Traducción Automática
SMT and NMT

Victoria Beltrán Domínguez

Enero 2022

Contents

1 Introducción

Este trabajo plantea construir el mejor traductor del inglés al español partiendo de un subconjunto del *Europarl-v7* que consiste en 35000 frases de entrenamiento y 1000 de evaluación. Para ello, deberemos utilizar tanto *Moses*[1] como redes neuronales recurrentes de NMT-keras[2].

Así, esta memoria se desglosa en diferentes partes: en la primera parte, nos centraremos en los datos y en cómo se han preprocesado. A continuación, se hablará de cómo se ha consturido el modelo para Moses y se mostrarán algunos resultados obtenidos. Seguidamente se presentará de qué manera hemos desarrollado el modelo de traducción con NMT-keras, destacando los resultados obtenidos en la experimentación. También se mostrarán los experimentos realizados en OpenNMT. Finalmente, se destacarán las lecciones aprendidas durante todo el trabajo y algunas conclusiones.

Antes de empezar, cabe destacar que muchos de los experimentos se han tenido que realizar de manera secuencial en polilabs (que muchas veces se desconectaba de repente), por lo que no se ha podido experimentar de manera tan amplia como habría gustado.

2 Dataset

Disponemos de cuatro ficheros, divididos en idiomas y en conjuntos de entrenamiento y test:

1. europarl-v7.es-en-train-red.en:

```
Resumption of the session
I declare resumed the session of the European Parliament  adjourned...
```

2. europarl-v7.es-en-train-red.es

```
Reanudación del período de sesiones
Declaro reanudado el período de sesiones del Parlamento Europeo, interrumpido...
```

3. europarl-v7.es-en-test-en

```
The Committee on Economic Affairs takes the view that quantitative economic targets may be us
```

4. europarl-v7.es-en-test-es

```
La Comisión de Asuntos Económicos adopta el punto de vista de que los objetivos económicos cu
```

Lo primero que debemos hacer es tokenizar los cuatro ficheros, utilizando el script `tokenizer.perl`. A continuación, debemos limpiar los ficheros de entrenamiento con el script `clean-corpus-n.perl`. De esta manera, todas las frases con más de 60 palabras han sido quitadas del corpus. Así, pasamos de 35000 frases de entrenamiento a 32350. Además, vamos a utilizar las últimas 1500 líneas del conjunto de entrenamiento para validación, por lo que pasamos a tener 30850 datos de entrenamiento.

A causa de que no disponemos de recursos muy potentes, además se ha creado un conjunto de datos más pequeño al que llamaremos Minieuroparl. Este se utilizará para realizar los diferentes experimentos, y encontrar qué parámetros nos ayudan a conseguir el mejor BLEU. Este dispondrá de 10000 muestras de entrenamiento, 500 de dev y 1000 de test.

Podemos encontrar un resumen de los dos corpus en la Tabla 2.

	Europarl	Minieuroparl
Train	30K	10K
Test	1K	1K
Val	1.5K	0.5K

3 Descripción del trabajo realizado y resultados obtenidos

En esta sección se procede a realizar todos los experimentos que se han realizado para cada ejercicio, así como a exponer los resultados obtenidos en estos:

3.1 Moses

En esta sección se procede a exponer todos los experimentos realizados con el fin de obtener el mejor modelo estadístico. Para sacar un valor de referencia, se ha entrenado el modelo con el corpus *Europarl* (todos los datos) con todos los parámetros por defecto y 5 epochs. Esto nos ha dado un BLEU de **26.65**.

El primer parámetro con el que se ha experimentado ha sido el orden de los n-gramas del modelo de lenguaje entrenado con SRILM [3]. Para esto, hemos entrenado modelos con diferentes n-gramas sobre el Minieuroparl utilizando 10 epochs. Los resultados de los experimentos son mostrados en siguiente tabla:

N-gramas	BLEU
2	21.63
3	22.89
4	**23.10**
5	22.68
6	21.96

De esta manera, a partir de ahora utilizaremos modelos de n-gramas de orden 4. Además, también hemos experimentado con el método de suavizado e interpolación óptimos.

4-gramas	Backoff	Interpolation
Original Kneser-Ney	**22.94**	22.87
Modified Kneser-Ney	22.24	22.72
Wittenbell	22.12	22.14

Vemos que, repitiendo los experimentos con diferentes suavizados e interpolaciones, utilizar backoff junto con kneser-ney original es la manera que consigue un BLEU mayor. Sin embargo, el resto no se queda especialmente lejos, así que podría ser que en otra ejecución obtuvieramos resultados completamente diferentes. Teniendo eso en mente y por simplicidad, vamos a utilizar backoff con descuento original Kneser-Ney.

Finalmente, hemos probado con algunos de los parámetros especificados para el modelo de alineamiento entrenado a partir de GIZA[4] en el manual de Moses.

Heurística utilizada para el alineamiento de palabras	BLEU
intersect	22.37
union	21.15
grow	22.41
grow-final	22.31
grow-diag	22.63
grow-diag-final	22.69
grow-diag-final-and	**23.07**
srctotgt	21.95
tgttosrc	22.34

Vemos que el modelo de alineamiento que obtiene mejor resultado es el que ya había por defecto. Antes de seguir y sólo por asegurar, se ha repetido el experimento con interpolación, obteniendo un BLEU de 22.75 que no consigue superar al backoff. También se ha experimentado con la configuración del modelo de reordenamiento, obteniendo nuevamente, la configuración que ya teníamos como la ideal, tal y como se muestra en la siguiente tabla:

Configuración del modelo de reordenamiento	BLEU
msd-bidirectional-fe	**23.07**
phrase-msd-bidirectional-fe	22.79
hier-mslr-bidirectional-fe	20.87
monotonicity-bidirectional-fe	22.57

Finalmente, se realiza una última prueba acerca de si mira puede llegar a mejorar el modelo. Entrenando nuevamente el modelo, obtenemos un BLEU de 22.89, por lo que decidimos no utilizar MIRA en el entrenamiento de nuestro modelo final.

Una vez encontrados los parámetros óptimos para este problema, y entrenando con 15 epochs todos los datos, obtenemos un BLEU de **27.45**, superando así nuestro BLEU inicial de 26.65. Aún se podría mejorar más este modelo dejándolo unos cuantos epochs más, pero los recursos y el tiempo del que disponemos, desafortunadamente no nos permiten un experimento más largo.

3.2 NMT-keras

A continuación, pasamos a experimentar con redes neuronales para traducción automática. Para ello, con el fin de sacar valores de referencia, se ha entrenado el modelo con la configuración por defecto y todos los datos, obteniendo un BLEU de 18.59. Por otro lado, entrenando con el conjunto Minieuroparl y todos los parámetros por defecto, obtenemos un BLEU de 7.84.

Así, a continuación pasamos a buscar los parámetros ideales para posteriormente entrenar el modelo final. El primer experimento realizado ha consistido en ver el impacto del tamaño de los embeddings en el BLEU. Cabe destacar que sobretodo se han probado aquellos que dieron mejor en la práctica para poder ahorrar tiempo. Los resultados se muestran en la siguiente tabla:

Tamaño embeddings fuente	Tamaño embeddings destino	BLEU
128	128	9.71
256	256	10.55
512	512	**10.73**

Así, a partir de ahora vamos a utilizar un tamaño de 512-512. También se ha experimentado con los diferentes compiladores, obteniendo que Adam con 0.001 es el mejor, tal como se muestra en la siguiente tabla:

Optimizadores	Learning Rate	BLEU
Adagrad	0.01	1.65
Adam	0.001	**10.73**
Adam	0.0002	6.20
Adadelta	1.0	4.42
SGD	0.1	2.91

Siguiendo con la dinámica, se ha experimentado también con el tamaño de la capa oculta del encoder y del decoder.

Encoder hidden size	Decoder hidden size	BLEU
64	64	10.73
128	128	11.77
256	256	13.56
512	512	**14.11**
1024	1024	11.19

Como vemos, el tamaño que nos da mejor BLEU es de 512 y por lo tanto es el que se va a utilizar para entrenar el modelo final. De esta manera, entrenando con quince epochs con todo el conjunto de datos obtenemos un BLEU de 21.17, mejorandolo así en 2.58.

3.3 Opcional: Transformer NMT-keras

A continuación, pasamos a experimentar utilizando la arquitectura del Transformer de NMT-Keras. Dado el alcance del proyecto y los recursos disponibles, solamente se ha entrenado 5 epochs los diferentes experimentos. Dicho esto, como hemos hecho en experimentos anteriores, entrenando 5 epochs con todos los datos obtenemos un BLEU de 7.15, valor que utilizaremos posteriormente para cuantificar la mejora. Pasando a los experimentos realizados con los datos de Minieuroparl, lo primero que hemos hecho ha sido experimentar con los tamaños de los embeddings fuente y destino. Podemos ver los resultados en la siguiente tabla:

Tamaño embeddings fuente	**Tamaño embeddings destino**	**BLEU**
64	64	1.69
128	128	**2.98**
256	256	2.52
512	512	0.53

Como podemos ver, el mejor resultado se obtiene al aplicar un tamaño de 128 a los embeddings. Con este resultado, se ha decidido alterar también el número de cabezas, que por defecto tenía un valor de 8. De esta manera, obtenemos que:

Número de cabezas	**BLEU**
2	2.41
4	**2.83**
16	2.73

Ninguno de estos valores es superior al obtenido con 8 cabezas, por lo que se ha dejado este valor al previamente utilizado. Esta es toda la experimentación que se ha realizado con esta arquitectura. De este modo, entrenando con todos los datos durante 5 epochs obtenemos un BLEU de 9.86, superando aproximadamente en 2 puntos a nuestro modelo inicial cuyo BLEU era 7.15.

3.4 Opcional: openNMT

Para llevar a cabo esta tarea, primero hemos tenido que investigar cómo utilizarla. Al revisar la documentación, se ha decidido utilizar OpenNMT-tf en nuestra máquina en local, dada la experiencia con el lenguaje en otras asignaturas. Sin embargo, hacer traducciones era realmente lento, llegando a tardar 5 horas o más en el corpus pequeño y los resultados obtenidos eran bastante malos (se repetían muchísimas frases al traducir, por ejemplo *El parlamente del parlamento del parlamento....* Ante la incertidumbre de que el experimento fuese incorrecto y dado que se encontró dentro del foro de OpenNMT la posibilidad de utilizar OpenNMT-py desde Google Collab, hemos decidido utilizarla al ser una opción mucho más flexible y cómoda de ejecutar.

Así, hemos subido el corpus a Collab, hemos realizado los cambios necesarios para utilizar nuestro Minieuroparl y hemos cambiado un fichero de configuración para entrenar nuestros modelos. Como se ha hecho en el resto de apartados, hemos ejecutado el modelo por defecto durante 10000 *train steps*, obteniendo un BLEU de 18.91. Una vez conseguido el valor como referencia, pasamos a la parte experimental con el Minieuroparl. Hemos entrenado durante 5000 *train steps*, realizando 500 *dev steps* y con un *batch size* de 256 a partir del modelo que utiliza OpenNMT-py por defecto. Los resultados han sido:

Tamaño embeddings fuente	**Tamaño embeddings destino**	**BLEU**
64	64	8.76
128	128	**8.90**
256	256	8.32
500	500	7.41
512	512	7.86

Como podemos observar, nuestro mejor resultado para este modelo por defecto basado en RNNs se obtiene con un tamaño de embeddings de 128. De este modo, entrenando durante 10000 *train steps* y con el mismo *batch size*, el resultado del modelo final es de 20.54 de BLEU. Este modelo consigue superar al modelo inicial, cuyo BLEU era de 18.91.

4 Discusión resultados

En esta sección se pretende comentar los resultados obtenidos previamente.

Poniendo nuestra atención en Moses, se ha experimentado con n-gramas, con el método de suavizado, con las configuraciones del modelo de alineamiento y el de reordenamiento. Para estos, vemos que uno de los experimentos más relevantes ha sido el de los n-gramas de orden 4. Para el resto de experimentos, sólo conseguimos rascar un poco más de BLEU pero no conseguimos mejoras más allá. Además, a causa de que muchos de los resultados son diferenciados en tan poco BLEU y dado el comportamiento indeterminista del entrenamiento del modelo, estamos algo inseguros de haber escogido los parámetros ideales para el entrenamiento del modelo final.

En cuanto a NMT-Keras, ni el modelo de atención recurrente ni el transformer consiguen superar el BLEU obtenido por el modelo estadístico. Asumimos que es debido a la magnitud de los datos, y que si hiciéramos un expermiento con un conjunto de datos aún más grande llegaría un momento en el que sería mejor. Para el modelo de atención recurrente, se ha experimentado con los tamaños de los embeddings fuente y destino, el optimizador, su learning rate inicial y el tamaño de la capa oculta. En cuanto al tamaño de los embeddings, se ha elegido seguir con 512 a pesar de que un tamaño de 256 da un BLEU muy similar y quizás nos ayudaría a generalizar. En cuanto a los optimizadores, se han probado Adagrad, Adam, Adadelta y SGD, siendo Adam aquel que mejores resultados da. Finalmente, para el tamaño de la capa oculta obtenemos que el mejor resultado con un tamaño de 512. Pasando al transformer, obtenemos que el tamaño de los embeddings óptimos son de 128 en comparación a los modelos de atención. También experimentamos con el número de cabezas, aunque ninguno consigue superar el valor por defecto. En openNMT, tras algunas dificultades, conseguimos experimentar con el tamaño de los embeddings, obteniendo que el mejor es de 128.

Finalmente, destacar que en la competición con los datos ocultos, hemos obtenido un BLEU de 27.7 en SMT y un BLEU de 20.9 en NMT.

5 Conclusiones

En este trabajo se ha experimentado con modelos estadísticos y neuronales para construir modelos de traducción automática. En cuanto al modelo estadístico, nos ha sorprendido que 4-gramas funcionase mejor que 3-gramas, pero después de un periodo de reflexión se entiende que en el español podría llegar a funcionar bien (incluso 5-gramas y 6-gramas). Sin embargo, el modelo estadístico tiene un aspecto no determinista que hace que no estemos seguros de que los resultados obtenidos consigan generar el mejor modelo posible.

Por otro lado, jugando con modelos neuronales, podemos observar que el aspecto más relevante de nuestros entrenamientos y lo que más ha afectado al rendimiento final es el tamaño de los embeddings, algo que consideramos que tiene bastante sentido al ser el tamaño de características que van a tener nuestras palabras. Sin embargo, quizás en un ambiente en el que tuviéramos más datos, puede que este aspecto no fuera tan crítico, y otros aspectos como la arquitectura cobrasen más relevancia.

Finalmente, quiero destacar que en los apartados opcionales la experimentación realizada no ha sido muy profunda, pero en todos los ejercicios, se ha conseguido mejorar el modelo que teníamos inicialmente.

6 Bibliografía

[1] Philipp Koehn et al. "Moses: Open Source Toolkit for Statistical Machine Translation." In: *ACL*. Ed. by John A. Carroll, Antal van den Bosch, and Annie Zaenen. The Association for Computational Linguistics, 2007.

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

[3] Andreas Stolcke. "SRILM – An extensible language modeling toolkit". In: *IN PROCEEDINGS OF THE 7TH INTERNATIONAL CONFERENCE ON SPOKEN LANGUAGE PROCESSING (ICSLP 2002*. 2002, pp. 901–904.

[4] Franz Josef Och and Hermann Ney. "A Systematic Comparison of Various Statistical Alignment Models". In: *Computational Linguistics* 29.1 (2003), pp. 19–51.