Informe Hito 1

Pablo Cleveland
Camilo Escobar

Diego Garrido
Pablo Miranda

Abstract

Language Modeling es una técnica ampliamente usada en el mundo del procesamiento de lenguaje natural, debido a que sirve de alimento para modelos que resuelven otras tareas, prácticamente en todas las que requieren generar texto dado un contexto. En este trabajo se implementará un modelo basado en redes neuronales: *Gated Convolutional Neural Network* (GCNN) sobre el *dataset* de WikiText-103 que tiene alrededor de 103 millones de *tokens* con un vocabulario de 260.000 palabras. Siendo el *output* de este modelo la representación: **H**. Como baseline se ocupará el modelo de trigramas con interpolación lineal y *perplexity* como métrica de evaluación.

1. Motivación

Los lenguajes naturales emergen, no son diseñados, a diferencia de los lenguajes de programación. Para estos últimos, contamos con definiciones formales de todas las palabras reservadas y las formas en que pueden ser usadas de manera válida, lo mismo no es cierto para los lenguajes naturales. A pesar de que existen reglas formales y heurísticas para partes del lenguaje, no existe un conjunto de estas que permita especificarlo de manera plena. Por esta razón, son interesantes los modelos de lenguaje, los cuales tratan de alguna u otra forma entender el funcionamiento del lenguaje.

El objetivo principal de *Language Modeling* es aprender la función de probabilidad conjunta de secuencias de palabras en un lenguaje para así generar una secuencia de palabras dado un contexto (otra secuencia de palabras). Para este fin, una representación del lenguaje es necesaria con el fin de que sea entendible para la máquina y permita la utilización de texto en otros modelos. A menudo, los modelos de lenguaje, sirven como la piedra angular de otras tareas. Por ejemplo, en speech recognition usualmente es emparejado con un modelo acústico el cual genera una lista de potenciales oraciones con sus respectivas probabilidades, las cuales luego son reordenadas mediante la probabilidad de ocurrencia de esas oraciones de acuerdo con el *language model*.

Dentro de las aplicaciones más comunes en las que se usa *language modeling* tenemos: *machine translation, spell correction, speech recognition, summarization, question answering, sentiment analysis, tagging, handwriting recognition, information retrieval, parsing, optical character recognition*, entre otras.

2. Definición de Language Modeling

Los modelos de lenguaje (*Language Models*) estiman la distribución de probabilidad de una secuencia de palabras (también pueden ser sentencias o letras), modelando la probabilidad de la siguiente palabra dado las palabras anteriores, esto es

$$P(w_0, \dots w_N) = P(w_0) \prod_{i=1}^{N} P(w_i | w_0, ..., w_{i-1})$$

donde w_i es una palabra del vocabulario. El *input* en *language modeling* es una secuencia ordenada de palabras y el *output* es un vector de probabilidades sobre el vocabulario, con la probabilidad de ocurrencia de cada una de las palabras del vocabulario dadas las palabras anteriores. El empleo de un modelo de lenguaje entrenado consiste en entregarle una secuencia de palabras y comenzar a generar nuevas palabras a partir del vector de probabilidad de ocurrencia, la generación puede ser un *sampling* aleatorio sobre el vector o la palabra más probable.

3. Dataset

El *dataset* a utilizar es el WikiText-103 un *dataset* con alrededor de 103 millones de *tokens* y con un vocabulario de aproximadamente 260.000 palabras (Merity et al., 2016), en este corpus cada secuencia es un párrafo. El *dataset* cuenta con 28,475 artículos para *training*, 60 para *validation* y 60 para *testing* (ver Tabla 1 con el resumen). A los *tokens* con una ocurrencia inferior a 3 se les asignó el *token* $< unk >$ como también a los *tokens* presentes en el conjunto de *validation* y *test* que no están presentes en el conjunto de *train*, los *tokens* que quedaron fuera del vocabulario (Out of Vocabulary) representan el 0.4 % del vocabulario.

	Train	**Valid**	**Test**
Articles	28,475	60	60
Tokens	103,227,021	217,646	245,569
Vocab	267,735		
OoV	0.4 %		

Table 1: Estadísticas del dataset WikiText-103.

4. Métrica de evaluación

La métrica de evaluación escogida es el *perplexity*, el cual es el estándar para evaluar modelos del lenguaje,

$$l = \frac{1}{\sum_{d \in D} |S_d + 1|} \sum_{d \in D} \sum_{j \in S_d \cup </S>} - \ln P(w_j | \ldots, w_{j-1})$$

$$\text{perplexity} = e^l$$

donde D es el conjunto de documentos o párrafos, S_d es la secuencia de palabras del documento d, w_j es la palabra j-ésima de un documento y $< /S >$ representa el último término de un párrafo.

La expresión anterior se puede simplificar considerando que el total de *tokens* a predecir es $N = \sum_{d \in D} |S_d + 1|$ que corresponde a la suma de las cardinalidades de los párrafos más el *token* de término del párrafo, quedando la siguiente expresión,

$$perplexity = e^{\frac{1}{N} \sum_{i=1}^{N} -ln\ p(w_i|...,w_{i-1})}$$

La *perplexity* es una medida de cuán bien un modelo probabilístico predice una muestra, se espera que en un buen modelo del lenguaje la probabilidad de que una palabra observada ocurra dado las palabras observadas anteriormente ($p(w_i | \ldots, w_{i-1})$) sea alta. Dado que las probabilidades se mueven en el dominio [0,1] el logaritmo de una probabilidad se mueve entre (-∞,0] al multiplicar esto por -1 se tiene que el exponente es un número no negativo y la exponencial de un número no negativo tiene dominio [1, ∞), se maximiza cuando el exponente es ∞ y se minimiza cuando el exponente es 0 (todas las probabilidades son 1), por tanto, un modelo de lenguaje es mejor que otro si este tiene una menor *perplexity*.

Además es posible demostrar una cota superior práctica a la *perplexity* de un modelo probabilístico del lenguaje (un caso extremo sería considerar un modelo de trigrama sin interpolación lineal, por lo que si un brigrama no existe la perplexity dará ∞), para esto consideremos un modelo de unigramas que asume independencia entre las palabras y que asume una distribución uniforme sobre las palabras, es decir, las palabras son equiprobables con $p(w_i) = \frac{1}{|V|}$, donde V es el vocabulario, se tiene que $perplexity = |V|$, por tanto el dominio efectivo es [1, $|V|$].

5. Modelo

5.1. Baseline

El modelo que se ocupará como baseline será el modelo de lenguaje de trigramas con interpolación lineal. El modelo de trigramas consiste de un vocabulario $\mathcal{V}$ y un parámetro $q(w|u, v)$ por cada trigrama u, v, w tal que $w \in \mathcal{V} \cup \{< /S >\}$ y $u, v \in \mathcal{V} \cup \{*\}$.

Luego, para una secuencia de palabras $w_0, \ldots, w_N$, la probabilidad de la secuecia está dada por

$$p(w_0, \ldots, w_N) = \prod_{i=0}^{N} q(w_i | w_{i-2}, w_{i-1})$$

De manera análoga, se obtienen las predicciones para para el modelo de bigramas y unigramas. Con estas 3 predicciones, el modelo de trigramas con interpolación lineal da una predicción para la secuencia igual a

$$p(w_0, \ldots, w_N) = \prod_{i=0}^{N} (\lambda_1 \times q(w_i | w_{i-2}, w_{i-1}) + \lambda_2 \times q(w_i | w_{i-1}) + \lambda_3 \times q(w_i))$$

donde $\lambda_1, \lambda_2\ y\ \lambda_3$ son hiperparámetros que satisfacen que $\lambda_i \geq 0$ y $\lambda_1 + \lambda_2 + \lambda_3 = 1$

5.2. Modelo escogido

El modelo escogido corresponde a una Gated Convolutional neural network (GCNN) basado en (Dauphin et al., 2017). En el estado del arte lo más frecuente es el uso de redes neuronales recurrentes, por esta misma razón resulta interesante el enfoque escogido que usualmente es empleado en el ámbito de procesamiento de imágenes y, no obstante, logra obtener resultados competitivos en términos de perplexity (fue el estado del arte en Google Billion Word *dataset*) con una notable eficiencia computacional en comparación a enfoques basados en redes recurrentes (ver Tabla 2).

	Throughput (CPU)	Throughput (GPU)	Responsiveness (GPU)
LSTM-2048	169	45,622	2,282
GCNN-9	121	29,116	29,116
GCNN-8 Bottleneck	**179**	**45,878**	**45,878**

Table 2: Velocidad de procesamiento en *tokens/s* en *test* para una LSTM con 2048 unidades y GCNNs en Google Billion Word dataset. La GCNN con bottlenecks la *responsiveness* 20 veces mientras mantiene un alto *throughput* (Dauphin et al., 2017).

Los modelos neuronales de lenguaje producen una representación

$$\mathbf{H} = [h_0, \ldots, h_N]$$

del contexto para cada palabra $w_0, \ldots, w_N$ para predecir la siguiente palabra $\mathbb{P}(w_i|h_i)$ (Bengio et al., 2003). En el modelo convolucional $\mathbf{H}$ se obtiene haciendo la convolución del input con una función f

$$\mathbf{H} = f * w$$

A grandes rasgos, la arquitectura del modelo escogido se refleja en la figura 1. Como se observa en la figura 1 las palabras se representan por un *vector embedding* guardado en una *lookup table* $\mathbf{D}^{|\mathcal{V}|\times e}$ donde $|\mathcal{V}|$ es el número de palabras contenidas en el vocabulario y e es el tamaño del embedding. El modelo recibe como entrada una secuencia de palabras $w_0, \ldots, w_N$ que son representadas por word embeddings

$$\mathbf{E} = [D_{w_0}, \ldots, D_{w_N}]$$

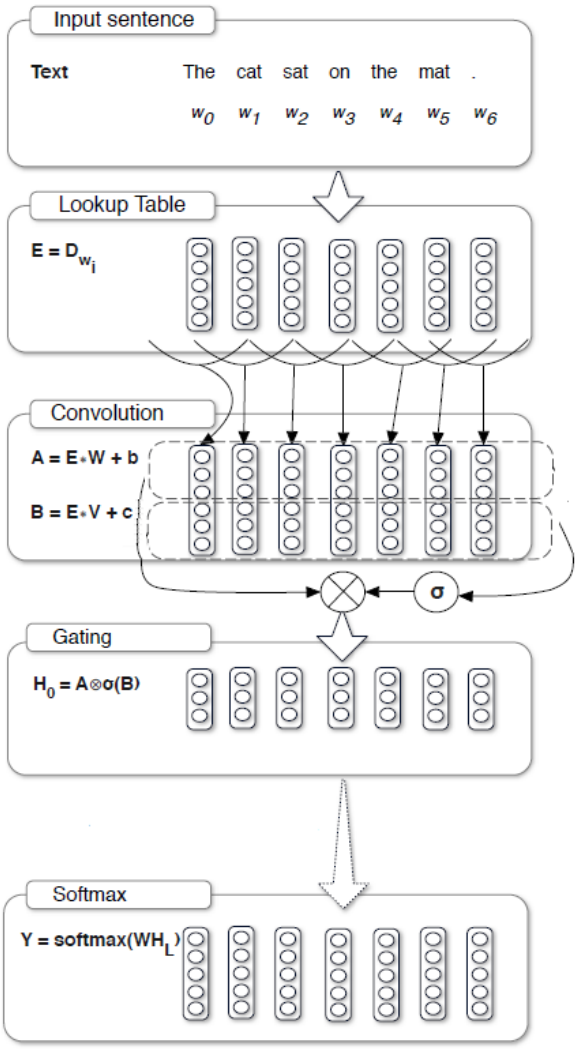

Figure 1: Arquitectura de la red

Las capas ocultas $h_0, \ldots, h_L$ se calculan como sigue:

$$(\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c}) \qquad (1)$$

donde $\mathbf{X} \in \mathbb{R}^{N\times m}$ corresponde al input de la capa l, m corresponde al número de inputs de la capa l. $\mathbf{W}, \mathbf{V} \in \mathbb{R}^{k\times m\times n}$, $\mathbf{b}, \mathbf{c} \in \mathbb{R}^n$ son parámetros aprendidos, donde k es el largo del kernel y n el número de outputs de la capa. Por último, σ es la función sigmoide y $\otimes$ es la multiplicación elemento a elemento de matrices. Para evitar ocupar información acerca del contexto futuro (palabras siguientes) (Oord et al., 2016) cuando se aplican las convoluciones se usa *zero-padding* al inicio de cada secuencia con $k - 1$ elementos. Por último, el output de la capa l se obtiene sumando el input al resultado obtenido en la ecuación 1

$$h_l(\mathbf{X}) = \mathbf{X} + (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c})$$

Para las predicciones se usa *adaptive softmax* que asigna mayor capacidad a las palabras frecuentes y menor capacidad a las palabras infrecuentes (Grave et al., 2017).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org.

Edouard Grave, Armand Joulin, Moustapha Cissé, Hervé Jégou, et al. 2017. Efficient softmax approximation for gpus. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1302–1310. JMLR. org.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. *arXiv e-prints*, page arXiv:1609.07843.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.