UNIVERSITAT POLITÈCNICA DE VALÈNCIA

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

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital
Universitat Politècnica de València

Traducción estadística y neuronal para el corpus Europarl

Traducción Automática

Autor: Juan Antonio López Ramírez

Curso 2019-2020

Índice general

Índice general 1

1 Introducción 3

2 Ejercicio Básico 5

2.1 Preparación de los datos 5

2.2 Entrenamiento de los modelos de lenguaje de salida (español) 5

2.3 Entrenamiento del modelo de traducción 5

2.4 Ajuste de los pesos del modelo log-lineal 6

2.5 Proceso de traducción y evaluación 6

3 Ejercicio Avanzado 7

3.1 Traductor con post-edición automática 7

3.2 Traductor estadístico con la herramienta Thot 7

3.3 Traductor neuronal con openNMT 8

4 Conclusiones 9

CAPÍTULO 1
Introducción

El objetivo de este trabajo es realizar los dos ejercicios propuestos en el boletín de la página https://www.prhlt.upv.es/~fcn/Students/ta/ejercicios.html, es decir, construir sistemas de traducción del inglés al español a partir de un corpus que es un subconjunto del europarl-v7.es-en.

Para entrenar modelos estadísticos de traducción de textos desde un idioma de entrada a otro de salida, hemos usado la herramienta **Thot** y, sobre todo, **Moses**. Además, para realizar el último apartado del ejercicio avanzado, hemos empleado **openNMT** para construir un traductor neuronal.

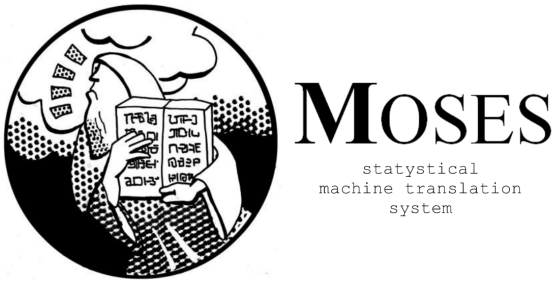

Figura 1.1: Herramienta Moses para traductores estadísticos.

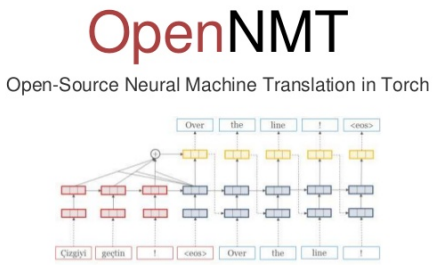

Figura 1.2: Herramienta openNMT para traductores neuronales.

Figura 1.3: Herramienta Thot para traductores estadísticos.

CAPÍTULO 2
Ejercicio Básico

En este primer ejercicio hemos construido un traductor con Moses a partir del corpus del Europarl que se nos proporcionaba.

2.1 Preparación de los datos

Primeramente, hemos preparado los datos de forma que, de las 50000 frases de entrenamiento, hemos separado 3000 para el ajuste de los pesos del modelo log-lineal. De esta forma, tenemos:

1. 47000 frases de entrenamiento en los ficheros que hemos renombrado como *training.es* y *training.en*. Estos han sido tokenizados, limpiados y guardados en *training.clean.tok.es* y *training.clean.tok.en*, respectivamente.
2. 3000 frases para el ajuste de pesos del modelo log-lineal que hemos guardado en los ficheros *development.es* y *development.en*. Igual que los anteriores, estos también han sido tokenizados y limpiados, por lo que se han guardado en *development.clean.tok.es* y *development.clean.tok.en*, respectivamente.
3. 1000 frases de test, limpias y tokenizadas, en los archivos *test.clean.tok.es* y *test.clean.tok.en*. Alternativamente, tenemos el fichero de test en inglés que debíamos enviar el 15 de enero al que hemos llamado *test.tok.clean.en*.

2.2 Entrenamiento de los modelos de lenguaje de salida (español)

Una vez tenemos esto, hemos procedido a crear nuestro traductor. En primer lugar, hemos creado y entrenado un modelo de lenguaje utilizando el software SRILM y el *training.clean.tok.es*. Este modelo está basado en cuatrigramas, con suavizado por interpolación y descuento Kneser-Ney. Esto se debe a que, con esta configuración, se conseguía el mejor resultado en los ejercicios de la práctica de Moses. Dicho modelo se ha guardado en el directorio *lm*, tal y como hacíamos en la práctica.

2.3 Entrenamiento del modelo de traducción

Posteriormente, hemos construido la tabla de segmentos (*phrases*) y los modelos de reordenamiento con los datos de entrenamiento obtenidos anteriormente: "training.clean.tok.en"

y "training.clean.tok.es". Para construir la tabla de segmentos, primero hay que alinear a nivel de palabra las dos partes del conjunto de entrenamiento y para ello hemos usado el software GIZA++. Los modelos de traducción se han guardado en el directorio *work/model/*.

2.4 Ajuste de los pesos del modelo log-lineal

A continuación, empleando las 3000 frases de los ficheros *development.clean.tok*, hemos obtenido los pesos del modelo log-lineal mediante Mert, ya que los experimentos prácticos proporcionaban mejores resultados que cuando utilizábamos MIRA. En cuanto al número de iteraciones, se ha tenido en cuenta el coste temporal, de modo que aunque con 7 iteraciones obteníamos el mejor resultado, el proceso de ajuste habría tardado demasiado tiempo, por lo que hemos fijado el número de iteraciones máximas a 5, cuyo resultado tampoco distaba mucho del máximo. El modelo ha sido guardado en la carpeta *mert-work*.

2.5 Proceso de traducción y evaluación

Aquí hay que tener en cuenta que se han seguido dos caminos. Por un lado, hemos evaluado el traductor con el conjunto de test que se proporcionaba, es decir, con *test.clean.tok.en*. Por otro lado, hemos traducido las frases de *test.tok.clean.en* para la tarea del 15 de enero.

Para el primer caso, las traducciones resultantes han sido guardadas en *test.hyp* y la evaluación obtenida ha sido un BLEU de **28.15**, un resultado que supera el 25.5 que se sugería en el enunciado del boletín.

En el otro caso, las traducciones resultantes han sido guardadas en *LopezRamirezJuanAntonio.hyp*, fichero que se envío como solución al ejercicio básico de este trabajo, y cuya la evaluación obtenida fue un BLEU de **28.29**, un resultado que, al igual que el anterior, supera tanto el 25.5 que se sugería en el enunciado del boletín como el 28.15 que se obtenía con el conjunto de prueba que se nos proporcionaba.

CAPÍTULO 3
Ejercicio Avanzado

3.1 Traductor con post-edición automática

En esta tarea se nos pide construir un traductor que realice una post-edición automática utilizando el corpus intermedio que se nos proporciona.

Primero, hemos traducido *europarl-v7.es-en-train-red-PE.en* con el traductor del ejercicio básico. El resultado lo hemos guardado en *europarlPE.hyp*.

Luego, hemos construido un traductor con Moses tomando como entradas las traducciones obtenidas de *europarl-v7.es-en-train-red-PE.en*, es decir, el fichero *europarlPE.hyp*; y como salidas *europarl-v7.es-en-train-red-PE.es*.

Para ello, hemos creado, primero, el modelo de lenguaje con la misma configuración que el del ejercicio básico (cuatrigramas, Kneser-Ney, interpolación...); entrenamos el modelo de traducción con el *europarlPE.hyp*; ajustamos los pesos del modelo log-lineal con el archivo *dev.ejeravanzado.origen*, que contiene las 1000 primeras frases del *europarl-v7.es-en-train-red-PE.es* y con un número máximo de 5 iteraciones mediante Mert.

Finalmente, hemos traducido el conjunto de test del ejercicio básico y el BLEU obtenido ha sido de **29.75**, superando el 28.15 que se conseguía sin post-edición automática. Por lo tanto, podemos concluir que el proceso de post-edición ha sido satisfactorio para mejorar los resultados obtenidos por el traductor del ejercicio básico.

3.2 Traductor estadístico con la herramienta Thot

El segundo apartado del ejercicio avanzado consistía en realizar un experimento parecido al hecho en el ejercicio básico, pero usando el toolkit Thot en lugar de Moses. Así, una vez se disponga de los resultados de ambas herramientas, se puede hacer una comparación del BLEU que proporciona cada una de ellas.

El proceso realizado con Thot es similar al seguido con Moses. Primero, se hace un procesado del corpus, tokenizándolo para pasar todo a minúscula y limpiando frases que resultan muy extensas.

Cuando el corpus ya está preparado, se realiza el proceso de entrenamiento de los modelos y el ajuste de parámetros. Primero, se ha entrenado el modelo de lenguaje del idioma de salida (español). Para hacer esto, Thot nos ofrece un conjunto de herramientas con las cuales, partiendo del corpus limpio, podemos entrenar un modelo de lenguaje en español. En nuestro caso, igual que en el ejercicio básico, hemos entrenado un modelo de lenguaje basado en cuatrigramas, con suavizado y descuento por Kneser-Ney e interpo-

lación. Esta es la estructura que se ha seguido debido a que, con ella, en los experimentos prácticos obteníamos los mejores resultados.

A continuación, gracias a las herramientas de Thot que nos permiten conseguir modelos estadísticos, hemos entrenado un modelo de traducción de inglés a español. Después, mediante el método *Nelder-Mead* proporcionado por Thot, se ha realizado el ajuste de pesos. Los datos empleados para estos dos pasos han sido las 3000 primeras frases del conjunto de entrenamiento, separadas de las 47000 frases utilizadas en training.

Por último, hemos realizado un filtrado del modelo de frases para desechar información de poco interés que puede contener nuestro modelo. Esto ha sido posible gracias a que podemos filtrar aquellos parámetros que sean relevantes para llevar a cabo la traducción.

Cuando ya disponemos de nuestro modelo entrenado, ajustado y filtrado, realizamos la traducción y evaluamos, obteniendo un BLEU de **26.45**. Comparando este resultado con el BLEU obtenido con Moses, que era de 28.15, podemos afirmar que, aunque se consigue superar de forma holgada la barrera del 25.5 que se sugería en el boletín del trabajo, queda lejos de la solución que nos proporcionaba Moses; por lo que Thot es una alternativa viable para alcanzar los requisitos mínimos que se nos plantean pero no es superior al trabajo original.

3.3 Traductor neuronal con openNMT

En cuanto al último experimento, este se ha fundamentado en el entrenamiento de modelos de traducción basados en redes neuronales. Para ello, se ha hecho uso de la herramienta openNMT, como habíamos comentado al principio de este trabajo. Este toolkit permite trabajar con redes neuronales con comodidad, debido a que podemos configurar una gran cantidad de parámetros de nuestra red de forma fácil.

Sin embargo, al realizar este experimento en nuestro ordenador local (en lugar de en el escritorio remoto como hicimos con la sesión de prácticas), hemos empleado los recursos de nuestro portátil, en específico la GPU NVIDIA GeForce GTX 1050 Ti, con la que se han podido acelerar las ejecuciones empleadas para realizar el ejercicio.

Para nuestro modelo hemos seguido los pasos que nos llevaron a conseguir el mejor BLEU en las prácticas, esto es, utilizando la siguiente configuración:

- Modelo Encoder-Decoder.
- 64 unidades LSTM.
- Tamaño de embedding de 256.
- Algoritmo de aprendizaje ADAM.

Además, se ha añadido *learning rate annealing* para ir reduciendo el valor del factor de aprendizaje, de manera que este comienza siendo de 0.01, hasta que llega a la mitad de epochs, y entonces pasa a valer 0.001.

El resultado obtenido ha sido un BLEU de **29.81**, es decir, el mejor BLEU conseguido de todos los experimentos realizados.

CAPÍTULO 4
Conclusiones

La realización de este trabajo, desde un punto de vista experimental, ha resultado bastante satisfactoria y útil para poder evaluar distintos métodos de traducción automática. Además, el hecho de trabajar con conjuntos de datos de un tamaño considerable permite apreciar el complejo procedimiento de la traducción, sobre todo desde un punto de vista temporal, ya que estos procesos suelen ser costosos, por lo que es realmente importante y útil saber cuál es la mejor configuración para nuestro traductor, tanto desde un punto de vista teórico (conocimiento de los modelos de lenguaje, saber cuántas iteraciones son las justas y necesarias para según que proceso de entrenamiento...) como práctico, para este último han servido las sesiones y experimentos del laboratorio.

Finalmente, recalcar que los resultados que se han obtenido (que pueden observarse en la figura 4.1) reflejan un panorama muy positivo para las traducciones basadas en redes neuronales, que supera, al menos en los experimentos que se han realizado en este trabajo, a los traductores estadísticos.

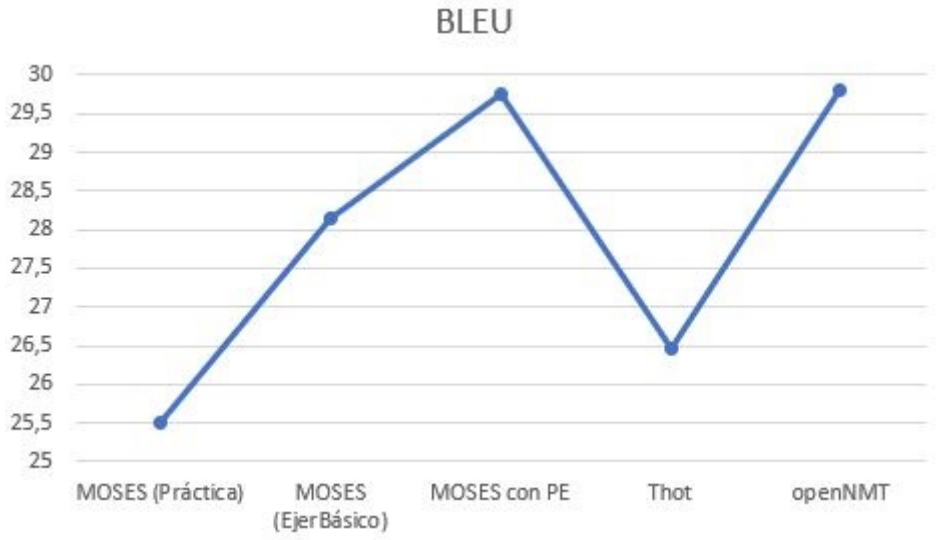

Figura 4.1: Gráfica que representa los resultados (BLEU) obtenidos con los traductores estadísticos y el traductor neuronal.