MEMORIA DEL TRABAJO FINAL

Traducción Automática

Autora
Aitana Menárguez Box
Febrero 2024

Índice

1 Introducción

El objetivo de esta memoria es reflejar el trabajo realizado para la asignatura de Traducción Automática. En este caso se ha experimentado con el entrenamiento de modelos de traducción automática para traducciones del inglés al español. En concreto, se han probado dos enfoques distintos: modelos estadísticos (a partir de Moses[2]) y modelos neuronales (a partir de OpenNMT[3]).

2 Datos utilizados

Para realizar la experimentación y entrenamiento de los modelos, se ha usado el conjunto de datos de *European Parliament Proceedings Parallel Corpus 1996-2011*[1]. Será necesario, antes de utilizar estos datos, procesarlos de forma adecuada.

2.1 Preparación de los datos

La preparación de los datos se basa en realizar, primero, una *tokenización* y, después, una limpieza de los archivos. Este proceso se ha completado de forma automática a partir de la herramienta que se facilita en [1]. El resultado de este procesamiento, a partir de los ficheros originales, serán tres conjuntos de datos: uno para entrenamiento (*train*), otro para valicación (*dev*) y otro para pruebas (*test*).

2.1.1 Limpieza para OpenNMT

Por otro lado, se ha probado el uso de una limpieza alternativa a partir del repositorio *MT-Preparation*[2] que permite, al igual que con la limpieza original, separar en tres el conjunto de datos. Para la evaluación final de los modelos y el envío de esta tarea, se deberán utilizar los datos procesados originales. Esta forma alternativa procesamiento se ha hecho con la finalidad de experimentar otras maneras de entrenar los modelos.

3 Marco experimental

A continuación se detallan las características de los modelos entrenados y las pruebas realizadas con cada uno de ellos a partir de la configuración de las diferentes herramientas proporcionadas, así como los resultados obtenidos en cada caso.

3.1 Moses

Por un lado, se han probado diferentes valores de entrenamiento del modelo de lenguaje: número de n-gramas (2, 3 y 5) y suavizado (*Unmodified Kneser-Ney* y *Witten-Bell*). También, se han variado las iteraciones de entrenamiento para la estimación de los pesos del modelo log-lineal (entre 5 y 20). Además, se ha empleado para los dos modelos con mejores resultados obtenidos el uso de Moses monótono.

[1] Disponible a través de `https://www.statmt.org/europarl/`
[2] Disponible a través de `https://github.com/ymoslem/MT-Preparation/tree/main`

3.2 Open NMT

En los casos en los cuales se ha escogido `transformer` como tipo de *encoder* y *decoder*, y se ha permitido el *encoding* posicional. La arquitectura interna (a nivel de número de capas *feed forward*, etcétera) se ha ido modificando según la experimentación Además, se han inicializado los parámetros con la inicialización `glorot`. Por otro lado, en todos los entrenamientos se ha añadido el parámetro `early_stopping` igual a 4 para detener el entrenamiento si el modelo no mejora durante las cuatro últimas evaluaciones. Además, se ha utilizado el optimizador *Adam* con un `adam_beta1` de 0.9 y un `adam_beta2` de 0.998, así como un *decay method* `noam`, una tasa de aprendizaje de 2.0 y un *dropout* de 0.2. El valor para el nivel de *batch* es de palabras y su tamaño es de 4069.

Para el entrenamiento con los datos procesados con la limpieza alternativa propuesta también se ha seguido la configuración anterior. Esta configuración ha sido basada en la de [5] con algunas modificaciones.

3.2.1 Uso de LSMTs y GRUs

En este caso, se debe cambiar el tipo de *encoder* y *decoder* a `rnn`, para que se trate de una red neuronal recurrente y se elegirá el tipo de RNN según se desee emplear GRU o LSTM. La configuración de estas redes se ha elegido en base a [4].

En primer lugar, se ha usado una red recurrente para el *decoder* y una red recurrente *bidireccional* con dos capas LSTM para el *encoder*, con 1024 nodos cada una de ellas. También se ha usado un tamaño de *word embedding* de 512 y *dropout* de 0.3. El *batch size de 64* a nivel de frases y la tasa de aprendizaje de 1.0, con optimización SGD.

Por otro lado, se ha entrenado una red más pequeña con capas GRU. El *encoder* cuenta con dos capas y el *decoder* una. Se ha utilizado la optimización *Adam* con un *learning rate* inicial de 0.0002.

3.2.2 Uso de modelos preentrenados

Se ha utilizado el modelo Helsinki[3] a partir del *framework* MarianMT para la traducción del inglés al español.

4 Resultados obtenidos

A continuación se muestran los resultados obtenidos en los entrenamientos de los diferentes modelos explicados anteriormente.

[3] Accesible a tavés de `https://huggingface.co/Helsinki-NLP/opus-mt-tc-big-en-es`

4.1 Moses

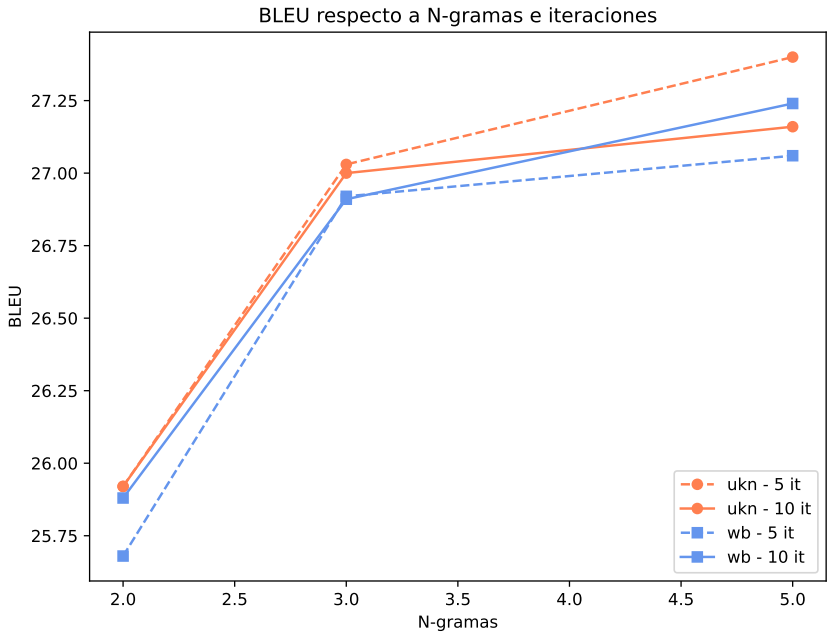

Figura 1: Experimentación con Moses. Evolución del BLEU recpecto del número de n-gramas, suavizado (`ukn` para *Unmodified Kneser-Ney* y `wb` para *Witten-Bell*) e iteraciones.

Se han probado las dos mejores combinaciones de Moses con más iteraciones. En este caso los resultados de BLEU han sido: **27.42** para un modelo de 5-gramas con suavizado *Unmodified Kneser-Ney* y 15 iteraciones y 27.15 para un modelo de 5-gramas con suavizado *Witten-Bell* y 20 iteraciones.

Con los parámetros de los mejores modelos de la Fig. 1 se ha probado Moses monótono y se ha obtenido un BLEU de 26.29 para el de 5 iteraciones y 26.34 para el de 5.

4.2 OpenNMT

Tabla 1: Resultados de la experimentación con **OpenNMT** con la limpieza de datos estándar

Embeddings	**Capas**	**FF-dim**	**Cabezas**	**BLEU**
128	8	64	4	20.2
128	4	64	2	**23.2**
512	6	2048	8	21.5

El BLEU obtenido con el uso de LSTM ha sido **4.1** y con GRU 2.8. Con el modelo preentrenado, se ha obtenido un resultado de 36.7.

Tabla 2: Resultados de la experimentación con **OpenNMT** con la limpieza de datos propuesta

Embeddings	**Capas**	**FF-dim**	**Cabezas**	**BLEU**
128	2	64	2	8.9[a]
128	4	64	2	24.2
512	6	2048	8	**24.9**

[a] Este modelo ha sido entrenado con un tamaño de *batch* igual a 50, no como en el resto de casos donde era de 4096.

5 Conclusiones

Llegados al final de esta memoria, es posible extraer diferentes reflexiones sobre los resultados obtenidos y el trabajo realizado. En primer lugar, se puede apreciar que los modelos estadísticos utilizados han dado mejores resultados en cuanto a rendimiento que los modelos neuronales.

Este dato sugiere que para ciertos conjuntos de datos y tareas, los enfoques estadísticos pueden resultar competitivos en comparación con las técnicas neuronales más recientes. Sobre todo teniendo en cuenta las condiciones experimentales, donde no se ha dispuesto de máquinas con una gran capacidad computacional sobre las cuales realizar experimentación masiva con los parámetros. Esta cuestión de recursos también ha sesgado las posibilidades de exploración. Entrenar modelos neuronales con una amplia variedad de parámetros y arquitecturas puede requerir recursos significativos, como el acceso a unidades de procesamiento gráfico (GPU) de alto rendimiento o servicios de computación en la nube con capacidades de escalado que, en este caso, no han sido accesibles.

Una forma de intentar abordar esta problemática puede ser el uso de modelos preentrenados para tareas específicas. Por un lado, se ha visto en este caso una mejora significativa de la calidad de las traducciones a partir de este tipo de sistemas. Esta observación resalta el potencial de transferencia de conocimiento y la utilidad de estos modelos. Sin embargo, si se quisiera afinar todavía más su precisión, sería necesario *fine tunear* estos modelos y para esta tarea es necesario, también, disponer de una gran capacidad de cómputo

También se puede destacar la sensibilidad de los modelos neuronales a los cambios en su configuración. Los resultados sugieren que cambios, a priori insignificantes, pueden tener un gran impacto en su rendimiento final y por ello requieren de un ajuste cuidadoso.

Por otro lado, la preparación y limpieza de los datos de entrenamiento se han identificado como aspectos críticos en el proceso. La calidad y la consistencia de los datos de entrada pueden influir significativamente en el rendimiento del modelo final, así como su procesamiento. Se debe prestar especial atención a la *tokenización*, la normalización y otros procesos de limpieza para garantizar la coherencia y la precisión de los resultados.

Como trabajo futuro sería una buena idea explorar el uso de técnicas de aumento de datos y la aplicación de enfoques de aprendizaje semi-supervisado para mejorar el rendimiento del modelo, especialmente en escenarios donde los conjuntos de datos disponibles son limitados.

References

[1] Miguel Domingo. *dockerfiles: Personalized Dockerfiles for different tasks*. `https://github.com/midobal/dockerfiles/tree/master`.

[2] *Moses - Main/HomePage*. `http://www2.statmt.org/moses/`. Accessed: 2024-2-7.

[3] *OpenNMT*. en. `https://opennmt.net/`. Accessed: 2024-2-7.

[4] Jean Senellart et al. "OpenNMT System Description for WNMT 2018: 800 words/sec on a single-core CPU". In: *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2018.

[5] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).