Unifying Global and Near-Context Biasing in a Single Trie Pass

Iuliia Thorbecke[1,2(✉)], Esaú Villatoro-Tello[1(✉)], Juan Pablo Zuluaga[1], Shashi Kumar[1,3], Sergio Burdisso[1], Pradeep Rangappa[1], Andrés Carofilis[1], Srikanth Madikeri[2], Petr Motlicek[1,4], Karthik Pandia[5], Kadri Hacioğlu[5], and Andreas Stolcke[5]

[1] Idiap Research Institute, Martigny, Switzerland
{iuliia.nigmatulina,evillatoro}@idiap.ch
[2] University of Zurich, Zurich, Switzerland
[3] EPFL, Lausanne, Switzerland
[4] Brno University of Technology, Brno, Czech Republic
[5] Uniphore, Palo Alto, USA

Abstract. Despite the success of end-to-end automatic speech recognition (ASR) models, challenges persist in recognizing rare, out-of-vocabulary words—including named entities (NE)—and in adapting to new domains using only text data. This work presents a practical approach to address these challenges through an unexplored combination of an NE bias list and a word-level n-gram language model (LM). This solution balances simplicity and effectiveness, improving entities' recognition while maintaining or even enhancing overall ASR performance. We efficiently integrate this enriched biasing method into a transducer-based ASR system, enabling context adaptation with almost no computational overhead. We present our results on three datasets spanning four languages and compare them to state-of-the-art biasing strategies We demonstrate that the proposed combination of keyword biasing and n-gram LM improves entity recognition by up to 32% relative and reduces overall WER by up to a 12% relative.

Keywords: Contextualisation and adaptation of ASR · real-time ASR · Aho-Corasick algorithm · Transformer-Transducer

1 Introduction

Traditional methods for incorporating contextual text data can be broadly categorized into two approaches: (1) integrating the information during decoding without modifying the ASR architecture, such as *rescoring* and *shallow fusion* (SF) [2,10,11,13,14,22,30,32,35], and (2) training the model to accept contextual input dynamically when needed [12,23,24].

This work was supported by Idiap Research Institute and Uniphore collaboration project. Part of this work was also supported by EU Horizon 2020 project ELOQUENCE (grant number 101070558).

K. Ekštein et al. (Eds.): TSD 2025, LNAI 16029, pp. 170–181, 2026.
https://doi.org/10.1007/978-3-032-02548-7_15

We focus on the first group of methods, which can be applied to any ASR model and are considered the least costly methods of context integration.

SF means log-linear interpolation of the score from the E2E model with an external contextual LM at each step of the beam search:

$$y^* = \text{argmax} \log P(y|x) + \lambda \log P_C(y), \tag{1}$$

where $P(y|x)$ is the probability of sequence y predicted by the E2E model based on the input acoustic feature x. $P_C(y)$ is an in-domain or context-biased *contextual* LM and λ is a hyperparameter to control the impact of the contextual LM on the overall model score [14,35]. For the speed and convenience of decoding, contextual information is usually presented as a graph, such as an n-gram LM as a weighted finite-state transducer (WFST) [2,20,21] or a bias list as a prefix trie [13], and a string-matching algorithm is usually used to detect relevant context entities in the hypotheses.[1]

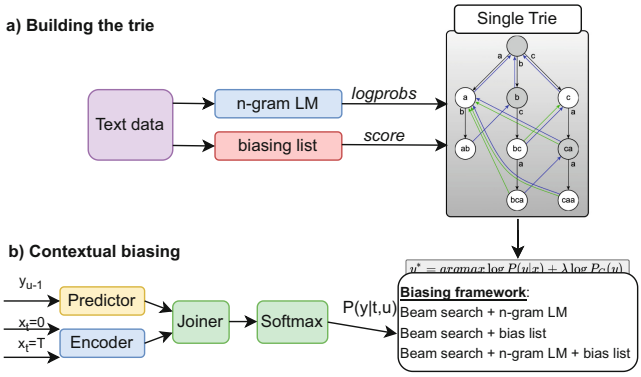

Fig. 1. Proposed biasing approaches at beam search time with Aho-Corasick string matching algorithm, including biasing list and n-gram LM statistics.

In previous studies on SF, typically either the effect of an external LM or a keyword bias list is investigated. In [14], subword-level and grapheme-level neural network language models (NNLMs) are used for rescoring. In [11], the authors combine the statistics over characters and words by rescoring with character- and word-level recurrent NNLMs (RNNLMs) at the same time. Both approaches, however, are unsuitable for online recognition and fast context adaptation due to the size of the NNLMs. In [30], the n-gram word-level LM is adopted by converting subwords to words before rescoring; this allows on-the-fly decoding but, as a character-level sequence can be mapped to different word sequences, during the decoding, the authors first have to convert character-level partial hypothesis into a word lattice.

On the other hand, SF with a list of target entities [13,17] can bias hypotheses towards particular words or NEs, such as proper names, terminology, or geographical

[1] https://github.com/kensho-technologies/pyctcdecode.

names. Context keywords can be integrated directly with the *keyword prefix trie* search algorithm without a need to train any LM [13]. Keyword biasing (KB), however, can lead to a degradation of the overall performance. Zhao et al. [35] improve SF with biasing at the subword unit level instead of word level and use a set of common prefixes (*"call"*, *"text"*) to avoid irrelevant bias.

In our approach, we enhance KB while maintaining robust overall performance by leveraging *"near-context"* n-grams from an n-gram LM alongside the *"global-context"* entities. Concretely, we unify existing SF methods by integrating a word-level language model with the SF of context entities within a *single trie pass*. First, we construct a lightweight n-gram LM by dynamically converting words into subwords *on-the-fly* and adjusting LM weights to correspond to the new subword units. Next, keyword weights are modified within the same trie, allowing for seamless integration of both LM n-grams and KB.

This approach extends the KB method from [10] by merging keyword information with LM n-grams to form a unified context graph (Fig. 1). We employ the Aho-Corasick (AC) algorithm for string matching, which efficiently handles search failures in the trie by managing mismatched cases. Given its proven efficiency in NLP tasks [29] and prior success in ASR KB [10], we demonstrate its effectiveness for SF as well, ensuring robust and efficient context integration during decoding.

In sum, our main contributions are as follows: *(i)* efficient contextual adaptation using a word-level n-gram LM and KB within a *single context trie*; *(ii)* comprehensive analysis of performance across named entities (NEs), out-of-vocabulary (OOV) NEs, and real-time factor (RTF) metrics; (iii) extending the use of the AC algorithm to integrate word-level n-gram LMs, previously applied only for KB; and *(iv)* benchmarking multiple SF strategies across four languages and three datasets.

2 Methodology

2.1 Preliminaries: The Aho-Corasick Algorithm

The AC algorithm proposed by [1] is a text pattern-matching algorithm that operates in linear time. It utilizes three data structures to represent the search set, or *transition diagram*: a trie, an output table, and a failure function. The output table stores suffixes reachable from any node that corresponds to the target search strings. The failure function handles cases where some search strings are suffixes of others [19]. For example, if a trie contains the word "CAN" but not CAT", a failure transition will backtrack "CAT" to the prefix "**CA-**" from "CAN" upon a mismatch. The algorithm is implemented as a finite state machine with backoff arcs that not only return to the root or a lower-order n-gram (e.g., from a 4-gram to a 3-gram) when a string end is reached, but also perform *failure transitions* to backtrack upon a mismatch. This mechanism allows partial match detection while keeping the trie's structure sparse, improving efficiency by avoiding repeated prefix matches.[2]

[2] Our work is based on the existing Aho-Corasick implementation available in the k2/icefall framework (https://github.com/k2-fsa/icefall/blob/master/icefall/context_graph.py.

2.2 Word-Level n-Gram LM for Transducer-Based ASR

One of the key contributions of our approach is the integration of a *word-level* n-gram language model, henceforth referred to as WL-3gramLM, into the Transducer architecture, a non-straightforward task given that the model operates at the BPE unit level [27]. Unlike traditional SF methods that rely on subword-level LMs, our proposed solution enables direct incorporation of a word-level n-gram LM, leveraging its richer word-level statistics and improved transparency for adaptation and biasing.

To achieve this, we propose an enhanced SF approach that utilizes the AC algorithm to construct a prefix trie directly from LM n-grams and their corresponding log probabilities. Since the ASR model produces hypotheses at the BPE level, we first convert the WL-3gramLM n-grams into sequences of BPE units using SentencePiece.[3] During decoding, when a hypothesis string matches a stored n-gram in the context trie, we incorporate the LM probability score as an additional bias score into the log probability of the matched candidate. Algorithm 1 illustrate how the LM probability score (`bias_score`) is obtained before the context trie is created.

A key challenge in integrating an ARPA-formatted n-gram LM is determining how to convert its word-level scores into a bias score (bonus) that can be effectively combined with the ASR model's subword-level log-probability scores. An n-gram LM in ARPA format provides scores as log-base-10 probabilities, $s_w = \log_{10}(P_{LM}(w))$. Applying s_w directly as a bias cost would lead to poorly scaled biasing. However, through empirical evaluation on our development sets, we found that a non-linear transformation of the LM score yielded superior performance. We systematically compared several functions and discovered that the most effective bias score is derived by directly exponentiating the original log-base-10 score:

$$\text{bias\_score}(w) = \exp(s_w) = \exp(\log_{10}(P_{LM}(w))) \tag{2}$$

This transformation is mathematically equivalent to $P_{LM}(w)^{\frac{1}{\ln(10)}}$, which is approximately $P_{LM}(w)^{0.434}$. This effectively compresses the dynamic range of the original LM probabilities, emphasizing higher probabilities less aggressively than a direct linear scaling, and sharply diminishing the influence of less probable ones. While the exponent $1/\ln(10)$ arises naturally from converting a log-base-10 probability into the argument of the natural exponential function, we acknowledge that this implicit compression factor could be considered an arbitrary choice and, in a more generalized framework, might be treated as a tunable hyperparameter γ (e.g., $P_{LM}(w)^{\gamma}$). However, for the scope of this work, this specific transformation proved robust and effective across our datasets.

Crucially, the bias_score(w) obtained through this transformation is a value in *probability space*, not log-probability space. When this score is applied, it is added to the log-probability of the ASR model's hypothesis. Specifically, we treat this bias_score(w) as a positive bonus that directly increases the overall log-likelihood of the matched word sequence during beam search. This unconventional direct addition of a probability-space value to a log-probability serves as an aggressive non-linear scaling, heavily

[3] https://github.com/google/sentencepiece.

rewarding high-probability n-grams while allowing for more nuanced integration than a simple linear addition in the log-domain.

A second implementation challenge is to apply this word-level bias_score(w), to a word w that has been tokenized into a sequence of subwords $(b_1, b_2, \ldots, b_k)$. We explored two strategies: (a) distributing the bonus evenly across all subwords (i.e., adding bias_score$(w)/k$ at each subword step) and (b) applying the entire bonus at a single point. Our experiments showed that the latter approach was more effective. Specifically, the entire bonus is added to the hypothesis score upon the emission of the final subword, b_k, of the word w.

To further analyze the impact of word-level statistics on subword-level decoding, we evaluate the model's ability to recognize OOV named entities (Sect. 3.2). The AC algorithm's capability to efficiently locate partial matches proves especially beneficial for improving OOV word recognition, reinforcing the advantage of our word-level LM integration over conventional subword-only approaches.

2.3 Unifying n-Gram LM and KB via AC

In this section, we present the integration of a word-level n-gram LM with KB during the decoding process. Specifically, we introduce a novel implementation strategy using a *single context trie* to combine LM n-grams with target keywords, while maintaining traversal at the BPE level.

During KB in decoding, a fixed positive score is added to the log probability of a candidate in the hypothesis when it matches any of the context entities. To facilitate this, we employ a trie structure derived from the AC algorithm, which enables efficient lookup and biasing. This trie contains both LM n-grams and the KB information.

The integration of keywords with LM n-grams is described in Algorithm 1. First, all n-grams from the LM are inserted into the trie with their respective scores. Then, for each keyword, a *bias score* is applied depending on whether the keyword's n-gram is present in the LM (*"inLM"*) or not (*"outLM"*). If the keyword n-gram is already present in the LM, its bias score is adjusted based on its associated LM score, while for keywords absent in the LM, a fixed bias score is applied.

The bias score is tuned on the development set, with different values for when the keyword n-gram is in the LM versus when it is not:

$$final\_bias\_score = \begin{cases} \alpha_{outLM} & \text{if NE is not in LM,} \\ exp(LMs) + \alpha_{inLM} & \text{if NE is in LM,} \\ exp(LMs) & \text{other words in LM.} \end{cases}$$

where LMs is the LM score, and α represents the fixed bias score. We conducted a grid search to determine the optimal values of the α hyperparameters. We evaluated different biasing scores for both cases: $\alpha_{inLM} \in \{0.5, 1.0, 1.5, 2.0\}$ and $\alpha_{outLM} \in \{0.5, 1.0, 1.5, 2.0\}$. Our results consistently show that the combination of $\alpha_{inLM} = 0.5$ and $\alpha_{outLM} = 1.5$ achieves the best performance across all experimental settings.

Algorithm 1: Compute Context Scores from ARPA and Keyword Files

```
Function context_file, arpa_file, in_lm_cost, not_in_lm_cost:
    Initialize empty list contexts
    Initialize empty dictionary contexts_scores
    Initialize empty dictionary keywords
    foreach line in context_file do
        keywords[line.strip()] ← "not in the lm"
    end
    foreach line in arpa_file do
        Split line into [weight, ngram]
        Append ngram to contexts
        bias_score ← e^float(weight)
        if ngram in keywords then
            keywords[ngram] ← "in the lm"
            contexts_scores[ngram_ids] ← bias_score + in_lm_cost
        end
        else
            contexts_scores[ngram_ids] ← bias_score
        end
    end
    foreach (entity, label) in keywords.items() do
        if label is "not in the lm" then
            Append entity to contexts
            contexts_scores[ngram_ids] ← not_in_lm_cost
        end
    end
    context_trie ← ContextTrie(contexts_scores)
    context_trie.build(SentencePiece.encode(contexts))
```

3 Experimental Setup

3.1 Data

Dataset Description. Our experiments require keyword lists for biasing in addition to audio and transcripts. Publicly available test sets meeting this criterion are limited, and mostly in English. We evaluate our biasing approach on one private dataset (DefinedAI[4]) covering banking, insurance and healthcare domains, and two public datasets: Earnings21[5] (stock market) [26] and CommonVoice [3]; see Table 1. DefinedAI provides manually annotated NEs. While Earnings21 includes two biasing lists from NER,[6] CommonVoice lacks gold NEs, requiring us to generate our own bias lists.

Biasing Lists for CommonVoice. We generate bias lists using HuggingFace BERT models [18,33] fine-tuned for NER per language.[7] Extracted NEs are filtered to remove unigrams, reducing noise. See Table 1 for statistics. Experiments on CommonVoice assess (1) SF performance on languages other than English (EN) and (2) the effect of large biasing lists, e.g., the German (DE) subset has 2k unique entities. Notably, DE and EN contain more unique NEs than French (FR) and Spanish (ES).

[4] Website: https://www.defined.ai.

[5] Audio split into 3-minute segments for decoding as in [6].

[6] We use only the *oracle* list from [6].

[7] Models: ES: https://www.mrm8488/bert-spanish-cased-finetuned-ner, EN: https://www.dslim/bert-base-NER-uncased, FR: https://www.cmarkea/distilcamembert-base-ner, DE: https://www.fhswf/bert_de_ner.

Table 1. Test sets statistics with context information. †Number of utterances with at least one named entity.

Test set	Size	Duration	Biasing entities		
		(hours)	unique	NE-utts†	OOV
DefinedAI	2K utt.	6	367	486	16
Earnings21	18K utt.	39	1013	-	-
CV-EN	16K utt.	27	1173	1125	169
CV-DE	16K utt.	27	1985	1906	304
CV-FR	16K utt.	26	600	549	225
CV-ES	15.5K utt.	26	122	135	15

3.2 Base ASR Model

Transformer-Transducer Training. For all experiments, we utilize the stateless version of the Zipformer transducer model [7,34]. For evaluation on the DefinedAI and Earnings21 test sets, we employ the pretrained Zipformer model trained on the Gigaspeech-XL dataset [4].[8] This choice is justified because the Earnings21 dataset provides only test data, while for DefinedAI, we have access to limited training data (40 h). Our choice of the model aligns with prior work on Earnings21, where the authors also used the Gigaspeech dataset for training [6]. This setup can be viewed as a domain adaptation scenario, given that both DefinedAI and Earnings21 consist of domain-specific data.

For experiments on CommonVoice, we train Zipformer models for each language using the corresponding training set. Training is performed from scratch using the latest Icefall Transducer recipe with its default hyperparameters. This includes the *ScaledAdam* optimizer [15] and a learning rate scheduler with a 500-step warmup phase [31], followed by a decay phase determined by the total number of steps (7,500) and epochs (3.5) [34]. The neural Transducer model is jointly optimized using an interpolated loss function, combining simple and pruned RNN-T loss [8,16] with CTC loss [9] ($\lambda = 0.1$). The peak learning rate is set to 5.0×10^{-2}, and each model is trained for 30 epochs on a single RTX 3090 GPU.

3.3 Experiments and Evaluation

For all experiments, we utilize their respective Zipformer-Transducer models (70M), with variations occurring solely in the decoding strategies. We benchmark our proposed approach against established methods for integrating external context during decoding.

Baselines. We implement three baseline methods, ranging from standard beam search to advanced SF (**SF**) techniques incorporating neural network language models (NNLMs) for contextual adaptation. • **BeamS**: Standard beam search decoding without

[8] Gigaspeech-XL: 10k hours of transcribed audio data, model: https://www.yfyeung/icefall-asr-gigaspeech-zipformer-2023-10-17.

Table 2. Biasing experimental results on DefinedAI and Earnings21 datasets. Results are reported in terms of WER and NE recognition accuracy (NE-a). Best performance appears in **bold** font, the second-best result appears underlined. †The result is statistically significant with $p < 0.07$.

Decoding type	Context source	DefinedAI		Earnings21	
		WER (↓)	NE-a (↑)	WER (↓)	NE-a (↑)
BeamS	n/a	10.4	68.0	14.4	59.5
SF	NNLM	10.2	69.3	14.9	58.5
	BPE-5gramLM	10.2	68.2	16.8	59.0
SF + AhoC	WL-3gramLM	**10.0**	70.0	**12.9**	61.7
	KB	10.4	**77.9**†	16.7	**63.5**†
	WL-3gramLM + KB	**10.0**†	73.3†	13.1†	62.8†

any external contextual information (i.e., no SF). • **NNLM**: A conventional SF approach integrating a Transformer-based NNLM. • **BPE-5gramLM**: A SF setup using a 5-gram LM trained at the BPE level.

Context Biasing with Aho-Corasick. (**SF+AhoC**.) We evaluate three variations of our proposed contextual adaptation method, leveraging the AC algorithm for efficient integration of external context. • **WL-3gramLM**: SF with a 3-gram word-level LM, integrated via an AC-based trie for fast lookup and retrieval. • **KB**: Context biasing using an AC-based trie for keyword matching, without an explicit LM. • **WL-3gramLM+KB**: a unified fusion approach that integrates a 3-gram word-level LM and a KB list through a single AC-based decoding trie pass.

LMs. For training the NNLM used in our baselines, we train Transformer-based [31] LMs. We use GigaSpeech-XL for DefinedAI and Earnings21, while for CommonVoice, we use the corresponding language-specific training sets. Each Transformer LM is trained for 10 epochs and consists of approximately 38M parameters.[9] Regarding the BPE-5gramLM, we train a 5-gram BPE-based LM using the SRILM [28] toolkit.
Similarly, for the word-level LMs required for the **SF+AhoC** experiments, we train 3-gram word-level LMs with SRILM. To construct these models, we use text data from the corresponding training sets for all test sets except Earnings21. For Earnings21, we instead use transcriptions from Earnings22 [5], a dataset from the same domain, ensuring domain consistency.

Evaluation. In addition to the word error rate (WER) metric, we evaluate the accuracy only on NEs (*NE-a*). To calculate the NE metrics, only strings containing NEs in the references are taken into account. Accuracy is calculated in a binary manner: "yes" – when the NE is completely recognized correctly, "no" – when at least one error occurs within the NE. For NE-A results with KB and WL-3gramLM+KB experiments and the WER results with WL-3gramLM+KB experiments, we also report statistical

[9] Transformer-LM recipe: https://github.com/k2-fsa/icefall/tree/master/icefall/transformer_lm.

Table 3. Results of SF experiments on 4 languages (English, German, French, Spanish) of CommonVoice reported in terms of WER, recognition accuracy of NEs (NE-A) and OOV (OOV-A) words. Recent Whisper (small-S and medium-M) performance is given as a reference [25]. Best performance appears in **bold** font, while the second-best result appears underlined. †The result is statistically significant with $p < 0.07$.

Decoding type	Context source	WER(↓)	NE-A(↑)	OOV-A(↑)	WER(↓)	NE-A(↑)	OOV-A(↑)
		CommonVoice-English			**CommonVoice-German**		
Whisper-S (244M)/Whisper-M (769M)		14.5/11.2	-	-	13.0/8.5	-	-
BeamS	n/a	13.5	45.2	6.5	7.7	55.5	19.4
SF	NNLM	13.3	46.7	6.0	7.6	55.5	18.8
	BPE-5gram-LM	13.3	45.4	6.6	7.6	55.8	20.4
SF + AhoC	WL-3gram-LM	13.3	46.9	5.9	7.6	57.0	19.7
	Keyword boosting (KB)	13.7	**69.3**†	**36.0**	7.7	**79.0**†	**53.4**
	WL-3gram-LM + KB	**13.2**†	59.3†	24.1	**7.4**†	70.8†	39.7
		CommonVoice-French			**CommonVoice-Spanish**		
Whisper-S (244M)/Whisper-M (769M)		22.7/16.0	-	-	10.3/6.9	-	-
BeamS	n/a	10.0	35.6	12.4	7.8	67.7	20.0
SF	NNLM	**9.8**	36.1	11.5	7.7	67.6	21.4
	BPE-5gram-LM	**9.8**	35.8	12.9	**7.6**	66.9	23.1
SF + AhoC	WL-3gram-LM	**9.8**	37.3	13.2	**7.6**	66.9	18.8
	Keyword boosting (KB)	9.9	**69.8**†	**53.9**	7.8	**86.8**†	**58.3**
	WL-3gram-LM + KB	**9.8**†	53.6†	33.0	**7.6**†	80.2†	35.7

significance tests against the baseline (BeamS). For WER evaluation on Earnings21, we use the *fstalign tool*[10] and for NE accuracy we used only "PERSON" and "ORG" categories. To evaluate OOV performance, we measured the recognition accuracy of OOV-NEs for German and French, as their datasets have the highest numbers of OOV-NEs (see the last column in Table 1).

As fast and flexible context integration is critical in many practical scenarios, we include an estimate of decoding time using the *inverse real-time factor (RTFX)*, which is the ratio between the length of the processed audio and the decoding time. RTFX is measured on the DefinedAI test set with one RTX 3090 GPU.

4 Results

The Impact of Word-Level n-Grams. To distinguish between out-of-domain and in-domain performance, we present results on DefinedAI and Earnings21 (Table 2) separately from CommonVoice (Table 3) experiments. For both setups, the results of fusion n-gram LM into a context trie (i.e., WL-3gramLM) lead to relative WER reduction w.r.t.

[10] Provided by authors of [26] as the dataset references are in a special NLP-format: https://github.com/revdotcom/fstalign.

Table 4. Ablation of decoding speed (RTFX; higher better).

Decoding type	Context source	RTFX(↑)
BeamS	n/a	**120.7**
SF	BPE-5gramLM	77.8
SF + AhoC	WL-3gramLM	111.1
	KB	113.5
	WL-3gramLM + KB	<u>117.3</u>

beam search alone: 3.8% for DefinedAI, 10.4% for Earnings21 (see Table 2), 1.5%, 1.3%, 2%, and 2.6% for EN, DE, FR, and ES from CommonVoice respectively (see Table 3). Moreover, for the out-of-domain sets, it improves the performance compared to the fusion with Transformer-LM (NNLM): i.e., from 10.2 to 10.0 for DefinedAI and from 14.9 to 12.9 for Earnings21. In addition to improved accuracy, training n-gram LMs is fast and easy and can be done even with a small corpus (e.g., 40 h in the case of DefinedAI), making it suitable for low-resource scenarios.

Trade-Off Between NE-Accuracy and WER. The biggest improvement on NEs is always achieved with the KB setup: 14.6% for DefinedAI and 6.7% for Earnings21 of relative improvement w.r.t. BeamS baseline (Table 2). However, the overall WER does not improve or even degrades, e.g., to 16.7 WER in the case of Earnings21. The overall WER is improved when KB is combined with word-level n-gram LM (WL-3gramLM + KB): relative improvement w.r.t. KB alone is by 3.8%, 21.6%, 3.6%, 3.9%, 1%, and 2.6% for DefinedAI, Earnings21, and CommonVoice EN, DE, FR and ES, respectively. It is important to note that the datasets vary in the number of NEs and the proportion of utterances containing them (see Table 1). This variation explains the smaller impact of fusion on WER for the FR and ES sets, which contain the fewest NEs. Nevertheless, even in these cases, our approach maintains WER performance without degradation.

RTFX. The RTFX results in Table 4 show that decoding with *WL-3gramLm + KB* is only a bit slower compared to BeamS approach alone and thus it can be used on-the-fly: RTFX of the *WL-3gramLm + KB* method is 6.0% lower than the *baseline* (BeamS) and there is no degradation w.r.t *KB*. Finally, while the WER performance of BPE-5gramLM and WL-3gramLM is generally comparable across datasets, a key distinction lies in their computational overhead. WL-3gramLM is 31% faster than BPE-5gramLM, with an RTFX of 77.8 compared to 111.1, offering a significant advantage for production-level ASR solutions.

OOV Performance. The incorporation of word-level statistics via WL-3gramLM does not adversely affect OOV recognition at the subword level (as indicated by the **OOV-A** column in Table 3). Despite the KB yielding optimal performance, it may not be the most suitable option if overall WER is a concern. Similar to named entities, the WL-3gramLM+KB configuration strikes an effective balance, preserving overall ASR performance while enhancing OOV accuracy by up to 51% for the German dataset. These outcomes substantiate that, even with the integration of word-level data into the rescoring process, the benefits of fusion extend to unseen words, given that the biasing occurs at the subword level.

5 Conclusion

We present an efficient approach for integrating word-level n-gram LMs with a Transformer Transducer ASR model using SF and a trie based on the Aho-Corasick algorithm. The method enables faster decoding than standard SF maintaining competitive WER and real-time factor (RTF) efficiency. Additionally, we show that combining n-gram LMs with KB in a unified context graph improves overall WER and NE accuracy, including for OOV words. Experiments across four languages and three datasets confirm the effectiveness of our approach in both in-domain and out-of-domain scenarios.

References

1. Aho, A.V., Corasick, M.J.: Efficient string matching: An aid to bibliographic search. Commun. ACM **18**(6), 333–340 (1975)
2. Aleksic, P., et al.: Bringing contextual information to Google speech recognition. In: Interspeech, pp. 468–472 (2015)
3. Ardila, R., et al.: Common voice: A massively-multilingual speech corpus. In: LREC, pp. 4218–4222. European Language Resources Association, Marseille, France (2020)
4. Chen, G., et al.: Gigaspeech: An evolving, multi-domain ASR corpus with 10,000 hours of transcribed audio. In: Proc. Interspeech 2021 (2021)
5. Del Rio, M., Ha, P., McNamara, Q., Miller, C., Chandra, S.: Earnings-22: A practical benchmark for accents in the wild. arXiv preprint arXiv:2203.15591 (2022)
6. Drexler Fox, J., Delworth, N.: Improving contextual recognition of rare words with an alternate spelling prediction model. In: Interspeech, pp. 3914–3918 (2022). https://doi.org/10.21437/Interspeech.2022-10991
7. Ghodsi, M., Liu, X., Apfel, J., Cabrera, R., Weinstein, E.: RNN-Transducer with stateless prediction network. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7049–7053. IEEE (2020)
8. Graves, A.: Sequence transduction with recurrent neural networks. arXiv preprint arXiv:1211.3711 (2012)
9. Graves, A., Graves, A.: Connectionist temporal classification. Supervised sequence labelling with recurrent neural networks, pp. 61–93 (2012)
10. Guo, Y., Qiu, Z., Huang, H., Siong, C.E.: Improved keyword recognition based on Aho-Corasick automaton. In: 2023 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE (2023)
11. Hori, T., Watanabe, S., Hershey, J.R.: Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition. In: 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 287–293. IEEE (2017)
12. Jain, M., Keren, G., Mahadeokar, J., Zweig, G., Metze, F., Saraf, Y.: Contextual RNN-T for open domain ASR. In: Interspeech (2020)
13. Jung, N., Kim, G., Chung, J.S.: Spell my name: keyword boosted speech recognition. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6642–6646. IEEE (2022)
14. Kannan, A., Wu, Y., Nguyen, P., Sainath, T.N., Chen, Z., Prabhavalkar, R.: An analysis of incorporating an external language model into a sequence-to-sequence model. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5828. IEEE (2018)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2014)

16. Kuang, F., et al.: Pruned RNN-T for fast, memory-efficient ASR training. In: Interspeech (2022)
17. Le, D., Keren, G., Chan, J., Mahadeokar, J., Fuegen, C., Seltzer, M.L.: Deep shallow fusion for RNN-T personalization. In: Spoken Language Technology Workshop (SLT), pp. 251–257. IEEE (2021)
18. Lhoest, Q., et al.: Datasets: A community library for natural language processing. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 175–184 (2021)
19. Meyer, B.: Incremental string matching. Inf. Process. Lett. **21**(5), 219–227 (1985)
20. Mohri, M., Pereira, F., Riley, M.: Weighted finite-state transducers in speech recognition. Comput. Speech Lang. **16**(1), 69–88 (2002)
21. Nigmatulina, I., et al.: Implementing Contextual Biasing in GPU Decoder for Online ASR. In: Interspeech, pp. 4494–4498 (2023). https://doi.org/10.21437/Interspeech.2023-2449
22. Prabhavalkar, R., Hori, T., Sainath, T.N., Schlüter, R., Watanabe, S.: End-to-end speech recognition: A survey. IEEE/ACM Trans. Audio, Speech, Lang. Process. **32**, 325–351 (2024). https://doi.org/10.1109/TASLP.2023.3328283
23. Pundak, G., Sainath, T.N., Prabhavalkar, R., Kannan, A., Zhao, D.: Deep context: end-to-end contextual speech recognition. In: Spoken language technology workshop (SLT), pp. 418–425. IEEE (2018)
24. Qiu, D., Munkhdalai, T., He, Y., Sim, K.C.: Context-aware neural confidence estimation for rare word speech recognition. In: Spoken Language Technology Workshop (SLT), pp. 31–37. IEEE (2023)
25. Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I.: Robust speech recognition via large-scale weak supervision. In: International conference on machine learning, pp. 28492–28518. PMLR (2023)
26. Rio, M.D., et al.: Earnings-21: A practical benchmark for ASR in the wild. In: Interspeech (2021)
27. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1715–1725 (2016)
28. Stolcke, A.: SRILM—an extensible language modeling toolkit. In: Seventh international conference on spoken language processing (2002)
29. Svete, A., Dayan, B., Cotterell, R., Vieira, T., Eisner, J.: Algorithms for acyclic weighted finite-state automata with failure arcs. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (eds.) EMNLP, pp. 8289–8305. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (2022). https://doi.org/10.18653/v1/2022.emnlp-main.567
30. Tian, J., Yu, J., Weng, C., Zou, Y., Yu, D.: Improving mandarin end-to-end speech recognition with word n-gram language model. IEEE Signal Process. Lett. **29**, 812–816 (2022)
31. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
32. Wang, W., et al.: Contextual biasing with the Knuth-Morris-Pratt matching algorithm. arXiv preprint arXiv:2310.00178 (2023)
33. Wolf, T., et al.: Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)
34. Yao, Z., et al.: Zipformer: A faster and better encoder for automatic speech recognition. In: ICLR (2024)
35. Zhao, D., et al: Shallow-fusion end-to-end contextual biasing. In: Interspeech, pp. 1418–1422 (2019)