# Traducción Automática

Traducción automática basada en redes neuronales

Victoria Beltrán Domínguez

Enero 2022

# Índice



# 1. Introducción

En esta práctica nuestro objetivo es experimentar con redes neuronales (y particularmente con el toolkit nmt-keras basadoo en Tensorflow y Keras [1]) para construir sistemas de traducción a partir de conjuntos de pares de frases bilingües.

Para ello, lo primero que se hará es seguir la guía detallada en el marco de las prácticas. Seguidamente, se procederá a realizar los ejercicios propuestos, analizando los resultados obtenidos.

## 2. Descripción del trabajo realizado en la sesión

El trabajo realizado en prácticas con el fin de construir un sistema de traducción basado en redes neuronales ha seguido una serie de pasos:

1. Puesto que particularmente hemos trabajado en *PoliLabs*, lo primero que hemos tenido que hacer ha sido definir todas las variables de entorno especificadas en la práctica para acceder al toolkit nmt-keras.
2. Una vez preparado el toolkit, hemos copiado los datos de la práctica anterior en el directorio *Data/EuTrans*.
3. Ya preparados el toolkit y los datos, debemos entrenar un modelo. Esto se ha conseguido utilizando el *main.py* del toolkit.
4. Para facilitar el acceso a los modelos entrenados, se ha creado un link simbolico.
5. A través del comando especificado en la práctica para generar traducciones utilizando el modelo, volcamos los resultados en un fichero.
6. Contrastamos los resultados obtenidos, obteniendo así la puntuación *BLEU* correspondiente. En este experimento, hemos obtenido una puntuación *BLEU* de 93.7.
7. Finalmente, se indica que si copiamos localmente el archivo de configuración, podemos ajustar los diferentes parámetros del modelo.

# 3. Resolución ejercicios propuestos

## 3.1. Ejercicio 1

| 1. Probar otros tamaños de representación de las palabras fuentes y destino (word embeddings) |
|---|

Podemos visualizar los resultados de la experimentación con tamaños de embeddings de 32, 64, 128, 256, 512 y 1024 en la siguiente tabla:

| **Tamaño palabra fuente** | **Tamaño palabra destino** | **BLEU** |
|---|---|---|
| 32 | 32 | 93.8 |
| 32 | 64 | 92.33 |
| 64 | 32 | 93.75 |
| 64 | 64 | 93.71 |
| 128 | 128 | 94.78 |
| 256 | 256 | 93.8 |
| 512 | 512 | 94.95 |
| 1024 | 1024 | 93.67 |

En esta podemos observar que los tamaños óptimos para estos están entre 128 y 512, pero podemos desmentir que cuando más grande el tamaño del embedding mayor será la puntuación *BLEU* porque el tamaño 1024 obtiene peores resultados.

## 3.2. Ejercicio 2

| 2. Probar otros tamaños de red recurrente para el codificador y decodificador |
|---|

Experimentando con los tamaños de red recurrente, se observa que el mejor resultados se obtiene con 128, aunque 64 y 256 también obtienen puntuaciones *BLEU* cercanas.

| **Tamaño codificador** | **Tamaño decodificador** | **BLEU** |
|---|---|---|
| 32 | 32 | 93.18 |
| 32 | 64 | 92.89 |
| 64 | 32 | 93.33 |
| 64 | 64 | 93.71 |
| 128 | 128 | 94.87 |
| 256 | 256 | 94.75 |
| 512 | 512 | 93.33 |
| 1024 | 1024 | 92.63 |

## 3.3. Ejercicio 3

| 3. Probar otros algoritmos de aprendizaje: Adagrad y Adadelta |
|---|

Los resultados de la experimentación se muestran a continuación:

| Algoritmo aprendizaje | BLEU |
|---|---|
| Adam | 93.71 |
| Adagrad | 93.99 |
| Adadelta | 92.5 |

Para este problema, *Adagrad* obtiene mayor puntuación *BLEU*, aunque bien podría ser un resultado un tanto arbitrario debido o bien a los pocos epochs con los que estamos entrenando o bien a la cantidad de datos.

### 3.4. Ejercicio 4

| 4. Probar Transformer |
|---|

Con el modelo de atención, previamente hemos obtenido una puntuación *BLEU* de 93.71. Probando ahora con transformers, hemos obtenido una puntuación *BLEU* de 93.49. Nuevamente, y dado el poco margen que las separa, no podemos sacar conclusiones acerca de cuál es mejor.

## 4. Conclusiones

En esta práctica hemos aprendido a construir un sistema de traducción automática basado en redes neuronales. También hemos experimentado con los parámetros del modelo. Considero, que las conclusiones más destacables podrían ser que un tamaño de word embedding de entre 128 y 512 es el óptimo y que los tamaños óptimos del codificador y decodificador de la red recurrente podrían variar entre 64 y 256.

No se han podido sacar conclusiones claras ni acerca del mejor algoritmo de aprendizaje ni acerca del uso de modelos de atención o transformers, pues los resultados obtenidos son muy cercanos y pueden deberse a un factor arbitrario debido al número de epochs o a la cantidad de datos.

# 5. Bibliografía

[1] Álvaro Peris y Francisco Casacuberta. «NMT-Keras: a Very Flexible Toolkit with a Focus on Interactive NMT and Online Learning». En: *The Prague Bulletin of Mathematical Linguistics* 111 (2018), págs. 113-124. ISSN: 0032-6585. DOI: 10.2478/pralin-2018-0010. URL: https://ufal.mff.cuni.cz/pbml/111/art-peris-casacuberta.pdf.