Cost-Effective Development of Custom Wake Word Detection Models for Low-Resource Languages in Embedded Devices

Adib Vali, Zahra Razavi, Zeinab Borhanifard, Romina Oji
Ussistant company, University of Rochester, University of Tehran, University of Tehran
Adib.vali@ussistant.ir, srazavi@cs.rochester.edu, borhanifardz@ut.ac.ir, romina.oji@ut.ac.ir

Abstract

Creating a reliable wake word detection system for custom wake words poses a significant challenge, particularly in low-resource languages where the scarcity of available data sources is a major hurdle. Moreover, collecting an adequately voluminous dataset that includes both positive and negative samples entails substantial financial costs and significant time expenditures. To address this problem, we propose a cost-efficient approach to enrich a small set of collected custom samples. We provide a range of techniques for preprocessing, data augmentation, and noise synthesis to expand the positive samples. In addition, we automatically extracted specifically chosen negative samples from an existing speech dataset. The augmented data is utilized for the training of a neural network-based detector through the utilization of Mycroft Precise. The results demonstrate an improved production-grade performance, which can be vastly used in embedded devices and custom virtual assistants.

Keywords: Wake-word detection, Data augmentation, Synthesizing, Embedded devices, Confusing words

1. Introduction

Wake word detection involves the identification of a particular spoken word or phrase that triggers the activation of a speech recognition system, such as smart virtual assistants Jose et al. (2020). In recent times, there has been a rising trend where companies and individuals are creating their own special wake word detection systems to make their products better and link them to specific brands or products. Such systems must exhibit a high degree of accuracy and low latency. However, creating a reliable wake word detector for personalized wake words on embedded devices is challenging due to resource constraints, far-field detection, and the costs of collecting custom training data. Additionally, achieving a minimal false alarm rate is essential in smart devices to uphold user privacy and reduce unnecessary or disruptive device activations.

Several approaches have been suggested for wake word detection in the past years Sainath and Parada (2015); Chen et al. (2014); Wilpon et al. (1991); Huggins-Daines et al. (2006); Panchapagesan et al. (2016). In the majority of these methods, a key factor in obtaining an accurate low-latency detector involves training the model with an extensive amount of wake-word-specific data. The dataset should include samples from a variety of speakers in a variety of noisy environments, along with negative samples that closely resemble wake words. Creating such a dataset can be costly. However, employing blind sampling methods like those in Sainath and Parada (2015); Chen et al. (2014) doesn't lead to an optimized model. To address this challenge, we propose a method for training a wake-word model using a significantly limited set of gathered wake word samples for Persian, a low-resource language. We collected 1350 wake-word utterances (30 minutes) from 70 speakers, and did not utilize any self-collected negative data. Instead, we extracted all negative samples from available speech datasets like Common Voice [1], which is a crowdsourced speech recognition dataset supporting more than 100 languages. To enhance the dataset, we integrate a range of augmentation and generation techniques found in existing literature Ghosh et al. (2022); Gao et al. (2020), along with our own discoveries, to expand the limited training dataset. The suggested methods proved effective in enhancing the robustness of the wake word detector. We utilize a Mycroft Precise DNN-based model [2], and our methods were used to train and develop the model.

In this paper, we present a model and several data augmentation methods to train a robust wake word detector with just a limited number of self-collected wake word samples. Our approach provides a model development framework to help researchers and companies reduce the time and cost of collecting training data for custom wake word detection.

2. Related work

Several wake word detection models and engines have been developed in recent years. Some of them such as Porcupine[3] are not open source although show high performance. HMM models are also widely used for wake word detection Rose and Paul (1990); Wilpon et al. (1991); Huggins-Daines et al. (2006). A highly used example of HMM is

[1] https://commonvoice.mozilla.org
[2] https://github.com/MycroftAI/mycroft-precise
[3] https://picovoice.ai/platform/porcupine/

Pocketsphinx Huggins-Daines et al. (2006). In recent years, DNN-based models, including CNN and RNN models, have improved performance compared to HMM-based ones Panchapagesan et al. (2016); Wöllmer et al. (2013). Additionally, TDNN-HMM Sun et al. (2017) from Amazon Alexa and Apple's Siri Sigtia et al. (2018) presents a production-grade low-resource model that trains DNN-HMM model with a million utterances of the trigger phrase. Mycroft Precise is also a renowned open-source engine that employs lightweight RNN architecture.

In terms of training data, however, most of the available wake word detection models require a large number of custom wake word utterances (e.g. more than 10 hours) which could be both costly and time-consuming to collect. Some research proposed solutions to train models with a low number of samples, such as Hossain and Sato (2021). They mainly rely on crowd-sourcing to collect all negative and positive utterances, which is still expensive and time-consuming. Others, such as Gao et al. (2020), present a method to extract close fake words using ASR systems from untranscribed speech datasets. This approach needs a stable and accurate ASR system, which may not be available for all languages, especially low-resource ones. Another approach suggested in Ghosh et al. (2022) is called Knowledge Distillation, in which student wake word detectors are trained using a large teacher model that, in turn needs more than 2000 hours of speech data; a dataset that cannot be easily found in many low resource languages. To overcome the training data constraints we explored techniques for enriching a small set of collected data to maximize its utility.

3. Data Preprocessing and Augmentation

In this section, we describe our approach for generating a wake word detection model for low-resource languages (i.e. Persian) with a limited number of wake word samples. As shown in Figure 1, our approach consists of several steps, including data preprocessing, augmentation, and incremental training techniques to improve the model's performance.

We did not collect any recorded negative voice samples. Instead, the negative samples are extracted from Common Voice transcribed dataset in three rounds of negative collection.

3.1. Confusion words Extraction

As suggested in Gao et al. (2020), confusion words are extracted based on their phonetics, IPA, and Levenshtein distance Navarro (2001) from the wake word phonetics. They are phonetically similar words in the language and can be recorded or extracted from available speech datasets.

To ensure that the wake word detection model can distinguish words that are similar but not equal to the wake word, we extract confusion words from the transcribed data. We use a Persian phonetic dictionary and compute the Levenshtein distance between the wake word and all the words in the dictionary. In this process, the weight of Persian vowels such as 'a', 'e', 'u', 'i', 'o' is increased, as we noticed that vowels have a greater impact on word pronunciation, and doing this can help to identify similar words to the wake word better.

Subsequently, we assemble a list of words closely phonetically resembling the wake word and locate them within the Common Voice dataset, enabling us to extract corresponding audio recordings containing these similar words. To extract the audio containing only the target word, we divide the audio into the number of characters in the sentence and extract the sound corresponding to the word based on its position in the sentence. In contrast to employing a ASR system, this approach not only offers faster and more cost-effective results but is also particularly optimal for low-resource languages lacking high-quality ASR systems. The reason is that, despite the fact that we have low-quality ASR systems, we can still benefit from a limited properly transcribed voice dataset, such as Common Voice.

3.2. Negative learning over large negative set

Identifying and utilizing confusion voice patterns that can cause false positives is another significant aspect of our approach. These patterns do not emerge when individual words are uttered. They can be heard in more extended pieces of speech where a stream of words contains patterns indistinguishable from the target wake word.

We employ a negative learning and incremental training approach to retrain the model using false positives. The confusion voice patterns are extracted from the speech pattern of a large audio set that does not contain the exact wake word. We repeat the training several times; after each, the false positive samples are fed into the model as part of the training data.

3.3. Wake Word-based Negative Data

To enhance the ability of the model to distinguish speeches similar to the wake word from a true wake word instance, we generate wake-word-based negative samples. For this, we randomly select one or two phonemes (300 milliseconds) from true wake word samples, mix them with (300 milliseconds) of the noise audio, then use the new noise audio as negative data. This leads the model to become

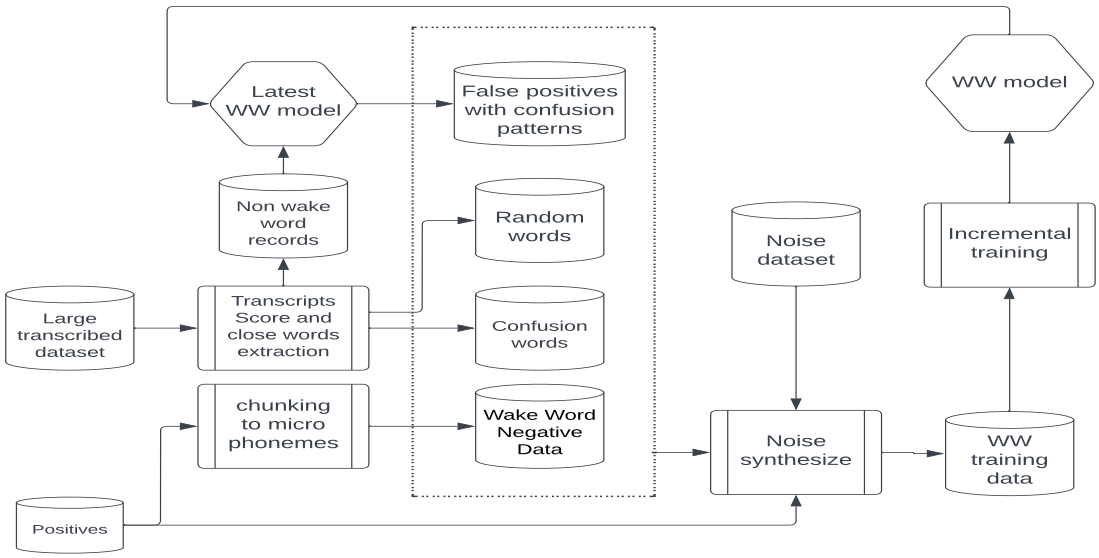

Figure 1: Wake word training system.

more sensitive to the presence of all parts of the wake word and do not get tricked with just stressed micro phonemes. This idea is inspired by the approach introduced in Wang et al. (2023) for face recognition where they generate negative samples by masking pieces of image in positive samples, and the results showed an enhanced model performance.

3.4. Noise synthesize

Considering prior research findings that highlight the advantages of incorporating noise and simulating various environmental sounds Seltzer et al. (2013); Gibson et al. (2018), we explored its impact on our wake word detection model. In fact, smart assistants are expected to be able to recognize the wake word in diverse noise conditions, while simultaneously avoiding erroneous detections of noise as positive wake words. In order to enrich our available set of positive samples, we incorporate five different signal-to-noise (SNR) ratios. We combine each training sample, both positive and negative, with four different randomly selected noise types with varying levels of SNR (5, 15, 25, and 35), as well as with a real-world noise. This approach has shown a significant effect on boosting the model performance.

4. Experiment

4.1. Model Architecture

The baseline model we use in this work is a modified version of the Precise engine , which employs a single recurrent neural network - specifically, a GRU - for wake word detection. We fine-tuned the hyperparameters of the network, in order to reduce the bias and enhance the discrimination ability of the model. However, we observed that increasing the robustness of the train and test sets to better reflect real-life conditions led to an increase in bias, making it difficult for the model to distinguish between positive and negative instances. After testing various hyperparameters, we found that increasing the number of units to 113 led to the best model optimization. Our model resembles the Precise engine model pipeline [4],in that 16000 sample rate audio is buffered with 1.5 seconds sliding windows at a time interval of 0.1 seconds before being passed through a 13 n_mfcc and 512 n_fft feature extractor. The extracted features are then fed into the GRU with 113 units.

4.2. Dataset

To ensure the robustness of our wake word detection model, we carefully prepared our dataset. We chose the wake word "Smarta" and collected a total of 1350 utterances from 70 speakers. The utterances were recorded using a close-talk microphone by native Persian speakers in a quiet environment. Each audio sample is two seconds long. To create negative samples for training, we extracted confusion samples from the Common Voice dataset, as described in Section 3.1.

Since "Smarta" does not exist in the Persian language, we consider all the words in the Common Voice dataset as potential negative data samples. This allows us to generate a diverse set of chal-

[4]https://github.com/MycroftAI/mycroft-precise

lenging negative samples that are not easily distinguishable from positive samples.

Samples	Label	Train	Test
Smarta	P	950	400
Random negative words	N	900	400
Confusion words	N	900	400
False positives generated in Negative learning	N	3600	1500
Wake word-based negative data	N	1200	600
Random environment noise	N	2000	1000

Table 1: clear dataset (without noise) statistics (P: positive, N: negative)

4.3. Experimental Setup

We offer two evaluation datasets for the presentation of our findings. In both test datasets, the data was meticulously gathered to ensure that none of it was utilized in the model's training process.

1. The Details of test sets are shown in Table 1. A key note to be mentioned is that to test the robustness of our model in different noise conditions, we randomly synthesized these samples with the Demand noise dataset Thiemann et al. (2013) in snr5, snr15, snr25, and snr35. We combined the generated noisy samples with the original samples.

2. The second test set, known as the Persian Vox dataset [5], encompasses 56 hours of Persian speech data. Our evaluation includes measurements of the model's performance and the rate of false positives per hour across the entire Vox dataset. The dataset is diverse and representative of real-world scenarios, which ensures the effectiveness of the proposed wake word detection model.

We trained and tested five wake word detection models using different training setups in our experiments. The baseline setup used "Smarta" and randomly selected negative words for training, as shown in Table 1. In the NS setup, we added noise-synthesized versions of Smarta and negative samples at four different SNR levels to the training data. For NS+NL, we additionally included noise-synthesized versions of false positives generated during negative learning. The NS+NL+CW setup further included noise-synthesized versions of confusion words in addition to the previously mentioned samples. Finally, in the NS+NL+CW+WWB setup, we also included noise-synthesized versions of wake word-based samples in addition to the other samples.

4.4. Results

As seen in Figure 2, each method contributes to improving the robustness of the model. Adding False

[5] https://librivox.org/

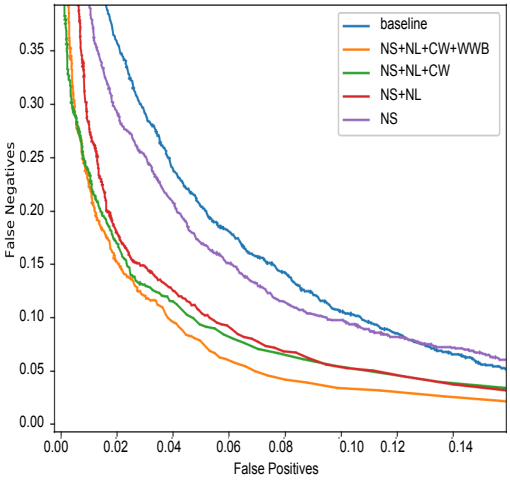

Figure 2: Wake word training system.

positives generated in Negative learning shows the highest improvement, and adding each part of the negative set helps the model to better classify between the wake word and confusion voices. Additionally, the model that uses all the presented techniques outperforms the other models on the second test set, as shown in Table 2.

In Table 2, we can observe a performance comparison between our top-performing model and the Porcupine wake word detector. Our model demonstrates superior performance, boasting a 7.6% lower false rejection rate when compared to Porcupine.

Training set	WER rate
Porcupine	0.32
Baselines	0.313
NS	0.29
NS + NL	0.28
NS + NL + CW	0.27
NS + NL + CW + WWB	0.244

Table 2: Experimental results

5. Conclusions

In this study, we introduced a cost-effective approach to enhance a wake word detection model for the low-resource language, Persian. We expanded our dataset with preprocessing, data augmentation, and noise synthesis, using both positive and negative samples. Our neural network-based detector, trained with this enriched data using Mycroft Precise, demonstrated significant performance improvements. This approach proves effective for enhancing model performance in the absence of extensive datasets.

6. Bibliographical References

Guoguo Chen, Carolina Parada, and Georg Heigold. 2014. Small-footprint keyword spotting using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091.

Yixin Gao, Yuriy Mishchenko, Anish Shah, Spyros Matsoukas, and Shiv Vitaladevuni. 2020. Towards data-efficient modeling for wake word spotting. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7479–7483.

Arindam Ghosh, Mark Fuhs, Deblin Bagchi, Bahman Farahani, and Monika Woszczyna. 2022. Low-resource Low-footprint Wake-word Detection using Knowledge Distillation. In *Proc. Interspeech 2022*, pages 3739–3743.

Matthew Gibson, Christian Plahl, Puming Zhan, and Gary Cook. 2018. Multi-condition deep neural network training. In *Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung 2018*, pages 77–84. TUDpress, Dresden.

Delowar Hossain and Yoshinao Sato. 2021. Efficient corpus design for wake-word detection. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 1094–1100.

D. Huggins-Daines, M. Kumar, A. Chan, A.W. Black, M. Ravishankar, and A.I. Rudnicky. 2006. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I.

Christin Jose, Yuriy Mishchenko, Thibaud Senechal, Anish Shah, Alex Escott, and Shiv Vitaladevuni. 2020. Accurate detection of wake word start and end using a cnn. *arXiv preprint arXiv:2008.03790*.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88.

Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Björn Hoffmeister, and Shiv Vitaladevuni. 2016. Multi-Task Learning and Weighted Cross-Entropy for DNN-Based Keyword Spotting. In *Proc. Interspeech 2016*, pages 760–764.

R.C. Rose and D.B. Paul. 1990. A hidden markov model based keyword recognition system. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 129–132 vol.1.

Tara N. Sainath and Carolina Parada. 2015. Convolutional neural networks for small-footprint keyword spotting. In *Proc. Interspeech 2015*, pages 1478–1482.

Michael L. Seltzer, Dong Yu, and Yongqiang Wang. 2013. An investigation of deep neural networks for noise robust speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7398–7402.

Siddharth Sigtia, Rob Haynes, Hywel Richards, Erik Marchi, and John Bridle. 2018. Efficient voice trigger detection for low resource hardware. In *Proc. Interspeech 2018*, pages 2092–2096.

Ming Sun, David Snyder, Yixin Gao, Varun Nagaraja, Mike Rodehorst, Sankaran Panchapagesan, Nikko Ström, Spyros Matsoukas, and Shiv Vitaladevuni. 2017. Compressed time delay neural network for small-footprint keyword spotting. In *Interspeech 2017*.

Joachim Thiemann, Nobutaka Ito, and Emmanuel Vincent. 2013. The Diverse Environments Multichannel Acoustic Noise Database (DEMAND): A database of multichannel environmental noise recordings. In *21st International Congress on Acoustics*, Montreal, Canada. Acoustical Society of America. The dataset itself is archived on Zenodo, with DOI 10.5281/zenodo.1227120.

Zhongyuan Wang, Baojin Huang, Guangcheng Wang, Peng Yi, and Kui Jiang. 2023. Masked face recognition dataset and application. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, pages 1–1.

J.G. Wilpon, L.G. Miller, and P. Modi. 1991. Improvements and applications for key word recognition using hidden markov modeling techniques. In *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, pages 309–312 vol.1.

Martin Wöllmer, Björn Schuller, and Gerhard Rigoll. 2013. Keyword spotting exploiting long short-term memory. *Speech Communication*, 55(2):252–265.