Sistema de reconocimiento de habla continua extremo a extremo

Daniel de la Osa Fernandez D.OSA@ESTUDIANTES.MATCOM.UH.CU
Grupo C-411

Wendy Diaz Ramires W.DIAZ@ESTUDIANTES.MATCOM.UH.CU
Grupo C-412

Dayrene Fundora Gonzales D.FUNDORA@ESTUDIANTES.MATCOM.UH.CU
Grupo C-411

Jose Luis Alvarez de la Campa J.ALVAREZ@ESTUDIANTES.MATCOM.UH.CU
Grupo C-412

Resumen

Presentamos un sistema de reconocimiento del lenguaje con las mas recientes técnicas que conforman el estado del arte sobre el tema en estos momentos, desarrollado usando **extremo-a-extremo** y **deep learning**. Nuestra arquitectura es mucho mas simple que los sistemas tradicionales, capaz de adaptarse con facilidad a las diferentes restricciones que posee este problema como es la independencia del hablante o sonido ambiental. Todo esto se logra mediante una robusta función que aprende a lidiar con esto. No se necesita un diccionarios de fonemas, todo se logra con una **red neuronal recurrente (RNN)** bien optimizada entrenada usando una **tarjeta gráfica (GPU)** así como técnicas para para la síntesis de de los datos usados para el entrenamiento.

1. Introducción

Los sistemas de voz tradicionales utilizan muchas etapas de procesamiento altamente diseñadas, incluyendo características de entrada especializadas, modelos acústicos y **Modelos Ocultos de Markov (HMM)**. Para mejorar estas etapas, los expertos en el dominio deben invertir un gran esfuerzo para ajustar sus características y modelos. La introducción de los algoritmos de aprendizaje profundo [1, 2, 3, 4] ha mejorado el rendimiento del sistema de voz, generalmente mejorando los modelos acústicos. Si bien esta mejora ha sido significativa, el aprendizaje profundo aún juega solo un papel limitado en las técnicas para el reconocimiento del habla tradicionales. Para mejorar el rendimiento en tareas tales como reconocer el habla en un entorno ruidoso en los sistemas tradicionales, se debe diseñar ingeniosamente para alcanzar la robustez. Por el contrario, este sistema aplica el *aprendizaje profundo* de **extremo-a-extremo** utilizando **RNN** aprovechando la capacidad que brindan los sistemas de aprendizaje profundo para aprender a partir de grandes conjuntos de datos. El modelo está entrenado de extremo a extremo para producir transcripciones y, por lo tanto, con suficientes datos y potencia de cálculo, pueden aprender la robustez al ruido o variación de la voz del hablante por su cuenta.

Este sistema se desarrolló con la intención de resolver el problema de reconocer el habla continua. Por lo que se trabajó primero en como obtener a partir de las ondas sonoras emitidas por el hablante información manejable matemáticamente, para lograr extraer sus características. Estas últimas luego servirían para el entrenamiento, es decir se debía encontrar una descripción matemática de los sonidos emitidos en idioma español para que la función pudiera clasificar de cierta manera los sonidos y así ir generando la secuencia de caracteres del lenguaje. Para el entrenamiento se uso Heroicos Corpus que contiene alrededor de 15000 muestras sonoras de varios hablantes de ambos sexos y de diferentes países de habla hispana.

La mayor motivación es el hecho de que los sistemas tradicionales para lograr resolver este problema generan modelos muy complejos que requieren para su correcto funcionamiento capacidades importantes de cómputo, por lo que pueden llegar a ocupar gigas de espacio en las unidades de almacenamiento por lo que su uso en dispositivos móviles pasa por que estos se conecten a Internet y manden el audio digital para su procesamiento remoto y esperar la respuesta. Debido a este inconveniente se quería desarrollar un sistema que pudiera ser capaz de generar un modelo que solo ocupe unas decenas de megabytes en disco lo que permitiera su incorporación en dispositivos móviles para su uso sin acceso a Internet. Nuestro sistema lo decidimos nombrar "**Linda Deep Speech**".

2. Desarrollo

Para construir el sistema pasamos por varios pasos de ingeniería, entrenamiento y prueba. Estos serán discutidos paso a paso.

2.1 Preprocesamiento

Para alimentar nuestra red neuronal primero se hizo un pre procesamiento del conjunto de datos, ya que había que generar un espectrograma de la señal sonora para extraer las características que permitieran la predicción. Para esto se utilizaron los 26 coeficientes cepstrales en frecuencias mel, los cuales describen numéricamente estas señales. Luego con esto se creo el vector de características que luego se normalizo para que todos tuvieran la misma longitud ya que todas las grabaciones no duraban la misma cantidad de tiempo. Junto con esto se creo también el vector de etiquetas asociadas a cada archivo de audio es decir a cada vector de características.

2.2 Arquitectura

El núcleo de nuestro sistema es una **red neuronal recurrente (RNN)** entrenada para ingerir espectrogramas de voz y generar transcripciones de texto en español. Sea x un archivo de audio de audio procesado y y su respectiva etiqueta de transcripción, del conjunto de entrenamiento $X = \{(x^1, y^1), (x^2, y^2), ...\}$ cada componente x^i es una serie de tiempo de longitud T^i donde para cada fragmento de tiempo es un vector de características del audio x_t^i, $t = 1, 2, 3, ..T^i$. De manera que como usamos las *frecuencias mel* lo que vamos a encontrar en $x_{t,p}^i$ es el poder de coeficiente p en el tiempo t. El objetivo de nuestro **RNN** es convertir una secuencia de entrada x en una secuencia de probabilidades de los caracteres para la transcripción y, con $\hat{y_t} = \mathbb{P}(c_t|x)$, donde $c_t \in \{a, b, c, d, ..., space, blank\}$.

Se crearon diferentes arquitecturas de modelos que se pueden ver en archivo *models.py*. Pero la que alcanzo mejores resultados en las pruebas es la que vamos a describir. Esta **RNN** posee 5 capas de unidades ocultas. Para la entrada x, las unidades ocultas en la capa l se denota por h^l con la convención de que h^0 es la entrada. Las primeras tres capas no son recurrentes. La salida de h^1 la salida, para cada tiempo t, depende del espectrograma x_t y C intervalos de tiempos de contexto. Las siguientes capas no recurrentes operan en sobre datos independientes para cada t. Entonces para cada tiempo t las tres capas son computadas por:

$$h_t^l = g(W^l h_t^{l-1} + b^l) \tag{1}$$

Donde $g(z) = \text{mín}\{\text{máx}\{0, z\}, 20\}$ es la función de activación cortada de **rectificacion linear (ReLu)**, y W^l, b^l son la matriz de pesos y el parámetro de *bias* para la capa l.

La cuarta capa es una capa *bidireccional recurrente*. Esta capa incluye dos conjuntos de unidades ocultas recurrentes: uno con recurrencia hacia delante h^f (2), y otro con recurrencia hacia atrás h^b (3):

$$h_t^f = g(W^4 h_t^3 + W_r^f h_{t-1}^f + b^4) \tag{2}$$

$$h_t^b = g(W^4 h_t^3 + W_r^b h_{t-1}^b + b^4) \tag{3}$$

La quinta capa que es no recurrente toma la unidades de *hacia adelante* y las *hacia atrás* como entrada $h_t^5 = g(W^5 h_t^4 + b^5)$ donde $h_t^4 = h_t^f + h_b +$. La capa de salida es una función *softmax* estándar devuelve la probabilidad del caracter predicho por cada t y cada caracter k en el alfabeto:

$$h_{t,k}^6 = \hat{y_{t,k}} = \mathbb{P}(c_t = k|x) = \frac{exp(W_k^6 h_t^5 + b_k^6)}{\sum exp(W_j^6 h_t^5 + b_j^6)} \tag{4}$$

Donde en W_k^6 y b_k^6 k denota la k-ésima columna.

2.3 Ajuste del Modelo y Optimización

Ya tenemos la primera predicción es decir tenemos dos los valores de $\mathbb{P}(c_t|x)$, producto del recorrido y transformación de las entradas al pasar las capas de la red. Ahora para lograr el aprendizaje debemos ver que tanto se equivocó para luego reajustar los parámetros y reinventar de nuevo.

Para esto primeramente se calcula el **CTC loss** [6] para luego mediante el algoritmo de backpropagation poder calcular y propagar el gradiente de la funcion de costo para luego mediante optimizadores; en este caso **ADAM**, poder calcular que tanto hay que moverse, es decir ajustar los W_i^j para lograr una mejor evaluación de la de la función de costos (**CTC loss**), o sea *reducir el error* en la predicción de los transcritos.

Como podemos ver este problema en esencia es un problema de optimización con decenas de miles de parámetros; incluso pudieran llegar a ser millones, en nuestro caso las experimentaciones tuvieron que trabajar con 541000 parámetros. Y el método para resolverlo radica simplemente en moverse en *dirección contraria* al *gradiente* de la *función de costos* tratando de buscar un *mínimo global* lo cual en muchos casos no podemos asegurar encontrar ya que los problemas tiene que ser *convexos* y en la práctica esto no sucede con frecuencia.

Luego el optimizador de **ADAM** es el estado de arte para muchos de estos problemas ya que define que tanto me voy a mover en la dirección del *gradiente* descendiente de forma eficiente y sorteando obstáculos como son que se caiga en un punto mínimo local malo y otros obstáculos que crean la optimización de estos problemas. Por esto este optimizador entra en la clasificación de *algoritmos gradientes-descendientes*. Muy utilizados en hoy en día.

2.4 Datos de Entrenamineto

La búsqueda de un corpus suficientemente grande para lograr el entrenamiento fue una de las tareas más arduas ya que para nuestro idioma no existen muchos gratis. Se trabajó después de mucha búsqueda sobre el West Point Heroico Spanish Speech que es bastante discreto en cuanto a longitud pero aún así se logró alcanzar 70 % en el WER. Con un aumento del conjunto de entrenamiento es bastante posibles alcanzar mucho mejores resultados.

3. Conclusiones

Hemos presentado un sistema de reconocimiento del habla usando las técnicas de extremo-a-extremo y

aprendizaje profundo para lo cual se ha indagado en profundidad en el problema de modelar y optimizar un modelo acústico desde cero exponiendo las ventajas que se supone frente a los sistemas tradicionales para resolver este problema. Se alcanzo alrededor de un 70 % en el WER creando predicciones que nos asombraron bastante y esto se logro con escasos recursos de computo usando una *GPU NVIDIA 960* con $4gb$ de *RAM*. Esto incita a seguir trabajando en el tema con nuevas ideas para optimizar más aún el modelo. Otro punto importante es que se logró en parte el objetivo de lograr reducir el tamaño en disco del modelo ya que ocupa solo $6mb$.

4. Recomendaciones

A partir de lo estudiado existen varias ideas sobre como proseguir el estudio de este tema. Primero hacerse con un corpus mucho mayor en cuanto a tamaño, el cual si pudiera ser de hablantes nacionales incluso mejor para crear un producto más autóctono y adaptado a nuestro acento. Esto nos permitiría lograr de seguro mayores resultados según el estado del arte sobre el tema.

Otro elemento sería la incorporación de un modelo linguístico en el proceso de decodificación de las perdiciones de manera que las probabilidades de que una secuencias de palabras (*1-ngram*,*2-ngram*, etc.) aparezca en la lengua española también influyan en escoger la secuencia más probable y así mejorar la predicción y la revisión ortográfica de las predicciones; ya que pudiera existir ambigüedad en las pronunciaciones. Por ejemplo la palabra *caza* y *casa*, en contexto de secuencias de n-gramas se pudiera deducir a través del modelo del lenguaje a cual palabra se refiere el hablante.

Por último introducir ruidos en los audios del conjunto de datos de entrenamiento para lograr mayor robustez en el sistema frente al sonido ambiental, con el objetivo final de poder usarlos en los diferentes ambientes donde puede encontrarse el dispositivo móvil que en definitiva es el dispositivo principal donde se desea ejecutar este sistema. Todo esto es posible dada la forma en que se entrena el modelo, la función objetivo se adaptaría a estos ruidos.

Referencias

[1] *R. Grosse, R. Raina, H. Kwong, and A. Y. Ng. Shift-invariance sparse coding for audio classification.* arXiv preprint arXiv:1206.5241, 2012.

[2] G. Dahl, D. Yu, L. Deng, and A. Acero. *Context-dependent pre-trained deep neural networks for large vocabulary speech recognition.* IEEE Transactions on Audio, Speech, and Language Processing, 2011.

[3] G. Hinton, L. Deng, D. Yu, G. Dahl. *Deep neural networks for acoustic modeling in speech recognition.* IEEE Signal Processing Magazine, 29(November):82–97, 2012.

[4] A. Mohamed, G. Dahl, and G. Hinton. *Acoustic modeling using deep belief networks.* IEEE Transactions on Audio, Speech, and Language Processing, (99), 2011.

[5] H. Lee, P. Pham, Y. Largman, and A. Y. Ng. *Unsupervised feature learning for audio classification using convolutional deep belief networks.* In Advances in Neural Information Processing Systems, pages 1096–1104, 2009.

[6] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. *Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks.* In ICML , pages 369– 376. ACM, 2006.