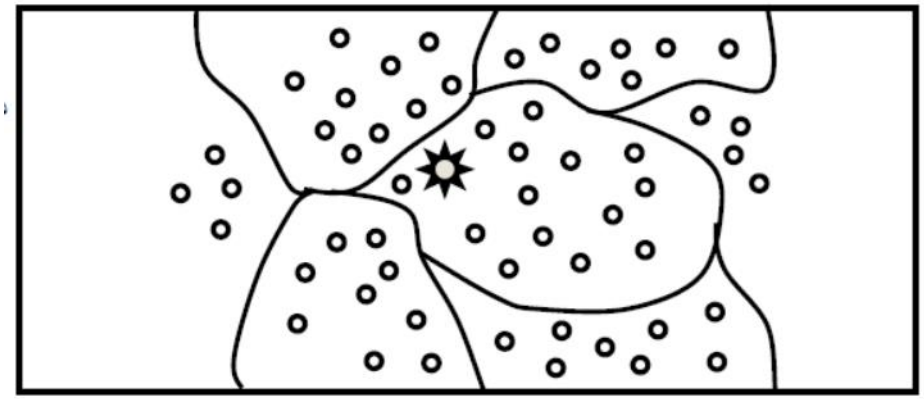
Búsqueda en audio

1. Representación de la señal
2. Spoken Term Detection
3. Query-by-example

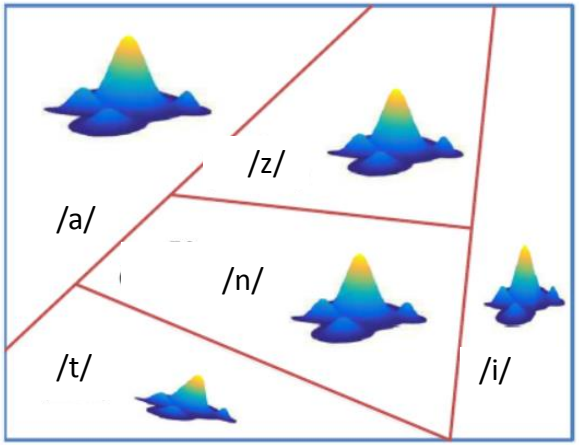
Representación de la señal

Cepstral Coefficients



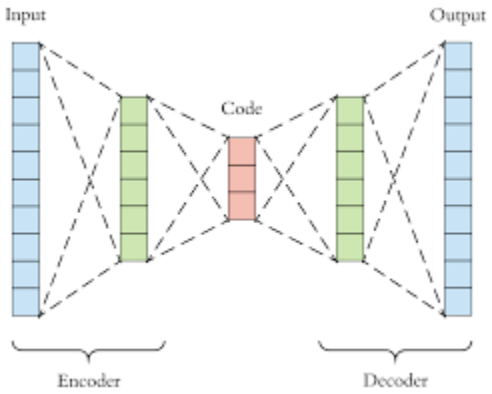
Phonetic posteriorgrams

Se utilizan modelos fonéticos “universales” para dividir el espacio de características acústicas. Cada *frame* se representa por un vector de posteriors correspondiente a los distintos fonemas considerados.



Bottelneck features

Mediante las bottleneck se obtiene una representación en un espacio de menor dimensionalidad de los vectores de características acústicas (o de los vectores de posteriorgrams fonéticos)

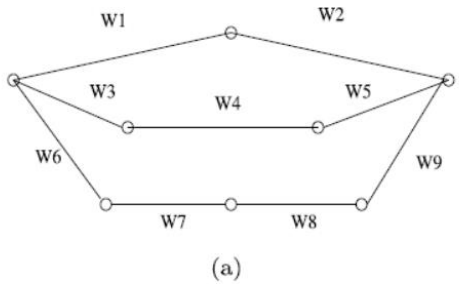


Spoken Term Detection

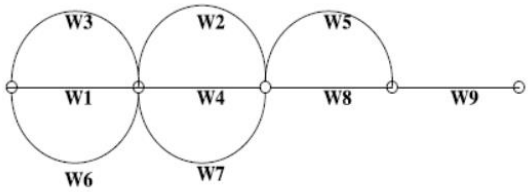
Problema: Dada una palabra escrita, encontrar sus apariciones en un audio.

Palabras in-vocabulary (INV): Se puede usar un sistema de Reconocimeinto de habla que genere hipótesis sobre las palabras que aparecen en el audio: 1-best o lattices. Posteriormente un módulo de deteción/decisión genera la solución.

Lattices



Word Confusion Network



Palabras fuera del vocabulario (OOV): Se pueden utilizar

- Reconocedores de unidades subléxicas (fonemas, sílabas)

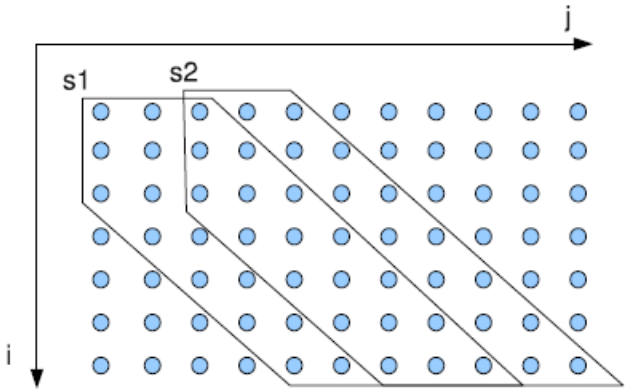
- Búsqueda de proxy-words, es decir palabras del vocabularios que se parecen fonéticamente a la palabra buscada.

Otra aproximación válida en ambos casos: Acoustic keyword spotting, es decir utilizar el modelo acústico de la palabra buscada (por ejemplo el HMM formado con la concatenación de sus fonemas), y buscar los segmentos que hacen matching con ese modelo (sin información de modelo de lenguaje).

Query by example

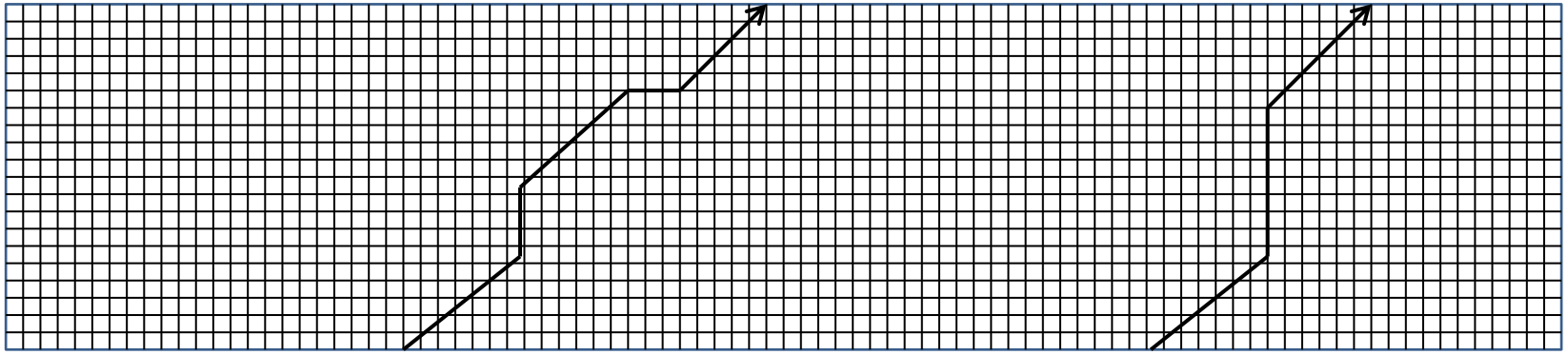
Problema: Dada la señal acústica correspondiente a una pronunciación de una palabra, segmento, o frase, buscar en un audio, las distintas apariciones de esa palabra, segmento, o frase.

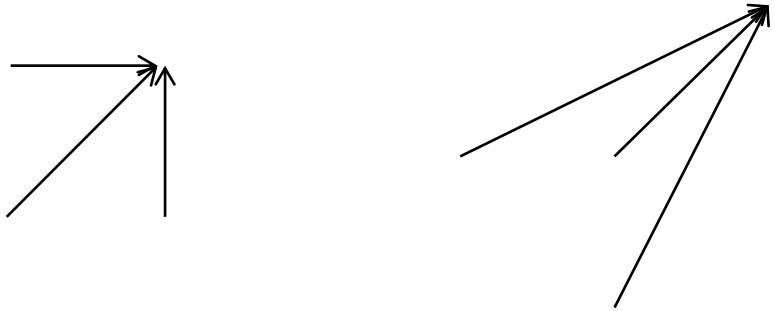
Segmental DTW



Inconveniente: Elevado coste computacional $O(n^2.m)$

SubsequenceDTW





(1,1,1) Subsequence DTW Sin normalizar

$$M_{i,j} = \begin{cases} c\left(q_i, u_j\right) & \text{if} \quad i = 0 \\ c\left(q_i, u_j\right) + M_{i-1,0} & \text{if} \quad i > 0,\ j = 0 \\ c\left(q_i, u_j\right) + M^*(i,j) & \text{else,} \end{cases}$$

$$M^*(i,j) = \min\left(M_{i-1,j}, M_{i-1,j-1}, M_{i,j-1}\right)$$

Distancias:

- Euclidea
- Coseno
- Kullback-Leibler
- Pearson

(1,1,1) Subsequence DTW normalizando

$$M_{i,j} = c\left(q_i, u_j\right) + M\left(i - x', j - y'\right)$$

$$\left(x', y'\right) = \arg\min_{(x,y)} \frac{M(i - x, j - y) + c(q_i, u_j)}{L(i - x, j - y) + 1}$$

Donde $c(q_i, d_j)$ representa la distacia entre las frames correspondientes del query y del audio y L es la longitud del camino hasta ese punto.

Evaluación

Actual Term-Weighted Value (ATWV)

$$ATWV = \frac{1}{|\Delta|} \sum_{K \in \Delta} \left(\frac{N_{\text{hit}}^K}{N_{\text{true}}^K} - \beta \frac{N_{\text{FA}}^K}{T - N_{\text{true}}^K} \right)$$

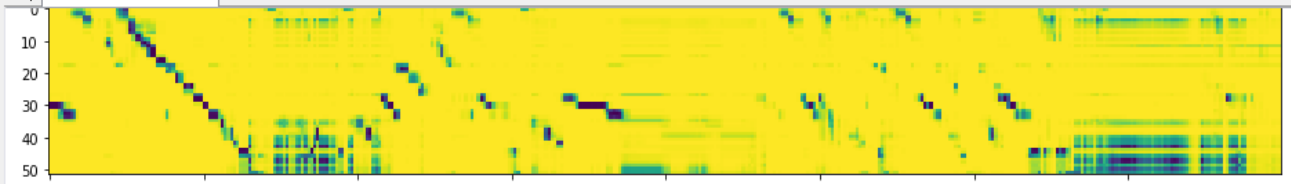
Donde

- Δ representa el conjunto de queries y | Δ | es el número de queries de ese conjunto.
- N^K_{hit} y N^K_{FA} representan el número de hits y de falsas alarmas , respectivamente, para el query K.
- N^K_{true} es el número de ocurrencias de K en el audio.
- T es la longitud del audio en segundos.
- β es un factor de pesado, cuyo valor propuesto por el NIST es 999.9, y que enfatiza el recall frente a la precisión en un ratio de 10:1.

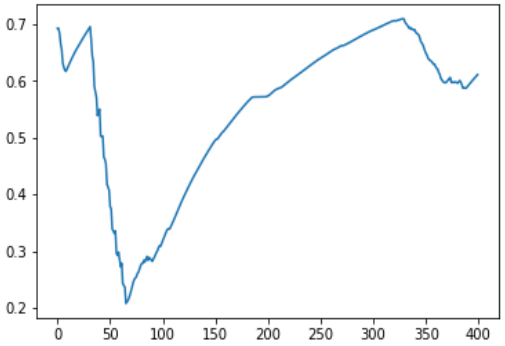
El ATWV representa el Term-Weighted Value (TWV) para el umbral usado por el sistema para decidir qué respuestas se incluyen en la solución, y que se determina habitualmente a partir del development set. Existe una métrica adicional, que es la Maximum Term-Weighted Value (MTWV) que es la major ATWV que el sistema proporcionaría si escogiera el umbral óptimo.

Terminal de IPython

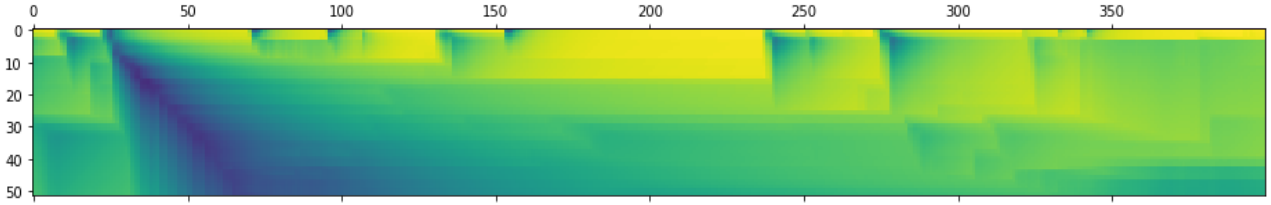
Terminal 1/A

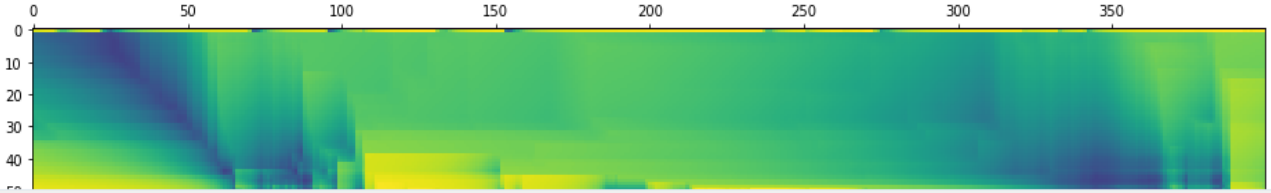


```
52 30000
(52, 30000, 4)
Elapsed time: 31.3820002079 seconds.
```

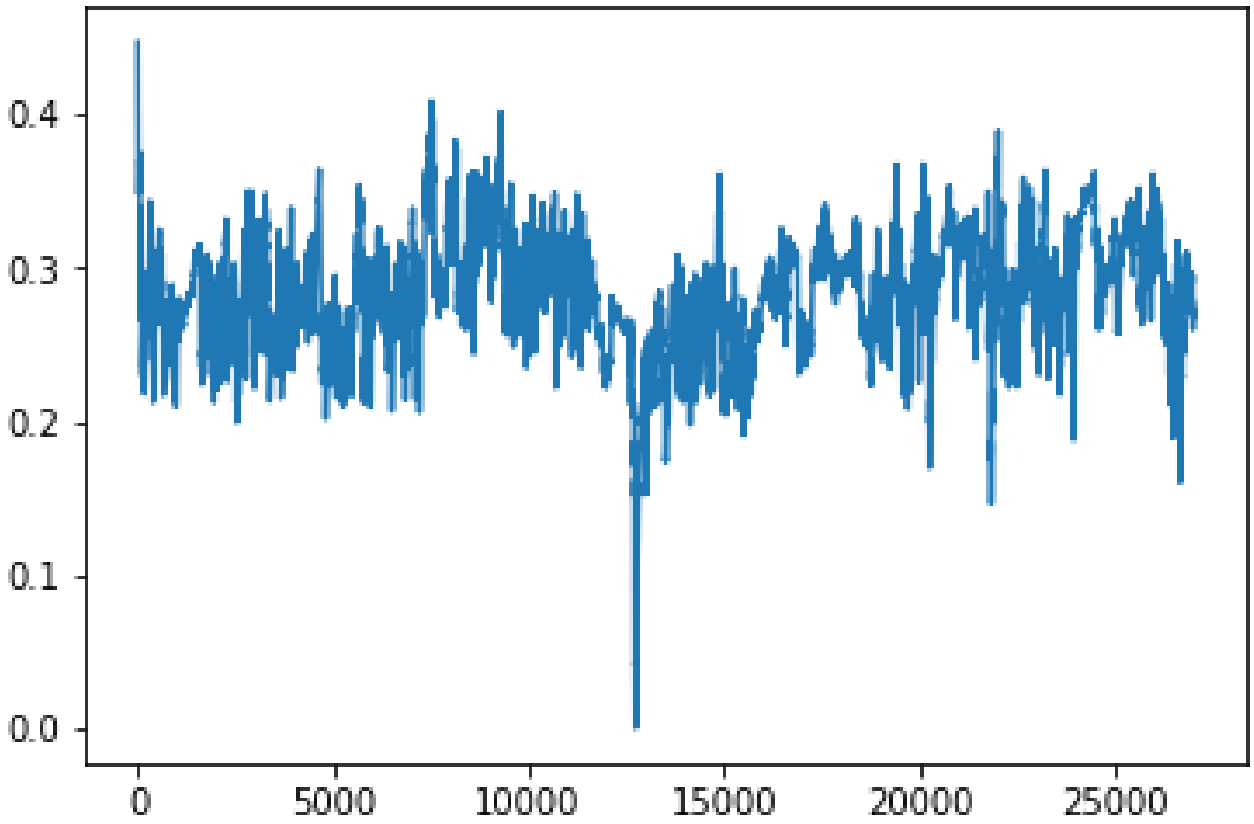


```
(52, 30000)
```





Permisos: **RW** | Fin de línea: **CRLF** | Codificación: **UTF-8** | Línea: **62** | Columna: **20** | Memoria: **54 %**




```
(52, 30000)
```