Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Recolección online de grabaciones para el estudio de las variantes argentinas del español

Tesis de Licenciatura en
Ciencias de la Computación

Fernando Bugni

Director: Agustín Gravano

Codirector: Miguel Martínez Soler

Buenos Aires, 2014

Recolección online de grabaciones para el estudio de las variantes argentinas del español

El uso de la lengua siempre ha caracterizado a las personas que la utilizan. La forma como nos comunicamos no sólo posee la información del mensaje a transmitir, sino que también posee características del hablante. Estas características pueden describir al hablante de distintas formas. Algunas de ellas pueden ser: su cultura, su nivel socioeconómico y su región, entre otras. En particular, Argentina no es la excepción. Nuestro país posee una fuerte componente dialéctica en su habla. Esto quiere decir que podemos saber de qué lugar proviene el hablante analizando su tonada. Hay varias regiones con esta característica definidas en el país. Nos enfocaremos en distinguir diferencias entre las regiones de Córdoba y Buenos Aires. En el presente trabajo desarrollamos un sistema de grabación a través de Internet que nos permitió recolectar grabaciones de hablantes provenientes de ambos grupos. Con este sistema de grabación diseñamos un experimento que nos ayudó a comparar el habla de cada grupo haciendo hincapié en sus diferencias. Analizamos las dificultades técnicas que surgieron y cómo impactaron en el estudio final. Utilizando estos datos, analizamos efectivamente cuáles son las características más predominantes y cómo repercuten para una buena clasificación. Extrayendo estos atributos, utilizamos algoritmos de Machine Learning para la clasificación de hablantes en los dos grupos.

Agradecimientos

¡Uff! ¡Todo el tiempo que pasó desde que comencé la carrera! Mejor ni acordarse. Pero aunque haya pasado tanto tiempo lo bueno fue toda la gente copada que conocí...

Les voy a agradecer siempre a todos los amigos nuevos que conocí gracias a la facu. Tantas horas estudiando y divirtiéndose también hace que haya sido buenísimo pasarla juntos. Algunos de ellos son: *Alex Valencia*, *Pablo Echebarriga*, *Gonza Raposo*, *Checho González*, *Agustín Olmedo* entre otros... A todos los amigos de la lista *Gente de la Facu* que estudiamos juntos en varias oportunidades. Entre ellos *Pablo Laciana* y *Leo Rodriguez* que inclusive fuimos a ver un par de veces al *Matador de Victoria*. Seguramente me olvide de algunos que ahora no recuerdo pero bueno... las hojas son finitas :)

A mis compañeros y amigos del trabajo, que me dieron la oportunidad de trabajar con ellos y compartir muchísimas tardes juntos. Estos son: *Mariano Gonzalez*, *Francisco Tarulla*, *Lucas Bali*, *Leonardo Kuntscher*, *Graciela Defeo*, *Alejandro Aquesta*. Sin su buena onda y ayuda no hubiera sido lo mismo terminar la carrera.

A mi grupo de amigos, que la mayoría también van a la facu. Ellos son: *Midra*, *Dani*, *Mich*, *El cansado*, *Pablox*, *Nahuel*, *Maru*, *Luis*, *Vicky*, *El Punga*. Con la buena onda de ellos y su ayuda cuando uno andaba medio mal no hubiera sido lo mismo. Voy a agradecer siempre lo increíble que son cada uno de ellos como persona.

¡A mis directores de tesis, que me han soportado todo este tiempo! A *Agustín* que es un genio explicando y le pone una increíble pasión a lo que hace. A *Miguel* que me ha ayudado todo este tiempo realizando este trabajo. También a la gente del *Laboratorio de Investigaciones Sensoriales* que nos prestaron su diccionario para realizar el experimento. ¡Gracias por todo! :P

Y por último, a mi familia. Mi vieja, *Camila*, que sin su apoyo y ganas no hubiera podido hacer nada en mi vida. Mi viejo, *Quique*, que me dio todo su cariño y me cuida desde el cielo. Mi hermano que es el mejor compañero que tengo de mi vida. Y a mi novia, *Emilia*, que me bancó todo este tiempo realizando esta tesis. *¡Te amo!*

A mi viejo, que me ayuda desde el cielo.
A mi vieja, que gracias a ella soy lo que soy.
Y a mi hermano, que es el compañero de mi vida.

Índice general

Capítulo 1

Introducción

El uso de la lengua siempre ha caracterizado a las personas que la utilizan. La forma en que nos comunicamos no sólo posee la información del mensaje a trasmitir, sino también características del hablante. Estudiar estas características del habla nos permite conocer mejor la cultura de las personas. Nos permite, por ejemplo, identificar a los hablantes para saber el lugar donde pertenecen.

Identificar y extraer características del habla son tareas muy difíciles de realizar. No sólo se deben obtener muestras muy variadas de muchos hablantes en distintas regiones, sino que también hay que prestarle importante atención a su edad, su sexo, su situación económica, grupo social, etc. Realizar un estudio de estas características es muy complejo y sobre todo, costoso. Además de estudiar cada grupo se deben utilizar muchos recursos: por ejemplo, se debe utilizar soporte para grabar en buena calidad las muestras, varios viajes para buscar los diferentes hablantes, analizar cada uno de las grabaciones de manera individual, entre otras cosas.

El objetivo de esta tesis es realizar un sistema que pueda facilitar la resolución de estos problemas. Vamos a enfrentar cada uno de ellos e intentar resolverlos de forma computacional. De los problemas descriptos el principal radica en obtener cada grabación. Si los grupos se encuentran muy alejados esto puede ser muy costoso por los viajes. También estas grabaciones deben ser de calidad aceptable como para realizar el estudio en cuestión. Se podría utilizar el teléfono para algunos experimentos pero hay que tener en cuenta que este posee calidad de audio muy baja. De hecho, se utiliza en algunos experimentos donde esta característica no es un inconveniente.

Es por esto que se desarrolló un sistema de grabación basado en Internet como herramienta para obtener muestras. De esta forma, se pueden realizar varias grabaciones sin necesidad de viajar a cada lugar. Es cierto que no todos los lugares poseen acceso a Internet y, si se realizara un experimento de estas características en lugares que no posean conexión, este sistema no sería útil. De cualquier forma, pensamos que su utilización soluciona muchos inconvenientes. Otra característica radica en que se puede manejar la calidad de la grabación. Utilizando distintas tecnologías a través de esta red se puede configurar la calidad para que sea lo más precisa posible para el experimento. El sistema desarrollado va a mejorar la forma de recolectar muestras

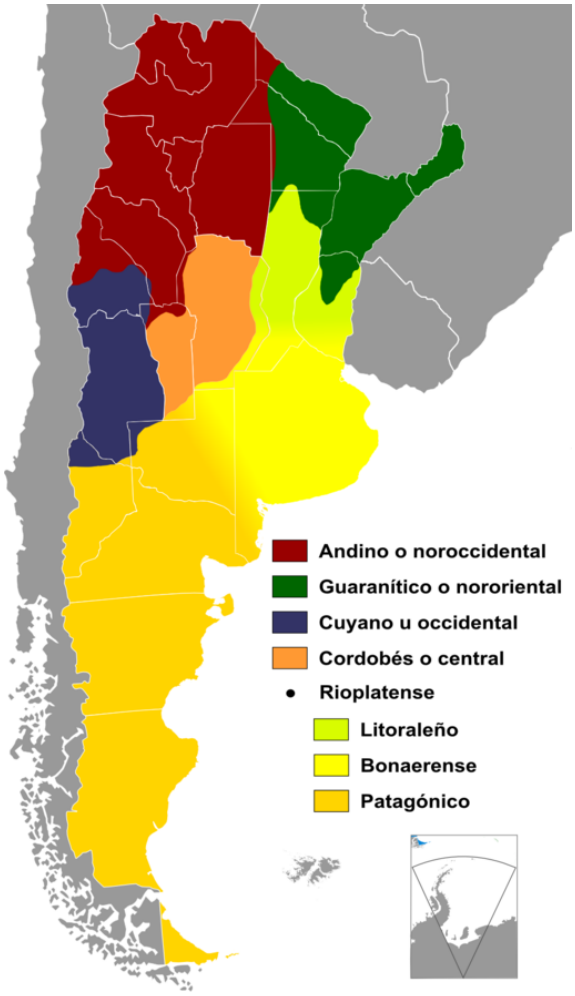


Figura 1.1: Dialectos del idioma español en Argentina

y lo utilizamos en un experimento para corroborar las ventajas y desventajas del mismo.

El objetivo del sistema es obtener muestras de habla para su posterior análisis o utilización en sistemas de procesamiento de voz. El experimento que tomamos como caso de estudio es medir las diferencias en el habla entre Córdoba y Buenos Aires. El primero de estos grupos se encuentra en la zona central de nuestro país mientras que el segundo cerca del Río de la Plata, como se puede observar en la figura 1.1. En la literatura existen estudios que explican estas diferencias, por ejemplo *El español en la Argentina* [Fontanella, 2000] de Beatriz Fontanella de Weinberg y *Español en la Argentina* [Battini, 1964] de Elena Vidal de Battini.

Fontanella de Weinberg recopila varios trabajos de colegas que analizan el español de cada región de Argentina. Cada región se describe en un capítulo distinto y entre ellas se encuentra uno para Buenos Aires y otro para Córdoba. Por ejemplo, en la descripción de estos capítulos se señalan las siguientes diferencias para Córdoba: un sonido /r/ más suave y corto, el cambio de sonidos de /y/ y /ll/ a /j/, y la aspiración

de la /s/ al terminar una palabra. También describe el estiramiento de la sílaba anterior a la acentuada en cada palabra como distintivo del acento cordobés. Por su parte, Vidal de Battini analiza región por región el uso de los fonemas importantes. Destaca la diferencia entre las dos regiones en el uso de la /r/, de la /s/ y de la /ll/, también referencia a la pronunciación de la /s/.

Extrayendo el análisis de estos libros se puede definir las reglas que describen a cada grupo. Las reglas son:

- **Regla 1: Los hablantes de Córdoba estiran la sílaba anterior a la acentuada mientras los de Buenos Aires no lo hacen.** Cada palabra posee una sílaba con su acento primario. Para cumplir esta regla se debe estirar la sílaba anterior a esta. Si la sílaba acentuada es la primera de la palabra, entonces no se estira. Ejemplo: 'Espectacular' posee su sílaba acentuada en '-lar'. La sílaba anterior, o sea '-cu-' se alarga solamente para hablantes de Córdoba.

- **Regla 2: Los hablantes de Córdoba aspiran y elisionan la /s/ al finalizar una palabra. Esto no sucede en Buenos Aires.** Para las palabras terminadas en /s/ se acorta la duración de éste último fonema en hablantes de Córdoba. Ejemplo: 'Pájaros' posee el fonema /s/ al final. Utilizando la dialéctica de Córdoba, la /s/ final sería más suave que una de Buenos Aires.

- **Regla 3: Para hablantes de Córdoba, la /s/ antes de la /c/ o /t/ suenan más suaves que para hablantes de Buenos Aires.** El fonema /s/, que precede a /c/ o /t/, suena más suave en cordobeces que en porteños. Ejemplo: 'Mosca' en la variante de Córdoba posee el fonema /s/ más suave que en Buenos Aires.

- **Regla 4: La 'c' antes de la 't' se pronuncia con menor frecuencia para hablantes de Córdoba que para hablantes de Buenos Aires.** El fonema /c/, que precede a /t/, no se debe pronunciar en el dialecto cordobés. Ejemplo: 'Doctor' no debe sonar el fonema /c/.

- **Regla 5: Para hablantes cordobeces la 'y' y 'll' se pasa a 'i'. No sucede esto para Buenos Aires.** Palabras con el fonema /y/ o /ll/ se pronuncian /j/. Ejemplo: 'lluvia' se debe pronunciar utilizando el fonema /j/.

- **Regla 6: En hablantes cordobeces la /r/ no vibra mientras que en Buenos Aires pasa lo contrario.** Palabras con el fonema /r/ deben ser suaves y no vibrar en el dialecto cordobés. Ejemplo: 'Espárrago' en su fonema /r/ debe ser suave en comparación con Buenos Aires.

Normalmente estas reglas se producen en el habla espontánea y raramente en habla leída. Algunas pueden agudizarse si se encuentran en lugares económicamente más vulnerables, pero en cualquier ambiente se cumplen.

En el próximo capítulo describiremos el diseño del experimento. Éste tiene como objetivo reconocer las diferencias planteadas con las reglas mediante la grabación de frases. Estas frases fueron grabadas tanto por hablantes de Córdoba como de Buenos Aires. También describiremos cuáles frases utilizamos y el medio empleado para grabar.

Capítulo 2

Diseño del experimento

Utilizando los estudios de [Fontanella, 2000] y [Battini, 1964] pudimos obtener las reglas definidas en el capítulo anterior. Recordemos que éstas describen la diferencia entre los dos grupos. En este capítulo proponemos realizar un experimento para extraer la información fonética de los mismos. La idea será realizar una serie de actividades donde el hablante sea grabado y que esas actividades hagan hincapié en estas diferencias descriptas. A continuación describimos el experimento en más detalle.

2.1. Elección de las frases

El acento se potencia cuando se realiza habla espontánea. Utilizando este concepto intentamos que el hablante diga las frases de la forma más natural posible. Es por ello que elegimos como actividad pronunciar frases popularmente conocidas. Si el hablante conoce la frase y la utiliza con frecuencia entonces es más fácil que su pronunciación sea espontánea. Con esta idea cubrimos las reglas 2 al 6.

Recordemos que en el capítulo anterior la regla 1 nos decía que había una diferencia en la duración de la sílaba previa a la acentuada: para hablantes de Córdoba esta duración es más larga que para hablantes de Buenos Aires. La sílaba acentuada varía según que tipo de palabra se refiere. No es lo mismo utilizar una palabra aguda que una esdrújula en esta regla. Para cubrir esta regla utilizamos un esquema de frases con una estructura fija pero que varía sus palabras según su entonación. Este esquema se llama AMPER [Van Oosterzee et al., 2004, Gurlekian et al., 2009] y lo veremos más en detalle en la sección 2.1.1.

Entonces los esquemas van a ser:

- **Frases utilizando esquema AMPER que cubre cada tipo de acentuación:** Estas corresponden a la regla 1 que hace énfasis en la longitud de la sílaba anterior a la acentuada. Utilizamos este esquema para cubrir todo tipo de acentuación.

- **Frases comunes que tratan de cubrir la espontaneidad:** Estas frases van a cubrir las reglas 2 a 6. Estas tienen que ver con la duración y la pronunciación de distintos fonemas. Utilizar frases comunes favorece la espontaneidad.

A continuación vemos estos dos esquemas en detalle.

2.1.1. Frases utilizando esquema AMPER

Utilizamos este esquema para analizar todas las variantes posibles de la regla 1. Recordemos que esa regla nos dice que hay que estirar la sílaba anterior a la acentuada. Esta regla define comúnmente la tonada cordobeza y puede aparecer de varias formas según su acentuación.

Para definir este esquema nos basamos en el trabajo de variabilidad rítmica en dos corpus [Gurlekian et al., 2009] que utilizan un esquema similar. Este trabajo estudió los acentos del español argentino utilizando un esquema de frase fija y intercambiando palabras para analizar todos sus casos. En nuestro trabajo tenemos un problema parecido por ello lo tomamos como referencia.

El esquema AMPER utilizado en este trabajo es:

Sujeto+“ salió ”+Adjetivo

- Objeto puede ser *“El canapé”, “El repollo”, “El espárrago”.*
- Adjetivo puede ser *“espectacular”, “delicioso”, “riquísimo”.*

Utilizamos estas palabras ya que cubren los tres tipos de acentuación: aguda, grave y esdrújula.

Por ejemplo: *“El canapé salió delicioso”*. Canapé tiene acento en la última sílaba, es una palabra aguda, mientras que delicioso es grave. En este ejemplo podemos analizar la sílaba anterior a la acentuada de los dos grupos estudiados, Córdoba y Buenos Aires. Armamos las combinaciones para obtener muchas variantes de dónde se encuentra el acento. De esta forma estudiamos en detalle la regla 1. Todas las combinaciones se pueden ver en el tabla 2.1.

Frase			**Aguda**	**Grave**	**Esdrújula**
El canapé	*salió*	*espectacular*	espectacular, canapé		
El canapé	*salió*	*delicioso*	canapé	delicioso	
El canapé	*salió*	*riquísimo*	canapé		riquísimo
El repollo	*salió*	*espectacular*	espectacular	repollo	
El repollo	*salió*	*delicioso*		repollo, delicioso	
El repollo	*salió*	*riquísimo*		repollo	riquísimo
El espárrago	*salió*	*espectacular*	espectacular		
El espárrago	*salió*	*delicioso*		delicioso	
El espárrago	*salió*	*riquísimo*			riquísimo

Tabla 2.1: Frases AMPER

2.1.2. Frases comunes

Como afirmamos antes, se utilizaron frases comunes para poder obtener los acentos de cada grupo de forma natural. Se pensó que si se graba una frase popular, el hablante, al estar acostumbrado a decirla, no iba a poder evitar impregnarle su acento. Todas las frases utilizadas se pueden ver en la tabla 2.2.

Algo interesante es que una misma frase permite extraer atributos para varias reglas. Por ejemplo: la frase *'En la pelea se conoce al soldado, sólo en la victoria se conoce al caballero'* extrae atributos para las reglas 4 y 5. La palabra *'victoria'* cubre la regla 4 que nos propone medir la duración de la */c/* antes de la */t/*. Sucede igual con la palabra *'caballero'* para la regla 5: el fonema */ll/* se pasa a */i/* cambiando su duración y sonido. En la tabla 2.2 podemos ver que la cantidad de frases utilizadas con respecto a sus reglas es despareja. Hay más frases para la regla 2 que para las demás. Más adelante veremos cómo impacta esto en las frases que pedimos grabar.

Orden de las frases

Ya definimos cuáles van a ser las frases, ahora debemos definir qué frases y en qué orden se deben decir durante el experimento. Diremos que una frase cubre una determinada regla si ésta la satisface. Sucede que el orden que utilicemos va a ser crucial para obtener muestras: no es lo mismo empezar por una frase que sólo cubre una sola regla que varias reglas al mismo tiempo. Si elegimos primero las frases que cubren varias reglas a la vez, en una sola grabación podremos obtener más cubrimiento de reglas. Entonces, en caso de poder, elegimos frases que cubran más de una regla.

¿Por qué querríamos cubrir más reglas en una misma frase? Más adelante veremos que una frase grabada, que cubre una regla, aportará información solamente a esa regla en particular. Si una frase cubre varias reglas a la vez estaríamos obteniendo más información ya que ésta aportaría para cada una de esas reglas. En ambas opciones utilizamos sólo una grabación. Por eso es importante que en cada elección de cada frase a grabar se maximice su cubrimiento de reglas.

El orden de las frases sigue el siguiente algoritmo:

```
1  OrdenDeFrasesConocidas:
2  Input: Frases (conj. de String)
3  Output: listaFrases (lista de String)
4  listaFrases = {}
5  DicPct <- Diccionario con porcentajes de cubrimiento para cada regla
6  Mientras Frases != {}:
7     regla <- DicPct.ObtenerReglaConMenorPorcentaje()
8     frase <- Frases.ObtenerLaFraseMasPonderada(regla)
9     listaFrases.agregar(frase)
10    DicPct.RecalcularPorcentajes()
11    Frases.quitar(frase)
12 Devolver listaFrases
```

Frase	**2 - Aspiración y elisión de /s/**	**3 - La 's' antes de la 'c' o 't' no suenan**	**4 - La 'c' antes de la 't' no suenan**	**5 - La 'y' y 'll' se pasa a 'i'**	**6 - La 'r' no debe sonar**
1 - 'No hay dos sin tres'	dos, tres				
2 - 'Más difícil que encontrar una aguja en un pajar'	más				
3 - 'Más perdido que turco en la neblina'	más				
4 - 'No le busques la quinta pata al gato'	busques	busques			
5 - 'Se te escapó la tortuga'		escapó			
6 - 'Todos los caminos conducen a Roma'	todos, los, caminos				
7 - 'No hay mal que dure cien años'	años				
8 - 'Siempre que llovió, paró'				llovió	
9 - 'Cría cuervos que te sacaran los ojos'	cuervos, los, ojos				
10 - 'La tercera es la vencida'	es				
11 - 'Calavera no chilla'				chilla	
12 - 'La gota que rebalsó el vaso'					rebasó
13 - 'La suegra y el doctor cuanto más lejos mejor'	más, lejos	doctor			
14 - 'A la mujer picaresca cualquiera la pesca'		picaresca			
15 - 'Quien siembra vientos recoge tempestades'	vientos				recoge
16 - 'La arquitectura es el arte de organizar el espacio'	es		arquitectura		
17 - 'El amor actúa con el corazón y no con la cabeza'			actúa		
18 - 'No dudes, actúa'			actúa		
19 - 'Perro que ladra no muerde'					perro
20 - 'La musica es sinónimo de libertad, de tocar lo que quieras y cómo quieras'	es, quieras				
21 - 'La belleza que atrae rara vez coincide con la belleza que enamora'				belleza	
22 - 'No esta mal ser bella, lo que está mal es la obligación de serlo'				bella	
23 - 'La batalla más difícil la tengo todos los días conmigo mismo'	más			batalla	
24 - 'El que no llora, no mama'				llora	
25 - 'En la pelea se conoce al soldado, sólo en la victoria se conoce al caballero'			victoria	caballero	
26 - 'La lectura es a la mente lo que el ejercicio al cuerpo'	es		lectura		
27 - 'El pez por la boca muere'	pez				
28 - 'En boca cerrada no entran moscas'	moscas	moscas			cerrada
29 - 'Más vale pájaro en mano que cien volando'	más				
30 - 'Río revuelto, ganancia de pescadores'	pescadores	pescadores			río, revuelto
31 - 'No hay qué pedirle peras al olmo'	peras				

Tabla 2.2: Frases conocidas

La idea del algoritmo es la siguiente: contabilizamos la cantidad de muestras por cada regla. Esto se realiza calculando, para cada regla, la cantidad de muestras que ya grabamos cubriendo esa regla; sobre la cantidad total de muestras que cubren esa regla. Entonces tendremos para cada regla su porcentaje de cubrimiento. Esto lo guardaremos en *DicPct*.

Mientras haya frases sin ser seleccionadas para grabar, elegimos una frase que corresponda a la regla que menos está cubierta. O sea, elegiremos como próxima regla a cubrir la que tenga menor porcentaje de cubrimiento. Esto se realiza llamando al método *ObtenerReglaConMenorPorcentaje* del objeto *DicPct*. Si hay varias con el mismo mínimo porcentaje, se elige una al azar.

Tenemos definida la próxima regla a cubrir. Debemos elegir la frase que cubra esa regla. Recordemos que hay muchas frases que cubren una determinada regla. Para definir cuál frase elegir optamos por la que cubra mayor cantidad de reglas, siempre que cubra a la regla seleccionada. Si hay varias frases en esta situación, elegimos una de estas al azar. Entonces no sólo aumentamos la cantidad de frases de la regla menos cubierta, sino que con esa frase cubrimos otras reglas. Esto se realiza en el método *ObtenerLaFraseMásPonderada*. Para terminar, agregamos la frase elegida y recalculamos los porcentajes de cada regla en las lineas 9 y 10.

Esta idea es importante ya que extraemos la mayor cantidad de información en cada grabación y al hablante le hacemos perder menos tiempo realizando el experimento.

Podemos ver un ejemplo de este algoritmo en la tabla 2.3. Este muestra los primeros 15 pasos que realiza en una ejecución. Descartamos los demás pasos por simplicidad. La columna "Tarea" nos muestra el número de orden de la frase a grabar, "Número de Frase" nos indica qué frase fue la elegida para esa tarea, y las columnas "Regla X" muestran el porcentaje de apariciones ya grabadas de esa regla. En el primer paso todas las reglas están en 0 %, lo cual sucede porque al comienzo no se grabó ninguna frase aún. Elegimos una frase que cumpla con la regla que tiene menor porcentaje de cubrimiento. Como todas las reglas tienen 0 % elegimos una al azar, en este ejemplo la frase 15, que cubre las reglas 2 y 6 (ver figura 2.2). Entonces el porcentaje de cubrimiento de esas reglas se incrementa. La frase 15 nos dice *"Quien siembra vientos recoge tempestades"* y, como dijimos, cubre la regla 2 y 6. Para la regla 2 hay 26 posibles muestras en todas las frases. La frase 15 sólo aporta 1 de ellas, entonces 1/26 nos aporta el 3 % de cubrimiento. Para la regla 6, tenemos 6 posibles muestras y esta frase nos provee sólo 1 de ellas. Su porcentaje entonces es $1/6 = 16\,\%$.

El ejemplo de la tabla 2.3 ilustra cómo el algoritmo, en cada paso, elige una frase que cubra la regla con menor porcentaje de cubrimiento. Por ejemplo: en la tarea 5, la regla 5 sólo tiene el 14 % de porcentaje grabado. Este cubrimiento es el menor de todos. En el próximo paso, el algoritmo elige una frase que cubra esa regla y, si es posible, que cubra otra regla más. La elegida es la frase 23 (ver figura 2.2). Como hay pocas frases de la regla 5, al agregar esta frase ya cubrimos el 28 % de las frases de esa regla. Y además se incrementa la regla 2 al 23 %. De este modo, intentamos

ir cubriendo paso a paso en forma pareja las distintas reglas.

Tarea	**Nro de Frase**	**Regla 2**	**Regla 3**	**Regla 4**	**Regla 5**	**Regla 6**
-	–	0 %	0 %	0 %	0 %	0 %
1	15	3 %	0 %	0 %	0 %	16 %
2	25	3 %	0 %	20 %	14 %	16 %
3	30	7 %	16 %	16 %	14 %	50 %
4	28	11 %	33 %	16 %	14 %	66 %
5	13	19 %	50 %	16 %	14 %	66 %
6	23	23 %	50 %	16 %	28 %	66 %
7	16	27 %	50 %	33 %	28 %	66 %
8	26	30 %	50 %	50 %	28 %	66 %
9	8	30 %	50 %	50 %	42 %	66 %
10	4	34 %	66 %	50 %	42 %	66 %
11	6	46 %	66 %	50 %	42 %	66 %
12	21	46 %	66 %	50 %	57 %	66 %
13	9	57 %	66 %	50 %	57 %	66 %
14	17	57 %	66 %	66 %	57 %	66 %
15	1	65 %	66 %	66 %	57 %	66 %
...						

Tabla 2.3: Pasos del algoritmo OrdenDeFrasesConocidas

Esta idea también se puede ver en la figura 2.1 que representa otro ejemplo del porcentaje de frases completadas. Teniendo en cuenta este algoritmo podemos notar que aproximadamente a partir de 10 grabaciones ya tenemos un buen porcentaje de cubrimiento. Por ejemplo, en la décima grabación la regla 4 ya tiene el 75 % de sus muestras grabadas, mientras que, la regla 5 el 50 % de sus muestras ya grabadas. En este ejemplo, en la grabación número 10 ya se grabó alrededor del 40 % de la cantidad total de muestras para cada regla.

2.1.3. Combinando los dos tipos de frases utilizando trazas

Definimos frases comunes y AMPER para grabar. Ahora debemos definir cómo intercambiarlas durante el experimento. Para ello definimos traza. Una traza es una lista de las frases que va a grabar un hablante en el experimento. Esta va a estar compuesta por entre 1 y 3 frases comunes extraídas del orden definido en *OrdenDeFrasesConocidas*, y luego una frase del esquema de AMPER. Tanto la cantidad de frases comunes como la elección de la frase AMPER se realiza al azar. Este patrón se repite sucesivamente hasta completar todas las frases. La idea es no cansar al hablante con frases repetitivas y evitar que sepa de antemano qué frase va a tener que grabar ya que si fuera el caso, podría modificar su entonación natural.

Al empezar el experimento, al hablante se le dará una traza que grabará sucesivamente en ese orden. Elegimos tener precalculadas las trazas para evitar cálculos costosos a la hora de empezar el experimento. Si no se precalcularan las trazas, deberíamos realizar los cálculos cada vez que empieza el experimento y podríamos retrasar la grabación del hablante. Es por eso que guardamos 10.000 trazas generadas.

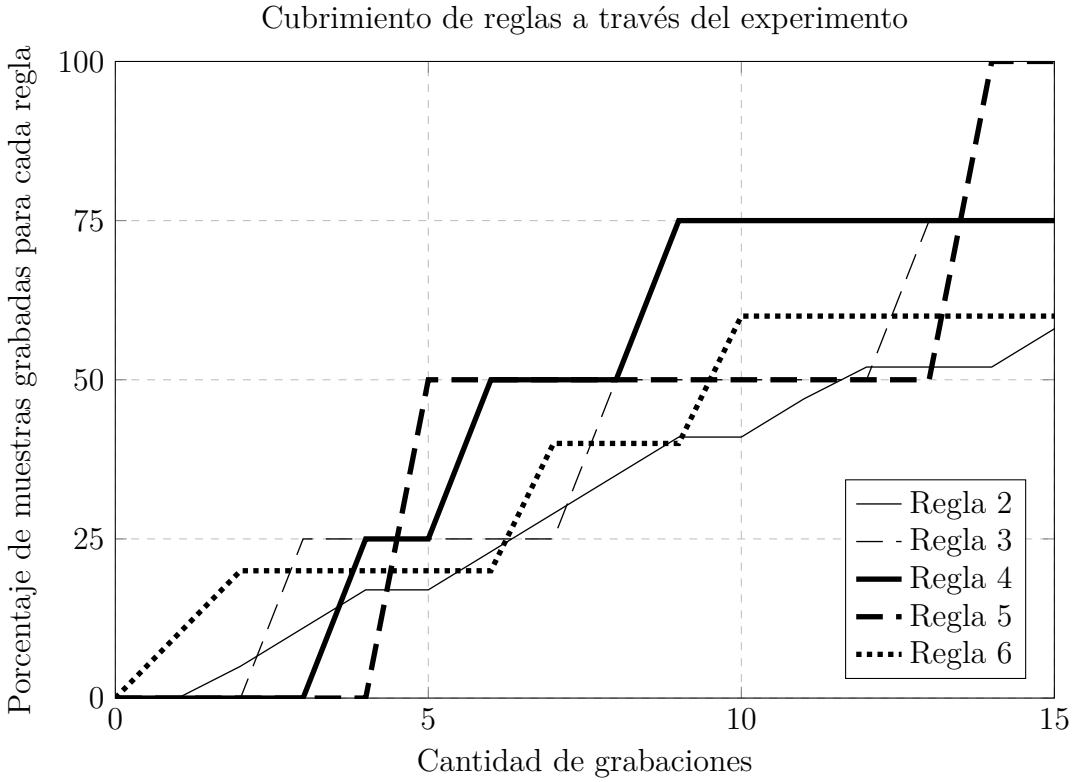


Figura 2.1: Porcentaje del total de frases grabadas por cada regla

La mínima cantidad de grabaciones que puede realizar un hablante son 5 grabaciones. Luego se le pregunta si quiere continuar grabando. Si acepta, se le agregan otras 5 grabaciones, siguiendo así sucesivamente hasta completar todas las frases a grabar de su traza. Elegimos grabar cada 5 grabaciones para que el hablante aporte el tiempo que tenga disponible y no obligarlo a grabar todas las frases. Aunque no grabe la totalidad de frases, las muestras van a servir en el experimento.

A continuación veremos cómo implementamos el sistema de grabación para soportar este experimento.

Capítulo 3

Sistema de grabación online

Para poder obtener grabaciones de distintas personas se desarrolló una página web. Esto es muy conveniente ya que nos permite grabar fácilmente desde cualquier lugar. En esta sección explicaremos la arquitectura del sistema y sus detalles técnicos.

La página web está desarrollada en Django, versión 1.4.2. Se eligió este framework por su facilidad a la hora de guardar objetos a la base de datos y también por la cantidad importante de bibliotecas que posee Python. La versión de Python que se utilizó es 2.7.3.

En la base de datos se guarda la información de cada hablante, las frases a grabar y las trazas. La base de datos elegida fue PostgreSQL versión 9.1 y se escogió ésta ya que es de código abierto entre otras cosas. Los audios se guardan en archivos *wav* por separado y también se guarda una referencia al nombre del archivo generado en la base de datos. Para el servidor HTTP se utiliza Apache versión 2.2.22. El servidor utiliza el sistema operativo Ubuntu 12.04.4 LTS.

3.1. Recolección de datos

Cuando un usuario visita nuestra página, primero debe llenar un formulario. Este le pregunta: género, fecha de nacimiento, lugar donde se crió y donde reside actualmente. Al confirmar el formulario, los datos son grabados en la base de datos en el servidor. Esto se puede apreciar en la figura 3.1. Luego se procede a realizar las grabaciones.

En la pantalla de grabación el usuario debe confirmar el acceso al micrófono que posee en su computadora, como se puede apreciar en la figura 3.2. Una vez hecho esto, se le explica las instrucciones del experimento y luego puede empezar a grabar.

Cada nuevo experimento utiliza una nueva traza del conjunto de trazas descriptas en el capítulo anterior. La interfaz que ve el usuario al grabar cada frase se puede ver en la figura 3.3. Las grabaciones pueden ser escuchadas por el usuario. Si el usuario al escucharla nota que no es una buena grabación, tiene la posibilidad de grabarla

Figura 3.1: Encuesta inicial del sistema

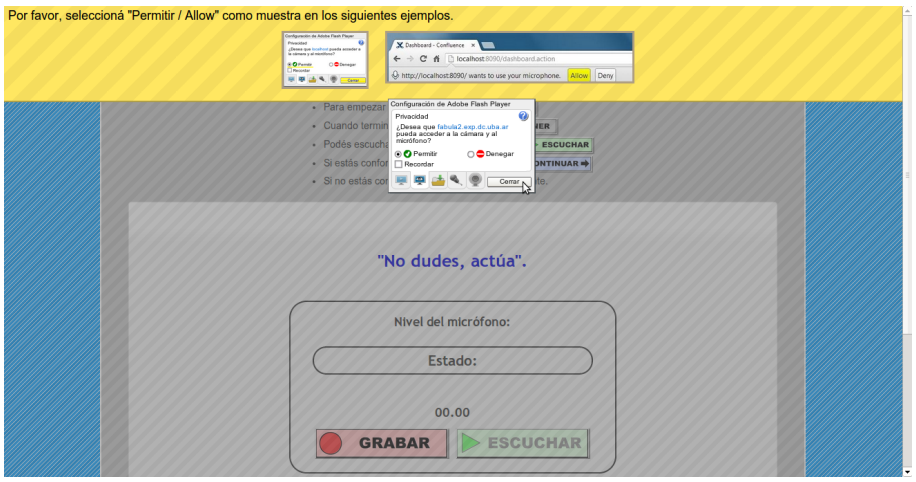


Figura 3.2: Se debe permitir micrófono para comenzar el experimento

otra vez. Lo importante es que la última grabación se escuche lo mejor posible. Para que el usuario reproduzca su grabación se aprieta el botón *Reproducir* como se ve en la figura 3.4.

Una vez que el hablante cree que su grabación se escuche bien, la confirma. Cada vez que se graba un audio, este se guarda en un archivo wav en el servidor. El archivo que se genera tiene una frecuencia de muestreo de 22050 Hz[1], cada muestra se analiza con 16 bits y posee un solo canal. Con estas características pudimos obtener un audio de buena calidad para el experimento que realizamos.

Recordemos que los hablantes graban en series de 5 frases. Una vez terminadas

[1]Necesitamos una frecuencia de muestreo que permita analizar las diferencias de los dos grupos sin ocupar excesivo espacio por cada grabación. Elegimos 22,05 KHz ya que nos permite realizar el experimento. Esta frecuencia de muestreo equivale a la mitad de frecuencia de un CD de audio.

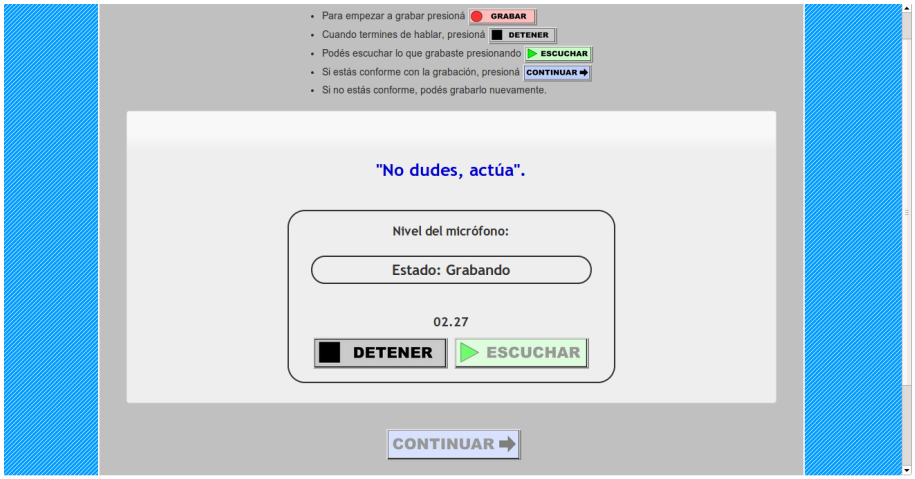


Figura 3.3: Grabando una frase

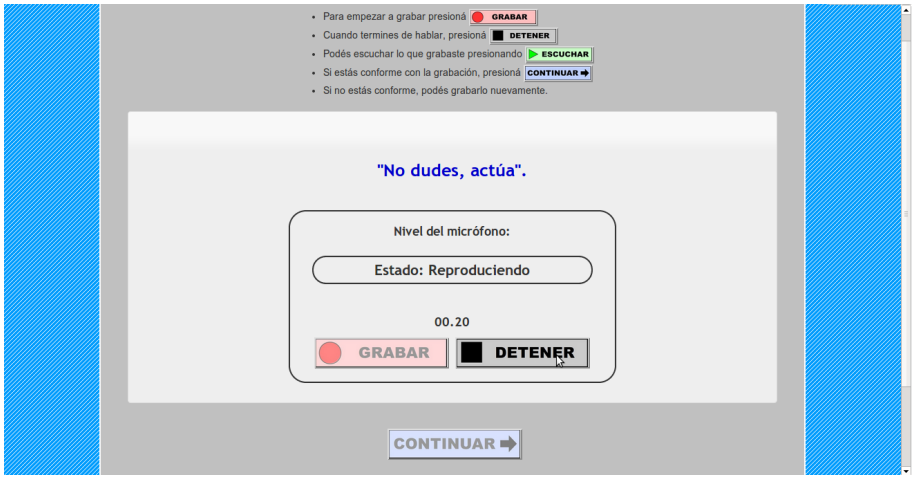


Figura 3.4: Reproduciendo la frase anteriormente grabada

estas 5 frases se le pregunta si querría seguir grabando o terminar el experimento. De esta forma, el hablante aporta grabaciones durante el tiempo que puede disponer.

3.2. Grabación a través del browser

Los navegadores actuales no permiten acceder al micrófono directamente. Actualmente está en desarrollo HTML5, tecnología que permitirá acceder al micrófono y a recursos similares de forma más fácil. No se eligió basarse en éste porque sólo los browsers más nuevos lo soportan. Al ser un estándar muy nuevo necesita que el usuario tenga instalada últimas versiones de software y utilizarlo hubiera excluido mucha gente. Teniendo en cuenta esto debimos utilizar una tecnología alternativa.

Encontramos un proyecto llamado Web Accessible Multimodal Interfaces [2] (WAMI). WAMI es una aplicación Flash que nos permite acceder al micrófono a través de JavaScript. Éste es muy utilizado en proyectos similares de procesamiento de habla. Esta herramienta nos permite definir dos URLs importantes: una que se utilizará para enviar el audio grabado y otra para escucharlo.

Cuando termina de grabar, se envía un mensaje POST al servidor a la URL configurada. El servidor obtiene el paquete de información y lo guarda como archivo wav. Cuando se quiere reproducir algún audio se envía un mensaje GET a la otra URL. El servidor lo responde con el audio requerido y se reproduce en el navegador. También con WAMI se puede configurar la calidad del audio grabado y analizar el nivel del volumen que posee. La biblioteca posee funciones que permiten saber qué nivel de volumen se encuentra en un instante preciso.

3.2.1. Requerimientos

Los requerimientos para participar del experimento son básicos: se debe disponer de micrófono y conexión a Internet. Tuvimos problemas con el browser que se utilizaba: WAMI necesita Flash versión 11.04 que no se encuentra en los repositorios tradicionales de Ubuntu. De esta manera, los navegadores que utilicen Flash instalado por el sistema operativo Ubuntu no podrán correr. Otros sistemas operativos como Windows o MacOS no tienen problemas con la versión de Flash instalada. De todas formas el navegador Chrome posee preinstalado la última versión de Flash, con lo cual este navegador puede correr perfectamente la aplicación sin importar el sistema operativo que se utilice.

3.3. Varias grabaciones por frase

Recordemos que para tener la mejor grabación de cada frase, le dimos la opción al hablante de que pueda escuchar como quedó. Esto requiere un ida y vuelta de paquetes entre el cliente (navegador) y el servidor.

Al grabar el cliente manda un mensaje al servidor con el audio de la grabación en crudo. La longitud de este paquete tiene un tamaño pequeño ya que las frases son de corta duración. Cuando el cliente quiere escucharlo envía un mensaje pidiendo ese mismo audio anteriormente grabado. El servidor envía el audio y es reproducido en el cliente. Este ida y vuelta de la grabación podría ser optimizada para que la grabación pueda ser escuchada sin tener interacción con el servidor. En nuestro experimento, no tuvimos problemas graves en lo que respecta al tiempo de retardo pero podría ser un punto débil del sistema que hay que tener en cuenta.

Puede resultar interesante analizar las anteriores grabaciones para ver por qué el hablante elige el último. Esta idea puede motivar algún trabajo futuro.

[2] Página web: https://code.google.com/p/wami/

3.4. Sistema de administración

Además de la interfaz pública para grabar, implementamos un sistema privado para administrar las grabaciones. Éste nos permite ver las grabaciones que fueron grabadas, la cantidad de grabaciones por cada frase que tenemos recolectada, la cantidad de trazas que todavía no se utilizaron, entre otras cosas. También nos permite escuchar y marcar las grabaciones para utilizarlas como primer filtro de las grabaciones. Explicaremos esto en más detalle a continuación.

3.4.1. Etiquetado de audio

Cuando varias personas terminan el experimento, los administradores pueden acceder a una página donde se puede escuchar cada audio que se va grabando. Los administradores escuchan las grabaciones y según su calidad los etiquetan con alguna de las etiquetas definidas. Las etiquetadas utilizadas esta vez son: 'Conservar', 'Sonido saturado', 'Mucho ruido de fondo', 'Problemas en el habla'. Esto se puede ver en figura 3.5.

Figura 3.5: Categorizando audios

Para acceder a los audios que fueron etiquetados de una determinada manera, el sistema tiene distintas URLs que nos permiten bajar todos esos audios en un archivo comprimido. Entonces si quisiéramos bajar todo el audio etiquetado con la categoría 'Conservar' podemos acceder a una URL y bajarlo sin necesidad de conectarse al servidor. Se necesita ser administrador del sistema para poder acceder a estas opciones.

3.5. Backups automáticos

El sistema posee backups que se generan a la noche, automáticamente. Los backups consisten en un volcado de información de toda la base de datos y en la sincronización de las grabaciones con una carpeta externa de backup.

3.6. Análisis del volumen

Un requerimiento primordial de este experimento es evitar grabaciones saturadas. Para ello medimos el volumen de la grabación mientras ésta se produce. El resultado es una serie de valores entre 0 a 100. Sobre estos valores calculamos el máximo y el mínimo. Si el primero es mayor a un cierto umbral arbitrario (o sea mayor a 80 por ejemplo) significa que la grabación saturó en algún momento. Si el mínimo es menor a un cierto umbral (o sea menor a 20 por ejemplo) quiere decir que hay mucho ruido ambiente. En cualquiera de los dos casos podemos pedirle al usuario que grabe nuevamente el experimento. De esta forma podemos filtrar grabaciones que no nos servirán para reconocer el acento.

Si bien la característica de filtrar por volumen fue programada, no fue utilizada en este experimento. El motivo fue que queríamos chequear cuán bien funcionaba la herramienta sin filtros y con completa participación de los usuarios. Otro motivo fue la paciencia de los hablantes: puede suceder que no se logre un ambiente beneficioso para grabar y el filtro rechace todas sus grabaciones, perdiendo un posible hablante. También notamos que había grabaciones que fueron rechazadas por el filtrado de volumen pero en realidad eran de buena calidad. Esto no lo queremos como primer experimento del framework. Por eso elegimos aceptar todas las grabaciones.

A continuación veremos cómo utilizamos esta información recolectada para el objetivo del experimento.

Capítulo 4

Extracción de información

Utilizando nuestra página web podemos obtener distintas muestras de Córdoba y Buenos Aires. ¿Cómo podemos analizar estas grabaciones correctamente? Un archivo wav, similar al que se genera en cada una de las muestras, posee muchísima información. Es por esto que debemos seleccionar correctamente qué partes de la información nos sirven y qué partes podemos descartar.

4.1. Alineación forzada

Una grabación a partir de una frase posee muchísima información. Debemos seleccionar qué parte de esta grabación nos interesa y qué parte puede ser descartada. Para ello etiquetamos en qué partes del audio se pronunció cada fonema y también, uniendo cada uno de estos fonemas, etiquetamos cada palabra. Por ejemplo: si tenemos la grabación de la frase *'El canapé salió espectacular'*, utilizamos un archivo aparte que nos dice *'«espectacular» se escucha entre el segundo 0.90 y 1.18'*. Lo mismo sucede para cada palabra y fonema de la grabación. Para marcar estas anotaciones utilizamos el formato de archivos TextGrid del programa Praat [Boersma and Weenink, 2013].

Un dato muy importante es que este etiquetado no debe tener que ser realizado con intervención de un humano. Si fuera el caso, tendríamos que hacerlo uno por uno, y al tener muchas grabaciones sería un trabajo muy arduo. De esto se encarga la alineación forzada. Las partes que debemos extraer de las grabaciones se encuentran en las diferencias de cada regla descripta anteriormente.

4.1.1. Prosodylab Aligner

Debemos tener una herramienta que nos permita obtener estos pequeños fragmentos de audio para analizar sus diferencias. Usamos una, llamada ProsodyLab Aligner [Gorman et al., 2011]. Su función es realizar alineaciones automáticas en cada una de las grabaciones de forma fácil. Analiza uno por uno cada grabación y mediante

un diccionario fonético determina en qué momento se dijo cada fonema y palabra.

Esta actividad se la conoce como alineación forzada. Básicamente consiste en marcar el tiempo desde que comienza y hasta que termina cada fonema, para todas las grabaciones. Esto se realiza de la siguiente forma: para cada grabación se agrega un archivo que enumera los fonemas que se escuchan en ella. La herramienta analiza las grabaciones y observa cada uno de los fonemas. Si un mismo fonema se utiliza en dos grabaciones, quiere decir que tiene haber características similares en la grabación en ese lapso de tiempo. Utilizando esta idea va prediciendo el lugar de cada fonema. Esta alineación no es perfecta, se puede equivocar en determinar dónde se escucha un fonema, pero es un buen comienzo.

Una particularidad que se destaca de esta herramienta es que no necesita datos de entrenamiento. Sólo con una hora de grabación es suficiente para correrlo y obtener resultados. Otra característica es que puede utilizarse para cualquier idioma. Esta herramienta está hecha en lenguaje Python (versión 2.5) y sirve como *wrapper* para utilizar HTK fácilmente. HTK es una biblioteca para trabajar con Modelos Ocultos de Markov (ver sección 4.1.2) fácilmente, y SoX, que nos permite trabajar con audio a través de la consola.

Los requisitos para utilizar ProsodyLab Aligner son: una hora de grabación con sus transcripciones fonéticas (no necesariamente alineadas a nivel fonético) y un diccionario fonético que nos provea para cada palabra los distintos fonemas que la componen. La hora de grabación la debíamos cumplir recolectando grabaciones de la página web. Esta meta era posible de realizar. La creación de un diccionario fonético era más complicada, ya que debía ser en español. Gracias al *Laboratorio de Investigaciones Sensoriales*[1] que nos prestó un diccionario, implementado por ellos, pudimos utilizar esta herramienta. Un diccionario fonético es básicamente un listado con las palabras que utilizamos y su transcripción en fonemas. Es importante esto ya que va a ser usado por el alineador para describir los fonemas de cada palabra en cada frase.

4.1.2. Modelos Ocultos de Markov

Los Modelos Ocultos de Markov [Rabiner, 1989] (en inglés Hidden Markov Model) son modelos estadísticos por los cuales tratan de predecir estados ocultos a partir de observaciones. Se llaman "ocultos" ya que el estado a predecir no se puede observar directamente. Sólo se puede predecir en qué estado oculto está el modelo a través de observaciones.

Un ejemplo de este modelo podría ser predecir la presión atmosférica observando sólo si el tiempo está lluvioso o seco. Los estados ocultos serían "Baja" o "Alta" presión atmosférica, que corresponde a lo que queremos saber. Las observaciones para predecir estos estados serían "Lluvia" o "Seco". En la figura 4.1 se puede apreciar este ejemplo. Para computar la secuencia más probable de estados ocultos dada una

[1]Este laboratorio depende de CONICET y UBA - Página web: http://www.lis.secyt.gov.ar/

secuencia de observaciones, se utiliza el algoritmo Viterbi.

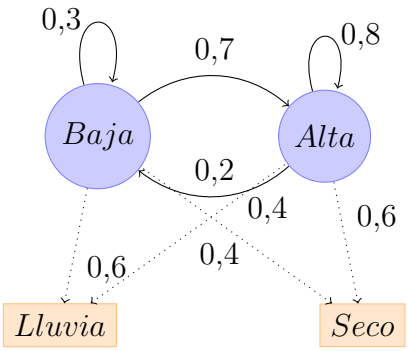


Figura 4.1: Ejemplo de HMM

Llevando esta idea a nuestro trabajo, los Modelos Ocultos de Markov tratan de predecir qué fonemas aparecen en cada parte de los audios utilizando la lista de fonemas pronunciada en cada grabación. Mediante este modelo matemático, el programa analiza en cuáles grabaciones, de la misma frase, se produce un mismo patrón de sonido. En las grabaciones que sucede esto, se marca como un fonema de la frase. Ese fonema va a ser marcado de igual forma en el TextGrid de cada una de las grabaciones con el mismo patrón de sonido. Los estados ocultos a predecir son los fonemas de cada frase, en cambio las observaciones provienen del análisis de cada audio. Entonces, a través de muestras va prediciendo los fonemas de las grabaciones.

A continuación se presenta un ejemplo de cómo funcionan estos modelos en nuestro trabajo. La figura 4.2 muestra las características de grabaciones de la misma frase producidas por dos hablantes distintos. Cada imagen tiene tres capas: el espectrograma de la grabación, su transcripción fonética (comienzo y fin de cada fonema en la grabación) y su transcripción ortográfica (comienzo y fin de cada palabra en la grabación).

Podemos observar que el espectrograma de ambas grabaciones es muy distinto. Si comparamos el mismo fonema en ambos espectrogramas, vemos que son muy diferentes. Aún así la alineación de los fonemas es similar. Si bien no son zonas iguales, poseen similares valores de atributos acústicos para esa región.[2] Los Modelos Ocultos de Markov analizan los atributos acústicos y teniendo en cuenta la secuencia de fonemas de la frase, que es idéntica para ambas grabaciones, puede predecir los tiempos de inicio y fin de cada fonema.

[2]Los atributos acústicos que se emplean son los MFCC, Mel Frequency Cepstral Coefficients, que serán descriptos más adelante, en la sección 4.2.2.

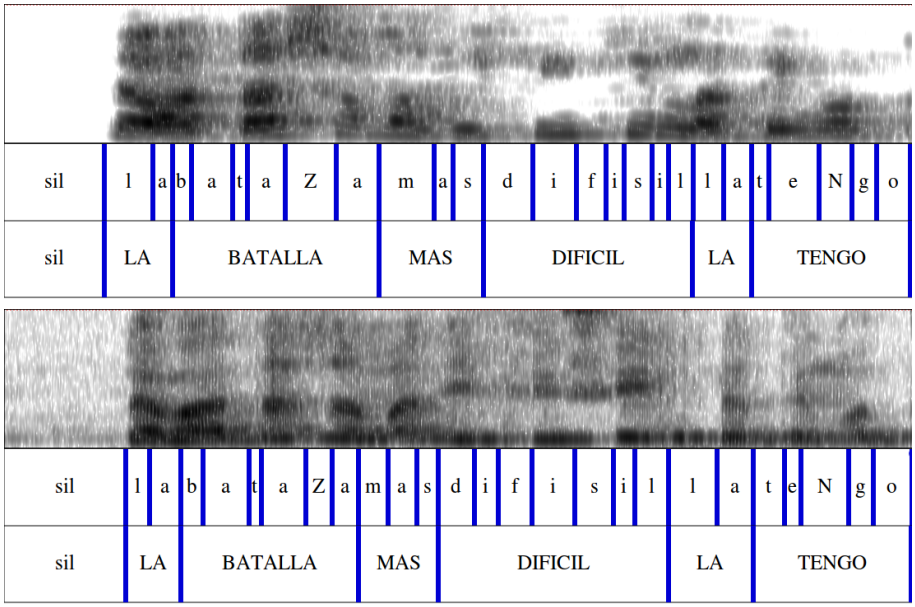


Figura 4.2: Espectrograma y transcripciones fonética y ortográfica de dos grabaciones producidas por hablantes distintos.

4.2. Extracción de atributos

La extracción de atributos fue realizada utilizando el lenguaje Python, que elegimos ya que es fácil de programar y tiene muchas bibliotecas útiles para este tipo de casos. Utilizamos una biblioteca muy conocida llamada Numpy (versión 1.6.1). Esta se utiliza para realizar cálculos matemáticos. Nosotros la utilizamos para tener buena precisión en el cálculo de los atributos.

Después de la alineación realizada, se ejecuta el extractor de atributos. Este posee como input los archivos wav y TextGrid que corresponden a las alineaciones temporales de cada fonema en cada audio. El diagrama de flujo del extractor se puede ver en la figura 4.3.

La rutina principal del programa toma una por una las grabaciones y su etiquetado asociado. Esta se representa en el componente "Extractor". Este llama a distintos componentes que van a calcular cada atributo. Para calcular estos atributos dividimos en dos componentes: "Extractor temporal" que se encarga de extraer *atributos temporales* y "Extractor acústico" que se encarga de extraer *atributos acústicos*. Los atributos temporales son calculados solamente utilizando el TextGrid, mientras que los atributos acústicos son calculados usando tanto el TextGrid como el archivo de audio. Si el atributo está presente en la grabación, tendremos ese dato en la extracción; si no, se dejará como desconocido. Luego de calculado todos sus atributos para cada grabación, juntamos todos los resultados y generamos el archivo Arff utilizando el componente "Generador de archivo Arff".

En el archivo Arff cada línea representa una grabación. Cada una de estas lineas posee todos los resultados del cálculo de los atributos separado por comas. Cada uno

de estos atributos corresponde al cálculo de una regla correspondiente. Necesitamos utilizar este formato ya que es necesario para ingresar datos en Weka, la plataforma que elegimos para correr algoritmos de *machine learning*. Veamos cada uno de los dos tipos de atributos.

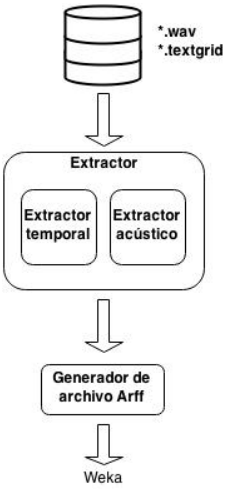


Figura 4.3: Diagrama de flujo

4.2.1. Atributos temporales

Los atributos temporales corresponden a atributos sobre la duración de los fonemas y las sílabas de cada grabación. Para calcularlos utilizamos como input el TextGrid generado en la alineación. Los atributos temporales se dividen en dos grupos: fonéticos y silábicos.

Básicamente para calcular un atributo se recorre el TextGrid buscando un patrón en particular y se mide su duración. También se mide la cantidad de ocurrencias que posee y luego se realiza su normalización de dos formas posibles.

La primera normalización, conocida como z-score, será utilizando la forma:

$$\frac{X - \mu}{\sigma}$$

donde:

- X es el valor a normalizar (por ejemplo: la duración de un fonema dado).
- μ es el promedio de duración de la unidad utilizada en la grabación.
- σ es el desvío estándar de la unidad utilizada en la grabación.

Y luego la segunda suponiendo que $\mu = 0$ (o bien, que la media no afecta negativamente los cómputos posteriores):

$$\frac{X}{\sigma}$$

Esta última tiene el nombre de Half-normal Distribution.

El valor a normalizar puede variar: mientras uno va a tener en cuenta fonemas, el otro tiene en cuenta sílabas. Debemos utilizar los datos normalizados ya que necesitamos atributos que nos muestren, para un hablante en particular, si el fonema en cuestión es relevante con respecto a los demás de la grabación. Al normalizar un atributo vemos cuán fuera de lo común resulta en el marco de *ese* hablante en particular en *esa* grabación. No importa si habla lento o rápido. Lo importante es la relación del fonema a medir con respecto a los demás. Lo mismo sucede para las sílabas. Si utilizáramos valores absolutos, se perdería esta relación ya que variaría con respecto a la velocidad del habla de cada hablante y cada grabación.

A continuación veamos los atributos en particular para cada uno de los grupos y cómo se realiza su cálculo. Para la regla 1 definimos atributos silábicos, ya que corresponde a reglas que están definidas para sílabas, mientras que para las demás reglas (2 al 6) definimos atributos fonéticos.

Atributos fonéticos

Los atributos que contabilizan fonemas son:

- **Duración de 'kt':** en este atributo buscamos el patrón /kt/ en los TextGrids y luego, en ese intervalo, medimos la duración del fonema /k/. Este atributo intenta extraer la diferencia explicada en la regla 4, que nos indica la duración de dicho fonema.
- **Duración de 'sc':** ídem con /sc/ y midiendo el fonema /s/. Este corresponde a la regla 3 que referencia a la duración del fonema /s/ anterior a /c/.
- **Duración de la 'll':** buscamos el patrón /ll/ y medimos su duración. Este atributo hace referencia a la regla 5 que mide dicho fonema.
- **Duración de 'rr':** ídem para /r/ fuerte. Referencia a la regla 6 que hace hincapié en este fonema.
- **Duración de 's' final:** ídem para las /s/ de final de palabra. Corresponde a la regla 2 que hace referencia a la aspiración de la /s/ de final de las palabras.
- **Duración de todos los fonemas:** este atributo suma la duración y la cantidad de todos los fonemas de una grabación y luego realiza su promedio. En este caso no se realiza normalización, ya que no se está tratando de analizar si un fonema en particular es destacado en comparación con los demás, sino que se trata de ver la duración en promedio de todos los fonemas en una grabación.

- **Duración de todas las vocales:** medimos la duración media de todas las vocales (juntas). Luego normalizamos para conocer la relación entre la duración de las vocales respecto de la duración de todos los fonemas.
- **Duración de todas las consonantes:** ídem anterior para todas las consonantes (juntas).

El cálculo de un atributo fonético se realiza de la siguiente manera: supongamos por ejemplo que queremos calcular la duración de 'kt' en la frase ***"En la pelea se conoce al soldado, sólo en la victoria se conoce al caballero"***. Analizamos el TextGrid asociado a la grabación que nos proveerá en qué instante se produjo cada fonema. Los fonemas en esta frase van a ser ***"en la pelea se konose al soldaDo solo en la biktorja se konose al kaBaZero"***. En la figura 4.4 se puede ver una representación gráfica del TextGrid marcando los tiempos para cada fonema[3]. Se marca en negro el segmento donde aparece la 'k'.

Los valores de la normalización serán: para μ se calculará como el promedio de duración de los fonemas en la frase en cuestión. Viendo la figura 4.4, será el promedio de los tiempos marcados para todos los fonemas. σ será el desvío estándar de la duración de todos los fonemas de la frase. Y X será el promedio de duración de los fonemas de la forma /k/ en el intervalo /kt/ correspondiente. La única aparición de este fonema es en la palabra "biktorja" y en el gráfico está marcado en negro. La misma idea se aplica en el cálculo de los demás atributos.

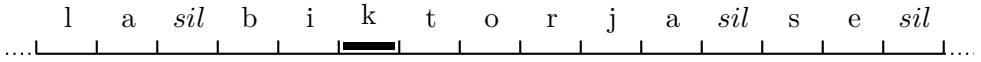


Figura 4.4: Ejemplo de cálculo de atributo

En definitiva, se busca el patrón definido por el atributo, se mide la cantidad de ocurrencias que posee y luego se realiza su normalización de las dos formas utilizando esos valores.

Atributos silábicos

Los atributos que contabilizan sílabas usados son:

- **Duración de la sílaba acentuada:** en cada una de las grabaciones buscamos la sílaba acentuada de cada palabra, medimos su duración y normalizamos con las demás sílabas de la frase. Ésto lo realizamos recorriendo su TextGrid asociado.
- **Duración de la sílaba anterior a la acentuada:** realizamos el mismo calculo anterior pero con la sílaba previa a la acentuada.

[3]Aclaración: la duración de los fonemas varía muchísimo. En el ejemplo se simplificó marcando todos los tiempos con el mismo tamaño para hacer más simple la figura.

Veamos cómo se realiza el cálculo de un atributo silábico: supongamos que queremos calcular el atributo que corresponde a la duración de la sílaba anterior a la acentuada y lo realizamos para la misma frase que en el caso anterior. μ representará el promedio de duración de las sílabas en la frase. σ será el desvío estándar de la duración de estas sílabas en la frase. Y finalmente X será el promedio de duración de las sílabas anteriores a las acentuadas. Para cada uno de estos valores se calculan los dos tipos de normalización. En la figura 4.5 podemos ver este ejemplo gráficamente[4]. No pudimos escribir toda la frase y todos sus acentos por cuestiones de espacio.

En la frase del ejemplo marcamos las sílabas anteriores a la acentuada para distinguirlas. El analizador, cuando calcule este atributo, va a identificar las sílabas acentuadas y tomará su antecesora.

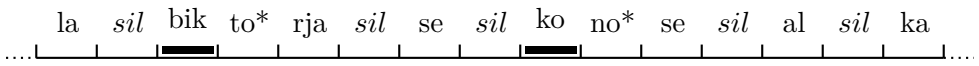


Figura 4.5: Ejemplo de cálculo de atributo

Para saber cuál es la sílaba acentuada se realizó una tabla que describe para cada frase cuáles son sus sílabas acentuadas. Ésta se encuentra en el apéndice de este informe. Estos atributos los usamos para poder medir la regla 1, la más prominente de la tonada cordobesa.

4.2.2. Atributos acústicos

Los atributos acústicos utilizan las propiedades de las grabaciones realizadas. Para ello debimos extraer información con algún método que permita medirlos. Elegimos el calculo de coeficientes cepstrales en escala de mel (en inglés MFCC: Mel Frequency Cepstral Coefficient) ya que tiene relación directa con la percepción auditiva humana. Veamos el cálculo de estos coeficientes.

La forma en que hablamos se produce por el movimiento de varias articulaciones en nuestra boca. Estas conectan a dientes, lengua, tráquea, etc. Estas articulaciones trabajan en conjunto para darle forma al sonido producido.

El tracto vocal suele modificarse de manera bastante lenta y se la puede considerar constante en intervalos de alrededor de 10 a 20 ms. Por otro lado, la señal de audio poseen muchas variaciones continuamente. Para minimizar las discontinuidades de la señal se aplica un ventaneo sobre la señal del habla.

Las ventanas consecutivas, aplicadas a la señal del habla, se solapan. Esto permite evitar las transiciones abruptas de la señal. Las ventanas empleadas son de 20 a 40 ms de duración, pero los vectores de atributos suelen calcularse cada 10 ms. Estos intervalos son conocidos como trama o frame. Este ventaneo se puede apreciar en la figura 4.6.

[4]Aclaración: al igual al caso anterior, la duración de las sílabas varía muchísimo. En el ejemplo se simplificó marcando todos los tiempos con el mismo tamaño para hacer más simple la figura.

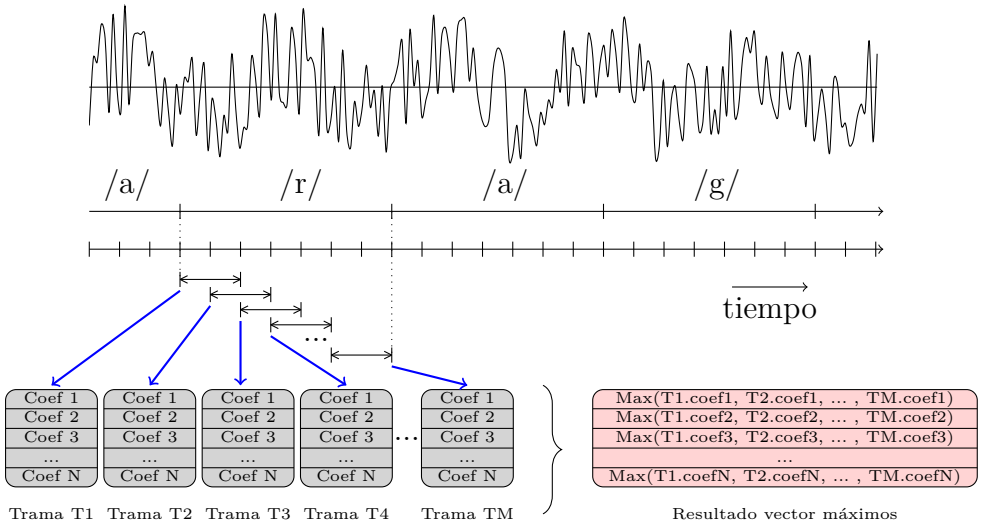


Figura 4.6: Representación del cálculo de MFCC: al encontrar el fonema /r/ aplicamos el ventaneo en toda su duración. Para cada uno de ellos, se calcula los coeficientes MFCC. Finalmente se arma una traza con los máximos de cada coeficiente. Ídem para el mínimo y promedio.

Luego de tener separada la señal en pequeñas tramas, para cada una de ellas se calcula la Transformada Discreta de Fourier. Posteriormente se utiliza un banco de filtros solapados y espaciados uniformemente sobre la escala de Mel (conocido en inglés como Mel Filterbank). Como dijimos antes, esta escala es importante ya que representa la percepción auditiva humana. A los valores de energía que superaron este filtro se les aplica logaritmo. Para finalizar, a estos se les aplica la Transformada Discreta del Coseno.

Terminado el algoritmo obtenemos 13 atributos acústicos de ese segmento. Agregamos también los coeficientes delta y delta-delta de estos atributos que representan las variaciones temporales. En total llegan a 39 atributos acústicos. Estos atributos se obtienen en una Trama.

Este cálculo de vectores de MFCC lo realizamos entre el principio y fin de cada atributo que represente una regla. Por ejemplo como observamos en la figura 4.6: si encontramos el atributo fonético /r/, calculamos los vectores MFCC en el intervalo donde suena este fonema. Con éstos vectores, calculamos el valor máximo, mínimo y promedio para cada enésima posición y armamos tres vectores con estos valores. O sea, el vector de valores máximos tendrá en su primer elemento el valor máximo de los primeros elementos de todos los vectores, así sucesivamente. Ídem para el vector mínimo y promedio.

Para realizar el cálculo de estos coeficientes se utilizó un script en Matlab[5]. Este

[5]El creador del script es Kamil Wojcicki y se puede descargar de la página web http://www.mathworks.com/matlabcentral/fileexchange/32849-htk-mfcc-matlab/

calcula los 39 componentes de cada trama. El extractor necesita estos valores para cada audio a extraer, es por eso que se conecta con Matlab a través de un wrapper para ejecutar el script y luego continuar con la extracción.

4.2.3. Atributos desconocidos

Una grabación no puede definir todos los atributos que declaramos. Cada frase define sólo algunos atributos y los demás no provee información para sabe cuál es su posible valor. Es por eso que debemos tomar una decisión sobre qué hacer con los atributos que no conocemos.

Determinamos que a los atributos que no pudieron ser extraídos en la grabación los marcaremos como desconocidos. Si para una grabación un atributo es desconocido, no sabremos ese valor para ese hablante. Debemos analizar otra grabación del mismo hablante que defina ese atributo.

4.2.4. Nomenclatura utilizada

Para referenciar cada uno de los atributos debimos definir una nomenclatura. La definición que tomamos es la siguiente:

TIPO+"_"+ATRIBUTO+"_"+NORMALIZACIÓN

- *TIPO:* puede ser *FON, SIL o ACU*. Esto corresponde al tipo de atributo, si es fonético, silábico o acústico.
- *ATRIBUTO:* puede ser *kt, ll, sc, rr, Sfinal, vowel o consonant* haciendo alusión a cada una de los atributos. También aquí se encuentran los atributos generados por MFCC cuyos nombres son de la forma *(Min | Max | Avg) + (KT | LL | SC | RR)*. Para hacer referencia a las reglas sobre atributos silábicos utilizamos los nombres de *syllableAccent* y *prevSyllableAccent* para la duración de la sílaba acentuada y de la sílaba anterior a esta.
- *NORMALIZACIÓN:* corresponde al tipo de normalización realizada. Estas pueden ser *norm* haciendo alusión a z-score o *normhd* haciendo alusión a normalización tomando $mu = 0$.

Por ejemplo: definimos *SIL_prevSyllableAccent_normhd* como la duración de la silaba anterior a la acentuada aplicando normalización con $mu = 0$. Todos los nombres y a qué atributo se refieren se pueden ver en el apéndice de atributos.

En la próxima sección veremos las grabaciones obtenidas en este experimento para luego analizar sus atributos.

Capítulo 5

Datos obtenidos

En este capítulo describimos los datos obtenidos a través del framework desarrollado.

Tuvimos algunos problemas al recolectar las grabaciones. El principal problema fue que el ambiente utilizado por muchos hablantes no estaba completamente en silencio como para realizar buenas grabaciones. Muchos errores surgieron en esa dirección. Otros errores comunes pero no tan frecuentes fueron: interpretaciones erróneas de la consigna, errores de volumen del micrófono y saturación.

5.1. Evaluación manual de las grabaciones

Como primer paso, se realizó una clasificación manual de las grabaciones para determinar si las mismas se realizaron correctamente. Para la misma se utilizó la herramienta de administración que vimos en el capítulo 3. Las clases que utilizamos fueron: "Conservar", "Sonido saturado", "Mucho ruido de fondo", "Problema en el habla". La cantidad de grabaciones de cada clase se puede ver en la tabla 5.1. Entre paréntesis se puede observar la cantidad de hablantes que produjeron esas grabaciones.

	Bs.As.	**Cba.**	**Total grabaciones**
Conservar	220 (19 hab.)	90 (8 hab.)	310
Problemas en el habla	33 (11 hab.)	15 (5 hab.)	48
Mucho ruido de fondo	2 (2 hab.)	12 (2 hab.)	14
Sonido saturado	2 (2 hab.)	0	2

Tabla 5.1: Evaluación manual de las grabaciones. Repreenta cantidad de grabaciones para cada clase y cantidad de hablantes entre paréntesis

Como puede observarse, los datos obtenidos están desbalanceados. No fue posible obtener la misma cantidad de grabaciones para los dos grupos. Esto se va a reflejar en la clasificación y en el análisis posterior.

Las categorías establecidas anteriormente describen los errores más frecuentes. Podemos observar que, del total de 374 grabaciones, 64 tuvieron algún problema. Esto representa alrededor del 17 % de las grabaciones. Es un número alto para ser un experimento guiado. La gran causa de este número es la falta de chequeo al aceptar un audio nuevo. La detección automática de errores en el momento de grabación es un tema que excede los objetivos de esta tesis y se propone como trabajo futuro.

El análisis que se presenta a continuación está basado solamente en las grabaciones clasificadas como "Conservar".

5.2. Alineación forzada

El alineador automático no realiza su función de forma perfecta. En ocasiones, el proceso de alineamiento forzado introduce errores. Es muy importante descartar las grabaciones mal alineadas, ya que si no lo hacemos, cuando los procese el extractor de atributos nos darían información errónea. Chequeamos cada grabación etiquetado como "Conservar", y analizamos con la aplicación Praat [Boersma and Weenink, 2013] si la alineación fue correcta. Los errores de alineamiento más comunes se debieron a:

- **Ruido de fondo:** las grabaciones donde el alineador se comporta de peor manera son aquéllas en las que se escucha ruido de fondo. En esos casos, las alineaciones resultan muy malas. Lamentablemente en nuestro caso esto es bastante común. En la figura 5.1 se puede ver un ejemplo utilizando Praat.

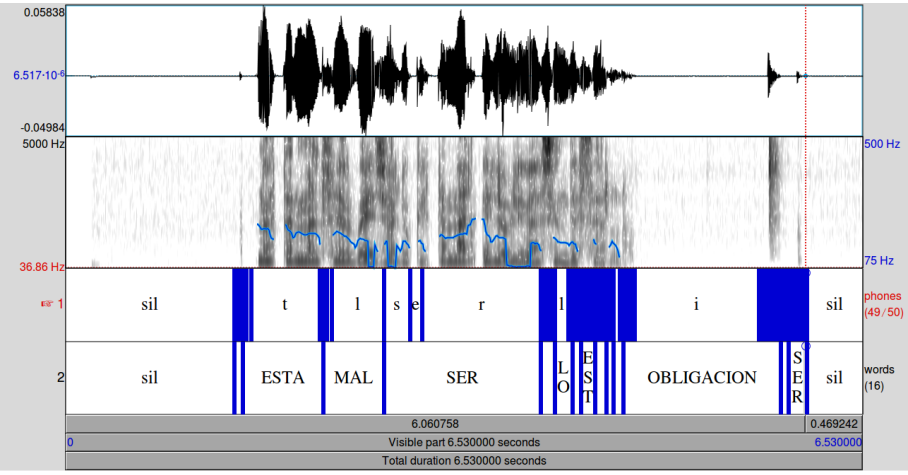


Figura 5.1: Ejemplo de alineación mala por ruido

- **"Mouse clic" al finalizar:** las grabaciones recibieron ruido del movimiento propio del hablante. Pasó en muchas oportunidades que el clic de finalizar del mouse se grabó como parte final en el audio (Ver figura 5.2). Ese sonido se grabó y afectó la alineación de forma tal que el alineador lo confunde con habla.

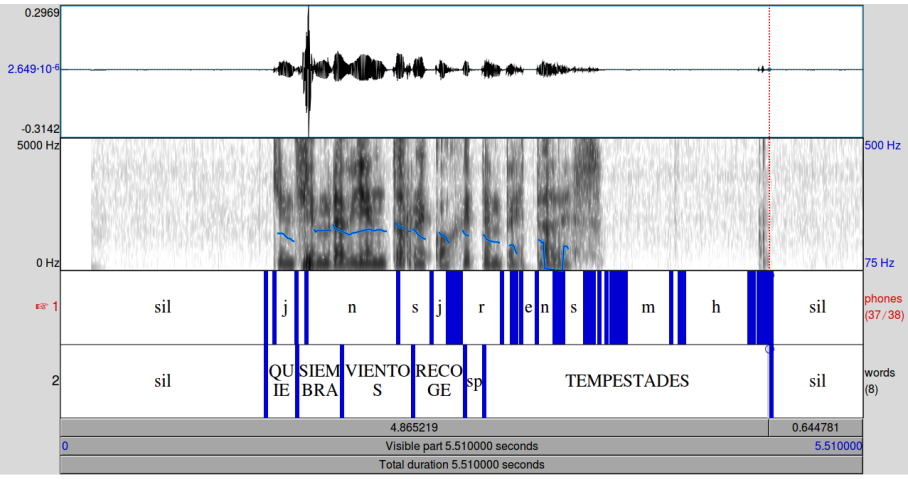


Figura 5.2: Clic al final

- **Saturación del micrófono:** el volumen del micrófono es configurado por el hablante. Es por ello que debemos confiar en su buena voluntad. Muchas veces la grabación fue buena pero en algunas partes la entonación tuvo mayor volumen que en otras; haciendo que, posteriormente, la alineación no sea precisa.

- **Entonación exagerada:** en algunas grabaciones se quiso exagerar la entonación. Por ejemplo, las palabras finalizadas en /s/ fueron grabadas en muchos casos sosteniendo ese fonema por tiempo prolongado. En la mayoría de las grabaciones no sucedió, por lo que no afectó el análisis del experimento. El problema que surgió en estos casos fue que el hablante no supo pronunciar la frase de la forma más natural posible.

5.3. Corrección de errores

Para corregir los errores descriptos debimos chequear manualmente cada uno de los TextGrids. Los resultados de la cantidad de alineamientos corregidos se pueden ver en la tabla 5.2.

	Bs.As.	**Cba.**	**Total**
Modificados	101	88	189
Correctos	119	2	121
Total	220	90	310

Tabla 5.2: Cantidad de alineamientos corregidos

Entonces las grabaciones que utilizamos salieron de estas 310 alineaciones.

Recordemos que cada hablante tenía la posibilidad de grabar varias veces la misma frase con la idea de que la última grabación fuera la mejor grabada. Debemos

quitar estos casos para no analizar frases que están mal grabadas. Contabilizamos las grabaciones repetidas y los mostramos en la tabla 5.3.

	Bs.As.	**Cba.**	**Total**
Todos los intentos	220	90	310
Último intento	**181**	**79**	**260**

Tabla 5.3: Cantidad de audios repetidos

Entonces las grabaciones extraídas con la herramienta que utilizaremos para clasificar son: 181 grabaciones para Buenos Aires y 79 para Córdoba. Estos audios corresponden a 8 hablantes de Córdoba y 19 de Buenos Aires. En la próxima sección veremos el análisis que realizamos con estas grabaciones.

Capítulo 6

Análisis

En esta sección, analizamos los datos obtenidos para entrenar clasificadores que distingan entre hablantes de Buenos Aires y Córdoba. Recordemos que tomamos las 260 grabaciones obtenidas a través de la página web y les aplicamos el extractor de atributos descripto en el capítulo 4. El resultado nos dio la descripción de cada grabación a través de los atributos que definimos.

Primero presentaremos el baseline que consideramos. Este nos servirá para tener una clasificación aceptable que luego trataremos de superar. Explicaremos los clasificadores utilizados para vencer esta marca y en base a tests estadísticos notaremos si aportan datos significativos. También describiremos el modelo de testing utilizando los datos recolectados. Por último, analizamos los atributos más descriptivos del habla de Buenos Aires y Córdoba.

Para el análisis de los datos, utilizamos la herramienta Weka[1]. Ésta provee varios algoritmos de machine learning. Para los tests estadísticos utilizamos la herramienta R versión 3.0.1.

6.1. Baseline

El baseline define el clasificador más simple, que posteriormente tratamos de vencer. No encontramos ningún trabajo que trate de distinguir entre porteños y cordobeses a partir de su habla. Es por eso que definimos el baseline utilizando el algoritmo **majority class**. Este algoritmo clasifica eligiendo siempre la categoría que en el conjunto es mayoritaria. Por ejemplo, si nuestro conjunto de datos de train tiene más muestras de Córdoba para la clasificación de nuevas instancias elegimos siempre Córdoba. La herramienta Weka provee un clasificador basado en majority class llamado **ZeroR**. Utilizaremos este para el cálculo del baseline.

Recordemos que, como vimos en el capítulo anterior, el conjunto de datos que obtuvimos posee más grabaciones de Buenos Aires que de Córdoba. Utilizando nuestros

[1]Página web: http://www.cs.waikato.ac.nz/ml/weka/

datos, este baseline tuvo una performance alrededor del 69 % de efectividad. Nos referimos a efectividad como la probabilidad de clasificar correctamente un hablante. Éste fue el porcentaje a superar. Si nuestro conjunto de datos estuviera debidamente balanceado este porcentaje sería exactamente del 50 % ya que, al elegir la clase mayoritaria, ninguna de las clases superaría a la otra. Lo ideal sería poder tener misma cantidad de los dos grupos. Al tener este desbalance, puede suceder que al clasificar a un hablante en un test se obtengan mejores resultados para Buenos Aires que para Córdoba, ya que tengo más muestras para identificar a un hablante. Esto se verá en detalle más adelante.

6.2. Clasificadores

Entrenamos varios clasificadores para poder determinar la procedencia de un hablante y superar la performance del clasificador baseline. Elegimos varios clasificadores de distintos tipos para éste propósito.

Recordemos que para cada grabación sólo se determinan los atributos que esa frase define y los demás quedan en valor desconocido. Es por ello, que nuestros clasificadores van a tener que manejar atributos desconocidos. Algunos clasificadores tienen la posibilidad de trabajar con estos atributos de forma natural, otros no.

Describimos cómo trabaja cada clasificador y además cómo trabaja con los valores desconocidos. Los clasificadores propuestos son:

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [Cohen, 1995] - Implementación JRip:

Este algoritmo intenta describir el conjunto de entrada definiendo pequeños grupos. Primero localiza un grupo que posee la característica a clasificar, y genera reglas que lo describan. Va agregando reglas de forma golosa. Luego cuando se supera una cierta condición (por ejemplo: cantidad de reglas), lo extrae y sigue con otro grupo. Finaliza cuando describe todos los grupos del conjunto de entrada. Entonces la forma de clasificar es definiendo una serie de reglas lógicas utilizando los atributos.

Cuando se producen las reglas en la fase de entrenamiento, las instancias que poseen atributo indefinido para la regla que se esta armando no son consideradas para el armado de la misma, o sea no se toman en consideración para las variables internas del algoritmo. Esto tiene la ventaja de que se dejan de lado para el final: primero se arman las reglas para las instancias que sí pueden ser clasificadas y luego, quedan solamente las instancias con valores indefinidos, pudiendo ser más fácil el armado de una regla que las identifique.

Si al momento de clasificar, una instancia posee un atributo desconocido y éste es necesario para una determinada regla de decisión, esta regla falla y sigue con la siguiente regla. Se evalúa la próxima regla de la misma forma hasta llegar a una posible clasificación.

C4.5 [Quinlan, 1993] - Implementación J48:

Este algoritmo se basa en un árbol de decisión. Tenemos nuestro conjunto de entrenamiento con instancias clasificadas y cada una de estas instancias consiste en un vector con sus atributos. El algoritmo realiza lo siguiente: para cada nodo del árbol se elige el atributo que efectivamente mejor separa a los dos grupos. El criterio para encontrar este atributo es utilizando la ganancia de información de cada atributo. El atributo con la mayor ganancia de información es utilizado como nodo del árbol. Luego se llama recursivamente por cada subconjunto que dividimos. Si las muestras pertenecen a la misma clase o los atributos no proveen información se crea sólo una hoja.

Las instancias que poseen valores indefinidos para el atributo que mejor separa en subgrupos no son separados ni en uno ni en otro subconjunto, sino que son utilizados en ambos. Entonces estas instancias en este paso no aportan diferencia entre una rama u otra. Lo interesante es que más adelante, calculando otro nodo del árbol, sí pueden ir a un subconjunto específico.

Si al momento de clasificar una instancia, el árbol utiliza un atributo desconocido se realiza lo siguiente: propaga por cada rama del nodo hacia abajo pero en cada camino sigue con un peso proporcional a la cantidad de elementos que fueron por cada rama en el entrenamiento. Luego, al llegar a los nodos hoja, combina los resultados utilizando esos pesos y elige la clase con mayor peso.

Support Vector Machine [Platt, 1998] - Implementación Function SMO:

Support vector machine define uno o varios hiperplanos para intentar clasificar muestras. Este hiperplano se construye utilizando transformaciones lineales de los datos de entrada y sirve para clasificar las muestras en dos grupos. Utilizando este hiperplano, se puede etiquetar cada dato de entrada con su clasificación observando de qué lado del hiperplano se encuentra.

Este algoritmo no puede ejecutarse si se posee instancias con atributos desconocidos. Es por eso que antes de empezar el entrenamiento a cada instancia con atributos que poseen valores desconocidos se la define con el valor de la media de ese atributo de las demás instancias. De esta forma, vamos a otorgarle un lugar aproximado en el hiperplano para ese atributo. Esto también sucede si al momento de clasificar la instancia posee un atributo desconocido.

Naive Bayes [Zhang, 2004] - Implementación homónima:

Un clasificador de tipo Naive Bayes supone que cada atributo describe una característica de su clase y no está relacionado con otro atributo. Cada uno de estos atributos contribuye de manera independiente a la clasificación de su clase. Se define una regla de decisión utilizando un modelo probabilístico basado en el teorema de Bayes para la clasificación de cada grupo.

Si un valor es indefinido en una instancia de entrenamiento, simplemente no es incluida para el cálculo de la clasificación. Los valores que son tenidos en cuenta son los que están definidos, o sea los que ocurren realmente. Esto es posible gracias a que los atributos son tenidos en cuenta como independientes.

6.3. Tests estadísticos

Utilizamos los resultados de cada clasificador para ver si son significativamente relevantes en la predicción de cada hablante en comparación con el baseline. Para realizar estos tests utilizamos para cada clasificador un vector con los porcentajes de instancias correctas para cada fold generado[2]. Los clasificadores utilizados son los descriptos en la sección anterior más el baseline. Para estos resultados realizamos dos tests principales: Prueba de rangos con signo de Wilcoxon y Test t de Student.

6.3.1. Test de Wilcoxon

Utilizamos el test de Wilcoxon ya que no estamos seguros que nuestros datos provengan de una distribución Normal. Este nos ayudará a determinar si hay razones estadísticas para decir si un clasificador es distinto que otro. Para realizar este test armamos un vector para cada clasificador incluyendo el baseline. Este vector tendrá el porcentaje de instancias correctas para cada uno de los folds. Entonces correremos el test estadístico utilizando el vector del clasificador baseline y el vector de otro clasificador, por ejemplo Support vector machine.

Las hipótesis fueron:

H_0: Clasificador alternativo no es diferente que ZeroR

H_1: Clasificador alternativo es diferente que ZeroR

Clasificador alternativo se refiere a los demás clasificadores descriptos.

Cada uno de los tests nos va a dar un p-valor. Si este es mayor 0,05, consideramos que no hay evidencia suficiente para determinar que el clasificador alternativo es mejor. Si de lo contrario es menor, sí podemos rechazar H_0 y asegurar que el alternativo es mejor.

6.3.2. Análisis Shapiro-Wilk Test

Utilizamos el test de Shapiro-Wilk para poder afirmar si un conjunto de datos proviene de una distribución Normal.

[2]Veremos cómo se componen los distintos folds más adelante en la sección 6.4

El test de Shapiro-Wilk se basa en plantear como hipótesis nula que la población está distribuida de forma Normal. Aplicamos el estadístico de este test: si el p-valor nos da menor a 0,05 entonces la hipótesis nula es rechazada y se afirma que los datos no provienen de una distribución Normal. Si, en cambio, es mayor a 0,05 no hay evidencia suficiente para rechazar H_0 y por ende se afirma que los datos siguen una distribución Normal.

Este test se realiza individualmente para cada vector resultado de porcentaje de instancias correctas para cada clasificador. O sea, chequeamos que los resultados de cada clasificador se asemejen a la distribución Normal. Por ejemplo si los resultados de ambos clasificadores ZeroR y J48 tuvieron en el test Shapiro-Wilk un p-valor mayor a 0,05, se puede realizar el t de Student para ellos dos. En caso contrario, se debe usar el test de Wilcoxon.

6.3.3. Student t Test

Para los vectores resultado provenientes de una distribución Normal se les aplica este test. Éste nos provee una forma de determinar si dos conjuntos de test son significativamente distintos, suponiendo que surgen de una distribución Normal. Para aplicar este test, los conjuntos de test deben tener muestras independientes entre sí. Para realizar este test utilizamos, como en el Test de Wilcoxon, dos vectores: uno para los resultados de Zero Rule y otro para un clasificador alternativo. También de la misma forma que planteamos la hipótesis del test de Wilcoxon, este va a tener las siguientes hipótesis.

H_0: No hay diferencias entre Zero Rule y clasificador

H_1: Hay diferencias entre Zero Rule y clasificador

La ventaja de usarlo es que, al saber qué distribución representa, obtenemos resultados más precisos. Aplicando el estadístico obtuvimos un p-valor. De la misma forma, si este es mayor a 0,05 no hay evidencia suficiente para rechazar H_0. De lo contrario, sí hay evidencia y rechazamos H_0.

6.4. Modelos de tests

En esta sección explicamos en detalle varios modelos de tests así como también qué resultados obtuvimos en cada uno de ellos. Éstos se basaron en la técnica de validación cruzada (en inglés cross-validation) por la cual utilizando el conjunto de los datos se arman varias iteraciones, donde en cada una de ellas se separa parte de los datos para entrenar y otra para testear.

Los siguientes modelos de tests son los que probamos:

- Clasificación por muestra: validación cruzada por cada grabación dejando un hablante afuera (sección 6.4.1)
- Clasificación por hablante: validación cruzada por hablante dejando uno afuera promediando los atributos de cada hablante (sección 6.4.2)

6.4.1. Clasificación por muestra

Definimos un modelo de test donde enfatizamos la idea de distinguir un hablante. Para cada fold generado, excluiremos un hablante y entrenaremos con todos los demás. Luego testeamos contra ese hablante excluido. Este esquema evita que tengamos grabaciones repetidas en los grupos de tests. Podemos ver este esquema en la tabla 6.1. También es conocido como **validación cruzada dejando uno fuera** (en inglés: Leave-one-out cross-validation)

En nuestro conjunto de datos tenemos 27 hablantes: 19 de Buenos Aires y 8 de Córdoba. Vamos a tener 27 folds distintos para cada uno de ellos. Cada uno de estos folds va a excluir todos los audios de este hablante, por eso cuando nos referimos a un hablante nos referimos a todos los audios grabados por él.

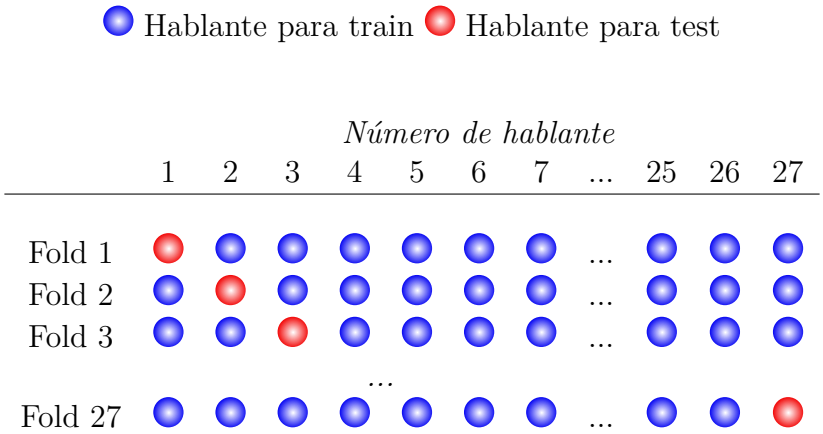


Tabla 6.1: Esquema de validación cruzada

Resultados

En la tabla 6.2 podemos observar los resultados de clasificación. No resulta relevante mostrar el porcentaje de clasificación correcta para cada fold. Expresamos el promedio de clasificación correcta de cada fold para cada clasificador como se muestra en la siguiente fórmula.

Promedio de porcentaje correcto en cada fold:

$$\frac{\sum_{i=1}^{\#\text{ de folds}} \text{Porcentaje instancias correctas en fold } i}{\#\text{ de folds}}$$

	Zero Rule	**Ripper**	**C4.5**	**SVM**	**NaiveBayes**
Promedio	70.37	69.47	70.37	71.34	71.46

Tabla 6.2: Clasificación correcta en porcentaje

Llama la atención que todos los clasificadores tuvieron porcentajes muy cercanos. Esto lo analizaremos más adelante.

Realizamos los test estadísticos de Wilcoxon y Test t de Student para este modelo de test. Estos resultados se pueden observar en la tabla 6.3.

	Student Test	**Wilcoxon Test**
ZeroR y Ripper	0.5816	0.6234
ZeroR y C4.5	1	1
ZeroR y NaiveBayes	0.4383	0.4042
ZeroR y SVM	0.4302	0.2646

Tabla 6.3: Resultados de cada test representado en p-valor

Los valores expresados corresponden al p-valor. Todos los clasificadores pasaron el test Shapiro-Wilk, entonces podemos afirmar que los resultados de cada clasificador corresponden a una distribución Normal. A pesar de esto, realizamos ambos tests estadísticos para cada uno. Estos test mostraron que las diferencias entre cada clasificador y el baseline no son estadísticamente significativos.

Recordemos que no todos los hablantes aportaron igual cantidad de grabaciones. Ellos tenían la opción de seguir aportando grabaciones o salir de la aplicación. Es por esto que cada porcentaje de fold no aporta igual cantidad de instancias correctas en cada grupo de testeo. Por ejemplo: puede suceder que la cantidad de instancias para un fold en su grupo de testeo sea de una grabación y, si esa muestra fue correctamente clasificada, el porcentaje de instancias correctas dará 100 %, mientras que, otro fold puede tener 10 instancias y clasificar correctamente 9, concluyendo que su porcentaje de instancias correctas dará 90 %. En la figura 6.1 podemos ver la cantidad de grabaciones en cada grupo de testeo.

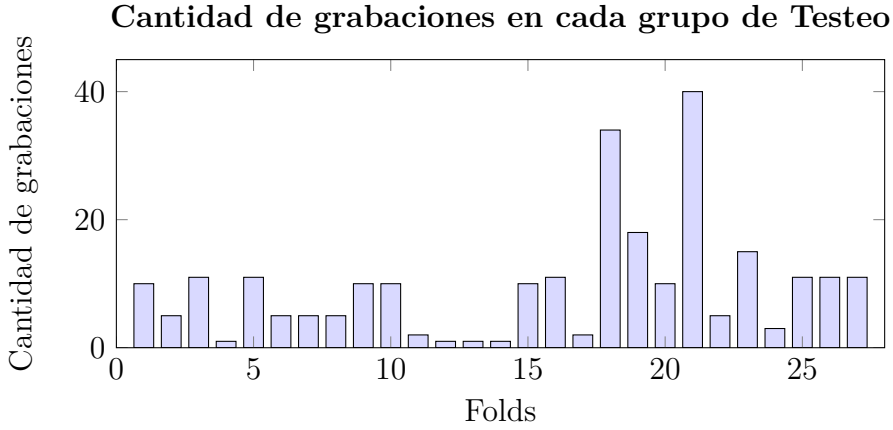


Figura 6.1: Cantidad de muestras para cada Test

Debemos tener una mejor métrica para equiparar la cantidad de instancias correctas en cada fold. Para evitar promediar por porcentaje correcto en cada grupo de test, realizamos el promedio sumando la cantidad de instancias correctas. Los resultados de cada clasificador utilizando esta nueva métrica se pueden ver en la tabla 6.4. La fórmula calculada para cada clasificador se puede ver a continuación:

Promedio de instancias correctas sobre instancias totales:

$$\frac{\sum_{i=1}^{\#\text{ de folds}} \text{Cantidad de instancias correctas en fold } i}{\#\text{ de instancias}}$$

	Zero Rule	**Ripper**	**C4.5**	**SVM**	**NaiveBayes**
Promedio	69	63	69	70	65

Tabla 6.4: Clasificación correcta en cantidad de instancias

Notamos que los resultados varían algunos puntos pero no a grandes rasgos. El clasificador que más varia es NaiveBayes con casi 5 % menos. A pesar de ello, no parece cambiar mucho el promedio de cada clasificador.

Análisis de los clasificadores construidos

Analizamos los clasificadores armados para cada fold. El clasificador Ripper generó reglas utilizando un conjunto acotado de atributos. En cambio, los clasificadores Support vector machine y NaiveBayes utilizaron todos los atributos ponderando cada uno por valores escalares. El clasificador C4.5 tuvo también una performance pobre por tener un árbol de un nivel solo. Podemos ver el conjunto de reglas que generó el clasificador Ripper para el fold 4:

- $(FON\_rr\_norm <= -6{,}901) and (ACU\_AverageRR\_7 <= 11{,}23) => place = cba(18{,}0/3{,}0)$

- $(FON\_ll\_norm <= -7{,}975) and (ACU\_AverageLL\_6 <= 4{,}308) => place = cba(15{,}0/0{,}0)$
- $else => place = bsas(222{,}0/49{,}0)$

Notamos que utilizan los atributos temporales y en menor medida los atributos acústicos.

A continuación, vemos el árbol de decisión generado por C4.5 para cada fold.

```
root
└── bsas(249,0/68,0)
```

Al armar un árbol de decisión de una sola rama y que siempre elige Buenos Aires, este clasificador elige de igual manera que ZeroRule. Por ende, ambos tienen igual performance.

Características del modelo de test

Analizando los distintos clasificadores, nos llama la atención que C4.5 obtuvo igual performance que ZeroRule. El clasificador C4.5 no saca provecho de los datos y termina creando un árbol de decisión eligiendo la clase mayoritaria.

Recordemos que el algoritmo C4.5 arma el árbol de decisión teniendo en cuenta la ganancia de información. Calcula esta métrica para cada atributo y el mayor de todos ellos se agrega como raíz del árbol. Luego descarta el atributo ya elegido y se llama recursivamente en cada rama.

A simple vista, el clasificador no tiene problemas al utilizar atributos indefinidos. Al calcular para cada atributo su ganancia de información estaría descartando estos atributos. Pensamos que los parámetros utilizados generan demasiado *prunning* en el árbol de decisión y termina quedando solamente una rama. Creemos que variando los parámetros se podría mejorar este árbol. Dejamos esta opción como trabajo futuro.

6.4.2. Clasificación por hablante

En este modelo de test variamos la instancia de clasificación. Para cada hablante juntamos sus grabaciones calculando su promedio para cada atributo. La idea es juntar los atributos de un mismo hablante en una sola instancia. De esta forma, evitamos los atributos desconocidos en cada una de las grabaciones y además podemos afirmar que las muestras son independientes ya que hay una muestra por hablante.

En la tabla 6.5 vemos un ejemplo de datos extraídos originalmente para entender mejor la idea.

Atributos		A1	A2	A3	...	AN
Hablante 1	**Audio1**	1	?	2		2
	Audio2	?	?	1	...	?
	Audio3	2	?	3		?
Hablante 2	**Audio1**	1	?	?	...	?
	Audio2	1	2	?		?

Tabla 6.5: Datos originales

Para cada hablante juntamos sus audios realizando el promedio de cada atributo. Por ejemplo: el Hablante 1 grabó 3 audios donde cada uno posee distintos atributos. Juntamos todos esos audios en uno promediando sus atributos: El Audio1 y Audio3 poseen el atributo A1 con valor 1 y 2 respectivamente. Entonces en la tabla 6.6 tendremos Audio1 con A1 definido como el promedio de estos valores: $(1 + 2) \div 2 = 1,5$. Ninguno de los audios grabados originalmente por Hablante 1 definió A2, entonces en este caso no definimos ningún valor y va a quedar como valor desconocido.

Atributos		A1	A2	A3	...	AN
Hablante 1	**Audio1**	**1.5**	**?**	**2**	...	**2**
Hablante 2	**Audio1**	**1**	**2**	**?**	...	**?**

Tabla 6.6: Atributos modificados

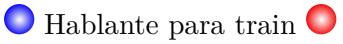
Esta variante la realizamos solamente utilizando los atributos temporales. Descartamos los atributos acústicos porque corresponden a grabaciones que en muchas oportunidades sufren de ruido y pensamos que también podía ser una causa por la cual los clasificadores tienen mal rendimiento.

Resumiendo, por cada hablante se define una fila de atributos. Esto minimiza la cantidad de valores desconocidos por cada hablante.

Variando la cantidad de hablantes

Además de esta modificación, realizaremos una variante más con respecto a la cantidad de hablantes de cada grupo. La primer variante será tener cantidad de hablantes aproximadamente pareja. La segunda variante será utilizando todo los hablantes.

En la primer variante tomaremos 9 hablantes de Buenos Aires y de 8 Córdoba. Nunca es bueno descartar datos pero debemos tener un conjunto de datos equilibrado para saber si utilizando los atributos que definidos podemos realizar una mejor clasificación que el baseline. En la tabla 6.7 vemos este esquema.



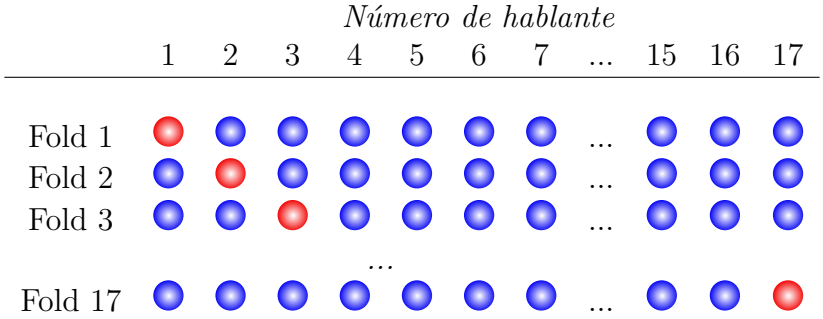


Tabla 6.7: Esquema de validación cruzada

Agregamos un hablante más de Buenos Aires ya que, de esta manera, entrenamos ZeroRule con instancias equilibradas. Con este esquema, cuando quitamos un hablante para testeo, el algoritmo ZeroRule entrenará la mitad de las veces con instancias equilibradas en el conjunto de entrenamiento. De esta forma, este clasificador clasificará correctamente la mitad de las veces.

Si utilizáramos igual cantidad de hablantes en ambos grupos, sucedería lo siguiente: al principio descartaríamos un hablante para testear. El clasificador ZeroRule entrenaría en el conjunto de hablantes y elegiría siempre la clase del hablante que no fue descartada para testeo, ya que es la clase mayoritaria. Cuando intente clasificar el hablante de testeo no dará correcto. Esto se repite en todos los folds haciendo que su performance sea 0 %.

Este cambio es menor y nos ayuda a que el análisis sea más correcto. Entonces el conjunto de datos posee sólo una unidad de diferencia entre ambos grupos. La elección de cada uno de estos hablantes fue al azar. Como el esquema anterior, tendremos un fold por cada hablante.

En la segunda variante no descartaremos datos. Realizaremos el mismo procedimiento teniendo en cuenta los 27 hablantes obtenidos.

Resultados

Los resultados de este cross-validaton podemos observarlos en la tabla 6.8 para la variante utilizando hablantes equilibrados, y en la tabla 6.9 para todos los hablantes. Para realizar el cálculo del promedio se realizó la suma del porcentaje de instancias correctas de cada fold sobre el total de folds. Al haber una instancia en cada grupo de test, el cálculo del promedio teniendo en cuenta las instancias correctas devuelve el mismo resultado que éste.

	Zero Rule	**Ripper**	**C4.5**	**SVM**	**NaiveBayes**
Promedio	53	53	76	94	76

Tabla 6.8: Clasificación correcta en porcentaje promediando los atributos por hablante de los 17 hablantes (utilizando 9 de Buenos Aires, 8 Córdoba)

	Zero Rule	**Ripper**	**C4.5**	**SVM**	**NaiveBayes**
Promedio	70	77	70	96	88

Tabla 6.9: Clasificación correcta en porcentaje promediando los atributos por hablante **sin descartar datos** (utilizando los 27 hablantes)

Podemos observar que el clasificador ZeroRule estuvo más cerca de un valor esperable para el tipo de clasificador que es. Todos los demás dieron por arriba de este valor. Se destacaron los dos clasificadores que utilizan todos los atributos ponderados: Support vector machine y NaiveBayes.

Los resultados sobre el p-valor de cada test estadístico de Wilcoxon y Test t de Student se puede ver en la tabla 6.10.

	Hablantes equilibrados		Todos los hablantes	
	Student Test	**Wilcoxon Test**	**Student Test**	**Wilcoxon Test**
ZeroR y Ripper	0.5	0.5229	0.2123	0.242
ZeroR y C4.5	0.08174	0.09083	0.5	0.5793
ZeroR y NaiveBayes	0.05186	0.06472	0.02855	0.0363
ZeroR y SVM	**0.002048**	**0.005367**	**0.002826**	**0.005367**

Tabla 6.10: Resultados de cada test representado en p-valor

Los clasificadores pasaron el test Shapiro-Wilk, entonces podemos afirmar que los resultados de cada clasificador corresponden a una distribución Normal y realizar el test t de Student.

Podemos observar que para el clasificador Support vector machine posee p-valor menor a 0,05 en ambas variantes. Esto quiere decir que para **Support vector machine hay evidencia suficiente de ser mejor que el baseline**. Por otro lado, los demás no pudieron lograr este cometido.

Análisis de los clasificadores construidos

Analizando los clasificadores notamos que el clasificador C4.5 armó árboles de decisión más elaborados. Notamos eso también viendo que su performance superó a Zero Rules.

Veamos un árbol de decisión generado por C4.5. Este corresponde al fold 7.

```
root
|__ FON_vowel_norm <= 7,221824
|   |__ FON_ll_norm <= −24,007 : cba(2,33/0,22)
|   |__ FON_ll_norm > −24,007 : bsas(18,67/0,89)
|__ FON_vowel_norm > 7,221824 : cba(5,0)
```

Los árboles de decisión generados esta vez utilizan mucho atributos como FON_ll_norm y FON_vowel_norm en varios folds. Los demás clasificadores también armaron sus reglas: SVM y NaiveBayes ponderaron cada atributo para su clasificación mientras que Ripper armó sus reglas parecidas a C4.5.

Características del modelo de test

Este modelo de test dio muy buenos resultados. El clasificador ZeroR tuvo una performance de alrededor del 50 % mientras que el clasificador C4.5 pudo armar árboles mejores.

Sin embargo, la forma que evitamos los valores desconocidos no es la mejor. El resultado de un fold para un determinado clasificador es 0 % o 100 %. Esto sucede porque es sólo una instancia la que representa.

Esto también se ve reflejado en las matrices de confusión de cada fold. Podemos ver en la tabla 6.11 la matriz de confusión del clasificador Ripper para el fold 1. En cada una de ellas, para cualquier clasificador, siempre se encuentra una sola instancia.

Buenos Aires	Córdoba	
1	0	Buenos Aires
0	0	Córdoba

Tabla 6.11: Matriz de confusión fold 1

6.5. Selección de atributos de forma automática

En esta sección aplicaremos evaluadores a los distintos atributos para analizar cuáles poseen mayor importancia. Para este análisis, utilizamos todo el conjunto de datos sin descartar ningún atributo. El evaluador utilizado fue la ganancia de información (InfoGain en Weka) para analizar la importancia de cada atributo. Utilizamos la opción Ranker que nos ordena los atributos de mayor a menor ganancia.

Este algoritmo trabaja de la siguiente forma: para cada atributo calcula la entropía de la clase y luego se calcula la entropía de la misma sabiendo el valor de este atributo. La ganancia de información de ese atributo es la resta de esos dos resultados. Esto se puede expresar como: $InfoGain(Class, Attribute) = H(Class) - H(Class|Attribute)$. Veamos cada uno de estos términos.

- $H(Class)$ representa el valor de la entropía de la clase a predecir. En otras palabras, mide la incertidumbre asociada a la clase sin tener en cuenta el valor de ningún atributo en particular. Recordemos que cuando decimos clase nos referimos a Buenos Aires o Córdoba.
- $H(Class|Attribute)$ representa el valor de la entropía de la clase sabiendo el valor del atributo *Attribute*. A esta se le llama *entropía condicional*. Si este atributo tiene información que ayude a predecir la clase, entonces la entropía condicional será menor a la entropía de la clase. O sea, $H(Class|Attribute) < H(Class)$. Cuanto menor sea la entropía condicional con respecto a la entropía de la clase, mayor será la ganancia de información para ese atributo.

La tabla 6.12 nos muestra que los más preponderantes se refieren a la duración de consonantes (utilizando nuestra nomenclatura FON_consonant_norm), vocales(FON_vowel _norm), duración de la sílaba acentuada (SIL_syllableAccent_normhd) y su sílaba anterior (SIL_prevSyllableAccent_normhd).

Ganancia de Información	**Atributo**
0.07231	FON_consonant_norm
0.07217	FON_vowel_norm
0.03963	SIL_syllableAccent_normhd
0.03963	SIL_prevSyllableAccent_normhd
0.02332	FON_ll_norm
0.02285	FON_Sfinal_norm
0.02226	ACU_MinLL_1
0.02144	ACU_AverageLL_1

Tabla 6.12: Resultados de InfoGain

Veamos si los valores de ganancia de información fueron altos. El valor mínimo de ganancia de información es 0, eso sucede cuando $H(Class) = H(Class|Attribute)$ ya que la entropía nunca puede ser menor a 0.

La entropía de la clase (o sea, $H(Class)$) es máxima cuando todas las clases son equiprobables. En este caso, $H(Class) \leq \log_b(n)$ donde $b = 2$ ya que esta calculada en bits y n cantidad de clases. Entonces $H(Class) \leq \log_2(2) = 1$. En nuestro caso, no todas las clases son equiprobables, así que $H(Class)$ puede ser menor a 1.

Viendo los valores de la tabla 6.12 notamos que el atributo con mayor ganancia de información posee un valor bastante alejado de 1 para nuestro conjunto de datos. Es por eso que notamos que un atributo sólo no es suficiente para una buena clasificación. En cambio, la conjunción de varios con buena ganancia de información realizan un buen trabajo.

El atributo sobre la duración de la sílaba y su anterior es entendible que aporten la mayor ganancia de información, ya que es la característica más conocida para distinguir los dos grupos. Son las primeras características que uno piensa al definir el habla de un cordobés. No es extraño encontrarlos entre los primeros lugares.

Los atributos sobre duración de consonantes y vocales sorprenden con sus valores pero luego de analizarlos son entendibles. Todas las reglas definidas, salvo la regla 1 sobre estirar la sílaba anterior a la acentuada, están definidas sobre consonantes y vocales. Esto quiere decir que todas las frases aportan a estos dos atributos. Es por esto que posee mayor ganancia de información.

En las tablas 6.13, 6.14 y 6.15 podemos observar el resultado de correr el algoritmo sólo para los tipos de atributos definidos. La ganancia de información no cambia ya que el cálculo es independiente del grupo de atributos que se le aplica. A pesar de esto, podemos notar cuáles atributos, según cada tipo, aportan mayor ganancia.

GI	Atributo
0.07231	FON_consonant_norm
0.07217	FON_vowel_norm
0.02332	FON_ll_norm
0.02285	FON_Sfinal_norm
0.01851	FON_Sfinal_normhd

Tabla 6.13: InfoGain para atributos fonéticos

GI	Atributo
0.02226	ACU_MinLL_1
0.02144	ACU_AverageLL_1
0.01438	ACU_MaxLL_5
0.01232	ACU_MaxKT_15
0.01219	ACU_MaxLL_6

Tabla 6.14: InfoGain para atributos acústicos

GI	Atributo
0.03963	SIL_syllableAccent_normhd
0.03963	SIL_prevSyllableAccent_normhd
0	SIL_prevSyllableAccent_norm
0	SIL_syllableAccent_norm

Tabla 6.15: InfoGain para atributos silábicos

La ganancia de información sobre los atributos fonéticos (tabla 6.13) nos muestra que el fonema *'ll'* y *'s'* para terminar una palabra aportan mayor cantidad de información. Sobre los atributos silábicos (tabla 6.15) notamos que los atributos calculados suponiendo $\mu = 0$ (o sea, con el prefijo *'normhd'*) nos aportaron ganancia de información, mientras que los atributos sin utilizar esta característica no aportaron. Sobre los atributos acústicos (tabla 6.14) notamos que el fonema *'ll'* aporta mayor ganancia de información utilizando varios componentes de MFCC. Le sigue el fonema *'kt'*.

Realizado estos análisis podemos concluir que los atributos silábicos sobre la silaba acentuada, anterior a la acentuada, la duración del fonema *'ll'* y *'s'* para terminar una palabra y los atributos acústicos sobre el fonema *'ll'* fueron, para nuestro conjunto de datos, los que mayor información aportaron para la clasificación.

Capítulo 7

Conclusiones y Trabajo Futuro

En este trabajo pudimos desarrollar una herramienta para el estudio de las variantes argentinas del español. Diseñamos desde cero una página web que nos permitió llevar a cabo experimentos del habla. Pudimos analizar las diferencias de dos variantes importantes de la Argentina: una radicada en Buenos Aires y la otra en Córdoba.

Al analizar las grabaciones recolectadas, nos enfrentamos a distintos problemas para poder utilizar esos datos. La elección de atributos a tener en cuenta y la precisa alineación de los mismos para cada grabación fueron actividades importantes que se reflejaron en los datos posteriores. Fue muy interesante desarrollar todos los pasos, desde la obtención de las grabaciones hasta el análisis de clasificación, para ver cómo influyen en el resultado final.

Una vez obtenidos los datos y extraídos los atributos, analizamos cómo testear debidamente los clasificadores entrenados. Pudimos definir un clasificador baseline que logramos superar proponiendo diferentes clasificadores. Encontramos que *Function SMO* tuvo mejor performance ya que es un clasificador que busca el hiperplano de margen máximo entre los dos grupos, es decir, con esta división generaliza mejor que los otros clasificadores probados. También analizamos qué atributos tienen más información a la hora de clasificar en cada grupo y comprobamos que estos son los mismos que uno intuye.

Realizando este trabajo aprendimos la dificultad que surge al tener un dataset desequilibrado. Esto fue un inconveniente a superar y se notó al probar los modelos de testing. Otra lección que aprendimos es que, si bien los atributos elegidos fueron correctos, creemos que se podrían mejorar los atributos acústicos; teniendo grabaciones más limpias y sin ruido, la calidad de estos sería mejor y por ende mejor la información que aportan.

A partir de este trabajo pueden surgir muchas mejoras que enumeraremos a continuación:

Chequeador cruzado: Una mejora, por parte del análisis de datos, podría ser que las grabaciones sean chequeadas entre los hablantes. Un hablante, intercalado con las grabaciones que realiza, podría escuchar un audio y su frase asociada de otro hablante que previamente realizó el experimento. Luego de escucharla, debería decidir si en la grabación se escucha correctamente la frase en cuestión. Si es así, se podría decidir que esa grabación es buena para el extractor y que se mantiene como conservada.

Si se logra que cada hablante pueda chequear que otra frase se dijo correctamente, permitiría que no sea necesario por parte de los administradores del sistema realizar este trabajo. De esta forma, se simplificaría mejor la recolección de grabaciones y su filtrado si están mal grabados.

Validación de calidad de sonido: En el momento de grabación, se podría analizar el audio grabado y rechazarlo si no supera un nivel aceptable auditivo. Esto puede implementarse de varias formas. Una posibilidad sería cuando se está grabando medir el volumen del micrófono cada cierta cantidad de tiempo (por ejemplo: 1 segundo). Si en esa medición el volumen no se encuentra entre un rango máximo y mínimo de volumen, descartar la grabación y pedirle al hablante que vuelva a grabar.

También se le podría dar más información al hablante. Sabiendo que el micrófono tuvo un pico de volumen, se podría pedir al hablante que baje el nivel de voz o se aleje del micrófono. Ídem si habla muy bajo. Otras posibles soluciones a este problema son analizar antes de empezar el experimento. Si el ruido ambiente genera saturación, pedir al usuario que realice el experimento en un lugar más silencioso.

Podemos realizar análisis más precisos sobre la calidad del audio cuando llega la grabación al servidor. En ese momento, el servidor ya puede obtener el archivo wav y realizarle todo tipo de análisis (por ejemplo: detección de ruido ambiente). Recordemos que el servidor esta implementado en Python, que posee muchas bibliotecas útiles para realizar esto. Al momento de terminar de procesar el audio en cuestión, deberá enviar la respuesta al hablante informándole si se debe realizar de vuelta la grabación o si fue exitosa. Es importante notar que esta solución necesita buena conexión al servidor.

Puntaje en alineaciones: En el informe analizamos manualmente si cada grabación fue alineada correctamente. Esto se pudo lograr gracias a que eran pocos audios. En un sistema automático que recolecte grabaciones y además extraiga cada atributo para luego entrenar los clasificadores, esta tarea no se podría realizar.

Una vez terminada una alineación, ProsodyLab-Aligner genera un archivo donde muestra cómo fueron esas alineaciones. Este archivo se llama *'.SCORES'* y en él se encuentra una lista de todos los audios seguidos de un puntaje. Si una alineación fue similar a otra va a tener aproximadamente un valor similar. En cambio, si posee una alineación muy distinta va a tener valores distintos. Este puede ser un buen filtro para saber si se pudo alinear bien.

Se podría definir un umbral para filtrar cuándo se realizó una alineación correcta. Si esta alineación no supera ese umbral, se debería descartar ese audio. Una cuestión importante a tener en cuenta es la cantidad de falsos positivos que pueden surgir. O sea, la cantidad de audios que no pasan el umbral pero están bien alineados.

Clasificación en vivo: Se podría realizar una prueba online de clasificación de hablantes. La misma consiste en realizar el mismo experimento para un hablante y que al final le devuelva como resultado si pertenece a Buenos Aires o Córdoba.

Esta clasificación en vivo necesitaría de los trabajos futuros relacionados con automatización del sistema. Cada vez que se realice el experimento para un hablante nuevo, se

deberá recalcular el clasificador. De esta forma, se le puede dar una respuesta sobre a qué grupo pertenece.

Mejores modelos y atributos: Se podría analizar si existen más y mejores atributos que distingan estos dos grupos. Por ejemplo: se podría analizar si la sílaba anterior a la sílaba acentuada se sigue estirando cuando éstas están en palabras distintas.

También se pueden mejorar los modelos de test incluyendo variantes que analicen de distinta forma los atributos desconocidos. Otro análisis interesante sería cómo mejorar el rendimiento de cada clasificador variando sus parámetros.

Apéndice: marcas prosódicas

En la tabla 7.1 podemos ver las marcas prosódicas de cada palabra en cada frase. La separación se realiza en sílabas y el asterisco marca donde ocurre el acento.

Frase	**Prosodia**
'No hay dos sin tres'	[['no'], ['aj'], ['dos*'], ['sin'], ['tres*']]
'Más difícil que encontrar una aguja en un pajar'	[['mas'], ['di', 'fi*', 'sil'], ['ke'], ['eN', 'kon', 'trar*'], ['una'], ['a', 'Gu*', 'xa'], ['en'], ['un'], ['pa', 'xar*']]
'Más perdido que turco en la neblina'	[['mas'], ['per', 'Di*', 'Do'], ['ke'], ['tur*', 'ko'], ['en'], ['la'], ['ne', 'Bli*', 'na']]
'No le busques la quinta pata al gato'	[['no'], ['le'], ['buh*', 'kes'], ['la'], ['kin*', 'ta'], ['pa*', 'ta'], ['al'], ['ga*', 'to']]
'Se te escapó la tortuga'	[['se'], ['te'], ['eh', 'ka', 'po*'], ['la'], ['tor', 'tu*', 'Ga']]
'Todos los caminos conducen a Roma'	[['to', 'Dos'], ['los'], ['ka', 'mi*', 'nos'], ['kon', 'du*', 'cen'], ['a'], ['Ro*', 'ma']]
'No hay mal que dure cien años'	[['no'], ['aj'], ['mal*'], ['ke'], ['du*', 're'], ['cien*'], ['a*', 'nos']]
'Siempre que llovió, paro'	[['sjem*', 'pre'], ['ke'], ['Zo', 'Bjo*'], ['pa', 'ro*']]
'Cría cuervos que te sacaran los ojos'	[['krj*', 'a'], ['kwer*', 'Bos'], ['ke'], ['te'], ['sa', 'ka', 'ran*'], ['los'], ['o*', 'xos']]
'La tercera es la vencida'	[['la'], ['ter', 'se*', 'ra'], ['es'], ['la'], ['ben', 'si*', 'Da']]
'Calavera no chilla'	[['ka', 'la', 'Be*', 'ra'], ['no'], ['Hi*', 'Za']]
'La gota que rebalsó el vaso'	[['la'], ['go*', 'ta'], ['ke'], ['Re', 'Bal', 'so*'], ['el'], ['ba*', 'so']]
'La suegra y el doctor cuanto más lejos mejor'	[['la'], ['swe*', 'Gra'], ['y'], ['el'], ['dok', 'tor*'], ['kwan', 'to'], ['mas'], ['le*', 'xos'], ['me', 'xor*']]
'A la mujer picaresca cualquiera la pesca'	[['a'], ['la'], ['mu', 'Cer*'], ['pi','ka', 'reh*','ka'], ['kwal', 'kje*', 'ra'], ['la'], ['peh*', 'ka']]
'Quien siembra vientos recoge tempestades'	[['kj*', 'en'], ['sjem*', 'bra'], ['bjen*', 'tos'], ['Re', 'ko*', 'Ce'], ['tem', 'peh', 'ta*', 'Des']]
'La arquitectura es el arte de organizar el espacio'	[['la'], ['ar', 'ki', 'tek', 'tu*','ra'], ['es'], ['el'], ['ar*', 'te'], ['de'], ['or', 'Ga', 'ni', 'sar*'], ['el'], ['eh', 'pa*', 'sjo']]
'El amor actúa con el corazón y no con la cabeza'	[['el'], ['a', 'mor*'], ['ak', 'tw*', 'a'], ['kon'], ['el'], ['ko', 'ra', 'son*'], ['i'], ['no'], ['kon'], ['la'], ['ka','Be*', 'sa']]
'No dudes, actúa'	[['no'], ['du*', 'Des'], ['ak', 'tw*', 'a']]
'Perro que ladra no muerde'	[['pe*', 'Ro'], ['ke'], ['la*', 'Dra'], ['no'], ['mwer*', 'De']]
'La musica es sinónimo de libertad, de tocar lo que quieras y cómo quieras'	[['la'], ['mu*','si','ka'], ['es'], ['si', 'no*', 'ni', 'mo'], ['de'], ['li','Ber','taD*'], ['de'], ['to', 'kar*'], ['lo'], ['ke'], ['kje*','ras'], ['i'], ['ko','mo'], ['kje*','ras']]
'La belleza que atrae rara vez coincide con la belleza que enamora'	[['la'], ['be','Ze*','sa'], ['ke'], ['a', 'tra*', 'e'], ['Ra*', 'ra'], ['Bes*'], ['kojn', 'si*', 'De'], ['kon'], ['la'], ['be','Ze*','sa'], ['ke'], ['e','na','mo*','ra']]

'No esta mal ser bella lo que está mal es la obligación de serlo'	[['no'], ['eh', 'ta*'], ['mal*'], ['ser'], ['be*', 'Za'], ['lo'], ['ke'], ['eh', 'ta*'], ['mal*'], ['es'], ['la'], ['o', 'Bli','Ga', 'sjon*'], ['de'], ['ser*', 'lo']]
'La batalla más difícil la tengo todos los días conmigo mismo'	[['la'], ['ba', 'ta*', 'Za'], ['mas'], ['di','fi*','sil'], ['la'], ['teN*', 'go'], ['to','Dos'], ['los'], ['dj*', 'as'], ['kon', 'mi*', 'Go'], ['mis*', 'mo']]
'El que no llora, no mama'	[['el'], ['ke'], ['no'], ['Zo*', 'ra'], ['no'], ['ma*', 'ma']]
'En la pelea se conoce al soldado, sólo en la victoria se conoce al caballero'	[['en'], ['la'], ['pe', 'le*', 'a'], ['se'], ['ko', 'no*', 'se'], ['al'], ['sol', 'da*', 'Do'], ['so*', 'lo'], ['en'], ['la'], ['bik', 'to*', 'rja'], ['se'], ['ko', 'no*', 'se'], ['al'], ['ka', 'Ba', 'Ze*', 'ro']]
'La lectura es a la mente lo que el ejercicio al cuerpo'	[['la'], ['lek', 'tu*', 'ra'], ['es'], ['a'], ['la'], ['men*', 'te'], ['lo'], ['ke'], ['el'], ['e', 'Cer', 'si*', 'sjo'], ['al'], ['kwer*', 'po']]
'El pez por la boca muere'	[['el'], ['pes*'], ['por'], ['la'], ['bo*', 'ka'], ['mwe*', 're']]
'El canapé salió espectacular'	[['el'], ['ka', 'na', 'pe*'], ['sa', 'ljo*'], ['eh', 'pek', 'ta', 'ku', 'lar*']]
'El canapé salió delicioso'	[['el'], ['ka', 'na', 'pe*'], ['sa', 'ljo*'], ['de', 'li', 'sjo*', 'so']]
'El canapé salió riquísimo'	[['el'], ['ka', 'na', 'pe*'], ['sa', 'ljo*'], ['Ri', 'ki*', 'si', 'mo']]
'El repollo salió espectacular'	[['el'], ['Re', 'po*', 'Zo'], ['sa', 'ljo*'], ['eh', 'pek', 'ta', 'ku', 'lar*']]
'El repollo salió delicioso'	[['el'], ['Re', 'po*', 'Zo'], ['sa', 'ljo*'], ['de', 'li', 'sjo*', 'so']]
'El repollo salió riquísimo'	[['el'], ['Re', 'po*', 'Zo'], ['sa', 'ljo*'], ['Ri', 'ki*', 'si', 'mo']]
'El esparrago salió espectacular'	[['el'], ['eh', 'pa*', 'Ra', 'Go'], ['sa', 'ljo*'], ['eh', 'pek', 'ta', 'ku', 'lar*']]
'El esparrago salió delicioso'	[['el'], ['eh', 'pa*', 'Ra', 'Go'], ['sa', 'ljo*'], ['de', 'li', 'sjo*', 'so']]
'El esparrago salió riquísimo'	[['el'], ['eh', 'pa*', 'Ra', 'Go'], ['sa', 'ljo*'], ['Ri', 'ki*', 'si', 'mo']]
'En boca cerrada no entran moscas'	[['en'], ['bo*', 'ka'], ['se', 'Ra*', 'Da'], ['no'], ['en*', 'tran'], ['moh*', 'kas']]
'Más vale pájaro en mano que cien volando'	[['mas'], ['ba*', 'le'], ['pa*', 'xa', 'ro'], ['en'], ['ma*', 'no'], ['ke'], ['sjen*'], ['bo', 'lan*', 'do']]
'Río revuelto ganancia de pescadores'	[['Rj*', 'o'], ['Re', 'Bwel*', 'to'], ['ga', 'nan*', 'sja'], ['de'], ['peh', 'ka', 'Do*', 'res']]
'No hay qué pedirle peras al olmo'	[['no'], ['aj'], ['ke'], ['pe', 'Dir*', 'le'], ['pe*', 'ras'], ['al'], ['ol*','mo']]

Tabla 7.1: Marcas prosódicas

Apéndice: nomenclatura de atributos

En la tabla 7.2 se puede ver la nomenclatura de los atributos elegidos.

Nomenclatura	**Referencia**
ACU_AverageKT_0 al ACU_AverageKT_38	Componentes promedio de MFCC del fonema /k/ anterior a /t/
ACU_AverageLL_0 al ACU_AverageLL_38	Componentes promedio de MFCC del fonema /ll/
ACU_AverageRR_0 al ACU_AverageRR_38	Componentes promedio de MFCC del fonema /r/ fuerte
ACU_AverageSC_0 al ACU_AverageSC_38	Componentes promedio de MFCC del fonema /s/ anterior a /c/
ACU_MaxKT_0 al ACU_MaxKT_38	Componentes máximo de MFCC del fonema /k/ anterior a /t/
ACU_MaxLL_0 al ACU_MaxLL_38	Componentes máximo de MFCC del fonema /ll/
ACU_MaxRR_0 al ACU_MaxRR_38	Componentes máximo de MFCC del fonema /r/ fuerte
ACU_MaxSC_0 al ACU_MaxSC_38	Componentes máximo de MFCC del fonema /s/ anterior a /c/
ACU_MinKT_0 al ACU_MinKT_38	Componentes mínimo de MFCC del fonema /k/ anterior a /t/
ACU_MinLL_0 al ACU_MinLL_38	Componentes mínimo de MFCC del fonema /ll/
ACU_MinRR_0 al ACU_MinRR_38	Componentes mínimo de MFCC del fonema /r/ fuerte
ACU_MinSC_0 al ACU_MinSC_38	Componentes mínimo de MFCC del fonema /s/ anterior a /c/
FON_phoneme	Duración del fonema
place	Lugar del hablante en la grabación

Tabla 7.2: Atributos normalizados

En la tabla 7.3 se muestra la nomenclatura de los atributos que se les aplicó normalización.

Nomenclatura	**Referencia**
FON_vowel_norm	Duración de las vocales
FON_consonant_norm	Duración de la consonantes
FON_Sfinal_norm	Duración de la /s/ final de palabra
FON_kt_norm	Duración del fonema /k/ anterior a /t/

FON_ll_norm	Duración de la /ll/
FON_rr_norm	Duración del fonema /r/ fuerte
FON_sc_norm	Duración del fonema /s/ anterior a /c/
SIL_prevSyllableAccent_norm	Duración de la silaba anterior a la acentuada aplicando normalización
SIL_syllableAccent_norm	Duración de la silaba acentuada aplicando normalización

Tabla 7.3: Atributos normalizados

En la tabla 7.4 se muestra la nomenclatura de los atributos que se les aplicó normalización tomando como $mu = 0$.

Nomenclatura	**Referencia**
FON_vowel_normhd	Duración de las vocales
FON_consonant_normhd	Duración de la consonantes
FON_Sfinal_normhd	Duración de la /s/ final de palabra
FON_kt_normhd	Duración del fonema /k/ anterior a /t/
FON_ll_normhd	Duración de la /ll/
FON_rr_normhd	Duración del fonema /r/ fuerte
FON_sc_normhd	Duración del fonema /s/ anterior a /c/
SIL_prevSyllableAccent_normhd	Duración de la silaba anterior a la acentuada aplicando normalización
SIL_syllableAccent_normhd	Duración de la silaba acentuada aplicando normalización

Tabla 7.4: Atributos normalizados

Bibliografía

[Battini, 1964] Battini, E. V. (1964). Español en la Argentina.

[Boersma and Weenink, 2013] Boersma, P. and Weenink, D. (2013). Praat: doing phonetics by computer [Computer program] Version 5.3.51, retrieved 2 June 2013 from http://www.praat.org/.

[Cohen, 1995] Cohen, W. W. (1995). Fast Effective Rule Induction. In *Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann.

[Fontanella, 2000] Fontanella, M. B. (2000). El español en la Argentina y sus variedades regionales.

[Gorman et al., 2011] Gorman, K., Howell, J., and Wagner, M. (2011). Prosodylab-Aligner: A Tool for Forced Alignment of Laboratory Speech. Canadian Acoustics. 39.3. 192–193. In *Proceedings of Acoustics Week in Canada, Quebec City.*

[Gurlekian et al., 2009] Gurlekian, J. A., Yanagida, R., Trípodi, M. N., and Toledo, G. (2009). Amper-Argentina: variabilidad rítmica en dos corpus.

[Platt, 1998] Platt, J. C. (1998). Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical Report MSR-TR-98-14, Microsoft Research.

[Quinlan, 1993] Quinlan, R. (1993). C4.5: Programs for Machine Learning.

[Rabiner, 1989] Rabiner, L. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Van Oosterzee et al., 2004] Van Oosterzee, C., Planas, A. M. F., Barrios, L. R., i Sabaté, J. C., Montserrat, J. E., and Celdrán, E. M. (2004). Proyecto AMPER: estudio contrastivo de frases interrogativas sin expansión del "tortosí" y del "lleidatá". In *Actas del VI Congreso de Lingüística General, Santiago de Compostela*, pages 1977–1990. Arco Libros.

[Zhang, 2004] Zhang, H. (2004). The Optimality of Naive Bayes. In *FLAIRS Conference*, pages 562–567.