INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

# Matching Latent Encoding for Audio-Text based Keyword Spotting

*Kumari Nishu, Minsik Cho, Devang Naik*

Apple

knishu@apple.com, minsik@apple.com, naik.d@apple.com

## Abstract

Using audio and text embeddings jointly for Keyword Spotting (KWS) has shown high-quality results, but the key challenge of how to semantically align two embeddings for multi-word keywords of different sequence lengths remains largely unsolved. In this paper, we propose an audio-text-based end-to-end model architecture for flexible keyword spotting (KWS), which builds upon learned audio and text embeddings. Our architecture uses a novel dynamic programming-based algorithm, Dynamic Sequence Partitioning (DSP), to optimally partition the audio sequence into the same length as the word-based text sequence using the monotonic alignment of spoken content. Our proposed model consists of an encoder block to get audio and text embeddings, a projector block to project individual embeddings to a common latent space, and an audio-text aligner containing a novel DSP algorithm, which aligns the audio and text embeddings to determine if the spoken content is the same as the text. Experimental results show that our DSP is more effective than other partitioning schemes, and the proposed architecture outperformed the state-of-the-art results on the public dataset in terms of Area Under the ROC Curve (AUC) and Equal-Error-Rate (EER) by 14.4 % and 28.9%, respectively.

**Index Terms**: keyword spotting, audio encoder, text encoder, audio-text sequence alignment

## 1. Introduction

Voice assistants on smart devices, like smart phones and small speakers, have become a ubiquitous part of our daily life by providing convenience in controlling personal/home devices while multi-tasking (e.g., driving). Thus, natural human-like interactions with such voice assistants are in greater demand than ever. Voice assistants commonly rely on Keyword Spotting (KWS) techniques to detect whether a custom trigger phrase and voice command was uttered in an input speech signal.

There are two broader categories of KWS tasks: fixed vocabulary KWS, where models work only for a fixed list of keywords [1, 2], and flexible KWS, where models are expected to work for any arbitrary keyword enrolled by users [3, 4, 5]. Flexible KWS is a highly challenging task as it involves detecting arbitrary keywords, which may not have been observed during model training. In flexible KWS, a user can specify their custom keywords either in the form of enrolled audio samples (also known as query-by-example, QbE KWS [3, 4]) or as the text. In this work, we focus on the latter where a custom keyword is provided as text, and a sequence of words.

Prior works in flexible KWS built on embedding-based neural network modeling have shown promising results [6, 7], but suffer from being inapplicable for heterogeneous modalities (like audio and text) [8, 9, 10], being ineffective on multi-word keyword detection [4, 11, 12], or failing to capture word-level semantics [13]. For more details, please refer to Section 2.

To address such challenges, we propose a novel embedding-based neural network architecture for flexible KWS, which consists of encoders, projectors, and an audio-text aligner. Unlike existing projection approaches in [4, 11, 12], our projection does not alter the sequence length but focuses on transformation of embeddings into a common space. Then, our audio-text aligner, powered by a novel Dynamic Sequence Partitioning (DSP) algorithm, finds the best audio sequence partitioning for high-quality audio-to-text mapping. Finally, all of the audio frames in each partition are aggregated for distance computation from the paired word in the text. Such a distance can be used to determine whether the target keyword (in text) also exists in the audio sequence. Our major contributions are as follows:

- We propose a novel Dynamic Sequence Partitioning (DSP) algorithm to compare two sequences of different lengths where **a)** contents are monotonically aligned, **b)** the indices from the shorter sequence have one-to-many relationship with the longer one, but **c)** the indices from the longer sequence have one-to-one relationship with the shorter one.
- We present a flexible embedding matching setup to map an acoustic embedding sequence to its corresponding textual word embedding sequence, even from two independently pre-trained encoders, such that the audio sequences for the same text are embedded similarly.
- We formulate a discriminative setting to train an end-to-end KWS model for both single and multi-word keywords. Our model consists of encoders, projectors and a novel DSP-based audio-text aligner.

## 2. Related work

One approach for flexible KWS is to use a well-known dynamic time warping (DTW) algorithm [8] to measure the distance between two embedding sequences of different lengths for the KWS task [9, 10]. DTW calculates the best match between two given sequences, where the indices of one sequence have a one-to-many relationship with the indices of the other sequence, which works well when we compare homogeneous modalities (like audio vs. audio) since an index from both ends represents the same entity (e.g., $25ms$ audio frame per index). However, DTW becomes less applicable or inaccurate when we compare heterogeneous modalities, such as audio (where each index represents one audio frame) and text (where each index represents one word token), which requires one index of a word to be mapped to multiple indices of audio frames but not vice versa. To overcome such limitations, [4, 11, 12] project both audio and text sequences into a fixed sequence length which makes the distance computation between two sequences sim-



10.21437/Interspeech.2023-478

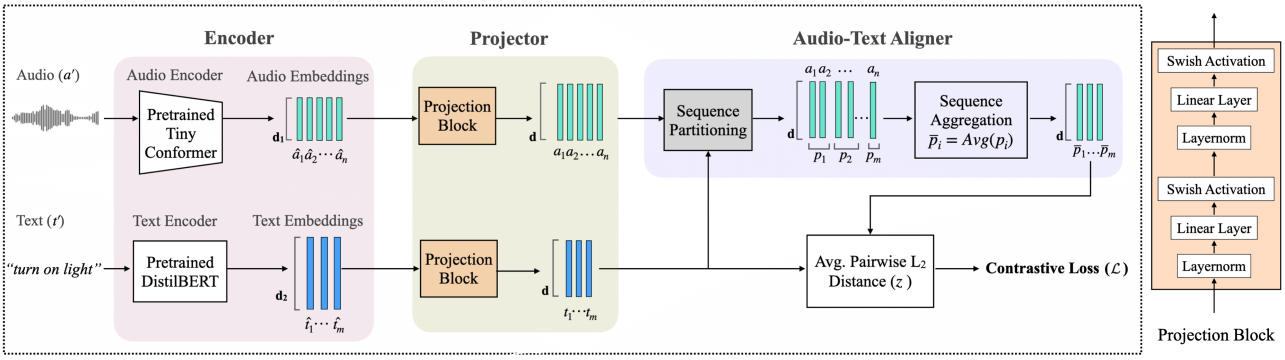


Figure 1: *Overall architecture of the proposed end-to-end KWS model i.e EMKWS (Encoding Matching based KeyWord Spotter). It has three modules- encoder, projector, audio-text aligner. Input to the model- (audio and text) pair, output- a binary label indicating if the audio content is same as the text.*

pler at the cost of losing critical semantic context during projection (i.e., projection does dimensionality reduction). Such projection-based schemes have been shown to learn word discrimination better than the DTW-based schemes but tend to deliver poor performance on multi-word keyword detection. Recently, [13] explored an audio-text based flexible KWS in a different way, where audio and text encoders generate flexible length sequences of embedding vectors per phoneme, which is fed into a GRU network to generate a single embedding vector for both the audio and text modalities. However, such phoneme-based embeddings do not effectively learn to discriminate words as they fail to capture word-level semantics.

The recent work in [5] measures the audio and text correspondence for flexible KWS task and have reported the state-of-the art results compared to CTC [14] and attention [15] based methods. However, the text encoder of [5] does not capture the semantics of the textual keyword as it is based on the grapheme and phoneme sequence, hence [5] is weak against similar sounding. Moreover, [5] learns explicit patterns between text and audio embeddings with monotonic matching loss, which compromises the alignment flexibility and hence it might not end up with fully optimized alignment.

# 3. Proposed Method

In this section, we describe our scheme, Encoding Matching based Keyword Spotter (EMKWS) shown in Fig. 1. EMKWS consists of three modules - an encoder, a projector, and an audio-text aligner, which drive our contrastive loss function. EMKWS model takes an audio and text pair as input and measures the similarity between the spoken content of the audio and the text to determine if they match. Let us denote an input example as $(a', t', l')$, where audio $a' = (a'_1, a'_2, \ldots, a'_{n'})$ is a sequence of audio frames, text $t' = (t'_1, t'_2, \ldots, t'_{m'})$ is a sequence of text tokens, and $l'$ is the ground truth binary label with $l' = 1$ for a positive input pair and 0 otherwise. We will loosely refer to text as a sequence of words for better clarity. The rest of this section describes each module in more detail.

## 3.1. Encoder

The encoder module consists of an audio encoder and a text encoder as shown in Fig. 1. We use the Conformer architecture [16] as the audio encoder to generate the embedding for a given audio input signal. This choice was inspired by the fact that the Conformer combines the power of self-attention [15] to learn global interactions and convolutions to capture local correlations efficiently. This is particularly useful for speech data as speech contains sequential information along with the global context in which the speech is being delivered. We employ a tiny version of the Conformer architecture described in Section 4. We first train this audio encoder for Automatic Speech Recognition (ASR) to produce a transcription for the given audio, we remove the last linear layer, and then take the remaining architecture as a frozen audio encoder. Given $a'$ as the input, let us denote the output from this audio encoder as $\hat{a} = (\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_n)$, where $\hat{a}_i \in \mathbb{R}^{d_1}, \forall i = 1, 2, \ldots, n$ and $n < n'$ as the Conformer module has a subsampling layer to reduce the encoded sequence length.

For the text sequence, we use a transformer-based model trained for natural language understanding, DistilBERT [17], to obtain a semantically focused text embedding for the input text. Given $t'$ as the input text, let us denote the output from this text encoder as $\hat{t} = (\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_m)$, where $\hat{t}_k \in \mathbb{R}^{d_2}, \forall k = 1, 2, \ldots, m$ and $m \geq m'$ as the tokenizer of DistilBERT may produce multiple sub-words as tokens for a single word.

## 3.2. Projector

The incoming audio and text embeddings from the encoder module are passed through the respective projection block in the projector module, as in Fig. 1. So far, the two modalities have been encoded in their separate embedding spaces. The role of the projector module is to project both embeddings to a common embedding space of dimension $d$. The projection block consists of two layer norms and two linear layers, as shown on the right side of Fig. 1. Given the audio and text embedding inputs $\hat{a}$ and $\hat{t}$, we denote the projected audio embedding as $a = (a_1, a_2, \ldots, a_n)$, where $a_i \in \mathbb{R}^d, \forall i = 1, 2, \ldots, n$ and the projected text embedding as $t = (t_1, t_2, \ldots, t_m)$, where $t_k \in \mathbb{R}^d, \forall k = 1, 2, \ldots, m$.

## 3.3. Audio-Text Aligner

Our audio-text aligner is composed of two parts, sequence partitioning and sequence aggregation. The audio and text embeddings generated from the projector module have the same embedding dimension $d$ at the different sequence lengths $n$ and $m$ in Fig. 1 (and, $n > m$ in most cases), and the aligner transforms the audio embedding into the same dimension as the text embedding (i.e., $\mathbb{R}^{m \times d}$) to compare them. Even though their

sequence lengths are different, the content alignment between the two is monotonic in nature. Our Dynamic Sequence Partitioning (DSP) described in Algorithm 1 leverages such monotonic characteristic to perform sequence partitioning of the audio sequence $a = (a_1, a_2, \ldots, a_n)$ into $m$ chunks. Then, each partition is aggregated for loss generation.

Let us denote a partition of the sequence $a$ into $m$ contiguous sub-sequences as $p = (p_1, p_2, \ldots, p_m)$, as shown in the Fig. 1. This partition can be represented using the $(m+1)$ indices $i_0, i_1, \ldots, i_m$ (where $i_0 = 1$, $i_m = n+1$) such that $p_1 = a_{i_0:i_1}$, $p_2 = a_{i_1:i_2}$, and similarly $p_m = a_{i_{(m-1)}:i_m}$, where we use the notation $a_{i:j}$ to denote the sub-sequence $\left(a_i, a_{(i+1)}, \ldots, a_{(j-1)}\right)$. Then, during the next sequence aggregation step, we aggregate each sub-sequence to create output as $(\overline{p_1}, \overline{p_2}, \ldots, \overline{p_m})$ where over-line $\overline{(.)}$ denotes the mean of a sequence of vectors. Note that the output is a sequence containing $m$ vectors of $d$ dimension each, and thus can be compared directly with the text embedding $t$. Finally, we measure the distance between these two sequences as the average pairwise $L_2$ distance, i.e. $\frac{1}{m}\sum_{k=1}^{m} \|\overline{p_k} - t_k\|_2$.

Since the model training is guided by such $L_2$ distance, our method captures the best possible partitioning solution by directly minimizing the $L_2$ distance using **Dynamic Sequence Partitioning (DSP) algorithm** in Algorithm 1. DSP finds the optimal partition such that the aggregated audio embedding and the text embedding have minimum average pairwise $L_2$ distance $z$ calculated as in the following formulation:

$$z(a,t) = \min_{p \in Partition(a,m)} \left( \frac{1}{m} \sum_{k=1}^{m} \|\overline{p_k} - t_k\|_2 \right) \quad (1)$$

where $Partition(a,m)$ denotes the set of all possible partitions of sequence $a$ into $m$ chunks.

Our optimal mapping from the audio to the text embeddings will offer the best possible view of the mapping throughout the training, and our experimental results in Fig. 2 suggest that our approach is effective, especially for the hard samples. The computational complexity of DSP is $\mathcal{O}(mn^2)$ but since $m$ is usually very small ($m \leq 4$ in our case), the complexity becomes $\mathcal{O}(n^2)$. Similarly, the space complexity is $\mathcal{O}(n)$.

### 3.4. Contrastive Loss

We use contrastive loss [18] as our training objective. The contrastive loss takes the average pairwise $L_2$ distance, $z$, from Equation 1 as input, which minimizes the distance for positive examples and maximizes the distance for negative examples. The loss $\mathcal{L}$ for an example $(a', t', l')$ is calculated as:

$$\mathcal{L} = l' \cdot \max(z - m_{pos}, 0) + (1 - l') \cdot \max(m_{neg} - z, 0) \quad (2)$$

where $l'$ is the ground truth label which is 1 for a positive example and 0 for negative example, $z$ is the computed distance, $m_{pos}$ is the positive margin over which positive examples will contribute to the loss, and similarly $m_{neg}$ is the negative margin under which negative examples will contribute to the loss.

## 4. Experimental Results

We implemented our method in PyTorch, and experimented on a x86 Linux machines with NVIDIA V100 GPU.

### 4.1. Experiment Setup

**Dataset:** We trained and evaluated the proposed method using LibriSpeech [19] and Libriphrase datasets [5]. The audio encoder in Fig. 1 is trained with *train-clean-100* and *train-clean-360* from the LibriSpeech dataset using character based tokenization. Following [5], we generated a phrase unit based Libriphrase dataset from LibriSpeech to train and evaluate the EMKWS model. The training set of Libriphrase was generated from *train-clean-100* and *train-clean-360*, and the evaluation set was generated from *train-others-500*. The Libriphrase dataset consists of 1, 2, 3, and 4 word-phrases. Each example is denoted by 3 entities: (*audio, text, target*) where the target value is 1 if the audio and text both represent the same content, otherwise 0. There are two types of negative examples, easy and hard, based on the Levenshtein distances [20] between the audio content and the paired text. To make comparison with [5], we built the same evaluation dataset, Libriphrase Easy (LE) and Libriphrase Hard (LH), consisting of 4391, 2605, 467, and 56 episodes of each word length respectively. Each episode has 3 positive and 3 negative audio-text pairs. The audio features are extracted using 80-channel filterbanks from a 25ms window with a stride of 10ms.

**Pre-trained Models:** We used the Conformer architecture [16] to build the audio encoder in Fig. 1. The Conformer hyper-parameters are {*6 encoder layers, encoder dimension of 144, convolution kernel of size 3, and 4 attention heads*}. Lastly, we added a Linear layer with output dimension of 29 and trained the resulting model with CTC loss [14]. The model was trained for 300 epochs with Adam optimizer [21] and transformer learning rate schedule [15] with 5k warm-up steps. After the training, we removed the Linear layer and froze the parameters of the audio encoder to use in EMKWS. For the text encoder, we use DistilBERT [17], a smaller version of the natural language understanding model BERT [22]. We took the pretrained DistilBERT model from the Hugging Face library [23].

**Algorithm 1** Dynamic Sequence Partitioning (DSP) Algorithm

**Input:**
- Audio Embedding, $a = (a_1, a_2, \ldots, a_n)$
- Text Embedding, $t = (t_1, t_2, \ldots, t_m)$

**Initialization:**
**1.** Let $D \in \mathbb{R}^{m \times n}$, $D[k][i] = 0, \forall\, 1 \leq k \leq m, 1 \leq i \leq n$
$D[k][i]$ will store the optimal loss to divide the array $a_{i:(n+1)}$ in $k$ chunks to match with the last $k$ text tokens
**2.** Set $D[1][i] = \left\|\overline{a_{i:(n+1)}} - t_m\right\|_2, \forall\, 1 \leq i \leq n$

**Procedure:**
**for** $k = 2, 3, \ldots, m$ **do**
  **for** $i = 1, 2, \ldots, n-k+1$ **do**
    Set $\delta_j = \left\|\overline{a_{i:j}} - t_{(m-k+1)}\right\|_2, \forall\, i < j \leq (n-k+2)$

    $D[k][i] = \min_{i<j\leq(n-k+2)} (\delta_j + D[k-1][j])$
  **end for**
**end for**
**Output:** $D[m][1]/m$, optimal loss to divide the sequence $a$ in $m$ chunks to best align with the $m$ tokens of sequence $t$

### 4.2. Training and Inference

We train the end-to-end EMKWS model with the frozen audio and text encoders. The output of encoder module gives audio and text embeddings of dimensions $d_1 = 144$ and $d_2 = 768$ respectively (see Fig. 1). The projector module projects them

| Method | AUC Score (%) | | EER (%) | |
|---|---|---|---|---|
| | LE | LH | LE | LH |
| **Baseline** [5] | 96.7 | 73.58 | 8.42 | 32.9 |
| Proposed (run 1) | 97.83 | 84.21 | 7.26 | 23.37 |
| Proposed (run 2) | 97.84 | 84.18 | 7.34 | 23.29 |
| Proposed (run 3) | 97.81 | 84.24 | 7.48 | 23.42 |
| **Proposed (avg)** | **97.83** | **84.21** | **7.36** | **23.36** |
| **Improvement (%)** | **1.2** | **14.4** | **12.6** | **28.9** |

Table 1: *Experimental results of the proposed EMKWS model on Libriphrase Easy (LE) and Libriphrase Hard (LH) datasets with comparison to the baseline model.*

to a common embedding space of dimension $d = 144$. For the contrastive loss in Eq. 2, we experimented with a combination of positive and negative margin hyper-parameters, with $m_{pos} \in \{0, 0.1, 0.2, 0.25, 0.4, 1.0\}$ and $m_{neg} \in \{2, 2.5, 4, 7, 10, 15\}$, and the best was $m_{pos} = 0.2$ and $m_{neg} = 7$. EMKWS is trained for 165 epochs with Adam optimizer [21] and transformer learning rate schedule [15] with 2k warm-up steps. EMKWS model uses the pre-computed embeddings of the enrolled keywords instead of keeping the text encoder during inference, which leads to 3.7M parameters in total.

### 4.3. Results and Discussion

We compared the proposed EMKWS model with the state-of-the-art results from baseline [5] in Table 1, which shows that our method outperforms the baseline in terms of Area Under the ROC Curve (AUC) and Equal-Error-Rate (EER) metric on Libriphrase Easy (LE) and Libriphrase Hard (LH) datasets. On the LE dataset, we see an improvement of 1.2% in AUC score and 12.6% in EER metric. The proposed model showed improvement on the LH dataset: AUC score improved by 14.4% and EER metric improved by 28.9%.

To understand the benefit of our dynamic programming scheme, we experimented with three different sequence partitioning algorithms: random, equal, and our DSP, and the experiment results are shown in Fig. 2. Although there are limited multi-word examples in the evaluation datasets, we see the DSP algorithm consistently superior to the other schemes.

We also examined how each word embedding in the text sequence is aligned with the embeddings of their corresponding chunks of audio frames, and visualized the correlation between text-to-audio and audio-to-audio embeddings resulted from the projector module in Fig. 3. We see three distinct blocks of strong monotonic correlation along the audio embedding axis, and each block is aligned with each word in Fig. 3 (a). At the same time, Fig. 3 (b) shows that embeddings of audio frames for each word are fully self-correlated. We conjecture that such intuitive patterns between the projected embeddings are due to the DSP algorithm. For the examples of Fig. 3 (a) and (b), the input audio length was 550ms with the words *"the"*, *"old"* and *"man"* spoken for 90ms, 230ms and 230ms, respectively. Thus, the relative portion of each word is approximately $(0.16 : 0.42 : 0.42)$ which matches the partition sizes $(2, 5, 5)$ from the DPS algorithm as highlighted with a dashed vertical line in Fig. 3 (a) and (b). We can make similar observations for the four word positive example *"of the United States"* in Fig. 3 (c) and (d).

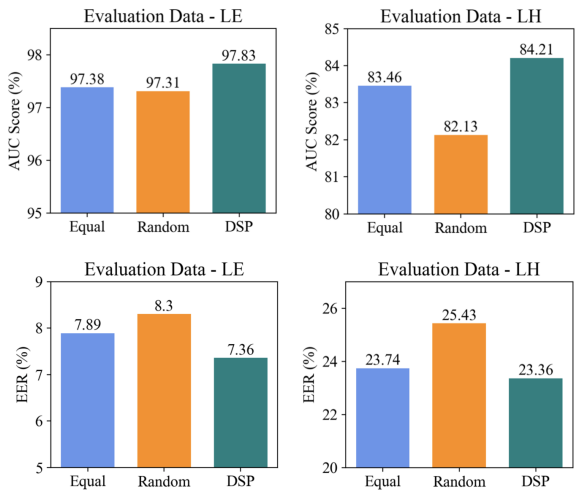


Figure 2: *Experimental results using different sequence partitioning schemes employed to partition the audio sequence to match its length with the text sequence. Proposed DSP algorithm shows superior performance on both LE and LH datasets in terms of AUC Score and EER metric.*

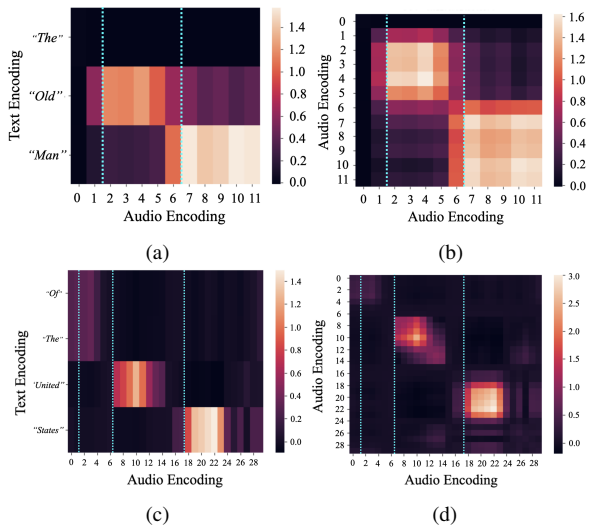


Figure 3: *Visualization of the projected audio and text embeddings using correlation between text-to-audio and audio-to-audio embeddings. (a) and (b) are for a positive example, "The Old Man". (c) and (d) are for a positive example, "Of The United States"*

## 5. Conclusions

In this paper, we proposed an audio-text-based end-to-end model architecture for flexible keyword spotting, consisting of encoder, projector, and audio-text aligner modules. We address the key challenge of comparing two sequences of different lengths by proposing a novel dynamic programming-based algorithm, Dynamic Sequence Partitioning (DSP), to optimally partition the audio embedding sequence into the same length as the text sequence. Experimental results showed that the proposed architecture outperformed the state-of-the-art results.

# 6. References


[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4087–4091, 2014.

[2] R. Tang and J. J. Lin, "Deep residual learning for small-footprint keyword spotting," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5484–5488, 2017.

[3] J. Huang, W. Gharbieh, H. S. Shim, and E. Kim, "Query-by-example keyword spotting system using multi-head attention and soft-triple loss," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6858–6862, 2021.

[4] K. R. R, V. Kurmi, V. Namboodiri, and C. V. Jawahar, "Generalized keyword spotting using asr embeddings," in *Interspeech*, 2022.

[5] H.-K. Shin, H. Han, D. Kim, S.-W. Chung, and H.-G. Kang, "Learning Audio-Text Agreement for Open-vocabulary Keyword Spotting," in *Proc. Interspeech 2022*, 2022, pp. 1871–1875.

[6] S. Settle and K. Livescu, "Discriminative acoustic word embeddings: Tecurrent neural network-based approaches," *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 503–510, 2016.

[7] S. Settle, K. Levin, H. Kamper, and K. Livescu, "Query-by-example search with discriminative neural acoustic word embeddings," in *Interspeech*, 2017.

[8] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 159–165, 1978.

[9] B. Yusuf and M. Saraçlar, "An empirical evaluation of dtw subsampling methods for keyword search," in *Interspeech*, 2019.

[10] T. S. Fuchs and J. Keshet, "Spoken term detection automatically adjusted for a given threshold," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, pp. 1310–1317, 2017.

[11] W. He, W. Wang, and K. Livescu, "Multi-view recurrent neural acoustic word embeddings," *ArXiv*, vol. abs/1611.04496, 2016.

[12] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H. yi Lee, and L.-S. Lee, "Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder," *ArXiv*, vol. abs/1603.00982, 2016.

[13] N. Sacchi, A. Nanchen, M. Jaggi, and M. Cernak, "Open-vocabulary keyword spotting with audio and text embeddings," in *Interspeech*, 2019.

[14] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," *Proceedings of the 23rd international conference on Machine learning*, 2006.

[15] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017.

[16] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," 2020.

[17] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *ArXiv*, vol. abs/1910.01108, 2019.

[18] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, 2006.

[19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[20] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet physics. Doklady*, vol. 10, pp. 707–710, 1965.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019.

[23] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6