Clasificación y verificación del hablante: experimentación

Antonio Bonafonte & Santiago Pascual

Resumen

En el diseño de un sistema de tecnología del habla suele requerirse una optimización de los parámetros del sistema a los datos que se disponen, ya que los parámetros óptimos suelen depender de la cantidad y del tipo de datos de que se dispone. Este curso, se propone que los distintos grupos de laboratorio se distribuyan el análisis de algunos parámetros o variantes del sistema de clasificación, incluyendo un clasificador de redes neuronales.
El último trabajo propone el uso de GMM para dados unos segmentos de audio sin etiquetas, definir cuántos locutores distintos hay y asignar segmentos según locutor. Los trabajos están ordenados, aproximadamente, por orden de dificultad. Los primeros no requieren desarrollar nuevas funciones pero sí la experimentación es laboriosa y computacionalmente costosa.

Se propone partir de un sistema de referencia que utilice los parámetros por defecto de la función `mfcc` del paquete `sptk` y comparar con las distintas variantes:

Las opciones de análisis que se proponen son (por orden de dificultad, aunque las primeras son laboriosas experimentalmente):

1. Optimizar los parámetros de un sistema basado en LPCC (cepstrum LPC): orden lpc, número de coeficientes cepstrum, y si tiene tiempo, preénfasis, duración y desplazamiento del tramo, tipo de ventana.

2. Optimizar los parámetros de un sistema basado en MFCC: número de filtros, número de coeficientes cepstrales, y si tiene tiempo, preénfasis, duración y desplazamiento del tramo, tipo de ventana.

3. Como el anterior, pero con la función `mcep`.

4. Análisis de GMM. Analizar como influye en la probabilidad el método de inicialización, el número de gausianas, las iteraciones. Modificar el programa de entrenamiento para que reserve parte de los datos de entrenamiento para validación e informe en cada iteración de probabilidad en datos de validación, además de en entrenamiento. Estudiar si guarda relación la mayor probabilidad con los resultados de clasificación/verificación. Repetir también varios entrenamientos de GMM (ejemplo, repetir 40 veces y estudiar las prestaciones del que da mejor y peor probabilidad).

5. Incorporar características dinámicas (Δ y Δ^2). Implementar su cálculo y estudiar en sistemas que usen estas características por separado, o de forma conjunta (con vectores mayores).

6. Este trabajo se apoya en el anterior. Una vez calculados los parámetros se utilizan varias GMM por hablante: una para coeficientes estáticos, otra para Δ, otra para Δ^2. En clasificación/verificación se ponderan los resultados obtenidos por cada GMM para la decisión. Puede realizarse experimentalmente o ajustar en datos de entrenamiento una regresión lineal.

7. Utilizar prácticas anteriores para analizar efecto de prescindir de tramos de silencio, de prescindir de tramos sordos y en ese caso, añadir pitch (se suele utilizar $\log F0$, al tener pdf más gausiana).

8. `PCA`: Se forman vectores de K tramas consecutivas (con desplazamiento una trama). Se aplica `pca` y `pcas` para reducir la dimensionalidad y correlación entre coeficientes.

9. Clasificación utilizando DeepLearning: experimentar con un *script* básico para *Keras* (número de neuronas, capas, función activación),

10. Clasificación utilizando DeepLearning: experimentar con un *script* básico para *Keras* (número de neuronas, capas, función activación),

11. Adaptación de GMM: tal y como comenta el paper, la mayoría de los sistemas suele partir un GMM general, para el conjunto de hablantes y realizar una adaptación MAP para tener el GMM de cada hablante.

12. *Clustering de locutores*. Esta ampliación introduce un problema distinto a la clasificación o verificación: el *clustering* o agrupación de segmentos de voz que pertenecen a un locutor. Se trata de dado un conjunto de segmentos de voz, detectar cuántos hablantes distintos hay, y asignar los segmentos a los hablantes. Se propone utilizar el criterio BIC y metodología del paper *Speaker, environment and channel change detection and clustering via the bayesian information criterion*, de Scott Chen, 1998 (apartado 4, y 2). Se aplicarán a todos los ficheros de *speecon* descartando previamente la identidad del hablante, que sólo se utilizará para evaluar.

Para los que dessen iniciarse en el DeepLearning, usando una red neuronal *fully connected* para clasificación, hemos preparado el siguiente prototipo, disponible en `github`[1] que una vez instalado python y pytorch debería funcionar.

https://github.com/santi-pdp/pav_spkid_pytorch

Para copiarlo, o bien descargar `.zip` o ejecutar
`git clone` https://github.com/santi-pdp/pav_spkid_pytorch

En el propio **README** hay algunas sugerencias de experimentación.

Opciones más avanzadas: utilizar redes recurrentes o convolucionales; abordar el problema de verificación.

Se puede ejecutar sin mirar el código, pero os animamos a intentar entenderlo (y con él, `pytorch`), incluso si no conocéis `python`.

[1] Si no conoces `git` es una oportunidad para empezar: es un software de control de versiones muy utilizado en proyectos de desarrollo, que permite evolucionar software, recuperando versiones anteriores, compartiendo entre usuarios, etc. Si ponéis vuestros proyectos en `github pueden ser más visibles, mostrarlos al buscar trabajo, etc.`
`Aprovecho para pediros que pongáis` *`estrellas`* `a los proyectos que os gusten!`