

Instituto tecnológico de Massachusetts
Departamento de ingeniería eléctrica e informática

6.345 Reconocimiento automático del habla
Primavera 2003

Publicado:11/04/03
Entregar : 23/04/03

Trabajo 9
Introducción al sistema de reconocimiento de voz SUMMIT

Introducción

En esta práctica entrenaremos y probaremos un sistema de reconocimiento de voz al completo, utilizando las oraciones del entorno **Pegasus** (solicitud de información sobre el estado de los vuelos). Procederemos paso a paso a través de todo el proceso, examinando la información/datos requerida en cada etapa, construyendo/entrenando los modelos necesarios y evaluando el rendimiento y el comportamiento de los mismos. En el proceso, usted se familiarizará con los componentes del sistema de reconocimiento de voz SUMMIT, así que podrá utilizarlo como motor de reconocimiento en su proyecto final. Al igual que en anteriores trabajos, las siguientes tareas (marcadas con **T**) deberían completarse durante su sesión de prácticas. Las respuestas a las preguntas (señaladas con **P**) deberían entregarse dentro del plazo correspondiente.

Software

En esta práctica utilizaremos un juego de programas que componen el sistema de reconocimiento SUMMIT. Utilizaremos la versión 3.7 del sistema especificado, ajustando la variable de entorno SLS_HOME a /usr/sls/sls/v3.7. Observe que esto se realiza automáticamente en el archivo .cshrc de los cómputos de la clase.

En el apéndice de las **herramientas de SUMMIT**, que se encuentra al final de esta fotocopia, se facilita un listado descriptivo de los programas utilizados en esta práctica. Rogamos examine la descripción de los programas antes de comenzar la práctica. Sea consciente también de que algunos de los componentes más intensivos computacionalmente están diseñados para ejecutarse de modo distribuido, utilizando el *servicio de envío del host* del grupo SLS. Examine la sección de procesamiento distribuido del apéndice.

Cómo comenzar

Para comenzar con esta práctica, escriba el siguiente comando en el la ventana de símbolo del sistema de UNIX:

```
% start_lab9.cmd
```

Esto creará un nuevo subdirectorio bajo su directorio local llamado `lab9` y lo rellenará con el grupo de archivos necesarios para esta práctica. Estos archivos contienen datos, modelos y especificaciones del parámetro utilizados durante la práctica. Las descripciones de los archivos se encuentran en el apéndice de los **archivos de SUMMIT** al final de esta fotocopia. En la primera parte de la práctica, observaremos algunos de estos archivos detalladamente.

Parte I: Archivos de datos

En esta parte de la práctica, estudiaremos los distintos archivos de datos necesarios para construir un sistema de reconocimiento de voz. Concretamente, nos centraremos en los siguiente:

- La partición de un corpus en entrenamiento inconexo, desarrollo y grupos de prueba.
- Formas de onda del audio.
- Transcripciones a nivel fonético y de palabra.
- Léxicos y diccionarios de pronunciación.

Corpus

El sistema SUMMIT utiliza un archivo de **corpus** que contiene especificaciones de todos los datos para varios entornos. El archivo de corpus ha sido ya creado y lo puede encontrar dentro del archivo `galaxy.corpus`. El programa `corpus tool` se puede utilizar para examinar las propiedades y grupos definidos en el archivo de corpus.

T1: Muestre el uso del programa `corpus tool` facilitando la opción de la línea de comandos `-help`. El corpus está dividido en varios grupos y, para cada uno, existen varios subgrupos. Un subgrupo corresponde a un único hablante (o llamada telefónica) y contiene varios enunciados.

Utilice la opción `-list sets` del programa para examinar la partición del corpus en distintos grupos de datos:

```
% corpus_tool -corpus galaxy.corpus -list_sets
```

Esto mostrará una lista de todos los grupos representados en `galaxy.corpus`. Fíjese en que los grupos para el dominio **Pegasus** se llaman `p *`. Utilice la opción `-set` para mostrar todos los subgrupos disponibles dentro de un grupo concreto:

```
% corpus_tool -corpus galaxy.corpus -set p_1
```

Igualmente, puede especificar el nombre de un subconjunto en el argumento `-set`. Examine sucesivamente “grupos de datos” más pequeños hasta que alcance la lista de enunciados individuales o “datum”. Utilice la opción `-datum` para ver las propiedades de un enunciado real. Por ejemplo:

```
% corpus_tool -corpus galaxy.corpus \  
-datum p_pegasus-phone-258-0301-19990107-006-000
```

Esto mostrará las propiedades para el datum individual y el lugar del archivo de la forma de onda.

- P1:** (a) ¿En cuántos grupos de datos distintos se ha dividido el corpus?
(b) ¿Cuántos hablantes (o llamadas telefónicas) existen en cada grupo **Pegasus**?
(c) ¿Cuáles son las propiedades definidas en este corpus?

Formas de onda

Tenga en cuenta que una de las propiedades del archivo de corpus es la especificación del lugar del archivo de la forma de onda del audio (`waveform file`) para cada enunciado (datum) del corpus. Estos archivos de forma de onda contienen el discurso digitalizado. Se guardan en formato de forma de onda NIST (`.wav`) con una cabecera ASCII de 1024 bytes. Podemos ver la información de la cabecera si utilizamos los comandos de localización estándar de UNIX: `more` o `less`. Podemos escuchar el archivo de audio con el programa `waveform play`.

T2: Observe la cabecera de uno de los archivos de la forma de onda. Utilice `waveform play` para oír la forma de onda. (Acuérdese de utilizar `-help` si necesita ver la función). Compare lo que oye con la transcripción a nivel de palabras proporcionada por la utilidad `orthography`.

Observe que estas oraciones fueron transcritas ortográficamente por un ser humano. Una de las tareas que más tiempo consumen en el desarrollo de un sistema de reconocimiento de voz es la recopilación y preparación de los datos necesarios para el entrenamiento y las pruebas.

Léxicos

Además de las transcripciones de palabras y las formas de onda de la voz, se requiere alguna información más para construir un sistema de reconocimiento de voz. Esto incluye el vocabulario o lista de palabras que el sistema puede reconocer y una transcripción fonética de las palabras. La construcción del vocabulario puede convertirse en una tarea bastante tediosa, si se tiene en cuenta la amplia variedad de preguntas que los usuarios pueden inquirir normalmente. La lista de palabras del vocabulario para esta práctica está en el archivo `pegasus.vocab`.

Una vez definido el vocabulario, debe precisarse la pronunciación de la forma base (o canónica) para cada palabra del vocabulario.

El modo más fácil de obtener las pronunciaciones de la forma base es buscándolas en un diccionario. Utilizamos una herramienta llamada `baseform tool.pl` para crear nuestro archivo de pronunciación de forma base inicial.

T3: Genere un archivo de forma base automáticamente, ejecutando el siguiente comando:

```
% baseform_tool.pl -vocab pegasus.vocab \  
-dict sls_pronlex.syl \  
-convert_stops \  
-rules baseform.rules \  

```

```
-out pegasus.baseforms \  
-baseform galaxy.baseforms
```

Este comando `baseform tool.pl` busca cada palabra en `pegasus.vocab`, y busca también su pronunciación de forma base predefinida en `galaxy.baseforms`. Si no se encuentra la forma base, buscará la pronunciación en un diccionario de pronunciación `sls pronlex.syl`, y tratará de generar la forma base.

La opción de la línea de comandos `-convert stops` provoca que el script convierta todas las etiquetas oclusivas en una variante alofónica concreta dependiente del contexto, basada en los fonemas que rodean a la oclusiva, en la posición de la oclusiva en la sílaba y en el patrón de acentuación de las sílabas de alrededor. El argumento `-rules` especifica un grupo de reglas de reescritura que elimina la información de acentuación y aplica otras convenciones de unidades fonémicas SLS.

El grupo de unidades fonémicas utilizadas en los archivos de forma base SLS se encuentra en el apéndice del **grupo de unidades fonémicas**. Se darán algunos mensajes de advertencia si algunas palabras del vocabulario no se encuentran en el diccionario. Será necesario que introduzca manualmente estas pronunciaciones si esto sucede.

T4: Examine la lista de palabras del vocabulario en `pegasus.vocab`, y sus transcripciones fonéticas en `pegasus.baseforms`.

P2: (a) ¿Cuál es el tamaño del vocabulario de reconocimiento?

(b) En el archivo automáticamente generado `pegasus.baseforms`, examine las entradas de pronunciación. ¿Puede sugerir algunas mejoras para las pronunciaciones para que éstas puedan reflejar mejor las variaciones reales de pronunciación de los usuarios? Las transcripciones fonéticas básicas facilitadas en `pegasus.baseforms` se “amplían” (a múltiples pronunciaciones) mediante el uso de un grupo de reglas fonológicas. Estas reglas tienen en cuenta variaciones dependientes del contexto en la realización fonética de las palabras. Normalmente presentan el mayor impacto en límites de palabra, ya que la variación interna a las palabras está representada con frecuencia en la pronunciación de la forma base. Las reglas fonológicas se encuentran en el archivo `pegasus.pron.rules`.

T5: Observe las reglas de pronunciación en el archivo `pegasus.pron.rules`. Trate de comprender el significado de las reglas. Para esta práctica, no es necesario que conozca todos los detalles de las reglas.

Transcripciones

Para entrenar los modelos acústicos para las unidades fonéticas de subpalabra, lo cual llevaremos a cabo en la parte III de esta práctica, es necesario encontrar los segmentos correspondientes de la forma de onda acústica que se asocien con fonos concretos. Dicho de otro modo, necesitamos una transcripción fonética con alineación temporal para cada oración del grupo de entrenamiento. Un modo de hacer esto consiste en llevar a cabo lo que se conoce como *alineamiento obligatorio*. Para cada oración, la ejecutamos a través del reconocedor de voz mediante un modelo acústico existente y un modelo de lenguaje *altamente restringido*.

El modelo acústico puede ser de otro corpus, si estamos simplemente comenzando en un nuevo dominio y queremos autocargar un dominio existente. Otra posibilidad es que el modelo provenga de una vuelta de entrenamiento anterior del dominio actual, si estamos tratando de mejorar iterativamente el modelo acústico. El modelo de lenguaje está restringido para ser sólo la secuencia deducible de fonos para la secuencia de palabras determinada en esa oración de entrenamiento concreta.

El programa `fst trans compute` (utilice `-shelp` para mensajes de ayuda) se utiliza para realizar alineamientos obligatorios. Emplea especificaciones del parámetro en el archivo `paths.espec` (observe el apéndice de **archivos de SUMMIT** para la descripción del contenido y el formato de este archivo). Los archivos de transcripción de palabra obligatoria y de alineamiento fonético (`.wrđ` y `.phn`) para todas las oraciones de entrenamiento han sido ya creados para que **no** tenga que volver a crearlos. Se encuentran en

```
/t/summit/forced_paths/6.345/<group>/<tag>.wrđ  
/t/summit/forced_paths/6.345/<group>/<tag>.phn
```

donde `<group>` toma valores especificados en las correspondientes propiedades del archivo de corpus y `<tag>` es la etiqueta de ese enunciado (datum del corpus).

Examinaremos gráficamente el alineamiento de las palabras y fonos con la forma de onda para algunos enunciados, utilizando el programa visual de transcripción `tv`. Este programa utiliza las especificaciones del parámetro en el archivo `tv.espec`. El archivo `espec` existente seleccionará aleatoriamente 50 oraciones del grupo `p 3` que va a ser examinado.

T6: Observe varios de los archivos de transcripción fonética y de palabras con alineación temporal:

```
/t/summit/forced_paths/pegasus/<group>/<tag>.wrđ  
/t/summit/forced_paths/pegasus/<group>/<tag>.phn
```

Examinaremos en **Sapphire** el alineamiento de palabras y fonos con la forma de onda. El programa se invoca con el script de `tcl`, `tv.tcl`:

```
% tv -ctl tv.espec
```

Utilice la información del espectrograma junto con la capacidad para reproducir segmentos seleccionados de la forma de onda (Ctrl-V) que correspondan a las palabras y fonos, para evaluar la calidad de los alineamientos fonéticos y de palabra. Indudablemente, se producirán algunos errores en el alineamiento.

P3: Desde una perspectiva cualitativa, ¿cómo son los alineamientos fonéticos y de palabra? ¿Qué tipos de errores de alineamiento observó en su ejemplo de oraciones?

Parte II: Modelos de lenguaje

Para construir un modelo de lenguaje, el sistema necesita un archivo de vocabulario y un grupo de datos de entrenamiento. El sistema ofrece también la opción de utilizar un grupo de reglas que coloca las palabras del archivo de vocabulario en clases de palabras.

T7: Examine el archivo `train_sents.txt`. Estas oraciones de entrenamiento son transcripciones de preguntas espontáneas de llamadas actuales al sistema **Pegasus**. Sería aconsejable que encontrara numerosos casos de efectos de habla espontánea, como son las palabras parciales, las pausas de relleno (um, uh) y discurso agramatical presente en los datos de entrenamiento. Fíjese en que se han transcrito todos los eventos acústicos, y no sólo las palabras habladas.

T8: Examine el archivo `pegasus.class.rules`. Este archivo contiene las clases de palabra que se han definido para este dominio. ¿Podría pensar en cualquier otra clase que pudiera ser útil para modelar? Observe que los nombres comúnmente utilizados, verbos y palabras funcionales, no se colocan en clases. ¿Por qué es útil crear clases de palabras para objetos semánticos como ciudades y compañías aéreas?

Usted ha explorado algunos modelos de lenguaje en la **Tarea 6**. Aquí utilizaremos un script escrito en PERL para crear todos los modelos de lenguaje bigrama y trigrama necesarios para el reconocedor.

T9: Entrene los modelos de lenguaje de **clase** ejecutando el siguiente comando:

```
% ngram_create.pl -prefix pegasus -sents train_sents.txt \  
-rules pegasus.class.rules
```

Próximamente en esta práctica, creará los modelos de lenguaje de **palabra**, pasando por alto las reglas de clase de palabras especificadas en el archivo `pegasus.class.rules`. Tendrá la oportunidad de comparar modelos n-grama de **palabra** y modelos n-grama de **clase** en el reconocimiento.

Parte III: Modelos acústicos

En esta parte de la práctica, entrenaremos y examinaremos modelos acústicos. Haremos lo siguiente:

- entrenar una rotación de componentes principales y una matriz de escalamiento.
- entrenar modelos de gaussianas de covarianza diagonal mixta para los rasgos acústicos de los distintos difonos.
- observaremos algunas de las propiedades del modelo acústico entrenado.

Realizaremos esto para modelos de límite difono dependientes del contexto. El reconocedor también tiene la capacidad de especificar modelos acústicos de segmentos. Se utilizan procedimientos de entrenamiento similares para entrenar modelos de segmentos. El archivo que “dirige” el proceso de entrenamiento y reconocimiento se denomina archivo de reconocimiento o archivo “rec”. Para esta práctica, este archivo es `pegasus.rec`. Este archivo enumera los módulos de procesamiento y sus especificaciones de parámetros para cada paso del proceso de entrenamiento y pruebas. (Consulte el apéndice de las **herramientas de SUMMIT** para ver los indicadores de las descripciones de los módulos de procesamiento, y el apéndice de los **archivos de SUMMIT** para una descripción más detallada de este archivo).

T10: Observe el contenido del archivo `pegasus.rec` y trate de comprender su contenido.

P4: (a) ¿Cuántas muestras existen en un marco (ventana) de la transformada de Fourier de tiempo breve (STFT)?

(b) ¿Cuántos coeficientes MFCC se están computando?

(c) ¿Cuáles son los componentes del vector característico de 112 dimensiones que se está computando para cada límite de difono?

(d) ¿Cuál es el número máximo deducible de componentes de mezcla para cada modelo `mixture diagonal gaussian`?

Agrupamiento de difono

Dado que en esta práctica estamos utilizando datos de entrenamiento limitados del entorno **Pegasus**, algunos difonos no presentarán suficientes muestras de entrenamiento pertenecientes al grupo de entrenamiento. Para paliar un problema de datos escasos como éste, se utiliza un procedimiento automático de agrupamiento para permitir que difonos parecidos compartan el mismo modelo de mezcla de gaussianas.

Dado que lleva mucho tiempo realizar este procedimiento de agrupamiento, le facilitaremos un archivo de etiqueta difono preagrupado `pegasus.blabels`. En este archivo, todos los difonos de la misma línea comparten el mismo modelo acústico. Las herramientas utilizadas para el agrupamiento se explican en el apéndice de **agrupamiento de difonos**.

T11: Examine el archivo `pegasus.blabels`. Preste atención a los difonos que comparten el mismo modelo acústico. ¿Puede identificar sus similitudes?

Análisis de los componentes principales y escalamiento

Una rotación y escalamiento de los vectores característicos basados en el análisis de componentes principales, se realiza para deshacer la correlación de los componentes de los vectores característicos y ajustar su varianza a uno. Estas operaciones hacen a los rasgos más propicios al modelado por mezclas de gaussianas de covarianza diagonal. La matriz de transformación se computa una vez a partir de todos los datos de entrenamiento y se utiliza para transformar los datos de prueba y de entrenamiento.

T12: Compute la matriz de escalamiento (pcs) de los componentes principales de los datos de entrenamiento utilizando el programa `train pcs` como se muestra a continuación:

```
% train_pcs -newpcs pegasus.bpcs \  
-bound \  
-blabels pegasus.blabels
```

Este programa utiliza la información de `pegasus.rec` junto con las especificaciones del parámetro en el archivo `trans.spec`. Cuando estamos entrenando modelos de difono, especificamos la opción de la línea de comandos `-bound`. El programa `pcs tool` puede utilizarse para examinar la matriz de transformación.

NOTA: Este programa crea archivos temporales que deberían limpiarse después de que haya acabado esta práctica con éxito. (Consulte el apéndice de los **archivos de SUMMIT** para una descripción de los archivos temporales y subdirectorios).

Modelos acústicos

Podemos ahora entrenar modelos de gaussiana de covarianza diagonal mixta para cada difono con el programa `train_models`. Este programa utiliza la información en `pegasus.rec` junto con las especificaciones del parámetro del archivo `trans.espec`.

T13: Entrene los modelos acústicos de los distintos difonos de los datos de entrenamiento con el programa `train_models` como se muestra a continuación:

```
% train_models -newmodels pegasus.bmodels \  
-pcs pegasus.bpcs \  
-bound \  
-blabels pegasus.blabels
```

Observe que este programa, al igual que `train_pcs`, también crea archivos temporales. El directorio temporal utilizado se muestra al principio de los mensajes de salida. Es aconsejable que no se olvide de este directorio ya que lo volverá a utilizar en la **T15**. De nuevo, ya que estamos entrenando aquí modelos de difonos, especificamos el indicador `-bound`. Otros tipos de modelos (segmento, por ejemplo) también pueden entrenarse con el mismo programa.

Después de entrenar los modelos, podemos observar algunas de las estadísticas para ver si los modelos se comportan adecuadamente.

T14: Utilice el programa `models_tool` para examinar el modelo de difono acústico `pegasus.bmodels`.

- P5: (a)** ¿Cuántos modelos distintos existen?
(b) ¿Qué modelos de difono crees que están bien entrenados?
(c) ¿Cuáles crees que están mal entrenados?

Podemos también examinar gráficamente si los modelos son buenos, superponiendo trazados de dispersión de los datos de entrenamiento y “trazados de contorno” bidimensionales de los modelos de mezcla de gaussianas. Esto se puede hacer con algunas de las herramientas de `MATLAB`.

T15: Convertiremos los modelos y mediciones al formato `.mat` de `matlab` a través de las siguientes herramientas:

```
% models_to_matlab -in pegasus.bmodels \  
-label "t(ah|m)" \  
-out models.mat
```

En este ejemplo, convertimos los parámetros del modelo para el difono “t(ah m)” (transición desde [ah] a [m]) al formato legible de `matlab`. `models.mat` contiene parámetros del modelo gaussiano diagonal mixto de nuestro difono deseado.

A continuación, cargue las mediciones de rasgos para el mismo difono desde el directorio temporal `temp/t1`. Observe que el directorio temporal puede ser distinto dependiendo de cuántas vueltas distintas de entrenamiento haya realizado. Debería ser un directorio utilizado por la última vuelta del programa `train models`. Esto se realiza con la herramienta `measurements to matlab`. Toma también un argumento en `-name` para los nombres de los archivos de datos del directorio temporal especificado.

```
% measurements_to_matlab -directory temp/t1 \  
-name bound_scores.vec \  
-labels_file pegasus.bllabels \  
-label "t(ah|m)" \  
-out measurements.mat
```

Para mostrar gráficamente los modelos entrenados, arranque primero MATLAB en una nueva ventana. Cargaremos las mediciones junto con los parámetros del modelo y crearemos entonces un trazado de dispersión de los datos de entrenamiento para dos dimensiones de rasgos especificadas (dimensión 1 y 2 en este ejemplo) y superpondremos el trazado de contorno del modelo gaussiano mixto correspondiente:

```
% matlab  
>> load measurements  
>> load models  
>> plot_clusters(measurements,1,2); hold;  
>> plot_mix_contour(model_means,model_variances,model_weights,1,2)
```

Parte IV: Reconocimiento y rendimiento

En esta parte de la práctica, utilizaremos los modelos acústicos y del lenguaje que entrenamos anteriormente, para llevar a cabo el reconocimiento de un nuevo grupo de oraciones (el grupo de desarrollo), y para medir el rendimiento del reconocimiento.

Concretamente, nos dedicaremos a:

- Realizar el reconocimiento 1-mejor utilizando los modelos de lenguaje bigrama de clase y de palabra.
- Realizar el reconocimiento N -mejor utilizando un modelo de lenguaje trigrama de palabra.
- Medir el rendimiento del reconocimiento y estudiar los errores de reconocimiento.

El programa `fst build recognizer.cmd` se utiliza para construir un transductor léxico completo de estado finito (FST) que represente el espacio de búsqueda léxica. El proceso de construcción del FST implica también el uso de un grupo de reglas fonológicas que expanden la pronunciación de cada forma base a todas las realizaciones fonéticas posibles para esa forma base. Las reglas fonológicas se encuentran en el archivo `pegasus.pron.rules`. Incorpora también los modelos de lenguaje que construyamos. El programa `fst test rec accuracy` se utiliza para llevar a cabo el reconocimiento. Utiliza la información en `pegasus.rec` además de las especificaciones del parámetro en el archivo `pegasus.espec`. Fíjese en la definición del grupo en el archivo `pegasus.espec` :

```
set: <p_7>*<[artifact == no]>*<[contains_p_oov == no]>
```

Especifica las oraciones del grupo de desarrollo que no presentan ni artefactos ni palabras no presentes en el vocabulario (OOV).

Reconocimiento 1-mejor

T16: Realizar el reconocimiento 1-mejor (búsqueda de Viterbi) en el grupo de desarrollo utilizando el modelo de lenguaje bigrama **de clase** que construimos en la **T9**. En primer lugar, construya el FST para el reconocedor con el siguiente comando:

```
% fst_build_recognizer.cmd pegasus 0 0
```

En este comando, el primer argumento es el nombre del dominio, el segundo es el peso de transición de una palabra (WTW) que se aplica a cada palabra, y el tercero es un valor booleano que indica si debería utilizarse un modelo de trigramas completo. Normalmente ajustamos el WTW a cero en la red, dado que el peso puede estar también controlado por un argumento en la operación de búsqueda en el archivo `pegasus.rec`. Este peso se utiliza para controlar las propiedades de inserción/eliminación del reconocedor.

Observe que el sistema realmente crea una colección de varios archivos FST. Cada FST cumple un objetivo distinto en la búsqueda completa. El archivo FST primario se llama `pegasus.ffst`. Éste contiene la red FST completa, que se utiliza para la búsqueda de avance inicial. El comando `fst test rec accuracy` se utiliza para probar el rendimiento del reconocimiento:

```
% fst_test_rec_accuracy -overwrite
```

Fíjese en que la opción `-overwrite` se facilita al comando `fst test rec accuracy` para borrar salidas de anteriores vueltas de reconocimiento.

El reconocimiento se realiza de forma distribuida sobre una colección de servidores SLS. Puede especificar la opción `-local` para obligar a que el reconocimiento se realice en su máquina local. Antes de que ejecute el reconocimiento en modo distribuido, sería aconsejable que utilizara la opción `-local` y que se asegurara de que el reconocedor se está ejecutando adecuadamente.

Observe que varios archivos son generados por este programa: un archivo de hipótesis oracional (`.hyp`), un archivo de referencia oracional (`.ref`) y un archivo del resumen del rendimiento (`.sum`). El nombre base de los archivos posee el siguiente formato: `<machine> <process id>`. La medida de rendimiento estándar para un sistema de reconocimiento de voz es la tasa de error por palabra, que está compuesta por la suma de los siguientes tipos de errores: sustituciones, eliminaciones e inserciones.

T17: Ejecute el reconocimiento utilizando el archivo del modelo acústico facilitado `galaxy.bmodels` que se ha entrenado con un corpus mucho más extenso:

```
% fst_test_rec_accuracy -bpcs galaxy.bpcs \  
-bmodels galaxy.bmodels \  
-blabels galaxy.blabeleds \  
-overwrite
```

P6: (a) Cree una tabla con los valores de los distintos errores (sustituciones, eliminaciones, inserciones y errores totales) para los dos experimentos de reconocimiento.

(b) ¿Cuál de los dos tiene un mejor funcionamiento?

(c) Examine el modelo acústico `pegasus.bmodels` y que acaba de entrenar y el modelo que le hemos facilitado `galaxy.bmodels` mediante `models tool`. Denos algunas razones de por qué uno funciona mejor que el otro.

T18: Ejecute el reconocimiento utilizando un modelo de lenguaje bigrama **de palabras**. Sería aconsejable que reconstruyera los modelos de lenguaje obviando el archivo de la regla de clase:

```
% ngram_create.pl -prefix pegasus -sents train_sents.txt
```

Luego, vuelva a construir los FST para el reconocedor:

```
% fst_build_recognizer.cmd pegasus 0 0
```

Ahora ya puede evaluar el reconocedor con el siguiente comando:

```
% fst_test_rec_accuracy -overwrite
```

P7: (a) Examine y compare los archivos del resumen de funcionamiento (`.sum`) para el bigrama de **palabra** y de **clase** utilizando el mismo modelo acústico, `pegasus.bmodels`, y haga una tabla de valores de los distintos errores (sustituciones, eliminaciones, inserciones y error total).

(b) ¿Cuál funciona mejor?

(c) ¿Concuerdan el resultado con sus conclusiones basadas en mediciones de perplejidad de la tarea 6?

(d) De su examen de los archivos de resumen de funcionamiento, ¿cuáles son las palabras que se insertan con más frecuencia?

(e) ¿Cuáles son las palabras que se eliminan con más frecuencia?

Reconocimiento *N*-mejor

T19: Realice el reconocimiento *N*-mejor (búsqueda A^*) sobre el grupo de desarrollo, utilizando el modelo de lenguaje trigrama y bigrama **de palabra**.

Dado que acabamos de crear modelos de lenguaje *n*-grama **de palabra** saltándonos las reglas de clase, podemos construir el reconocedor FST con los modelos de lenguaje bigrama y trigrama de **palabra**, utilizando el siguiente comando:

```
% fst_build_recognizer.cmd pegasus 0 1
```

Luego, pruebe el reconocedor utilizando el siguiente comando:

```
% fst_test_rec_accuracy -nbest 10 -overwrite
```

Fíjese en que además de los archivos de hipótesis (`.hyp`), referencia (`.ref`) y resumen del funcionamiento (`.sum`), se crearán también los archivos de hipótesis de la oración *n*-mejor para cada oración.

Si observa la especificación `-nbest out` del archivo `pegasus.espec`, verá que estos archivos se encuentran en el subdirectorio `temp/nbest`.

T20: Repita el reconocimiento *n*-mejor utilizando los modelos de lenguaje del bigrama de **clase** y del trigramma de **clase**.

Será necesario que reconstruya los modelos de lenguaje utilizando reglas de clase, los FST del reconocedor especificando la opción del trigramma, y que a continuación pruebe el reconocedor:

```
% ngram_create.pl -prefix pegasus -sents train_sents.txt \  
-rules pegasus.class.rules  
% fst_build_recognizer.cmd pegasus 0 1  
% fst_test_rec_accuracy -nbest 10 -overwrite
```

Examine los archivos del resumen de funcionamiento (`.sum`) para estos experimentos de reconocimiento *n*-mejor.

Observe también las listas *n*-mejores (en `temp/nbest`) y dése cuenta del parecido que comparten las hipótesis de las oraciones con las diez puntuaciones más altas.

P8: (a) ¿Cuáles son los valores de los distintos errores (sustituciones, eliminaciones, inserciones y errores totales) para estos experimentos del reconocimiento *n*-mejor?

(b) ¿De qué forma puede compararse el rendimiento del trigramma de clase con el del trigramma de palabra?

(c) ¿De qué forma se puede comparar el rendimiento con las vueltas de reconocimiento anteriores en las que no utilizábamos el modelo de lenguaje trigramma?

Apéndice: Herramientas de SUMMIT

Esta sección facilita información para la consulta de alguna documentación en línea disponible para las herramientas de SUMMIT, proporciona un índice descriptivo de las herramientas utilizadas en esta práctica y menciona algunos de los conceptos básicos que necesita conocer para completar la práctica.

Documentación en línea

Se encuentra disponible en línea un índice de muchas de las herramientas de SUMMIT, junto con una descripción de alto nivel. Puede acceder a esta información a través del navegador web de la URL:

```
file:/usr/sls/sls/v3.7/html/index.html.
```

En la URL: `file:/usr/sls/sls/v3.7/html/sapphire.html.`, puede encontrar un índice descriptivo de algunos módulos de SAPPHERE utilizados por muchas de las herramientas de SUMMIT, concretamente las especificaciones del archivo `pegasus.rec`.

Observe que estos archivos son sólo accesibles a máquinas dentro del grupo SLS.

Ayuda para el uso de la línea de comandos

Observe que para la mayoría de los programas, proveer el argumento de la línea de comandos `-help` (para programas regulares) y/o `-shelp` (para scripts de Sapphire Tcl), supondrá una enumeración de las funciones del programa junto con los argumentos válidos de la línea de comando. Esto es útil si olvida la sintaxis de uso y desea ver qué otras opciones hay disponibles.

Procesamiento distribuido

Algunos de los componentes más intensivos desde el punto de vista computacional, como `train pcs`, `train models`

y `fst test rec accuracy`, están diseñados para ser ejecutados de forma distribuida, utilizando el *servicio de envío de host* del grupo SLS.

Observe que estos programas pueden ejecutarse *localmente* en su estación de trabajo, especificando el argumento de la línea de comando `-local` cuando ejecute el programa. Cuando lo ejecute localmente, saldrá impresa por pantalla más información diagnóstica. Este “modo local” es útil para depurar experimentos y para asegurarse de que se ejecutan adecuadamente antes de usted distribuya los trabajos por todas partes. El único programa relacionado con el servicio de envío del host que es necesario que conozca es:

phd Muestra el estado actual de los hosts: `phd -status`.

En las siguientes URL puede encontrar más información detallada sobre el servicio de envío del host:

```
file:/usr/sls/sls/v3.7/html/phd.html
file:/usr/sls/sls/v3.7/html/dispatch.html.
```

Índice del programa y descripción

cfg create Crea una gramática libre de contexto (CFG) dado un grupo de clases/reglas (`.rules`) y un léxico de palabras (`.wlex`). La CFG se utiliza para el entrenamiento de modelos de lenguaje *n*-grama.

corpus tool Muestra y permite la modificación de las propiedades y conjuntos definidos en un archivo de corpus existente.

fst test rec accuracy Un script Sapphire Tcl para realizar reconocimiento y para evaluar el funcionamiento del sistema. Utiliza la información de archivo del dominio `rec` (ej., `pegasus.rec`) junto con las especificaciones del parámetro del archivo del dominio `spec` (ej., `pegasus.spec`). Muchos valores de parámetros pueden especificarse también a través de la línea de comandos. Utilice la opción `-shelp` para ver su función. Este programa genera varios archivos: un archivo de hipótesis oracional (`.hyp`), un archivo de referencia oracional (`.ref`) y un archivo de resumen de funcionamiento (`.sum`). El nombre base de los archivos tienen este formato: `<machine> <process id>`.

fst trans compute Un script de Sapphire Tcl para computar un conjunto de trayectorias obligatorias, facilitando la opción de la línea de comando `-forced`. Utiliza la información del dominio de archivo `rec` (ej., `pegasus.rec`) junto con las especificaciones del

parámetro en el archivo `paths.espec`. La salida puede utilizarse para entrenar PCS y modelos acústicos. Este programa no cuenta con la opción de ejecución de modo distribuido y generará archivos intermedios en el directorio temporal.

models tool Muestra alguna información sobre un archivo del modelo entrenado. Utiliza las opciones de la línea de comando `-detailed` y `-very detailed` para imprimir crecientes cantidades de información.

ngram create.pl Crea versiones de avance y retroceso de los modelos bigrama y trigrama. Cada uno de estos modelos se utiliza en varios puntos en la búsqueda del reconocedor.

ngram create random sentences Genera oraciones aleatoriamente, utilizando la estadística del modelo de lenguaje n -grama especificado. Esto es útil para evaluar cualitativamente un modelo n -grama.

ngram perplexity Computa la perplejidad de un grupo especificado de enunciados, utilizando un modelo de lenguaje n -grama determinado. Resulta útil para obtener una medición cuantitativa del rendimiento de un modelo n -grama.

ngram tool Muestra información sobre un archivo determinado del modelo n -grama. Permite también la comprobación de los valores del parámetro y el ajuste de los parámetros de suavizado.

pcs tool Muestra información sobre un archivo determinado de una matriz de escalamiento (`.pcs`) de los componentes principales.

train models Un script Sapphire Tcl para entrenar un nuevo grupo de modelos acústicos basados en transcripciones comprobadas obligatorias o a mano. Utiliza la información del dominio del archivo `.rec` (ej., `pegasus.rec`) junto con las especificaciones del parámetro en el archivo `trans.espec`. Este programa no cuenta con la ejecución en modo distribuido y generará archivos intermedios en el directorio temporal.

train pcs Un script Sapphire Tcl para entrenar un nuevo grupo de componentes principales basados en transcripciones generadas obligatorias o a mano. Utiliza información del dominio del archivo `.rec` (ej., `pegasus.rec`) junto con las especificaciones del parámetro del archivo `trans.espec`. Este programa no cuenta con la ejecución en modo distribuido y generará archivos intermedios en el directorio temporal.

tv Un script Sapphire Tcl para ver gráficamente los enunciados y las transcripciones alineadas (los archivos `.wrd` y `.phn`). Este programa utiliza las especificaciones del parámetro en el archivo `espec` determinado. Por ejemplo, `tv.espec` seleccionará aleatoriamente 50 oraciones del el grupo `p 3` que va a ser examinado. La herramienta muestra el espectrograma, la forma de onda, la transcripción fonética y de palabras de un enunciado con alineación temporal. Puede reproducir segmentos y palabras individuales situando el ratón sobre el segmento deseado y pulsando `Ctrl-v`.

waveform play Reproduce una forma de onda en formato NIST (`.wav`).

wlex create Crea un archivo de léxico de palabras (`.wlex`) a partir de un archivo básico de vocabulario (`.vocab`). El léxico de palabras se utiliza para construir modelos de lenguaje n -grama.

Apéndice: Archivos SUMMIT

Esta sección contiene un índice descriptivo de los archivos utilizados en la práctica. Estos archivos contienen datos, modelos y especificaciones del parámetro necesarias para la construcción de un reconocedor de voz.

Directorio y archivos temporales

Los programas `fst trans compute`, `train pcs` y `train models` crean subdirectorios temporales que guardan archivos de datos creados durante el proceso de entrenamiento. Estos subdirectorios se encuentran en `temp/t<n>` donde `<n>` es un número entero único que el sistema ha determinado para que entre en conflicto con directorios temporales existentes. Estos directorios suelen hacerse muy extensos, de modo que haga el favor de eliminarlos después de que haya ejecutado sus experimentos con éxito.

Archivos ESPEC

El tipo ESPEC, que significa especificación del experimento, se utiliza para especificar cualquier parámetro y archivo pertinente al uso de una herramienta de SUMMIT. Un archivo `espec` es un archivo simple de texto con una extensión `.espec`, que graba la información utilizada para ejecutar un determinado experimento. Se puede encontrar información detallada en línea sobre los archivos `espec` en las siguientes URL:

```
file:/usr/sls/sls/v3.1/html/espec_control_usage.html
file:/usr/sls/sls/v3.1/html/espec.html
```

paths.espec Especificaciones del parámetro experimental para `fst trans compute`. Especifica el grupo de datos para ejecutar un alineamiento obligatorio de hosts permitidos, el nombre del enunciado, la ubicación de la forma de onda, la secuencia de palabras de referencia y el nombre de los archivos de transcripciones fonéticas y de la palabra de salida de alineamiento temporal.

pegasus.espec Especificaciones del parámetro experimental para `fst test rec accuracy`. Especifica el conjunto de datos para realizar el reconocimiento, las máquinas del host permitidas en las que se ejecutan los trabajos distribuidos, el nombre del enunciado, la ubicación de la forma de onda y la secuencia de palabra de referencia (para medir el rendimiento).

trans.espec Especificaciones del parámetro experimental para `train pcs` y `train models`. Especifica el conjunto de datos para entrenar sobre los hosts permitidos, el nombre del enunciado, la ubicación de la forma de onda, la transcripción fonética de alineación temporal y la secuencia de palabra de referencia.

tv.espec Especificaciones del parámetro experimentales para `tv`. Especifica el grupo de datos para examinar y el lugar de los archivos necesarios: forma de onda y transcripciones fonéticas y de palabras de alineación temporal.

Índice del archivo y descripción

domain.info Definición y especificación de los valores del parámetro y de la variable que se utilizarán en este entorno. Este archivo es usado por la mayoría de los programas para obtener ajustes del parámetro por defecto.

galaxy.corpus El archivo de corpus para varios dominios experimentales, incluyendo **Mercury**, **Jupiter** y **Pegasus**, etc. Los grupos `p *` son para Pegasus. Contiene propiedades y definiciones de grupo para el conjunto de datos. Utiliza el programa `corpus tool` para examinar los contenidos.

pegasus.baseforms Diccionario de pronunciación fonética básica para todas las palabras del vocabulario. Cada línea presenta el siguiente formato:

word : phone1 phone2 ...

Las otras pronunciaciones se especifican mediante (*phone1, phone2*) y & se utiliza para indicar un límite de palabra.

pegasus.blabels Enumera las etiquetas (límite) de todos los difonos agrupados. Todas las etiquetas de la misma línea comparten el mismo modelo acústico. Los modelos acústicos se construirán para todos los grupos de difono.

pegasus.class.rules Grupo de palabras para asociaciones de clase o reglas. Se utiliza para crear un archivo CFG que se utiliza después para construir modelos de lenguaje *n*-grama de clase.

pegasus.pron.rules Grupo de reglas fonológicas utilizado para expandir (a múltiples pronunciaciones) las transcripciones fonéticas básicas en el archivo de las formas bases. Estas reglas tienen en cuenta la variación dependiente del contexto en la realización fonética de las palabras. Normalmente presentan el mayor impacto en límites de palabra, ya que la variación interna de palabras puede representarse normalmente en las pronunciaciones de la forma base.

pegasus.rec Este es el archivo que “dirige” el entrenamiento y el proceso de reconocimiento. Enumera los módulos de procesamiento y sus especificaciones del parámetro para cada paso en el proceso de pruebas y entrenamiento. Concretamente, especifica el procesamiento de la señal que va realizarse en la forma de onda, el tipo y tamaño de los vectores característicos que van a computarse, en análisis de los componentes principales que se va a realizar, el tipo de modelos acústicos que se van a entrenar y el tipo de búsqueda que se va a utilizar durante el reconocimiento.

pegasus.vocab Lista de todas las palabras del vocabulario del entorno.

train sents.txt Lista de enunciados del grupo de entrenamiento **Pegasus** y algunas oraciones adicionales. Se utilizan para entrenar los modelos de lenguaje.

Apéndice: Agrupamiento de difonos

Para realizar agrupamiento de difonos, es necesario que extraiga las etiquetas del difono del archivo FST de búsqueda de avance `pegasus.ffst`:

```
% get_blabels_from_fst.pl -fst pegasus.ffst
```

Escribirá las etiquetas del difono en el archivo `out.blabels`. Luego, se obtendrán los grupos de difonos con la herramienta `create blabels clusters.pl`:

```
% create_blabels_clusters.pl
-blabels out.blabels \
-slabeleds decision_tree/pegasus.slabeleds \
-initial_questions decision_tree/initial_diphone_clusters.txt \
-cluster_questions decision_tree/diphone_questions.txt \
-traindir temp \
-bpcs galaxy.bpcs \
-out new.blabels
```

Tenga en cuenta que primero tiene que borrar todos los archivos del directorio `temp`. Le llevará aproximadamente 50 minutos realizar el agrupamiento. El archivo de salida `new.blabels`, donde las etiquetas del difono del mismo grupo están colocadas en la

misma línea. El archivo es el mismo que el archivo `pegasus.blabels` que le facilitamos.

Apéndice: Conjunto de unidades fonémicas

VOCALES

aa → <u>tot</u>	er → <u>bert</u>
ae → <u>bat</u>	ey → <u>bait</u>
ah → <u>but</u>	ih → <u>bit</u>
ao → <u>bought</u>	iy → <u>beet</u>
ax → <u>about</u>	ow → <u>boat</u>
aw → <u>bout</u>	oy → <u>boy</u>
ay → <u>buy</u>	uh → <u>book</u>
eh → <u>bet</u>	uw → <u>boot</u>

SEMIVOCALES

l → <u>let</u>	y → <u>yes</u>
r → <u>rat</u>	w → <u>wet</u>

NASALES

m → <u>met</u>	ng → <u>song</u>
n → <u>net</u>	

OCLUSIVAS

p → <u>pet</u>	pd → <u>tap</u>
t → <u>tip</u>	td → <u>ta<u>t</u></u>
k → <u>kit</u>	kd → <u>tack</u>
b → <u>bet</u>	bd → <u>tab</u>
d → <u>debt</u>	dd → <u>tad</u>
g → <u>get</u>	gd → <u>tag</u>
p- → <u>speak</u>	tf → <u>butter</u>
t- → <u>steak</u>	df → <u>shudder</u>
k- → <u>skeet</u>	

FRICATIVAS

f → <u>fat</u>	v → <u>yat</u>
th → <u>third</u>	dh → <u>that</u>
s → <u>sat</u>	z → <u>zoo</u>
sh → <u>shoot</u>	zh → <u>asia</u>
hh → <u>hat</u>	

AFRICADAS

ch → <u>chew</u>	jh → <u>judge</u>
------------------	-------------------

SECUENCIAS ESPECIALES

ao r → <u>boar</u>
eh r → <u>bear</u>
ih r → <u>beer</u>
ix ng → <u>writing</u>
tq en → <u>written</u>
nt → <u>interstate</u>