

Como enseñar a las máquinas a leer y comprender

Sanchez Ramos, Lesly Dashiel
Universidad Nacional
de Ingeniería
Lima, Perú
lesly.sanchez.r@uni.pe

Yagi Vasquéz, Gladys Alesandra
Universidad Nacional
de Ingeniería
Lima, Perú
gladys.yagi.v@uni.pe

Resumen—Los modelos RNN utilizados en máquinas con el propósito de enseñarles a leer y comprender han ido en aumento en los últimos años. Sin embargo, sigue siendo un reto implementar un modelo completamente fiable para esta clase de propósito. Además, esta clase de modelo trabajan con un gran conjunto de datos.

En el presente trabajo se implementó un modelo basado en RNN para enseñarle a una máquina a leer y comprender un libro de niños.

Palabras Claves: NLP, RNN, Seq2Seq, CBT.

I. INTRODUCCIÓN:

Los avances en los algoritmos de Deep Learning, con el objetivo de darle a una máquina la capacidad de leer y comprender el lenguaje en documentos, han ido en aumento [1]. Esto se debe a que en los últimos años los conjuntos de datos para entrenamiento y prueba a gran escala para evaluar este tipo de capacidad han aumentado, así como la creación de frameworks, tales como ParlAI, que nos ofrecen una amplia gama de herramientas para entrenar modelos de lectura y comprensión así como modelos de diálogo y respuesta.

La capacidad de las máquinas para leer y comprender un determinado documento puede ser testada mediante su habilidad para poder responder preguntas y respuestas por medio de contenido de documentos, predecir palabras en historias incompletas, resumir reseñas de productos o predecir si una premisa y su hipótesis se contradicen o son neutrales entre sí.

Teaching Machines to Read and Comprehend [1] es uno de los papers más famosos y mencionados que implementa modelos basados en redes neuronales para probar la capacidad de un sistema para responder preguntas planteadas sobre el contenido de un documento que han visto. Uno de esos modelos es el Deep LSTM Reader, donde por medio un encoder y decoder Deep LSTM, se procesan las consultas y luego el documento, dando como resultado un modelo que

procesa cada par de consultas del documento como única secuencia larga.

Un conjunto de datos famoso por ser utilizado para la comprensión lectora de las máquinas es el CNN/Daily Mail.

Original Version	Anonymised Version
Context The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." ...	the <i>ent381</i> producer allegedly struck by <i>ent212</i> will not press charges against the " <i>ent153</i> " host, his lawyer said Friday. <i>ent212</i> , who hosted one of the most-watched television shows in the world, was dropped by the <i>ent381</i> Wednesday after an internal investigation by the <i>ent180</i> broadcaster found he had subjected producer <i>ent193</i> "to an unprovoked physical and verbal attack." ...
Query Producer X will not press charges against Jeremy Clarkson, his lawyer says.	producer X will not press charges against <i>ent212</i> , his lawyer says.
Answer Oisin Tymon	<i>ent193</i>

Figura 1. Versión original y anonimizada de un artículo extraído del conjunto de datos de validación del Daily Mail. Las entidades anonimizadas están constantemente permutándose durante la prueba y entrenamiento.

De la Figura 1 podemos observar como se extraen las entidades de oraciones clave de los artículos de las noticias para ser reemplazados con espacios en blanco. Para así poder entrenar a la máquina en responder preguntas dependiendo del texto restante.

En este trabajo, se buscará analizar un modelo de red neuronal recurrente que sea capaz de completar oraciones a partir de fragmentos de un libro para niño, se pretende que esto sea posible mediante el uso del framework ParlAI, la cual también es una plataforma que nos ofrece tutoriales de como añadir nuestros propios datasets así o usar los que nos proporcionan, crear nuestros propios modelos, entrenarlos y validarlos, así como implementar agentes que se encargen de diversas tareas.

Demostraremos facilidad que tiene esta plataforma así como la eficacia de nuestro modelo para esta clase de trabajos.

II. OBJETIVO DEL ESTUDIO

Comprender y analizar las redes neuronales recurrentes y como estas permiten que una máquina pueda leer y comprender un cuento para niños.

II-A. Objetivos Específicos

1. Implementar el modelo Seq2Seq en el framework ParlAI para enseñarle a una máquina a leer y comprender.
2. Evaluar la efectividad del modelo Seq2Seq para completar oraciones de un libro para niños.

III. MARCO TEÓRICO

III-A. Natural Language Processing

El procesamiento del lenguaje natural (NLP) es un campo de la ciencia de la computación que se ocupa del uso de las computadoras para la extracción, interpretación, análisis y manipulación del lenguaje humano. Algo relacionado al NLP es el Question answering (QA), el cual es una tarea del NLP encargada de responder a preguntas en el lenguaje natural, aprovechando los métodos de recuperación de información y representación del conocimiento [2]. Siendo de esta manera la forma en la que la máquina aprenderá a leer y comprender .

III-B. Recurrent Neural Network

Las redes neuronales recurrentes (RNN) tienen la capacidad de modelar la información contextual en secuencias largas. Para el procesamiento del lenguaje natural se desea capturar las dependencias a largo plazo; además de mantener el orden contextual entre las palabras para resolver el significado global de un texto. Otra ventaja de las RNN es su capacidad para aprender representaciones de secuencias de longitud variable, como frases, documentos y muestras de habla [2].

III-C. Long Short-Term Memory

La memoria a largo y corto plazo (LSTM) utiliza puertas de entrada, salida y olvido para controlar la propagación del gradiente en la memoria de la red neuronal recurrente (RNN). Esto permite proteger las celdas de memoria que lleven algún estado oculto al siguiente paso. Los estados de activación permiten que la red aprenda las condiciones para cuando olvidar o mantener la información en la celda de memoria [2].

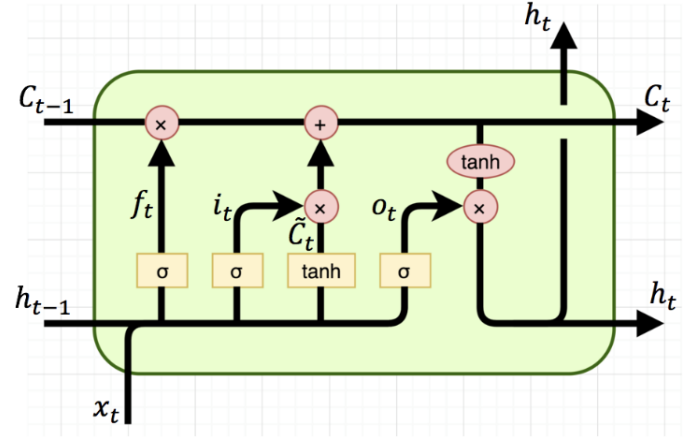


Figura 2. Diagrama de una celda LSTM

III-D. Seq2Seq

El modelo Sequence to Sequence se utiliza cuando tenemos una entrada de longitud variable que no es necesariamente una entidad asociada a otra. Un modelo Seq2Seq se compone más comunmente de dos RNN que trabajan de manera cooperativa: encoder y decoder. Ambas siendo modelos LSTM. Este modelo es usado comúnmente para máquinas de traducción, resumen de texto, modelos de conversación, predicción de palabras, y más.

III-D1. Encoder: Se encargará de codificar el input palabra por palabra en un vector de estado o de contexto (en el caso de LSTM estas son llamadas estado oculto o vectores de estado de las celdas).

III-D2. Decoder: Este es el estado oculto final por parte del codificador. Se encarga de encapsular todos los elementos de entrada para poder realizar una predicción.

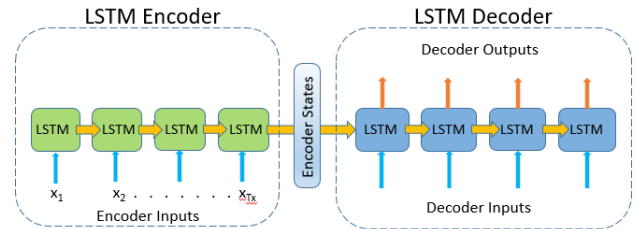


Figura 3. Modelo encoder-decoder con LSTM unidireccionales

En la Figura 2, el encoder codifica una secuencia de entrada pasándolas por las celdas LSTM, este encoder consta de los estados ocultos; luego, se pasa a la decodificación como estados iniciales junto con los

inputs del decodificador para poder hacer predicciones.

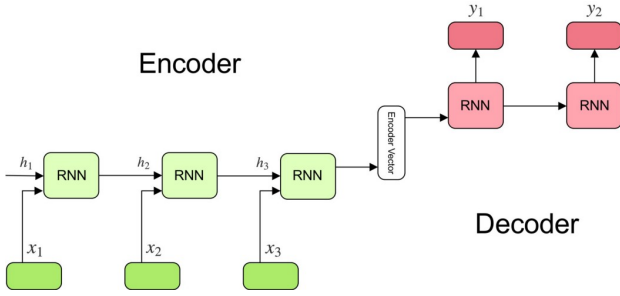


Figura 4. Encoder-decoder

Como podemos observar en la Figura 4, cada rectángulo representa una capa oculta en un paso de tiempo t .

La fórmula para calcular los elementos de salida de la capa oculta en cada paso de tiempo t es [4]:

$$h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t) \quad (1)$$

A continuación, se detalla los parámetros asociados a esta fórmula:

- $x_1, \dots, x_{t-1}, x_t, x_{t+1}, \dots, x_T$: Los vectores de palabras correspondientes a un documento con T palabras.
- W^{hx} : Matriz de pesos utilizada para condicionar el vector de palabras de entrada x_t .
- W^{hh} : Matriz de pesos utilizada para condicionar el vector de palabras de salida h_{t-1} .
- h_{t-1} : Salida de la función no lineal en el paso de tiempo anterior $t - 1$.
- σ : Función de no lineal (sigmoide).

Además, tenemos que la función para calcular la distribución de probabilidad de salida sobre el vocabulario en cada paso del tiempo t es:

$$y_t = \text{softmax}(W^s h_t) \quad (2)$$

Donde W_s es la matriz de pesos utilizada de acuerdo al vocabulario y a la palabra de salida h_{t-1} . e y_t es la siguiente palabra predicha dado el contexto del documento hasta el momento (es decir h_{t-1}) y el último vector palabra observado x_t .

IV. ESTADO DEL ARTE:

En la actualidad existen novedosas arquitecturas de redes neuronales para tareas de comprensión lectora de la máquina. En esta sección describiremos algunas de estas.

IV-A. Deep LSTM Reader

Deep LSTM Reader propuesto en el artículo [Teaching Machines to Read and Comprehend](#) [1] es un modelo que pone a prueba la capacidad de los Deep LSTM

encoder para manejar secuencias largas de inputs. Este modelo se alimenta palabra por palabra de artículos del CNN al Deep LSTM encoder y luego con un delimitador también se introducen consultas al codificador. Como resultado, se obtiene un modelo que procesa cada par de consultas del artículo como una única secuencia larga. Dado el artículo y la consulta, el modelo predice el token del artículo que responde a la consulta.

IV-B. ReasoNet

ReasoNet propuesto en el artículo [ReasoNet: Learning to Stop Reading in Machine Comprehension](#) [3] es una arquitectura de red neuronal llamada red de razonamiento la cual tiene el fin de realizar las tareas de comprensión de la máquina. Las ReasoNets hacen uso de múltiples giros para explotar eficientemente y razonar sobre la relación entre las consultas, los documentos y las respuestas. Con el uso de aprendizaje por refuerzo, las ReasoNets pueden determinar dinámicamente continuar con el proceso de comprensión tras digerir los resultados intermedios, o terminar la lectura cuando concluye que la información existente es adecuada para producir una respuesta.

IV-C. R-NET

ReasoNet propuesto en el artículo [R-Net: Machine Reading Comprehension with self-matching networks](#) [5] es un modelo de redes neuronales de extremo a extremo para responder a preguntas del estilo de la comprensión lectora, cuyo objetivo es responder a preguntas de un pasaje dado. En este artículo primero se empareja la pregunta y el pasaje con redes recurrentes basadas en la atención para obtener la representación del pasaje consciente de la pregunta. Luego propone un mecanismo de atención auto-ajustada para refinar la representación comparando el pasaje consigo mismo, lo que codifica eficazmente la información de todo el pasaje. Por último, emplea redes de punteros para localizar las posiciones de las respuestas de los pasajes.

V. METODOLOGÍA:

En esta sección se describirá los pasos del desarrollo del proyecto y las herramientas necesarias.

V-A. ParlAI

ParlAI es un framework de software de código abierto para la investigación de diálogos implementada en Python, disponible en <http://parl.ai>. Su objetivo es proporcionar un marco unificado para compartir, capacitar y probar modelos de diálogo; integración de Amazon Mechanical Turk para la recopilación de datos, la evaluación humana y el aprendizaje por refuerzo; además cuenta con un [repositorio](#) de modelos de aprendizaje automático para comparar con otros modelos y mejorar las arquitecturas existentes. También ofrece más de 20 tareas, incluye conjuntos de datos

populares como SQuAD, bAbI tasks, MCTest, WikiQA, QACNN, QADailyMail, CBT, bAbI Dialog, Ubuntu, OpenSubtitles y VQA.

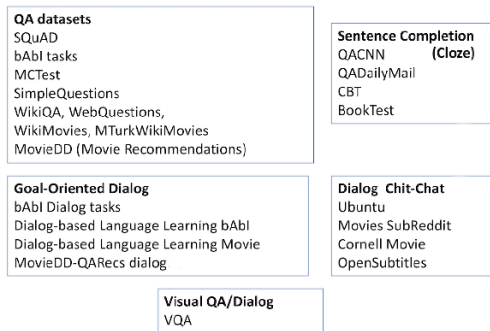


Figura 5. Tareas en la primera versión de ParlAI. Fuente: [ParlAI: A Dialog Research Software Platform](#)

Además de la amplia gama de datasets disponibles, ofrece una amplia gama de ayudantes para crear nuestros propios agentes.

ParlAI integra varios modelos, incluidos modelos neuronales como redes de memoria, Seq2seq y attentive LSTMs, los cuales implementaremos en este proyecto.

VI-B. Metodología de Trabajo

Pasos a seguir para el desarrollo del proyecto:

1. Primeros instalamos el framework ParlAI
2. Importamos el dataset CBT
3. Implementamos los agente para el modelo
4. Creamos el modelo Seq2Seq
5. Entrenamos el modelo

VI. EXPERIMENTACIÓN Y RESULTADOS

VI-A. Conjunto de Datos

Para la evaluación del modelo se tomó el dataset [Children'S Book Test \(CBT\)](#), ya que este era el de menor tamaño y esto haría que el modelo cargue más rapido el dataset, en comparación a otros datasets conocidos como CNN y Daily Mail.

CBT es un dataset que está diseñado para medir directamente qué tan bien los modelos lingüísticos pueden explotar un contexto lingüístico más amplio, por esta razón la usaremos para que la máquina aprender a leer y comprender. Cabe mencionar que la CBT se construye a partir de libros que están disponibles gratuitamente, gracias al Proyecto Gutenberg.

VI-B. Experimentación

Entrenamos el modelo limitandolo al tiempo de entrenamiento, obtuvimos los siguientes resultados:

1. Para 10 minutos de entrenamiento

Evaluación para los datos de validación.

```
07:56:39 | valid:
accuracy bleu-4 clen ctpb ctps ctrunc ctruncen exps exs f1 gpu_mem llen loss lr ltpb ltps
\
all .03812 3.813e-11 512.3 4098 15776 0 0 30.79 8000 .03812 .2582 2.013 5.521 .00053 16.1 61.98
cbt:CN 0 0 522.1 0 0 2000 0 0 2000 0 2.004 5.885
cbt:NE 0 0 489 0 0 2000 0 0 2000 0 2.047 7.126
cbt:P .1525 1.525e-10 528.9 0 0 2000 .1525 2 3.809
cbt:V 0 0 509.2 0 0 2000 0 0 2000 0 2 5.254

ltrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 0 0 463.3 .5141 .03787 80 4115 15838
cbt:CN 0 0 359.7 .4968 0
cbt:NE 0 0 1257 .4839 0
cbt:P 0 0 45.12 .5757 .1515
cbt:V 0 0 191.4 .5000 0
```

Figura 6. Tabla 1. Fuente: Autoría propia

Evaluación para los datos de test

```
08:02:18 | test:
accuracy bleu-4 clen ctpb ctps ctrunc ctruncen exps exs f1 gpu_mem llen loss lr ltpb ltps
\
all .0399 3.99e-11 524.4 4192 15498 0 0 29.55 10000 .0399 .2585 2.029 5.849 .00053 16.22 59.86
cbt:CN 0 0 539.4 0 0 2500 0 0 2500 0 2.004 5.992
cbt:NE 0 0 503.7 0 0 2500 0 0 2500 0 2.11 8.446
cbt:P .1596 1.596e-10 537.1 0 0 2500 .1596 2 3.83
cbt:V 0 0 517.3 0 0 2500 0 0 2500 0 2.001 5.128

ltrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 0 0 1318 .5112 .0396 80 4208 15558
cbt:CN 0 0 400.3 .4954 0
cbt:NE 0 0 4656 .4704 0
cbt:P 0 0 46.07 .5792 .1584
cbt:V 0 0 168.6 .4998 0
```

Figura 7. Tabla 2. Fuente: Autoría propia

2. Para 20 minutos de entrenamiento:

Evaluación para los datos de validación:

```
08:51:30 | valid:
accuracy bleu-4 clen ctpb ctps ctrunc ctruncen exps exs f1 gpu_mem llen loss lr ltpb ltps
\
all .03988 3.988e-11 512.3 8197 23889 0 0 46.63 8000 .03988 .2588 2.013 5.492 .00053 32.2 93.86
cbt:CN 0 0 522.1 0 0 2000 0 0 2000 0 2.004 5.96
cbt:NE 0 0 489 0 0 2000 0 0 2000 0 2.047 7.274
cbt:P .1595 1.595e-10 528.9 0 0 2000 .1595 2 3.428
cbt:V 0 0 509.2 0 0 2000 0 0 2000 0 2 5.307

ltrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 0 0 515.8 .5158 .03963 82 8229 23983
cbt:CN 0 0 387.7 .4968 0
cbt:NE 0 0 1443 .4841 0
cbt:P 0 0 30.81 .5793 .1585
cbt:V 0 0 201.6 .5000 0
```

Figura 8. Tabla 3. Fuente: Autoría propia

Evaluación para los datos de test

```
08:55:18 | test:
accuracy bleu-4 clen ctpb ctps ctrunc ctruncen exps exs f1 gpu_mem llen loss lr ltpb ltps
\
all .0405 4.05e-11 524.4 8363 23081 0 0 44.02 10000 .0405 .2596 2.029 5.835 .00053 32.35 89.29
cbt:CN 0 0 539.4 0 0 2500 0 0 2500 0 2.004 6.067
cbt:NE 0 0 503.7 0 0 2500 0 0 2500 0 2.11 8.642
cbt:P .1628 1.628e-10 537.1 0 0 2500 .1628 2 3.479
cbt:V 0 0 517.3 0 0 2500 0 0 2500 0 2.001 5.153

ltrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 0 0 1575 .5117 .0402 82 8395 23178
cbt:CN 0 0 431.5 .4968 0
cbt:NE 0 0 5665 .4706 0
cbt:P 0 0 32.42 .5804 .1608
cbt:V 0 0 172.9 .4998 0
```

Figura 9. Tabla 4. Fuente: Autoría propia

3. Para 30 minutos de entrenamiento:

Evaluación para los datos de validación:

```

09:05:05 | valid:
accuracy bleu-4 clen ctpb ctps ctrunc ctrunclen exps exs f1 gpu_mem llen loss lr ltpb ltps \
all .0406 4e-11 512.3 8197 23555 0 0 45.98 8000 .0406 .2587 2.013 5.479 .00053 32.2 92.54
cbl:CN 0 0 522.1 0 0 2000 0 2.004 6.005
cbl:NE 0 0 489 0 0 2000 0 2.047 7.331
cbl:P .1608 1.6e-10 528.9 0 0 2000 .1608 2 3.275
cbl:V 0 0 509.2 0 0 2000 0 2 5.304

litrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 0 0 548.2 .5161 .03975 83 8229 23648
cbl:CN 0 0 405.6 .4961 0
cbl:NE 0 0 1527 .4880 0
cbl:P 0 0 26.45 .5795 .1590
cbl:V 0 0 201.2 .5000 0

```

Figura 10. Tabla 5. Fuente: Autoría propia

```

11:27:59 | valid:
accuracy bleu-4 clen ctpb ctps ctrunc ctruncen exps exs f1 gpu_mem llen loss lr ltpb ltps
\
all .06838 6.842e-11 512.3 4098 16190 0 0 31.6 8000 .06846 .2583 2.013 5.749 .0010 16.1 63.61
cbl:CN 0 1.839e-13 522.1 0 0 2000 .0003333 2.004 5.837
cbl:NE 0 0 489 0 0 2000 0 2.047 8.261
cbl:P .2735 2.735e-10 528.9 0 0 2000 .2735 2 3.64
cbl:V 0 0 509.2 0 0 2000 0 2 5.256

litrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 0 0 1111 .5043 .06838 147 4115 16254
cbl:CN 0 0 342.7 .4910 0
cbl:NE 0 0 3070 .3989 0
cbl:P 0 0 30.09 .6332 .2735
cbl:V 0 0 191.8 .4943 0

```

Figura 14. Tabla 9. Fuente: Autoría propia

Evaluación para los datos de test

```

09:09:00 | test:
accuracy bleu-4 clen ctpb ctps ctrunc ctruncen exps exs f1 gpu_mem llen loss lr ltpb ltps
\
all .0405 4.05e-11 524.4 8363 23458 0 0 42.83 10000 .0405 .2596 2.029 5.821 .00053 32.35 86.88
cbl:CN 0 0 539.4 0 0 2500 0 2.004 6.118
cbl:NE 0 0 503.7 0 0 2500 0 2.11 8.406
cbl:P .1620 1.62e-10 537.1 0 0 2500 .1620 2 3.339
cbl:V 0 0 517.3 0 0 2500 0 2.001 5.129

litrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 0 0 1658 .5128 .0402 83 8395 22545
cbl:CN 0 0 454.1 .4962 0
cbl:NE 0 0 5570 .4750 0
cbl:P 0 0 28.2 .5804 .1608
cbl:V 0 0 168.8 .4998 0

```

Figura 11. Tabla 6. Fuente: Autoría propia

4. Para 40 minutos de entrenamiento:

Evaluación para los datos de validación

```

09:44:05 | valid:
accuracy bleu-4 clen ctpb ctps ctrunc ctruncen exps exs f1 gpu_mem llen loss lr ltpb \
all .04013 4.013e-11 512.3 8197 23830 0 0 46.52 8000 .04013 .2588 2.013 5.485 1.01e-05 32.2
cbl:CN 0 0 522.1 0 0 2000 0 2.004 6.005
cbl:NE 0 0 489 0 0 2000 0 2.047 7.456
cbl:P .1609 1.609e-10 528.9 0 0 2000 .1609 2 3.099
cbl:V 0 0 509.2 0 0 2000 0 2 5.328

ltps ltrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 93.62 0 0 596.3 .5103 .03988 86 8229 23924
cbl:CN 0 0 426.3 .4968 0
cbl:NE 0 0 1731 .4885 0
cbl:P 0 0 22.18 .5797 .1595
cbl:V 0 0 206 .5000 0

```

Figura 12. Tabla 7. Fuente: Autoría propia

Evaluación para los datos de test

```

09:47:51 | test:
accuracy bleu-4 clen ctpb ctps ctrunc ctruncen exps exs f1 gpu_mem llen loss lr ltpb \
all .0406 4.06e-11 524.4 8363 23303 0 0 44.44 10000 .0406 .2596 2.029 5.823 1.01e-05 32.35
cbl:CN 0 0 539.4 0 0 2500 0 2.004 6.106
cbl:NE 0 0 503.7 0 0 2500 0 2.11 8.804
cbl:P .1624 1.624e-10 537.1 0 0 2500 .1624 2 3.175
cbl:V 0 0 517.3 0 0 2500 0 2.001 5.126

ltps ltrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 90.15 0 0 1835 .5127 .0403 86 8395 23393
cbl:CN 0 0 485.7 .4962 0
cbl:NE 0 0 6661 .4740 0
cbl:P 0 0 23.92 .5806 .1612
cbl:V 0 0 168.4 .4998 0

```

Figura 13. Tabla 8. Fuente: Autoría propia

5. Para 80 minutos de entrenamiento:

Evaluación para los datos de validación

Evaluación para los datos de test

```

11:33:33 | test:
accuracy bleu-4 clen ctpb ctps ctrunc ctruncen exps exs f1 gpu_mem llen loss lr ltpb ltps
all .0685 6.85e-11 524.4 4192 15723 0 0 29.99 10000 .0685 .2584 2.013 5.914 .0010 16.22 60.83
cbl:CN 0 0 539.4 0 0 2500 0 2.004 5.973
cbl:NE 0 0 503.7 0 0 2500 0 2.11 8.928
cbl:P .2420 2.42e-10 537.1 0 0 2500 .2420 2 3.694
cbl:V 0 0 517.3 0 0 2500 0 2.001 5.06

litrunc ltruncen ppl token_acc token_em total_train_updates tpb tps
all 0 0 2032 .4961 .0683 147 4208 15784
cbl:CN 0 0 392.8 .4864 0
cbl:NE 0 0 7537 .3859 0
cbl:P 0 0 40.2 .6178 .2412
cbl:V 0 0 157.6 .4944 0

```

Figura 15. Tabla 10. Fuente: Autoría propia

VII. ANÁLISIS

Observamos las tablas 1, 3, 5, 7 y 9 para el análisis de la métrica accuracy que nos da la precisión del modelo. Analizando las tablas obtenidas al entrenar el modelo con los datos de evaluación cambiando el tiempo de entrenamiento para 10, 20, 30, 40 y 80 minutos, sus accuraccys respectivos son de 3.812 %, 3.988 %, 4.000 %, 4.013 % y 6.838 %; se nota un incremento del accuracy a medida que aumentamos el tiempo de entrenamiento.

Ahora observamos las tablas 2, 4, 6, 8 y 10 para el análisis de la métrica accuracy que nos da la precisión del modelo. Analizando las tablas obtenidas al entrenar el modelo con los datos de test cambiando el tiempo de entrenamiento para 10, 20, 30, 40 y 80 minutos, sus accuraccys respectivos son de 3.990 %, 3.988 %, 4.050 %, 4.060 % y 6.050 %; se nota un incremento del accuracy a medida que aumentamos el tiempo de entrenamiento.

Ambos incrementos del accuracy ocurrieron por la modificación del máximo tiempo de ejecución del entrenamiento, ya que esto hizo que el modelo aprenda más y se beneficie, aumentando su precisión.

VIII. CONCLUSIONES

Se logró comprender como trabajan las RNN en diferentes modelos y como estas permiten que las máquinas sean capaces de aprender a leer y comprender.

- Se logró implementar el modelo Seq2Seq en el ParlAI para poder leer y comprender un libro para niños.
- Se concluyó que el modelo Seq2Seq implementado necesita más tiempo de entrenanmiento para obtener mejores resultados.

REFERENCIAS

- [1] Teaching machines to read and comprehend. *Avances en los sistemas de procesamiento de información neuronal*.
- [2] Uday Kamath, John Liu, and James Whitaker. *Deep learning for NLP and speech recognition*, volume 84. Springer, 2019.
- [3] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055, 2017.
- [4] Richard Socher and Richard Socher Mundra. Cs 224d: Deep learning for nlp4. 2015.
- [5] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. R-net: Machine reading comprehension with self-matching networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, 2017.