UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Departamento de sistemas informáticos y computación

Traducción basada en redes neuronales recurrentes

Práctica Traducción automática

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor
Francisco Javier Gil-Terrón Rodríguez

2021 - 2022

Tabla de contenidos

1. Introducción

Esta práctica tiene por objetivo crear un sistema de traducción automática a través de nmt-keras, toolkit basado en Tensorflow y Keras. Para ello, se emplearán redes neuronales recurrentes y se partirá de un corpus paralelo en dos lenguajes, que será el mismo que se empleó en la práctica anterior basada en Moses.

Para el desarrollo del trabajo, se ha empleado docker sobre una máquina Windows, configurado para que las instancias de docker pueda hacer uso de la GPU, con una tarjeta gráfica NVIDIA GeForce GTX 1060 de 6GB.

En primer lugar, la primera prueba que se ha desarrollado durante la sesión de la práctica tenía la siguiente configuración: codificador y decodificador (bidireccional en el primer caso) LSTM de 64 neuronas, vector para codificar palabras fuente y destino (embeddings) de 64, learning rate 0.001 y limitado a 5 epochs.

Como se había comentado, los datos que se emplearán son los mismos que se usaron para Moses, que recordemos que ya están tokenizados. Con el script main.py se procede a entrenar el modelo de traducción siguiendo la configuración de config.py.

Una vez se ha generado dicho modelo, se traduce y finalmente se evalúa el BLEU de la misma manera que ocurría en el caso de Moses. Para el caso base con la configuración que se ha citado, se ha obtenido un BLEU de 94.79.

Para las siguientes pruebas, se han mantenido 5 epochs, y cada prueba se ha realizado 3 veces para reducir posible sesgo introducido por la aleatoriedad, siendo la media de estos el resultado que se mostrará.

2. Variación de tamaño embedding

En esta primera prueba se modificarán los tamaños de representación de las palabras fuentes y destino (word embeddings) donde la inicial era de 64. Para ello habrá que modificar las variables, SOURCE_TEXT_EMBEDDING_SIZE y TARGET_TEXT_EMBEDDING_SIZE.

Ya que el tamaño original es de 64, se probará a dividir y multiplicar este valor por 2 para estudiar la variación del BLEU. Los resultados se muestran en la siguiente tabla:

32	64	128	256	512	1024
90.3	94.79	97.54	97.89	97.91	97.03

Tabla 1 – Valores de BLEU en función del tamaño de embedding

Como se puede observar, al reducir el tamaño a 32, el BLEU se ve reducido consistentemente a aproximadamente 90, mientras que, al aumentarlo, asciende para todos los valores que se han probado. De esta manera, se probó a duplicar el tamaño de embedding hasta encontrar un máximo, que en las pruebas realizadas resultó ser tanto 256 como 512, dando ambos resultados muy similares entorno próximos a 98 de BLEU. Por el contrario, al incrementarlo a 1024 de tamaño, comienza a disminuir, posiblemente debido a que el conjunto de entrenamiento es demasiado pequeño para ese valor.

3. Variación tamaño red

En esta sección se evaluará la variación del BLEU en función del tamaño de la red, esto es, la cantidad de neuronas que componen el decodificador y codificador LSTM.

Para ello se modificarán las variables del archive config.py ENCODER_HIDDEN_SIZE y DECODER_HIDDEN_SIZE, siendo el caso original 64 neuronas para ambos, mientras que se fijará el tamaño de embedding a 256 (mejor obtenido en el apartado anterior) para que la variación del BLEU únicamente se produzca por la modificación del tamaño de la red.

32	64	128	256
86.05	98.14	98.25	97.83

Tabla 2 – Valores de BLEU en función del tamaño de la red

Como se puede observar en la tabla anterior y siguiendo el mismo procedimiento que en la variación del tamaño de embedding, se comenzó

probando con un valor más pequeño, pero ya que disminuía considerablemente el BLEU reduciéndolo a 86.05, se probó a incrementarlo. Así pues, se han obtenido mejores resultados al aumentar el tamaño de la red, alcanzando el máximo en 128 con un BLEU de 98.25, y se dejaron de hacer pruebas al comprobar que con 256 de tamaño de la red el BLEU volvía a reducirse por debajo de 98, probablemente debido a lo mismo que ocurría en el caso anterior, a que no hay suficientes datos para que funcione correctamente.

4. Otros algoritmos de aprendizaje

A continuación, se estimará la variación del BLEU al cambiar el algoritmo de aprendizaje empleado u optimizador. En estas pruebas se han utilizado los mejores valores de tamaño de embedding y red obtenidos, estos son 256 y 128 respectivamente.

Para ello se debe cambiar el parámetro OPTIMIZER, que viene de serie con el algoritmo Adam. De entre las opciones que acepta se han probado Adagrad y Adadelta, cuyos resultados se pueden ver en la siguiente tabla:

Adam	**Adagrad**	**Adadelta**
98.25	12.3	91

Tabla 3 – Valores de BLEU en función del algoritmo de aprendizaje

Tal como se puede ver en la tabla, los mejores resultados que se han alcanzado eran con el algoritmo propuesto originalmente: Adam. No obstante, cabe destacar que para el caso de Adagrad, no se encontró una configuración óptima que lograra un BLEU del mismo orden que el resto de los algoritmos probados, aunque puede deberse a la limitación de 5 epochs y que los resultados sean tan bajos debido a que su convergencia es mucho más lenta. Por el contrario, en el caso de Adadelta, bastó con modificar el factor de aprendizaje o learning rate a 1.0 para que su ejecución fuera óptima, logrando un BLEU de 91, pero que como se había comentado, sigue siendo inferior a Adam.

5. Transformer

Finalmente, en esta última sección, se modificará la estructura de la red inicialmente propuesta por un transformer, donde cabe estacar que para que funcione correctamente el tamaño de modelo y de embedding debe ser el mismo.

Una vez más, las pruebas que se han llevado a cabo han sido limitadas a 5 epochs y la variable a modificar es MODEL_TYPE en este caso. Los diferentes tamaños de modelo que se han probado se muestran en la tabla siguiente:

32	**64**	**128**	**256**
63.68	87.81	92.4	88.91

Tabla 4 – Valores de BLEU en función del tamaño del modelo

En este caso se ha seguido la misma lógica que en el estudio de la variación de tamaño de embedding y red: en primer lugar, reducir el tamaño (el original era de 64) e ir incrementando hasta alcanzar un máximo. De esta manera, se puede ver como con un tamaño de 32 se obtiene el peor BLEU con un valor de 63.68, que se incrementa considerablemente hasta 87.81 duplicando el tamaño del modelo. Si seguimos incrementándolo, en 128 se ha obtenido el máximo de este tipo de red con un valor de BLEU de 92.4, y se ha observado que al duplicarlo de nuevo el BLEU volvía a descender por debajo de 90.

A la vista de estos resultados se puede concluir con que la estructura de redes neuronales basada en encoder decoder con modelo de atención ofrecía mejores resultados para esta tarea, ya que ninguno de los tamaños de transformer propuestos alcanza los valores anteriormente mostrados.

6. Bibliografía

Casacuberta Francisco, & Domingo Miguel. (2021). *Boletín de la práctica*. Universitat Politècnica de València. https://www.prhlt.upv.es/~fcn/Students/ta/p1mt.pdf