FACULTAD DE CIENCIAS EXACTAS, INGENIERÍA Y AGRIMENSURA

Aprendiendo a pronunciar texto en Inglés

Jeremías Rodríguez

July 18, 2019

1 INTRODUCCIÓN

La pronunciación del lenguaje inglés ha sido estudiada extensivamente por linguistas, y mucho es sabido sobre las correspondencias entre letras y los sonidos del habla inglesa, llamados *fonemas*. El inglés es un lenguaje particularmente difícil de dominar por sus irregularidades en la pronunciación. Por ejemplo, la letra "a" en la palabra "gave" es una vocal larga, pero no en "have" o "read".

En este trabajo se busca aplicar varios de los contenidos estudiados en este curso (y el anterior) al problema de convertir texto en inglés a su correspondiente transcripción en símbolos fonéticos[1].

single vowels				diphthongs		
ɪ	iː	ʊ	uː	eɪ	ɔɪ	aɪ
ship	sheep	book	shoot	wait	coin	like
e	ɜː	ə	ɔː	eə	ɪə	ʊə
left	her	teacher	door	hair	here	tourist
æ	ʌ	ɒ	ɑː	əʊ	aʊ	/
hat	up	on	far	show	mouth	

unvoiced consonants							
p	f	θ	t	s	ʃ	ʧ	k
pea	free	thing	tree	see	sheep	cheese	coin
voiced consonants							
b	v	ð	d	z	ʒ	ʤ	g
boat	video	this	dog	zoo	television	joke	go
m	n	ŋ	h	w	l	r	j
mouse	now	thing	hope	we	love	run	you

Figure 1.1: Los 44 sonidos (fonemas) del inglés oral.

[1] El Alfabeto Fonético Internacional es un sistema de notación fonética creado por lingüistas. Su propósito es otorgar, de forma regularizada, precisa y única, la representación de los sonidos de cualquier lenguaje oral. Tiene aproximadamente 107 símbolos básicos, y cada idioma hace uso de un subconjunto particular.

El dataset utilizado se llama netTalk[2], usado por primera vez en el paper "Parallel Networks that Learn to Pronounce English Text"[1]. Los autores lograron entrenar, a partir de unos pocos ejemplos, redes neuronales que convierten strings de texto en inglés (e.g. *calculator*) en strings de fonemas ($kælkjωleIt@$) con muchísima precisión.

En este trabajo recrearé y analizaré varios experimentos llevados a cabo en el citado paper; y posteriormente extenderé la investigación utilizando herramientas estudiadas en el curso.

2 DATASET NETTALK

2.1 PRESENTACIÓN DEL DATASET

El dataset netTalk contiene una fila por cada una de las 20000 palabras del *Miriam Webster's Pocket Dictionary*. Por cada fila, el dataset contiene sólo dos columnas:

[written_representation]	[phonemic_representation]
argue	argY-
argumentation	argYmxnteS-xn
argumentative	argYmEntxtIv-
aright	xrA--t
arise	xrAz-
aristocracy	@rxstakrxsi
aristocrat	xrIstxkr@t
aristocratic	xrIstxkr@tIk
arithmetic	xrIT-mxtIk
arithmetical	@rIT-mEtIk-L
arithmetician	xrIT-mxtIS-xn

Ambas columnas son de tipo string, la primera columna sólo contiene caracteres del alfabeto inglés (abc..z), mientras que la segunda columna contiene caracteres que representan fonemas. El alfabeto elegido por los autores del paper original cuenta con 50 fonemas, incluyendo algunos fonemas extras utilizados en palabras extranjeras. Los 50 símbolos (fonemas) junto con ejemplos de palabras que utilizan cada uno de estos sonidos puede consultarse en netTalk.names (adjuntado).

En el extracto del dataset que he mostrado podemos ver que se ha forzado una correspondencia 1-1 entre letras de una palabra y fonemas de su transcripción: cada fonema corresponde al sonido producido por una de las letras de la palabra inglesa. Por ejemplo, la palabra "arithmetical":

```
a>@ ; r>r ; i>I ; t>T ; h>- ; m>m ; e>E ; t>t; i>I ; c>k ; a>- ; l>l
```

Esta correspondencia es forzada, pues la cantidad de fonemas usados para pronunciar una palabra podria ser menor a la cantidad de letras de la palabra. Los autores del paper original incluyen un fonema "vacio", representado con el símbolo '-', para indicar la ausencia de sonido. Utilizando este simbolo, se obliga a que la transcripcion fonetica tenga la misma cantidad de caracteres que la palabra original, lo cual es computacionalmente conveniente.

2.2 PREPROCESAMIENTO

Todos los métodos que hemos estudiado en Machine Learning y Data Mining requieren que los datos input estén expresados como una matriz de números reales. El dataset netTalk que se acaba de introducir es una representación a muy alto nivel de la información que queremos utilizar, y por ello los autores del paper

[2] `https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Nettalk+Corpus)`

original realizaron una codificación y transformación bastante compleja para obtener datos numéricos con los que trabajar.

Comencemos por definir claramente el problema a tratar. Deseamos hallar una función f que convierta strings con palabras en inglés, a strings con su correspondiente transcripción fonética:

$$f : Words \rightarrow Phonetics$$
$$f(\text{"}hypotenuse\text{"}) = \text{"}hApat - NYs - \text{".}$$

Utilizando el hecho de que cada letra del input se corresponde a un fonema en el output, podemos reducir el problema a hallar el fonema correspondiente de cada letra. Es decir, deseamos obtener una función g que se focalice en predecir el fonema de una sóla de las letras a la vez, obviamente teniendo en cuenta el entorno de ella. Asumimos que para predecir el sonido de una letra, sólo necesitamos saber las 3 letras anteriores y las 3 siguientes (si las hay):

$$g : L \times L \times L \times L \times L \times L \times L \rightarrow Phonems$$
$$g(-,-,-,h,y,p,o) = h$$
$$g(-,-,h,y,p,o,t) = A$$
$$g(-,h,y,p,o,t,e) = p$$
$$g(h,y,p,o,t,e,n) = a$$
$$g(y,p,o,t,e,n,u) = t$$
$$g(p,o,t,e,n,u,s) = -$$
$$g(o,t,e,n,u,s,e) = N$$
$$g(t,e,n,u,s,e,-) = Y$$
$$g(e,n,u,s,e,-,-) = s$$

Por supuesto, esta ventana de 7 letras puede no ser suficiente para algunas palabras donde se necesita considerar una porción más amplia para generar una correcta pronunciación; pero en general funciona bien. Finalmente, si obtenemos esta función g, podemos derivar f aplicando sucesivamente g a cada letra del input de f.

$$f(\text{"}hypotenuse\text{"}) = g(\text{"}---hypo\text{"})g(\text{"}--hypot\text{"})g(\text{"}-hypote\text{"})g(\text{"}hypoten\text{"})\ldots$$

Qué ventajas tiene g respecto a f? Con un input fijo de 7 caracteres y un output fijo de 1 fonema; se puede entrenar un algoritmo de machine learning dándole muchos ejemplos de 7 caracteres y un fonema output correspondiente al caracter central. Aprender la función g es sustancialmente más fácil, porque sólo se predice un caracter a la vez.

Por lo tanto, se convirtió el dataset netTalk a un nuevo dataset de 8 columas, donde dividimos cada palabra en varias filas como se ilustró arriba con "hypotenuse". En nuestro caso, de tener 10000 palabras pasamos a tener 146934 filas. Los scripts utilizados en esta fase pueden verse en /src/preprocessing.

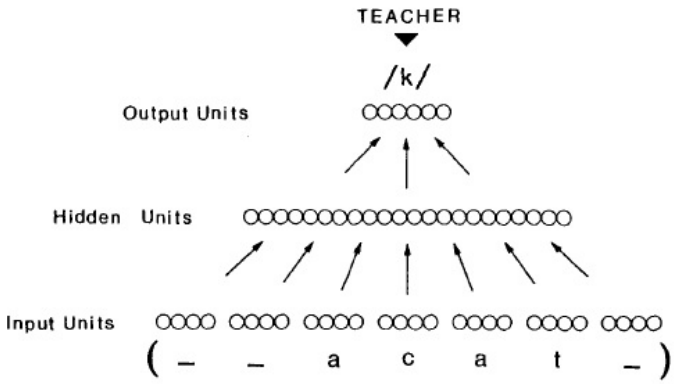
El paso restante es codificar todos los caracteres de las 146934 filas como números. Para ello usaremos una representación muy simple: Dado que tenemos 26 letras en el alfabeto inglés, reemplazaremos la columna de cada letra por 26 columnas binarias 1-0. Si la primera letra era una 'j', entonces la columna 10 tendrá un 1 y las otras 25 columnas un cero. Los fonemas se codificarán de forma similar, en 20 columnas binarias.[3]

[3] Los 50 fonemas se codifican en 20 columnas que representan "articulatory features" que los definen, como por ejemplo si tienen un tono alto, medio o bajo; si el ruido procede de la garganta, del paladar, etc; o si el ruido es con o sin voz.

Por lo tanto, nuestro dataset de 146934 filas tendrá 26 * 7 columnas binarias representando las 7 letras input; y otras 20 columnas binarias representando el fonema output. A este dataset resultante le aplicaré las distintas técnicas de aprendizaje y clustering.

3 Redes Neuronales

El primer paso es repetir los exprimentos que se realizaron usando redes neuronales en [1]. Para ello, utilizaré una red neuronal con 26 * 7 neuronas en la capa de entrada, y 27 en la capa de salida. Repetí los experimentos con distintas cantidades de neuronas en la capa intermedia, para estudiar como cambia la precisión de la red.



3.1 Optimización de parámetros

En un primer experimento dividí aleatoriamente las 20000 palabras originales en 1000 palabras para training/validación y 19000 para testear. Ambos conjuntos fueron preprocesados como se indicó en la sección anterior.

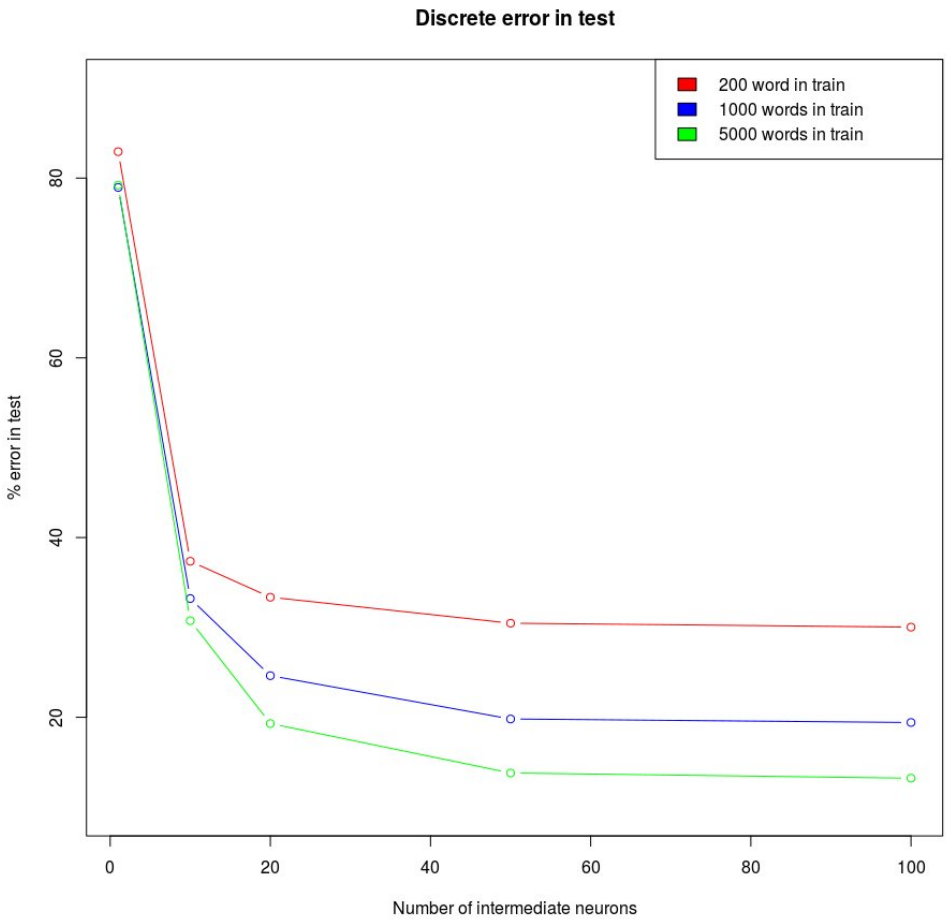
Los dos parámetros a optimizar, learning rate (η) y momentum (μ), fueron elegidos utilizando grid search. Realicé la búsqueda para μ y η entre $10^{-1} y 10^{-5}$ (25 combinaciones), y dado que entrenar cada red tarda un muy considerable y se deben realizar varias corridas, decidí utilizar sólo 20 neuronas en la capa intermedia para esta etapa de optimización de parámetros.

Utilicé el algoritmo backpropagation, en particular la implementación en C que utilizamos en el curso de machine learning. Dividí el conjunto de 1000 palabras en 750 para training y 250 para validación. Los parámetros óptimos fueron μ=0.0001 y η=0.01 y el código de toda esta sección puede verse en /src/ANNs

	η =.1	η =.01	η =.001	η =.0001	η =.00001
μ =.1	25.4	24.85	25.06	33.52	39.01
μ =.01	24.92	24.38	25.18	33.61	39.94
μ =.001	25.24	24.00	25.24	33.61	39.98
μ =.0001	25.26	23.99	25.26	33.60	39.98
μ =.00001	25.26	23.99	24.84	33.60	39.98

3.2 Resumen de resultados usando ANNs

Una vez seleccionados los parámetros óptimos, el siguiente paso fue entrenar redes con distintas cantidades de neuronas intermedias y con distintas cantidades de puntos en training. La siguiente grafica resume los resultados:
Algunas apreciaciones:



- Como vemos, usando una red con 20 o más neuronas podemos llegar a un accuracy realmente alto, donde más del 80% de los fonemas son generados correctamente.
- El mejor error en test se obtiene con 5000 palabras en training y 100 neuronas, donde el 86.78% de los fonemas son generados correctamente.
- A partir de 50 neuronas en adelante, continuar agregando más neuronas no mejora el resultado. Apartentemente no queda más información que pueda ser aprendida por una ANN llegado a ese punto, independientemente de la complejidad de la red.
- Similarmente, la ganancia de seguir agregando más palabras al dataset de entrenamiento se reduce cada vez más a medida que llegamos a unas 5000 palabras.

El "estancamiento" en la reduccion del error descripto en los ultimos dos items probablemente se deba a:

- Hay irregularidades y excepciones del lenguaje que probablemente no puedan ser aprendidas de otra forma que memorizando los casos puntuales.
- Estamos usando una ventana de 7 caracteres para pronunciar, y probablemente hay palabras cuyos patrones de pronunciacion podrian ser aprendidos considerando una ventana mayor.

Me parece asombroso llegar a predecir con tanta exactitud los fonemas, dado que este problema es muy complejo y posee una gran cantidad de reglas y excepciones. En el paper original se alcanzan porcentajes similares, e incluso se llega superar el 90% entrenando la red con más de 30000 épocas.

Lo que mas me sorprende de estos resultados es que con sólo 200 palabras de ejemplo del idioma somos capaces de predecir la pronunciación del resto del idioma con casi 80% de precisión.

3.3 Ejemplos y conclusión

Esta tarea exhibe una gran cantidad de regularidades globales junto con reglas más especializadas y casos excepcionales. Los resultados fueron muy buenos y las redes aprendieron exitosamente a generar una pronunciación cercana a la real. Veamos algunos ejemplos:

Written Representation	predicted Phonetics	real Phonetics
machine	m@C-en-	mxS-in-
learning	lI-rnIG-	l--RnIG-
unexpected	xnEXpEktxd	xnIXpEktxd
beautiful	bcc-tIfcl	bY--tIf^l
telescope	tIlEskxp-	tElxskop-
virus	varxs	vArxs

Como se puede apreciar, la red se confunde en varios fonemas, pero los errores son muy similares a los que cometemos las personas, pues la confusión se da generalmente entre vocales con sonidos similares.

Me interesaría mucho probar si este mismo procedimiento funcionaría para aprender cualquier idioma, provisto un dataset de palabras y sus transcripciones fonéticas.

4 Random Forest

Utilizando el dataset codificado, con 5000 palabras codificadas en el conjunto de training, decidí aplicar el algoritmo random forest.
Dado que hay 20 columnas binarias de output, entrené 20 clasificadores, cada uno especializado en predecir la columna i-ésima.
Finalmente, combiné los resultados de los 20 clasificadores para generar las predicciones completas del conjunto de 15000 palabras test. El tiempo de entrenamiento fue similar al de redes neuronales, pues hubo que entrenar 20 clasificadores. Utilicé 500 árboles por ensemble.
default p.

Cantidad de palabras en train	RF Error discreto en test %	SVM Error discreto en test
200	33.72	37.5
1000	22.14	27.64
5000	14.27	16.63

References

[1] Terrence J. Sejnowski and Charles R. Rosenberg *Parallel Networks that Learn to Pronounce English Text.* Complex Systems 1 (1987)