Modelos de secuencias: redes recurrentes y recursivas

César Olivares

Pontificia Universidad Católica del Perú
Maestría en Informática
INF659 - Técnicas avanzadas de data mining y sistemas inteligentes

2018

Redes Neuronales Recurrentes

- Las redes neuronales recurrentes (RNNs) son una familia de redes neuronales para el procesamiento de datos secuenciales $\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(\tau)}$.
- La extensión de las redes multicapas hacia las redes recurrentes se fundamenta en la idea de **compartir parámetros** en el paso de cada paso de la secuencia hacia el siguiente.

Grafos computacionales

- Un grafo computacional es una formalización de la estructura de un conjunto de operaciones computacionales.

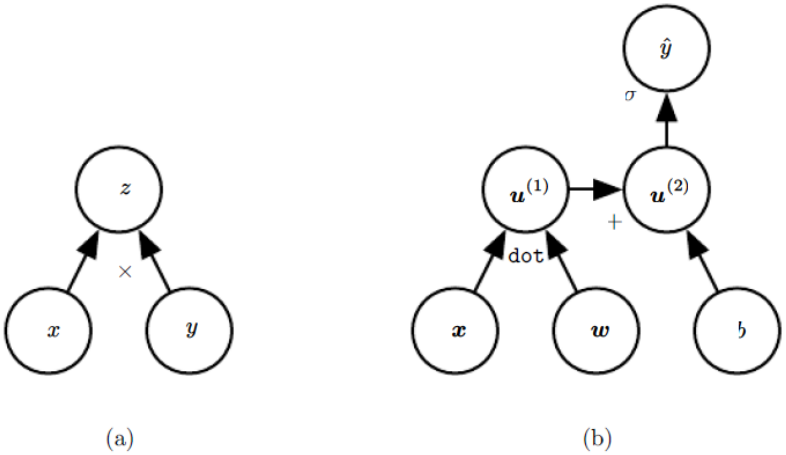


Figura 1: Ejemplos de grafos computacionales. (a) Uso de la operación $\times$ para calcular $z = xy$. (b) Cálculo de la predicción para regresión logística: $\hat{y} = \sigma(\boldsymbol{x}^\top \boldsymbol{w} + b)$. Algunas de las expresiones intermedias no llevan nombre en la expresión algebraica pero necesitan nombre en el grafo. (Goodfellow, 2016)

Grafos computacionales desplegados

- Cuando un cálculo tiene una estructura repetitiva, su representación desplegada (*unfolded* o *unrolled*) evidencia la presencia de parámetros compartidos.
- Por ejemplo, la forma clásica de un sistema dinámico es:

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}; \boldsymbol{\theta}) \tag{1}$$

donde $\boldsymbol{s}^{(t)}$ es el estado del sistema.

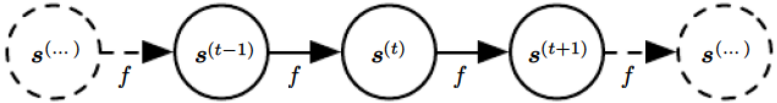


Figura 2: Grafo computacional desplegado de un sistema dinámico clásico. Cada nodo representa el estado en un tiempo t, y la función f mapea el estado en t al estado en $t+1$. La función f utiliza los mismos parámetros $\boldsymbol{\theta}$ en todos los pasos de tiempo. (Goodfellow, 2016)

RNNs como Grafos computacionales desplegados

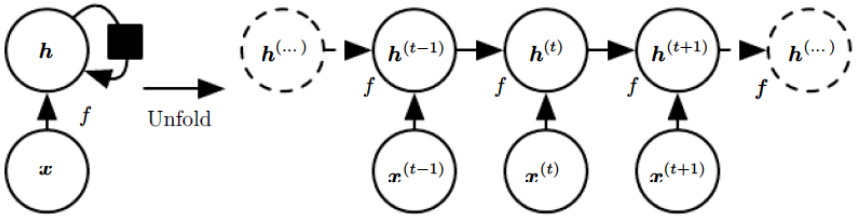
- En el caso de un sistema dinámico conducido por una señal de entrada externa $\boldsymbol{x}^{(t)}$, el estado recoge información sobre toda la secuencia previa:

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}) \tag{2}$$

- Así como casi cualquier función puede ser considerada una red neuronal *feed-forward*, cualquier función que implique recurrencia puede ser considerada una RNN.
- Para indicar que el estado son las unidades ocultas de la red, se usa la variable $\boldsymbol{h}$ para representar el estado:

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}) \tag{3}$$

- Para tareas predictivas, las RNNs aprenden a usar $\boldsymbol{h}^{(t)}$ como un sumario imperfecto de los aspectos relevantes del pasado.



(Goodfellow, 2016)

Ventajas de la recurrencia

- Podemos representar la recurrencia después de t pasos como una función $g^{(t)}$:

$$\begin{aligned} \boldsymbol{h}^{(t)} &= g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, ..., \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) \\ &= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}) \end{aligned} \tag{4}$$

- La estructura recurrente nos permite factorizar $g^{(t)}$ en la aplicación repetida de una función f. Ello nos trae las siguientes ventajas:
  1. El tamaño de las entradas al modelo es independiente del tamaño de la secuencia.
  2. Es posible usar la *misma* función de transición f con los mismos parámetros en cada paso de tiempo.
- Aprender un único modelo compartido nos permite generalizar el aprendizaje a secuencias de diferentes longitudes, con menos ejemplos de entrenamiento.

Tipos de secuencias modeladas por RNNs

- Las RNNs pueden modelar secuencias de entrada y de salida, así como mapear secuencias a puntos de datos individuales (y viceversa).

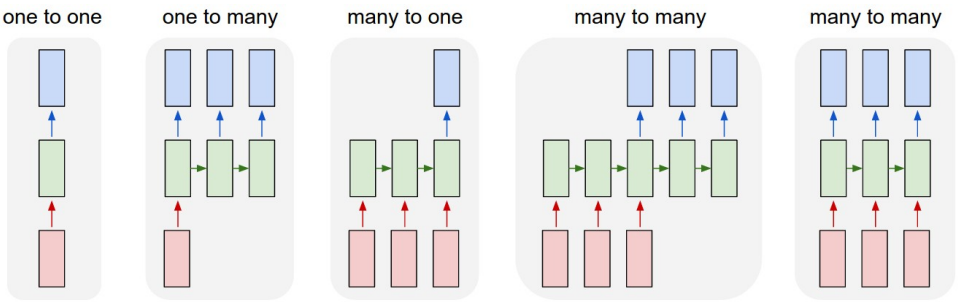


Figura 3: Modelos de secuencias modeladas por RNNs. Cada rectángulo es un vector de nodos. Cada flecha representa una función (p.ej. multiplicación de matrices). De izquierda a derecha: (1) Modelo de red feed-forward, no RNN. (p.ej. clasificación de imágenes). (2) Secuencia de salida (p.ej. anotación de imágenes). (3) Secuencia de entrada (p.ej. análisis de sentimiento). (4) Secuencias de entrada y salida (p.ej. traducción automática). (5) Secuencias sincronizadas (p.ej. clasificación de cada fotograma de video). (Karpathy 2015)

RNN «many-to-many» con conexiones «hidden-to-hidden»

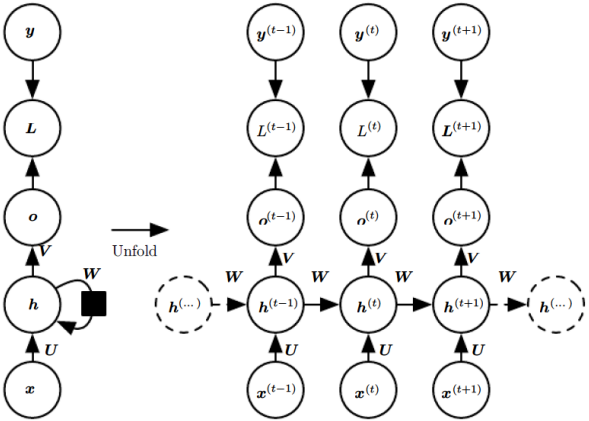
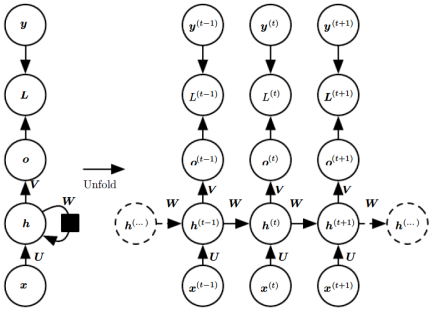


Figura 4: RNN que produce una salida $\boldsymbol{o}$ en cada paso de tiempo y emplea conexiones recurrentes entre las unidades ocultas. La pérdida L mide la distancia entre cada $\boldsymbol{o}$ y su correspondiente objetivo $\boldsymbol{y}$. Cuando se usa salidas *softmax*, asumimos que $\boldsymbol{o}$ son las log-probabilidades sin normalizar (*logits*). Las matrices $\boldsymbol{W}$ y $\boldsymbol{U}$ (más un bias $\boldsymbol{b}$) parametrizan el paso de un estado al siguiente. La matriz $\boldsymbol{V}$ (más un bias $\boldsymbol{c}$) parametriza las salidas. (Goodfellow 2016)

RNN «many-to-many» con conexiones «hidden-to-hidden» (2)



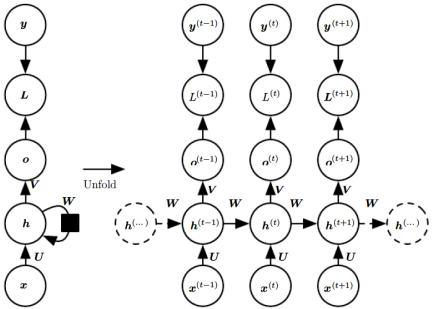
$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \tag{5}$$

$$\boldsymbol{h}^{(t)} = tanh(\boldsymbol{a}^{(t)}) \tag{6}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \tag{7}$$

$$\hat{\boldsymbol{y}}^{(t)} = softmax(\boldsymbol{o}^{(t)}) \tag{8}$$

RNN «many-to-many» con conexiones «hidden-to-hidden» (3)



$$
\begin{aligned}
L(\{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, ..., \boldsymbol{y}^{(\tau)}\}) &= \sum_t L^{(t)} \\
&= -\sum_t \log p_{model}(\boldsymbol{y}^{(t)} | \{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(\tau)}\})
\end{aligned} \tag{9}
$$

- El cálculo del gradiente de la función de pérdida es muy costoso: $O(\tau)$ tanto en tiempo de ejecución como en consumo de memoria.
- El algoritmo de retropropagación aplicado al grafo desplegado con costo $O(\tau)$ se denomina retropropagación en el tiempo (BPTT).

Redes con recurrencia desde la salida

- Las redes con recurrencia sólo desde la salida, son estrictamente menos potentes.
- Si **o** no tiene muchas dimensiones y riqueza representacional, la expresividad de la red queda severamente limitada.
- El entrenamiento es muy sencillo y completamente paralelizable pues el gradiente de cada paso se puede calcular aisladamente.

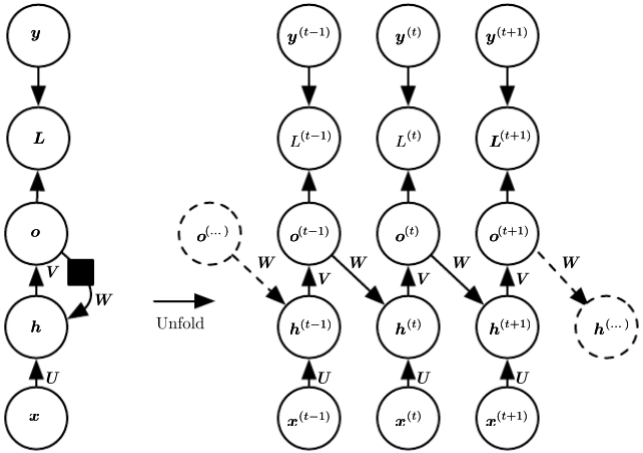
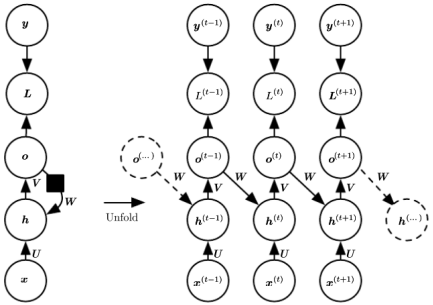


Figura 5: RNN con recurrencia desde la salida. (Goodfellow 2016)

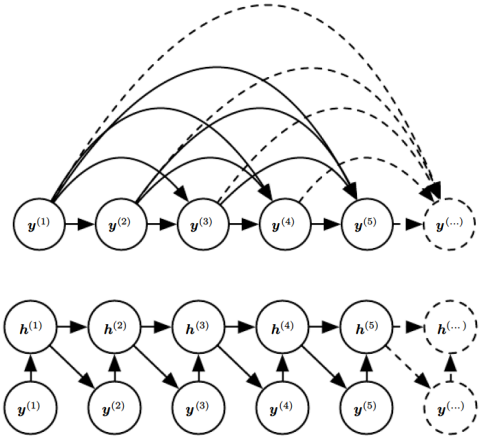
«Teacher forcing»

- Las redes con conexiones recurrentes desde las salidas pueden ser entrenadas con **teacher forcing**.
- *Teacher forcing* es el procedimiento por el cual el modelo recibe el valor conocido de $y^{(t)}$ como entrada en el momento $t + 1$.
- Este procedimiento permite optimizar la máxima verosimilitud del modelo.
- También puede incluirse en modelos que incluyan conexiones entre las capas ocultas, pero en estos casos sí se tendrá que usar la BPTT.
- Su desventaja es que si el modelo se usa en *open loop* (es decir, cuando no se sabe el valor real de $y^{(t)}$), la red será expuesta a entradas que pueden ser muy diferentes a lo que vio durante su entrenamiento.
- Se puede entrenar el modelo usando entradas reales y también predecidas, o usar una estrategia de *aprendizaje por currículo*.



RNNs en cuanto Modelos Gráficos Dirigidos

- Normalmente queremos interpretar la salida de la RNN como una distribución de probabilidades, y la función de pérdida como la entropía cruzada asociada a la distribución.
- Podemos interpretar la RNN como un modelo gráfico cuya estructura representa dependencias directas entre cualquier par de valores y.
- Si consideramos las unidades ocultas $h^{(t)}$ como variables aleatorias, el modelo revela que la RNN brinda una parametrización muy eficiente de la distribución conjunta sobre las observaciones ($O(1)$ vs. $O(k^\tau)$ para k valores posibles de y).



Estrategias para determinar la longitud de la secuencia de salida

- Para extraer muestras de la distribución de probabilidad se requiere un mecanismo que determine la longitud de la secuencia de salida.
  - Se puede añadir un símbolo especial que indique el final de una secuencia.
  - Se puede añadir una salida binaria al modelo que represente la decisión de continuar o interrumpir la generación de muestras.
  - Se puede añadir una salida extra que prediga la longitud τ de la secuencia de salida.

Modelado de secuencias condicionado al contexto: «one-to-many»

- Para condicionar una secuencia de salida de acuerdo a un vector $\mathbf{x}$ de contexto, de longitud fija (no secuencial), se puede alimentar $\mathbf{x}$:
  - como entrada adicional en cada paso de tiempo,
  - como estado inicial $h^{(0)}$, o
  - las dos anteriores.

Modelado de secuencias condicionado al contexto: «one-to-many» (2)

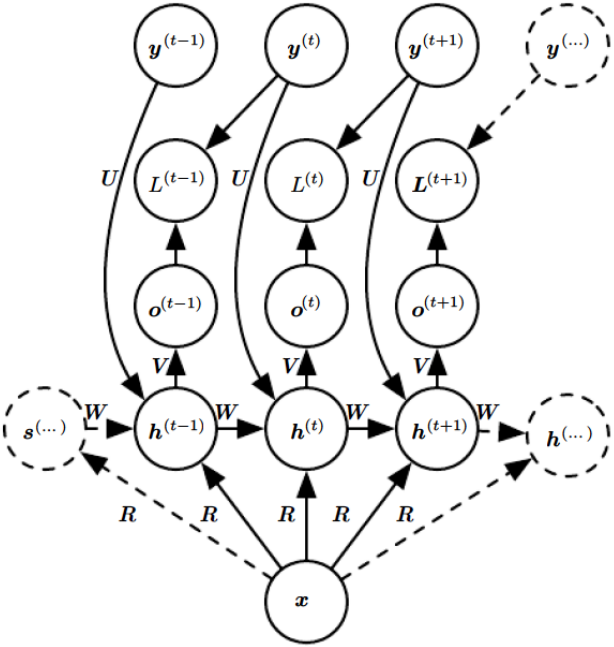


Figura 6: RNN que mapea un vector de longitud fija **x** a una distribución de secuencias **Y**. Apropiada para anotación de imágenes. Cada $h^{(t)}$ se usa a la vez como entrada en t y objetivo de entrenamiento para $t-1$. (Goodfellow 2016)

RNN «many-to-many» con conexiones «output-to-hidden»

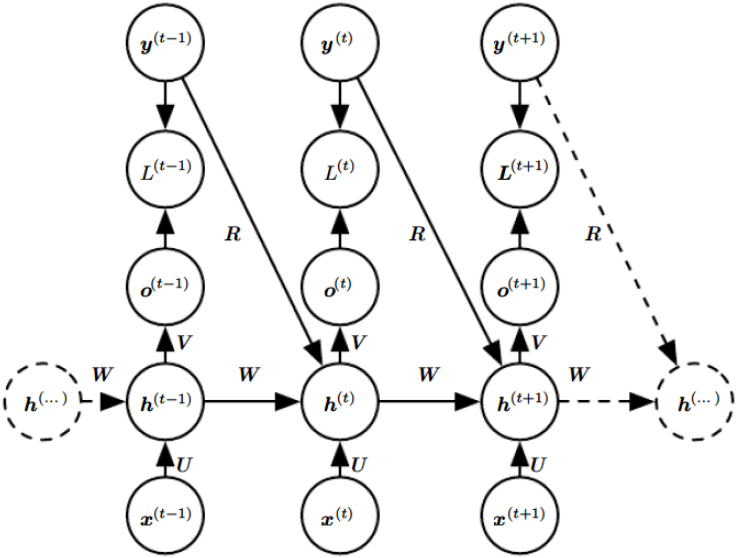


Figura 7: La adición de conexiones desde las salidas hacia las unidades ocultas permiten modelar distribuciones arbitrarias $P(y|x)$ (de la misma longitud) donde los valores y no son necesariamente independientes entre sí dados los valores x. (Goodfellow 2016)

RNNs bidireccionales

- Las RNNs anteriores tienen estructuras «causales»: cada estado de un momento t sólo contiene información del pasado y el presente.
- Las RNNs bidireccionales permiten modelar los casos en que cada predicción de $y^{(t)}$ depende de *toda la secuencia*, tanto de pasos previos como de pasos futuros, pero es más sensible a los valores alrededor de t.

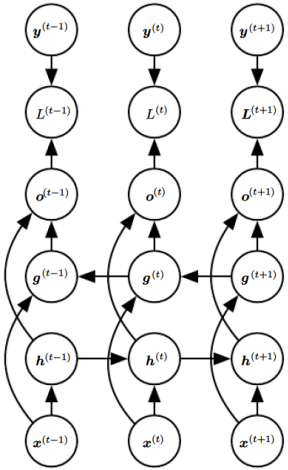
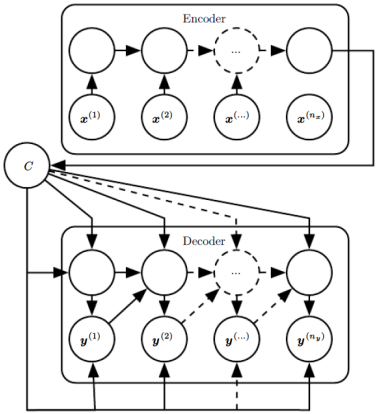


Figura 8: RNN bidireccional típica. Cada paso t se beneficia de un resumen relevante del pasado mediante h y del futuro mediante g. (Goodfellow 2016)

Arquitecturas «Sequence-to-Sequence»

- Las arquitecturas «Sequence-to-Sequence» modelan la probabilidad de una secuencia de salida $(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, ..., \boldsymbol{y}^{(n_y)})$ dada una secuencia de entrada $(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(n_x)})$.
- Una RNN codificadora genera una variable de contexto C, que representa un resumen semántico de la entrada y es alimentada como entrada a una RNN decodificadora, la cual genera la secuencia de salida (o calcula su probabilidad)
- La longitud n_x es independiente de n_y.
- C podría tener una longitud variable, y se puede añadir un mecanismo de atención.



RNNs profundas

- La mayoría de RNNs se componen de tres bloques de parámetros y transformaciones:
  1. De la entrada al estado oculto
  2. De un estado oculto al siguiente
  3. Del estado oculto a la salida
- Los tres bloques pueden beneficiarse con el uso de arquitecturas profundas.

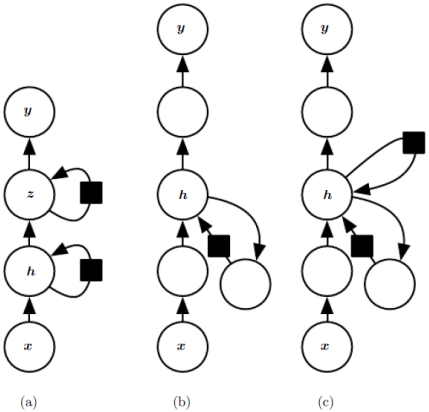
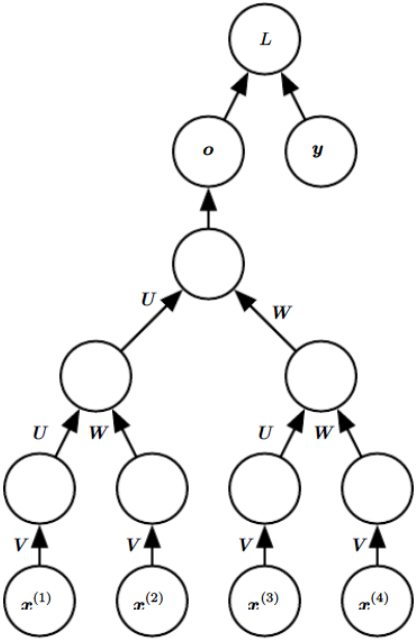


Figura 9: *(a)* Unidades ocultas organizadas en grupos jerárquicos. *(b)* Profundidad en los tres bloques. *(c)* «Skip connections» para mitigar el alargamiento en las conexiones recurrentes (Goodfellow 2016)

Redes Neuronales Recursivas

- Las redes **recursivas** forman árboles de conexiones con parámetros compartidos.
- Han sido usadas en procesamiento de lenguaje natural y visión computacional.
- Potencialmente podrían utilizarse para aprender a razonar (Bottou 2011).



Bibliografía

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. MIT Press. Retrieved from http://www.deeplearningbook.org/
- Graves, A. (2012). Supervised Sequence Labelling with Recurrent Neural Networks (1st ed.). Springer-Verlag Berlin Heidelberg.
- Karpathy, A. (2015). The Unreasonable Effectiveness of Recurrent Neural Networks. Retrieved September 21, 2017, from http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Lipton, Z. C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. CoRR, abs/1506.0, 1–38. https://doi.org/10.1145/2647868.2654889