UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Bachelor's degree in Data Science and Engineering

Spoken and Written Language Processing - 270225

Assignment 4 - Speech Recognition using Dynamic Time Warping

Sílvia Fàbregas
Eduard Morillo

Professors:
José Adrián Rodríguez
Carlos Escolano

May 23, 2024

Contents

Motivation and structure

Our motivations for this assignment are straightforward: first, we aim to understand the DTW algorithm, and second, we're tackling the task of recognizing spoken numbers in audio pieces through dynamic time warping.

The data we're working with is audio sampled at 16 kHz: `commands10x10` comprises 10 recordings of 10 digits, while `commands10x100` consists of 100 recordings of each of the 10 digits for validation. In addition, `free10x4x4` comprises 4 recordings of the 10 digits for each of four speakers (Jackson, Nicolas, Theo, and Yweweler).

This assignment is divided into various sections. In the first one, both the word error rate (WER) and the speaker error rate (SER) of the dataset `free10x4x4` are evaluated. The defined `wer` funcion compares by default the 'text' of the correct answer with the 'text' of the labeled recording that is close to each of the test wav files (using DTW distance). This function has been modified to compare the 'speaker' of the correct answer with the 'speaker' of the labeled recording that is the closest.

In Section 2, we compare the WER with respect to the number of cepstral coefficients.

In Section 3, the influence on the recognition accuracy of each of the cepstral normalization steps (mean and variance) with and without littering has been analysed.

Additionally, the MFCC (Mel Filter Cepstrum Coefficients) parameters have been extended with the first-order derivatives in order to improve the word error rate. The results are presented in Section 4.

In order to visualize the DTW algorithm, we have constructed plots with some examples of alignment, both when the test signal is different from the reference signal (different digit) or equal (same digit), see Section 5.

Finally, in the last section (6), the DTW algorithm has been compared with some of its variances, both in speed and accuracy.

Theoretical Background

Speech feature extraction

Speech feature extraction aims to convert raw audio signals into a format that is better suited for analysis and interpretation by machine learning algorithms. Even though audio signals are stored as sequences of numbers, so they could be inputted to the models right away, they contain a vast amount of data, much of which may be irrelevant or redundant for the task at hand, such as background noise or variations in pitch due to speaker differences.

The primary goal of speech feature extraction is therefore to capture the relevant characteristics of the speech signal while discarding irrelevant information. From raw audio signals, which consist of time-domain waveforms representing the amplitude of the sound signal at each point in time, a set of feature vectors that represent different aspects of the speech signal is *extracted*. The following sections delve deeper into the extraction process, particularized in the WAV file `eng2_m` to be found in [1]. The audio, that was sampled at 8,000 kHz, has been truncated to retain only the first 15,700 samples, aiming to enhance visual analysis. It corresponds to a male pronouncing *What if somebody decides to break it?*, and its time-domain plot is to be found below.

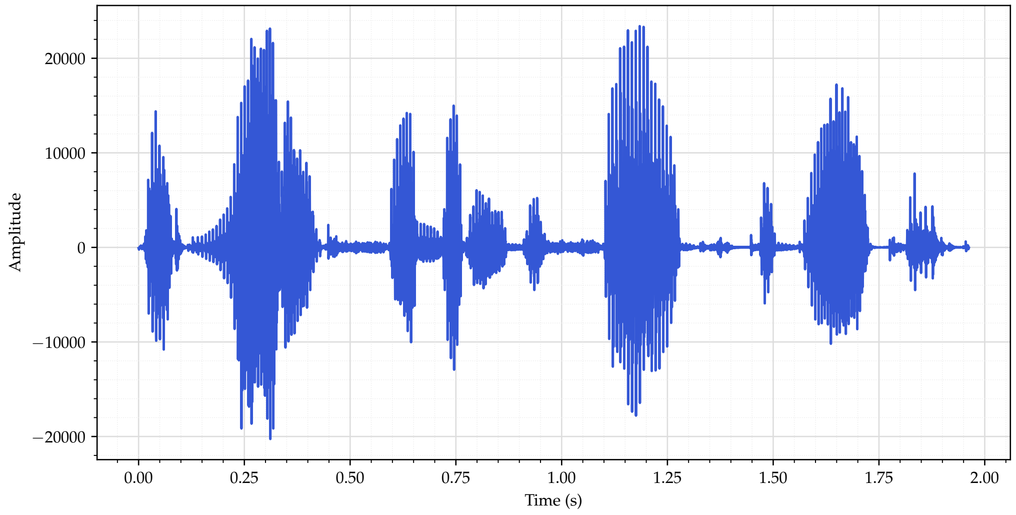

Figure 1: Amplitude of the speech signal over time.

Fourier Transform

The Fourier Transform allows us to analyze the frequency components present in the speech signal by breaking it down into its constituent frequencies. It serves as the basis for more advanced feature extraction processes that rely on the spectral components of the signal.

Mathematically, given a time-domain signal $x(t)$, the continuous Fourier Transform $X(f)$ is defined as:

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft}\, dt$$

where f represents frequency, and j is the imaginary unit. However, computers store and work with digital (discrete) signals, so the Discrete Fourier Transform (DFT) is used instead. The DFT transforms a sequence of N samples $x[n]$ into its frequency-domain representation $X[k]$, where k represents discrete frequencies:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi nk/N}$$

The following figure is a representation of the DFT of the signal in figure 1.

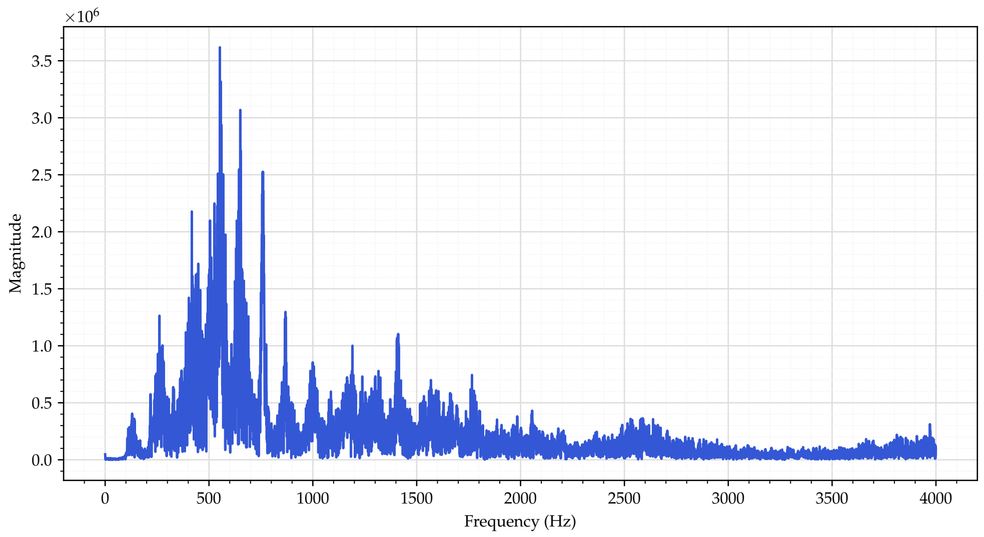

Figure 2: Frequency Spectrum of the speech signal.

Short-Time Fourier Transform

The frequency content of a signal often varies over time. In speech signals this is particularly true as the pitch and formants fluctuate due to phonetic transitions, prosodic variations, and speaker characteristics. This variability complicates the interpretation of the Discrete Fourier Transform (DFT), as it describes frequency components in the signal averaged over all time.

Therefore, in practice, the Short-time Fourier Transform (STFT) is employed. The STFT divides the signal into short, overlapping segments and computes the Fourier Transform for each segment independently. This is accomplished by multiplying the original signal, $x(t)$, by a window function, $w(t)$, that is brief in duration. A common choice for the window length N is around 25 milliseconds, with each window separated 10 milliseconds. Mathematically,

$$X(t, f) = \int_{-\infty}^{\infty} x(\tau) \cdot w(\tau - t) \cdot e^{-j2\pi f \tau} \, d\tau$$

where $w(\tau - t)$ weights the signal around the time t, and $X(t, f)$ represents the time-frequency representation of the signal. Again, to deal with digital signals, the discrete version of the STFT has to be employed:

$$X[m, k] = \sum_{n=0}^{N-1} x[n] \cdot w[n - m] \cdot e^{-j2\pi kn/N}$$

where $X[m, k]$ represents the complex-valued STFT coefficients at time index m and frequency index k. Applying it to three disjoint 400-sample segments on the example, the STFT looks as follows.

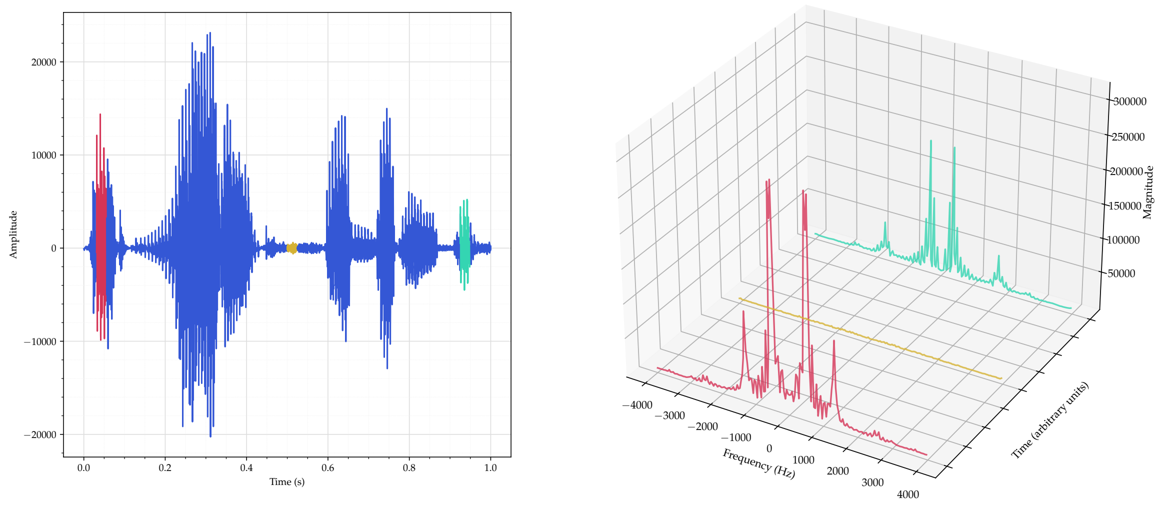

Figure 3: Segments of the original signal whose STFT has been computed (left), STFT with colours matching the highlighted segments (right).

Figure 3 (right) presents the DFT of three random 200-sample segments from the original signal, weighted by a Hamming window. They are snapshots of the signal's frequency content over specific time intervals. However, to gain a comprehensive view of how the frequency components evolve over time, this analysis is extended to include all segments of the signal. To enhance visualization, each segment's DFT multiplied by the corresponding window function is color-coded by its magnitude. Brighter colors indicate higher amplitudes, while darker hues represent lower amplitudes. This spectral representation of the signal's frequency content evolving through time is commonly known as a spectrogram.

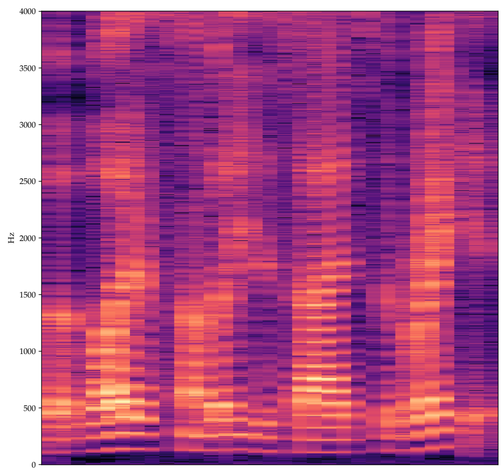

Figure 4: Spectogram of the speech signal.

Mel-scale Spectrograms

The representation of the frequency content of a signal that traditional spectograms offer does not entirely align with the human perception of sound.

The Mel scale introduces a nonlinear frequency scale more closely related to the human auditory system's response. It is defined based on psychoacoustic experiments that measure how humans perceive the distance between different frequencies, and can be approximated with:

$$M(f) = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right)$$

where f is the frequency in hertz and $M(f)$ is the frequency in Mel.

To obtain the Mel-scale Spectogram, a set of filters known as a Filterbank is applied to each DFT in the spectrogram. The Filterbank is constructed by evenly distributing a defined number of triangular filters across the Mel scale, each with a specified width.

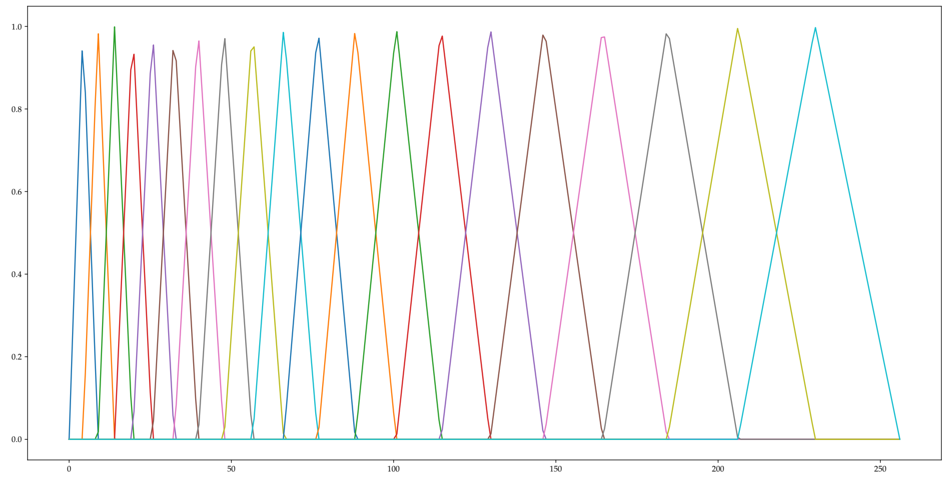

Figure 5: Sample Filterbank of 20 filters of width 257 that span frequencies between 0 and 4,000 for signals sampled at 8 kHz.

For each DFT in the spectrogram, the dot product is computed with each filter in the Filterbank. This process results in a reduced set of points representing the energy within perceptually relevant frequency bands for each time frame.

Optionally, a logarithmic function can be applied to the resulting Mel-scale spectrogram to compress the dynamic range, further aligning it with human auditory perception.

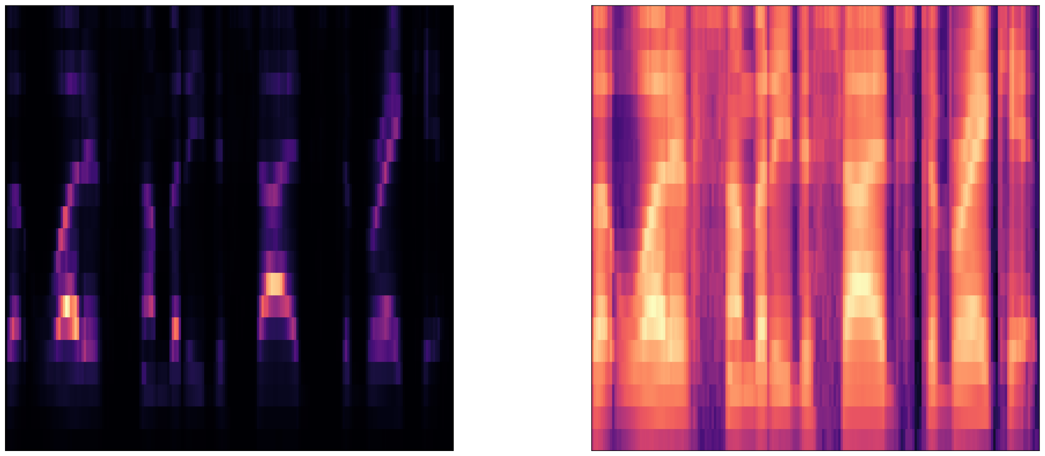

Figure 6: Mel-scale Spectrograms of the sample speech signal without taking the logarithm (left), and taking the logarithm (right).

Mel-frequency cepstral coefficients

Mel-frequency cepstral coefficients (MFCCs) built on the concepts explained above to, finally, capture the essential characteristics of a speech signal in a compact and discriminative manner. They are computed by applying the Discrete Cosine Transform to each column the logarithm of the Mel-Scale Spectrogram.

The DCT essentially decomposes the input signal into a weighted sum of cosine functions, each with a different frequency. In particular,

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot \cos\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right)$$

where $X[k]$ represents the k-th DCT coefficient, $x[n]$ is the input signal, and N is the length of the sequence. A finite number of coefficients is computed (eg. 13).

MCFFs don't capture how the speech signal *changes* over time. In such cases, an approximation of its first and/or second derivatives can be computed as follows:

$$\Delta c[n] = \frac{1}{2}(c[n+1] - c[n-1])$$
$$\Delta^2 c[n] = \frac{1}{2}(\Delta c[n+1] - \Delta c[n-1])$$

Dynamic time warping

Dynamic Time Warping (DTW) is a technique used to measure the similarity between two sequences, particularly time series data, that may vary in length and speed. It aims to find an optimal alignment between two sequences by stretching or compressing one of the sequences in time to minimize the overall distance between them. This makes it suitable for tasks where traditional measures of similarity, such as Euclidean distance, are inadequate due to temporal variations. In the case of the assingment, DWT will be used to compare the MFCCs of different signals in order to make predictions.

Working Principle

The fundamental idea behind DTW is to find the optimal warping path that minimizes the cumulative distance between corresponding elements of the two sequences. This is achieved through dynamic programming, where a matrix of accumulated distances is computed and the optimal path is traced back from the bottom-right corner to the top-left corner of the matrix.

Mathematical Formulation

Let $Q = \{q_1, q_2, ..., q_n\}$ and $C = \{c_1, c_2, ..., c_m\}$ be two sequences of lengths n and m respectively. The local distance between two elements q_i and c_j is typically calculated using a distance metric such as Euclidean distance:

$$d(q_i, c_j) = (q_i - c_j)^2$$

The accumulated distance matrix D is computed iteratively using the following recursion:

$$D(i, j) = d(q_i, c_j) + \min(D(i-1, j), D(i, j-1), D(i-1, j-1))$$

The optimal distance $DTW(Q, C)$ between sequences Q and C is the minimum cumulative distance along the optimal warping path.

Results

The results are described in the following sections.

Section 1

Here's an improved version of the text:
The error rate for the `free10x4x4` dataset, when `same_spk=False`, is 41.9%. However, when we set `same_spk=True`, the error rate significantly decreases to 12.5%. Furthermore, if we modify the code to detect the speaker rather than the word, the error rate remains relatively low at 14.4%. This indicates that enabling `same_spk=True` leads to a decrease in error because we obtain multiple recordings from each speaker and number.

It's evident that recordings from the same speaker and the same number exhibit the highest similarity, followed by recordings of the same digit but different speakers. Conversely, recordings of different digits from the same speaker show the lowest similarity. To further illustrate this effect, we constructed a smaller dataset, `free10x4x1`, containing only one recording per speaker-digit pair. Remarkably, the SER turned out to be 75%. This increase was primarily due to the DTW algorithm pairing recordings based on digits rather than speakers (19 out of 30 errors were of this nature).

Section 2

Various numbers of cepstral coefficients were experimented with. In the `commands10x100` dataset, optimal performance was achieved with 9 cepstral coefficients (see Figure 1). As the number increased or decreased, the WER worsened. In contrast, the performance of the `free10x10x4` dataset exhibited less stability. Nonetheless, it was observed that as the number of cepstral coefficients increased beyond 12, the WER consistently rose.

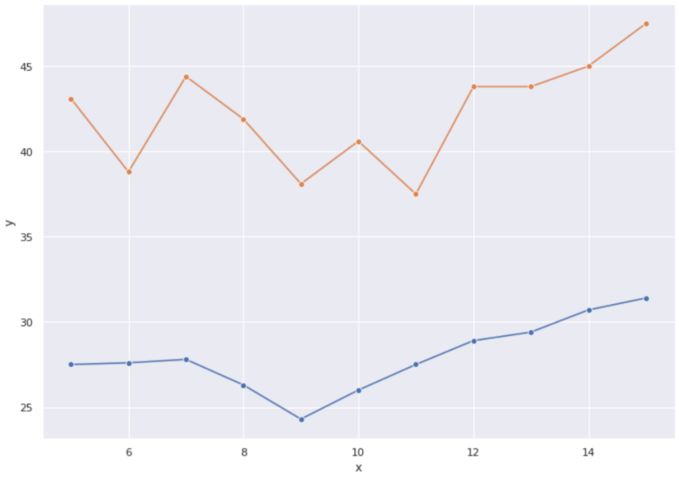

Figure 7: WER of `free10x4x4` (orange) and `commands10x100` (blue) for different number of cepstral coefficients.

Section 3

In the following tables we see the effect of activating and deactivating the normalization stages (mean and variance) with or without lifering (0 versus 22) for the dataset `commands10x100`. The best results are with the default values (no liftering and with normalization, see Table 2), followed by liftering with mean normalization but no variance normalization, see Table 1. Liftering compensates for the lack of normalization. Finally, when there is no liftering, it is imperative to normalize, or the results turn out to be very poor.

		cms	
		True	False
cmvn	True	38.2%	-
	False	28.2%	41.4%

Table 1: Results with liftering for dataset `commands10x100`.

Liftering is a technique commonly used in speech processing, particularly in the context of cepstral analysis. It involves applying a lifter function to the cepstral coefficients after they have been computed from the speech signal. The purpose of liftering is to emphasize certain coefficients while de-emphasizing others. It helps to enhance important

		cms	
		True	False
cmvn	True	24.3%	-
	False	39.3%	54.6%

Table 2: Results without liftering for dataset `commands10x100`.

characteristics of the signal by selectively amplifying certain coefficients.

Note: Setting `cmvn=True` and `cms=False` is equivalent to setting `cmvn=True` and `cms=True`.

Section 4

The MFCC parameters have been extended with the first-order derivatives. The code has been modified as follows:

```python
def dif(M):
    M_dif = []
    for i in range(len(M)):
        M_dif_i = []
        for j in range(1, len(M[i]) - 1):
            M_dif_i.append((M[i][j + 1] - M[i][j - 1])/2)
        M_dif.append(M_dif_i)
    return M_dif

# Compute MFCC
def get_mfcc(dataset, **kwargs):
    mfccs = []
    for sample in dataset:
        sfr, y = scipy.io.wavfile.read(sample['wav'])
        y = y/32768
        S = mfsc(y, sfr, **kwargs)
        # Compute the mel spectrogram
        M = mfsc2mfcc(S)

        M_dif = dif(M)
        M = np.concatenate((M, M_dif), axis=1)

        M = M.T
        mfccs.append(M.astype(np.float32))
    return mfccs
```

The WER of `commands10x100` has gone from 24.3% to 24.1% (in the submissions, it has gone from 84% for the baseline model whereas 85.6% for the model that includes first derivatives).

Section 5

```
path = {'x':[], 'y': []}

    while i > 0 and j > 0:
        path['x'].append(i)
        path['y'].append(j)
        min_option = min(D[i-1, j], D[i-1,j-1], D[i, j-1])
        if D[i-1, j] == min_option:
            i = i-1
        elif D[i, j-1] == min_option:
            j = j-1
        else:
            i, j = i-1, j-1
    path['x'].append(i)
    path['y'].append(j)
```

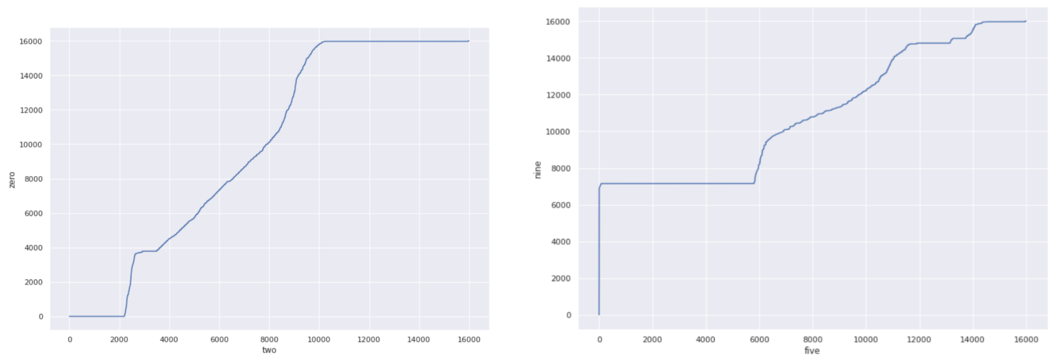

Figure 8: Examples where the test and the reference exhibit different numbers.

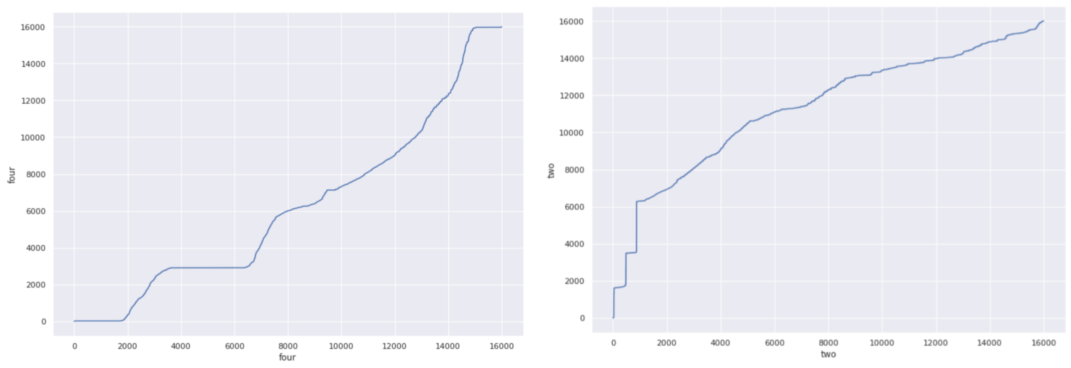

Figure 9: Examples where the test and the reference exhibit the same number.

Figure 8 shows two examples where the reference has a different number from the test audio, whereas Figure 9 portrays two instances where the reference and test numbers matched. One of the instances of mistaken pairs is "five" and "nine," which possess highly similar sound patterns.

This visualization enables us to observe how the alignment is being generated, and potentially identify the reasons behind certain errors.

Section 6

Many variants of DWT are to be found across the literature. In this section some of them will be implemented and compared on a common benchmark. The algorithm described in will be the baseline.

DTW - type 1

In this version of the DTW, the recursion that fills the cost matrix is as follows:

$$D(i,j) = d(q_i, c_j) + \begin{cases} \min D(i-2, j-1) + \eta \\ \min D(i-1, j-2) + \eta \\ \min D(i-1, j-1) \end{cases}$$

with η a penalty parameter for paths deviated from the 45-degree path.

Derivative Dynamic Time Warping

Derivative Dynamic Time Warping or DDTW for short, proposed in [2], first converts the original time series into a high level feature using the first-order derivative. This operation, for a point q_i, within a time series of length n, is defined as follows:

$$D'(q_i) = \frac{(q_i - q_{i-1}) + \frac{q_{i+1} - q_{i-1}}{2}}{2}, \quad 1 < i < n$$

Then classic DTW is applied to D'.

Comparison

The following table summarized the results obtained.

model	*Elapsed time*	*WER*	*Accuracy*
baseline	8min 49s	23.1%	81.00%
DTW-type1, $\eta = 0$	7min 18s	43.4%	79.33%
DDTW	39min 25s	-	14.33
match_anywhere	-	25.4%	85.00%

Table 3: Elapsed time, WER and accuracy obtained with variations of the baseline model.

Conclusions

Several variants of the DTW algorithm were implemented and compared on a common benchmark.

DTW-type1 with $\eta = 0$ demonstrated a notable increase in word error rate (WER) compared to the baseline, yet decreased in accuracy. This might be attributed to the absence of parameter tuning for η, highlighting the importance of hyperparameter optimization for better performance.

Derivative Dynamic Time Warping (DDTW), despite its theoretical appeal, exhibited disappointing results in our context. The extensive computation required for derivatives likely contributed to its prolonged training time.

In contrast, the `match_anywhere` variant showed promise, surpassing the baseline model in terms of accuracy and WER. By allowing alignment to start and end at 5 units fom the cost matrix corners, it proved beneficial for scenarios where audio segments may not perfectly align at fixed corners.

Overall, while some DTW variants offered enhancements over the baseline, careful parameter tuning and domain-specific considerations are crucial for maximizing their effectiveness. Further exploration with larger datasets and refined parameter settings could yield deeper insights into the comparative performance of DTW variants across different applications.

References

[1] Signalogic. *Speech Codec Wav Samples Keywords: Voice Codec Samples, Codec Sample Data, WAV File Samples*. URL: https://www.signalogic.com/index.pl?page=speech_codec_wav_samples (visited on 05/15/2024).

[2] Tomasz Górecki and Maciej Łuczak. "Using derivatives in time series classification". In: *Data Mining and Knowledge Discovery* 26 (2013), pp. 310–331.