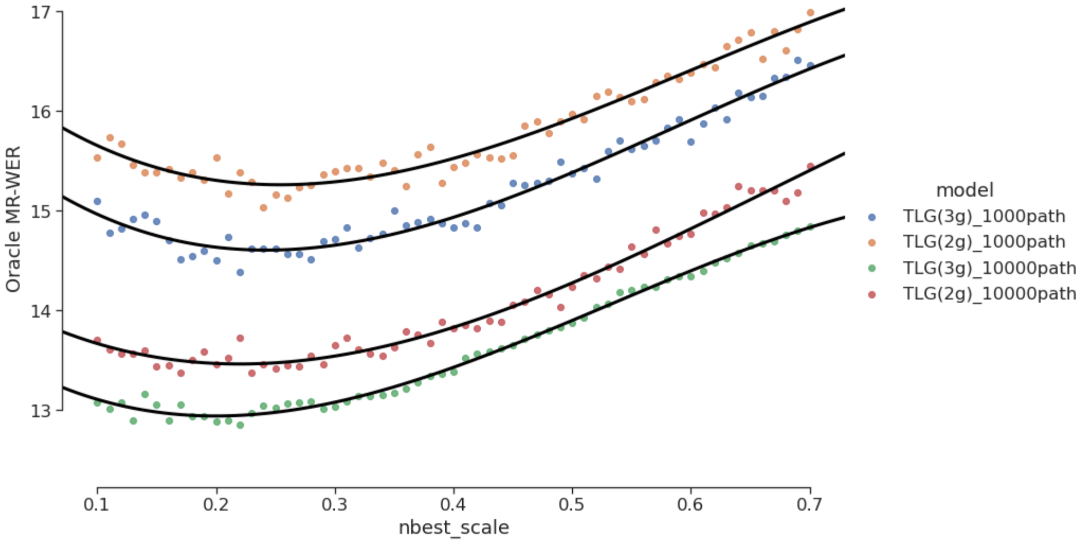

Fig. 14: Oracle MR-WER for all models against a range of `nbest_scales`

Whole-lattice Rescoring

With whole-lattice rescoring, we experimented the following combinations across various LM scale values:

- Using TLG(3 g) in the first pass, followed by rescoring with 4-gram
- Using TLG(2 g) in the first pass, followed by rescoring with 3-gram

- Using TLG(2 g) in the first pass, followed by rescoring with 4-gram

The results, as illustrated in Figure 15, reveal that rescoring with a 4-gram model offers minimal improvements. Surprisingly, the performance of a graph compiled with a bigram G is highly competitive, even when compared to a graph compiled with a trigram G, which is nearly twice the size. The marginal accuracy gains from trigram/4-gram LMs do not justify their computational overhead, particularly when considering the increased computational load. The effectiveness of the bigram G graph highlights the potential for optimizing ASR systems with simpler models without significantly compromising performance.

Attention Decoder

We evaluated the attention decoder rescoring method using the same hyperparameter combinations as those employed in whole-lattice rescoring. The `nbest_scale` was set to 0.2, as determined by `nbest_oracle`, while attention scales and `lm_scales` were varied. Figure 16 presents the MR-WER results for the "TLG(3g) + 4-gram G" model, demonstrating how varying `lm_scales` and `attention_scales` impact performance. The contour plot in Figure 16 visually maps the interaction between these two parameters, with the `lm_scale` on the x-axis and the `attention_scale` on the y-axis. The contour lines represent points of equal MR-WER values, providing insight into how these scaling factors influence accuracy. Warmer colors on the plot signify regions of lower error rates, indicating optimal parameter settings, while cooler colors correspond to higher error rates. This analysis highlights the critical balance between attention and language model scaling, necessary to achieve the best performance in ASR tasks.

Benchmarking and Comparison With the ESPNet-transformer Model

Table 8 summarizes the results achieved on the MGB3-dev dataset using different decoding approaches. Notably, the modified topology CTC-decoding method yielded an approximately 2% higher MR-WER than the regular CTC-decoding approach. Conversely, both whole-lattice and attention-decoder rescoring methods demonstrated substantial improvements in MR-WER. Additionally, it was observed that constructing TLG with larger n-gram G did not significantly enhance MR-WER, despite doubling the model size.

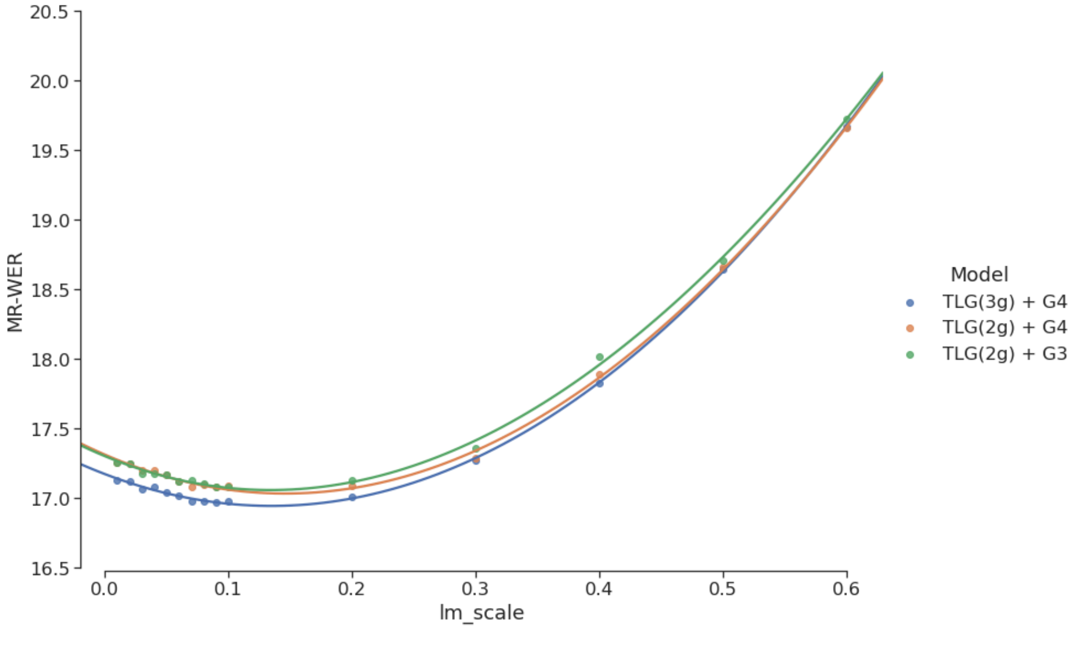

Fig. 15: Whole-lattice decoding MR-WERs for all models against a range of `lm_scales`

Table 8: Best results on MGB3-dev

Method	MR-WER%
CTC-decoding	18.81
CTC-decoding (modified topology)	20.98
Whole-lattice rescoring [TLG (2 g) + 4-gram LM]	16.97 (lm_scale=0.09)
Whole-lattice rescoring [TLG (2 g) + 3-gram LM]	17.08 (lm_scale=0.09)
Whole-lattice rescoring [TLG (2 g) + 4-gram LM]	17.08 (lm_scale=0.1)
Attention-decoder [TLG(3 g) + 4-gram LM + Attention]	16.33 (attention_scale=1.9, lm_scale=0.1)
Attention-decoder [TLG(2 g) + 3-gram LM + Attention]	16.5 (attention_scale=0.9, lm_scale=0.1)
Attention-decoder [TLG(2 g) + 4-gram LM + Attention]	16.45 (attention_scale=0.7, lm_scale=0.1)

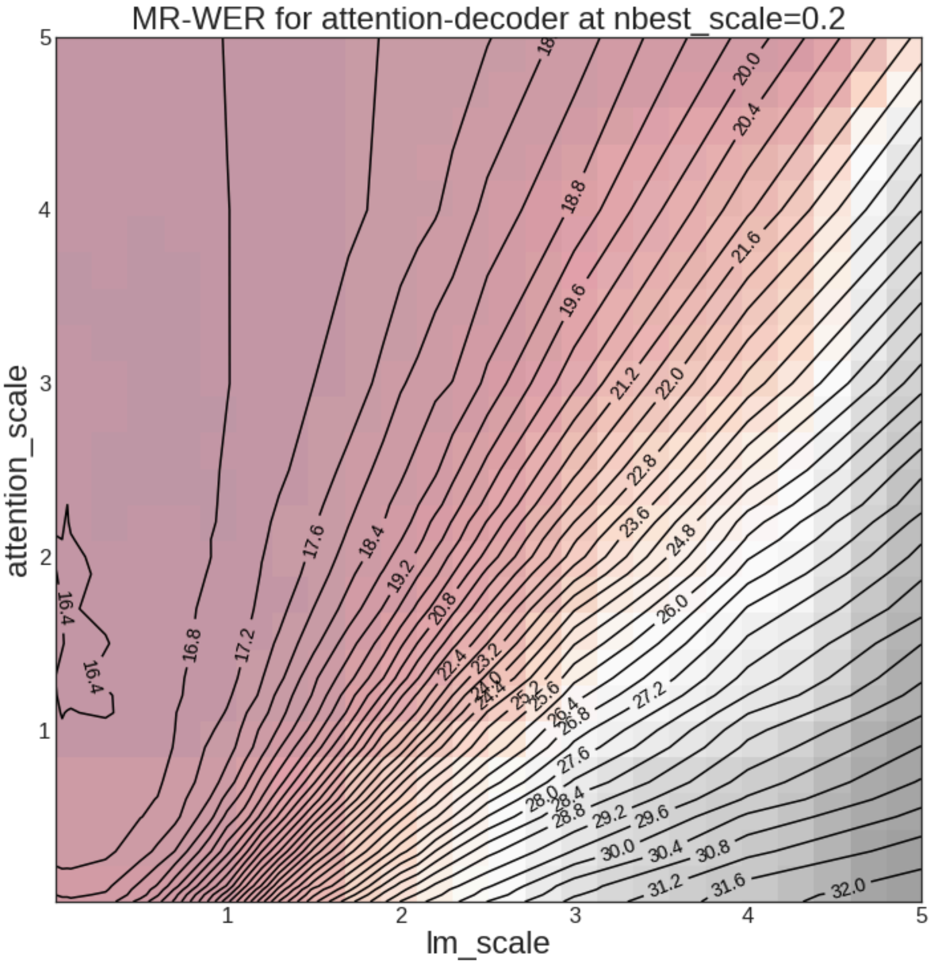

Fig. 16: Attention-decoder MR-WER for TLG(3 g) rescored with 4-gram models against ranges of `lm_scales` and `attention_scales`

Table 9: Comparison with the ESPNet-transformer model

Model	MR-WER	Inference speed test with 6-sec sample	
		CPU (I7 10700K)	GPU (RTX A6000)
ESPNet-transformer (trained with the same data)	14.18%	0.688 s	0.55 s
Attention-decoder [TLG(3 g) + 4-gram LM + Attention]	16.33%	0.138 s	0.048 s

Table 10: Results on the MASC test set

Dataset	ESPNet-model (Single-reference-WER)%	K2-model (Single-reference-WER)%
MASC test-clean	14.6	15.29
MASC test-noisy	28.3	31.86

As a final experiment, we trained an ESPNet transformer model (Watanabe et al., 2018) using the same quantity of data. Table 9 shows that the K2-based model, while still trailing by approximately 2% in WER, is 10 times faster, especially on a GPU, when tested on the MGB3-dev set. This speed advantage is crucial for real-time applications. And to assess the generalization ability the models, we tested them on another dataset, MASC (Al-Fetyani et al., 2021). Table 10 shows that the K2-based model continues to show consistent results, even though the difference between the two models' MR-WERs became lower than the difference in the case of the MGB3-dev set.

Memory Profile

As mentioned previously, memory is one of the most important concerns in graph-based decoding because of the size of the rich morphological language, as in Arabic. Table 11 presents a complete memory profile for our system's core components. This section provides a detailed analysis of the trade-off between model size and performance. Our analysis focuses on comparing two key configurations from our experiments:

- A high-accuracy configuration using a search graph built with a 3-gram language model, `TLG(3g)`, and rescored with a 4-gram LM

- An optimized configuration using a `TLG(2g)` graph and rescored with a 3-gram LM

As shown in Table 11, the `TLG(3g)` graph alone requires 11.6 GB of memory, while the `TLG(2g)` graph requires only 6.34 GB. This represents a 45% reduction in the search graph's size. When considering the full linguistic components required for decoding and rescoring, the high-accuracy system consumes $11.619GB$ (TLG) $+ 6.597GB$ (4-gram LM) $=$ $18.22GB$. This demonstrates a total memory saving of nearly 46% for the linguistic components. Critically, this substantial reduction in memory comes at a negligible cost to accuracy. As reported in Table 8, the MR-WER for the optimized system is 16.5%, compared to 16.33% for the high-accuracy system, a performance degradation of only 0.17%.

Table 11: Memory profile for different system elements

Item	Memory footprint after loading
Conformer-L model	1.695 G
4-gram LM (as FST)	6.597 G
3-gram LM (as FST)	3.475 G
TLG (3 g)	11.619 G
TLG (2 g)	6.34 G

Limitations

While this study demonstrates a significant advance in building efficient ASR models for Arabic, we acknowledge several limitations. First, our work focuses exclusively on Modern Standard Arabic and Egyptian Colloquial Arabic. The performance of our models on other major dialectal families, such as Levantine or Maghrebi Arabic, has not been evaluated and would require additional dialect-specific data. Second, our optimization efforts were concentrated on the linguistic components (TLG and n-gram LMs). Further memory and speed improvements could likely be achieved by applying model quantization or pruning techniques to the Conformer acoustic model itself, which was outside the scope of this work.

Conclusion

As the experiments showed, the K2-based model provides a good tradeoff between the WER and the inference speed in addition to the ability to enhance the model output by encoding the domain knowledge to an FST-based decoder. Throughout this work, we provided a method to optimize the construction of such FST-based decoders for the Arabic language. Our model shows results that are comparable to those of state-of-the-art models with faster decoding speeds. Along with this, we succeeded in reducing the memory footprint of a 1.5M vocabulary TLG to 6.34 GB with almost unnoticed WER degradation, which provides a method for deploying the ASR system on low-spec computing machines. For future work, we aim to investigate more advanced rescoring techniques, such as integrating neural language models, to further improve performance without significantly increasing computational demands.

Acknowledgment

The authors gratefully acknowledge the generous support of the Applied Innovation Center (AIC) at the Egyptian Ministry of Communication and Information Technology. The provision of necessary resources by the AIC was crucial for the successful completion of this study.

Funding Information

This work was supported by the Applied Innovation Center (AIC) at the Egyptian Ministry of Communication and Information Technology.

Authors Contributions

Wael A. Sultan: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition.

Mourad S. Semary and Sherif M Abdou: Writing - Review & Editing.

Ethics

This manuscript is an original work. The corresponding author certifies that all co-authors have reviewed and approved the final version of the manuscript. No ethical concerns are associated with this submission.

References

Al-Fetyani, M., Al-Barham, M., Abandah, G., Alsharkawi, A., & Dawas, M. (2023). MASC: Massive Arabic Speech Corpus. *2022 IEEE Spoken Language Technology Workshop (SLT)*, 1-6. https://doi.org/10.1109/slt54892.2023.10022652

Ali, A., Bell, P., Glass, J., Messaoui, Y., Mubarak, H., Renals, S., & Zhang, Y. (2016). The MGB-2 challenge: Arabic multi-dialect broadcast media recognition. *Proceeding of the IEEE Spoken Language Technology Workshop (SLT)*, 279-284. https://doi.org/10.1109/slt.2016.7846277

Ali, A., Magdy, W., Bell, P., & Renais, S. (2015). Multi-reference WER for evaluating ASR for languages with no orthographic rules. *Proceeding of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 576-580. https://doi.org/10.1109/asru.2015.7404847

Ali, A., Vogel, S., & Renals, S. (2017). Speech recognition challenge in the wild: Arabic MGB-3. *Proceeding of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 316-322. https://doi.org/10.1109/asru.2017.8268952

Collobert, R., Puhrsch, C., & Synnaeve, G. (2016). An end-to-end convnet-based speechrecognition system. *ArXiv:1609.03193*, *2*(1), 1-8. https://doi.org/10.48550/arXiv.1609.03193

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the 23rd International Conference on Machine Learning*, 369-376. https://doi.org/10.1145/1143844.1143891

Hannun, A. (2021). The history of speech recognition to the year 2030. *ArXiv*, arXiv:2108.00084. https://doi.org/10.48550/arXiv.2108.00084

Hannun, A., Case, Carl., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., & Coates, A. (2014). Deep speech: Scaling up end-to-end speech recognition. *ArXiv*. https://doi.org/10.48550/arXiv.1412.5567

Hannun, A., Pratap, V., Kahn, J., & Ning Hsu, W. (2020). Differentiable Weighted Finite-State Transducers. *ArXiv*. https://doi.org/10.48550/arXiv.2010.01003

Hussein, A., Watanabe, S., & Ali, A. (2022). Arabic speech recognition by end-to-end, modular systems and human. *Computer Speech & Language*, *71*, 101272. https://doi.org/10.1016/j.csl.2021.101272

Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 66-71. https://doi.org/10.18653/v1/d18-2012

Laptev, A., Majumdar, S., & Ginsburg, B. (2022). CTC Variations Through New WFST Topologies. *Interspeech 2022*, 1041-1045. https://doi.org/10.21437/interspeech.2022-10854

Li, J. (2022). Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, *11*(1), e8. https://doi.org/10.1561/116.00000050

Miao, Y., Gowayyed, M., & Metze, F. (2015). EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. *Proceeding of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 167-174. https://doi.org/10.1109/asru.2015.7404790

Mohamed, A., Dahl, G. E., & Hinton, G. (2012). Acoustic Modeling Using Deep Belief Networks. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(1), 14-22. https://doi.org/10.1109/tasl.2011.2109382

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, *23*(2), 269-311.

Povey, D., & Woodland, P. C. (2002). Minimum phone error and I-smoothing for improved discriminative training. *IEEE Xplore*, 105-108. https://doi.org/10.1109/ICASSP.2002.5743665

Povey, D., Zelasko, P., & Khudanpur, S. (2021). Speechrecognition with next-generation kaldi (k2,lhotse, icefall. *Interspeech: Tutorials*. International Speech Communication Association (ISCA), Brno.

Snyder, D., Chen, G., & Povey, D. (2015). Musan: A music, speech, and noise corpus. *ArXiv*. https://doi.org/10.48550/arXiv.1510.08484

Wang, D., Wang, X., & Lv, S. (2019). An Overview of End-to-End Automatic Speech Recognition. *Symmetry*, *11*(8), 1018. https://doi.org/10.3390/sym11081018

Wang, W., Wang, G., Bhatnagar, A., Zhou, Y., Xiong, C., & Socher, R. (2020). An Investigation of Phone-Based Subword Units for End-to-End Speech Recognition. *Interspeech 2020*, 2642-2646. https://doi.org/10.21437/interspeech.2020-1873

Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Enrique Yalta Soplin, N., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., & Ochiai, T. (2018). ESPnet: End-to-End Speech Processing Toolkit. *Interspeech 2018*, 2207-2211. https://doi.org/10.21437/interspeech.2018-1456

Xiang, H., & Ou, Z. (2019). CRF-based Single-stage Acoustic Modeling with CTC Topology. *Proceeding of Then 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5676-5680. https://doi.org/10.1109/icassp.2019.8682256