FACULTAD DE CIENCIAS EXACTAS, INGENIERÍA Y AGRIMENSURA

# Aprendiendo a pronunciar texto en Inglés

Jeremías Rodríguez

July 22, 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 fonemas[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: Algunos de los 51 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], fue usado por primera vez en el paper "Parallel Networks that Learn to Pronounce English Text"[1]. Los autores entrenaron redes neuronales que convierten strings de texto en inglés (e.g. *calculator*) en strings de fonemas ( $kælkjωleIt@$).

En este trabajo recrearé y analizaré varios experimentos llevados a cabo en el citado paper (con ANNs). Posteriormente aplicaré otros clasificadores (Random Forest, Gaussian Process, SVM y KNN) y compararé resultados.

# 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. Los autores del paper original eligieron un conjunto de 51 fonemas. (ver rawDs/netTalk.names para mas detalle).

En el extracto del dataset que he mostrado podemos ver que se ha forzado una correspondencia 1-1 entre letras de la primera columna y fonemas de la segunda: cada fonema corresponde al sonido producido por una de las letras . 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
```

Los autores del paper original incluyen un fonema "vacío", representado con el símbolo '-', para indicar la ausencia de sonido. Utilizando este simbolo, se completa las transcripciones foneticas para que tengan 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 numerica. 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 original realizaron una codificación y transformación bastante compleja para obtener datos numéricos con los que trabajar.

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

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:

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

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, la letra misma 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.

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

Qué ventajas tiene g respecto a f? Al tener un input fijo de 7 caracteres y un output fijo de 1 fonema; podemos aplicar los algoritmos de machine learning que hemos estudiado (tener inputs/outputs de longitud variable complicaria el problema) . Ademas, 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/encodedDs.

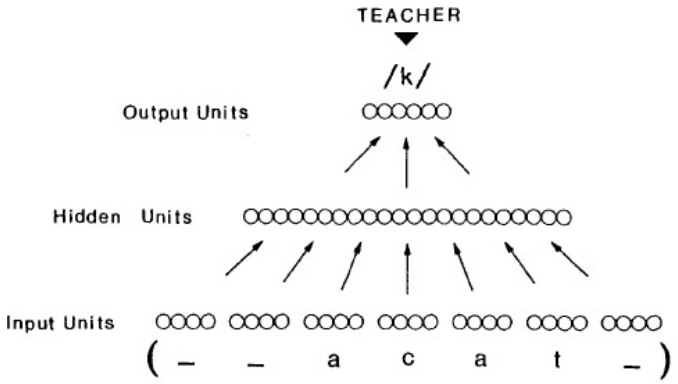
El paso restante es codificar todos los caracteres (letras y fonemas) de las 146934 filas como números. Para codificar las **letras del input** usaremos una representación muy simple: Dado que tenemos 26 letras en el alfabeto inglés, cada letra por 26 columnas binarias 1-0. Si la primera letra era una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

Por lo tanto, el dataset final de 146934 filas tendrá 26 * 7 columnas binarias representando las 7 letras input; y otras 18 columnas binarias representando el fonema output. A este dataset resultante le aplicaré distintas técnicas de aprendizaje que estudiamos en el curso.

[3] Los 50 fonemas se codifican en 18 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.

# 3 REDES NEURONALES

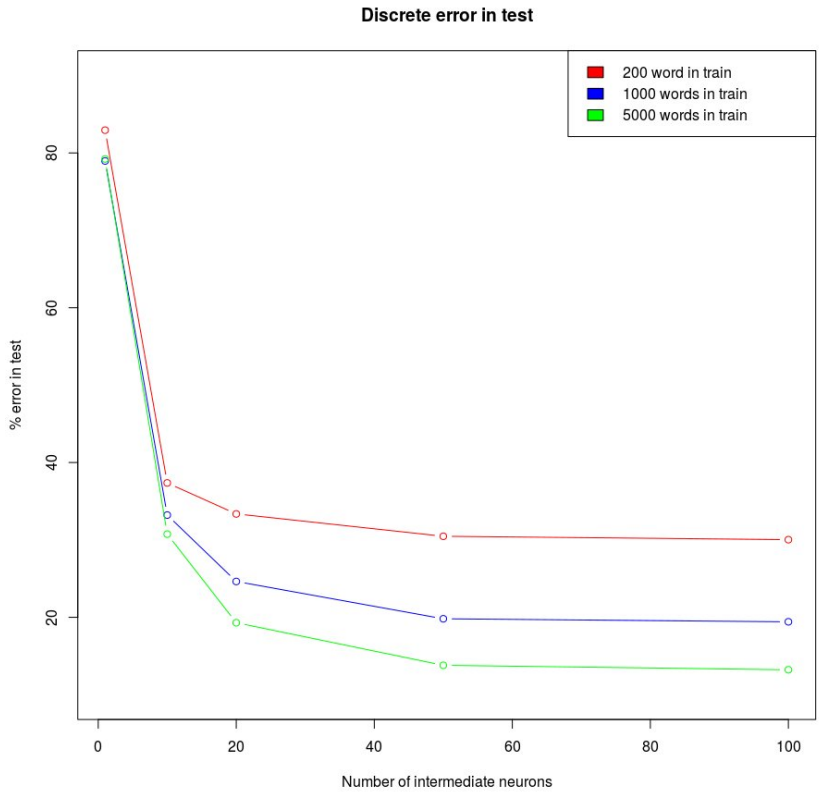
El primer paso es repetir los exprimentos que se realizaron usando redes neuronales en [1]. Para ello, utilizaré una red neuronal con 26 * 7 neuronas en la capa de entrada, y 18 en la capa de salida.




## 3.1 OPTIMIZACIÓN DE PARÁMETROS

Para optimizar parametros se usaron 1000 de las 20000 palabras del dataset original; divididas en 750 training y 250 validacion. Realicé la búsqueda en grid search para learning rate ($\eta$) y momentum ($\mu$) entre $10^{-1}y10^{-5}$ (25 combinaciones). Utilicé el algoritmo backpropagation, en particular una version modificada del codigo en C que utilizamos en el curso de machine learning. Los parámetros óptimos fueron $\mu$=0.0001 y $\eta$=0.01 y el código de toda esta sección puede verse en /src/ANNs

## 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, y los resultados pueden verse en la figura anterior.

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. (Para comprobar esto, entrene una red con 150 neuronas intermedias; y el error no mejoro aun entrenando con muchisimas epocas)
- 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 que:

- 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, altamente dimensional y posee una gran cantidad de reglas y excepciones.

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 (habilidad que a un ser humano puede llevarle realmente mucho tiempo y ejemplos adquirir).

### 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 OTROS METODOS DE MACHINE LEARNING

Me interesa analizar ahora la performance de otros algoritmos que hemos estudiado en el curso, y compararla con los resultados vistos en la seccion anterior. Los metodos que decidi aplicar a este problema de clasificacion son:

- **Nearest Neighbor**: Decidi usar este metodo tan simple para obtener un "baseline" del error. En este caso, si deseamos predecir el fonema generado por el input (y,p,o,t,e,n,u), se descartan las columnas de contexto y simplemente se busca en el dataset de training otro input donde la letra a predecir sea tambien t. Encontrado este nearest neighbor en train, se retorna el fonema asociado.

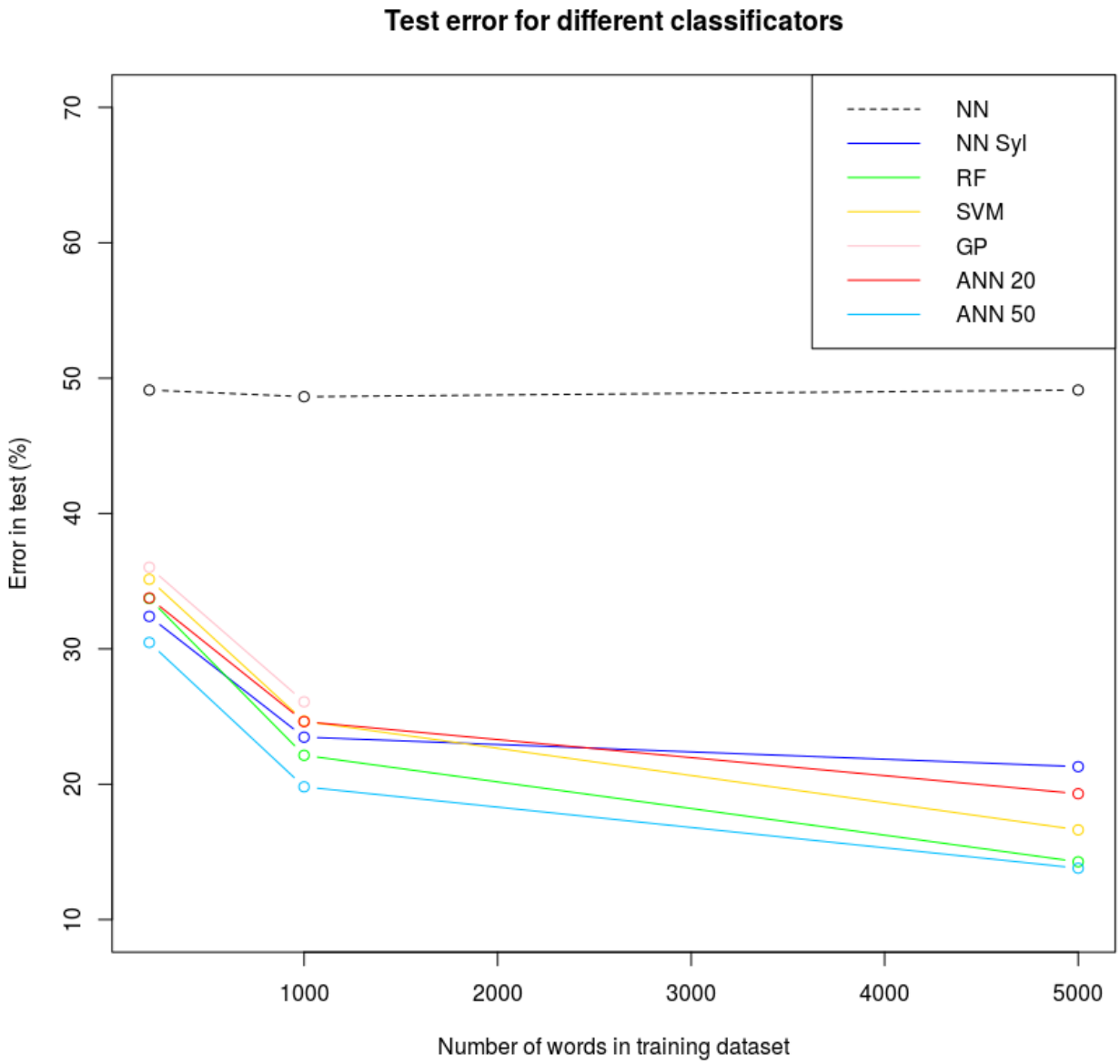
- **Nearest Neighbor silaba**:Similar al anterior. En este caso, si deseamos predecir el fonema generado por el input (y,p,o,t,e,n,u), buscamos el nearest neighbor en train usando las tres columnas centrales (o,t,e).

- **Random Forest**: Dado que hay 18 columnas binarias de output, entrené 18 clasificadores RF, cada uno especializado en predecir la columna de output i-ésima. Utilicé 500 árboles por ensemble.

- **SVM**: Al igual que con RF, entrene 18 clasificadores, cada uno especializado en predecir la columna de output i-ésima. Use un kernel Gaussiano, lo cual involucra la dificultad extra de optimizar parametros. Para cada una de las 18 columnas, optimice $C$ y $\sigma$ en grid search.

- **Gaussian Processes**: Si bien no estudiamos este metodo en el curso, me parecio interesante incluirlo(add reference). Al igual que para SVM y RF, tuve que entrenar un clasificador por columna. El kernel usado es Gaussiano, aunque la optimizacion de parametros es realizada automaticamente por kernlab.

Para cada uno de estos 5 metodos, repeti los calculos usando datasets de entrenamiento de distinto tamanyo: 200, 1000 y 5000 palabras. Las restantes (19800, 19000 y 15000 respectivamente) fueron usadas para calcular el error discreto en la prediccion de cada fonema.




*porque ANNS son mas suitable *cositas extra de c/metodo
anexo 1: tabla

## REFERENCES

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