Traducción automática: NMT-Keras

Marcos Esteve Casdemunt

Enero 2020

Contents

1 Introducción

El objetivo principal de esta práctica consiste en evaluar dos *toolkits* sobre un corpus paralelo. En primer lugar se tratará de construir un traductor estadístico utilizando el *toolkit* Moses. Posteriormente se construirá un traductor basado en redes neuronales utilizando el *toolkit* NMT-Keras.

2 NMT-Keras

NMT-Keras es un *toolkit* de traducción automática neuronal preparado para utilizar arquitecturas basadas en modelos de atención + redes recurrentes o bien modelos basados en Transformers. El toolkit proporcionará un modelo de traducción neuronal de textos de un idioma origen a un idioma destino.

Para realizar esta práctica se ha utilizado una tarjeta gráfica NVIDIA GTX 1660 para aprovechar de esta manera el paralelismo que aporta este tipo de hardware y reducir el tiempo de entrenamiento del modelo.

2.1 Ejercicio 1

Para el primer ejercicio se han explorado distintos tamaños de embedding para la capa de embeddings de entrada del encoder así como los embeddings del decoder.

Table 1: Evolución del BLEU al modificar el tamaño de los embeddings (5 epochs)

Input Embedding	**Target Embedding**	**BLEU**
32	32	90.45
64	64	94.9
128	128	98.2
256	**256**	**98.51**
512	512	97.36

A la vista de los resultados expuestos en la tabla 1 los mejores resultados se obtienen con un tamaño de embedding de 256. Además se observa una disminución del BLEU con un tamaño de Embedding de 512, lo que puede deberse a que con el reducido tamaño del conjunto de entrenamiento no se pueden entrenar unos embeddings que capten con una buena calidad las relaciones sintácticas y semánticas entre los distintos términos.

2.2 Ejercicio 2

Una vez fijados los tamaños de embedding input y output a 256 se ha realizado una exploración por distintos tamaños para el modelo de LSTM.

Table 2: Evolución del BLEU al modificar el tamaño de las capas LSTM (5 epochs)

Input LSTM	**Target LSTM**	**BLEU**
32	32	96.47
64	**64**	**98.51**
128	128	95.31

A la vista de los resultados en la tabla 2 se observa que el mejor valor se obtiene con 64 unidades LSTM. Un incremento en el número de unidades supone una disminución en el BLEU probablemente debido a que no existen suficientes datos como para realizar una buena estimación de los parámetros y no caer en un sobreajuste.

Una vez obtenidos los mejores parámetros para el modelo (tamaño embedding 256 + 64 unidades LSTM) se ha realizado un entrenamiento hasta la convergencia, esto es, hasta que el BLEU en entrenamiento es de 100%. Tras 27 epochs el algoritmo ha convergido y al realizar una evaluación sobre el conjunto de test se ha obtenido un BLEU de **99.87**

2.3 Ejercicio 3

Para la realización de este ejercicio se han probado otros metodos de aprendizaje como Adagrad o Adadelt, ambos evaluados con 5 epochs. La convergencia de ambos algoritmos es mucho más lenta que la convergencia obtenida por ADAM y por tanto los BLEU obtenidos en ambos casos son reducidos **12.82** y **1** respectivamente.

2.4 Ejercicio 4

Para la utilización del modelo basado Transformer es necesario que el tamaño del modelo y los tamaños de los embeddings coincidan. A continuación se muestra el estudio de tamaños realizado para este tipo de arquitectura.

Table 3: Evolución del BLEU al variar el tamaño del modelo Transformer (5 epochs)

Tamaño del modelo	**BLEU**
32	64.22
64	87.21
128	**92.59**
256	88.22

A la vista de los resultados expuestos en la tabla 3 se observa que el modelo basado en transformer y con un tamaño de 128 obtiene el mejor resultado. Cabe destacar que los resultados obtenidos por el transformer son inferiores, para esta tarea, al mejor modelo de redes neuronales y basado en la arquitectura encoder-decoder con modelo de atención.