Traducción estadística basada en frases: RNN

Ramon Ruiz Dolz

December 2018

1 Introducción

Esta segunda sesión de prácticas de traducción automática consiste en hacer uso de las redes neuronales para la tarea de traducción de textos. Concretamente se ha hecho uso de la herramienta *nmt-keras*[1] construida sobre *Theano* [2] y *Keras* [3]. A lo largo de esta segunda práctica, del mismo modo que en la anterior, se han entrenado distintos modelos de traducción automática expuestos en la sección 2 y se han comparado sus resultados en la sección 3. Se ha usado el corpus EuTrans para el entrenamiento de las redes. Para la evaluación de los resultados se ha hecho uso de la métrica BLEU [3]. Esta medida indica, a mayor valor, mejor calidad de la traducción.

2 Experimentos

En esta sección se presentan las pruebas realizadas con las redes neuronales recurrentes (RNN). Una vez preparado el entorno y los datos de entrenamiento y test se procede a entrenar el modelo de traducción neuronal. Se han realizado tres experimentos, un primer experimento inicial siguiendo el guión del boletín, un segundo experimento probando distintos tamaños para el *encoder-decoder* y finalmente probando distintos algoritmos de aprendizaje como lo son *Adagrad* y *Adadelta*.

2.1 Experimento inicial

Para el experimento inicial se ha lanzado una RNN compuesta por un *encoder* LSTM de 64 neuronas, con un vector para codificar las palabras fuente de talla 64. Así mismo, el *decoder* también es un LSTM de 64 neuronas, con un vector para codificar las palabras destino de talla 64. Inicialmente se ha configurado con un *learning rate* de 0.001 y 5 epochs.

[1] https://github.com/lvapeab/nmt-keras
[2] http://deeplearning.net/software/theano/
[3] https://keras.io/

2.2 Ajuste de los word embeddings

En este experimento se probará con distintos tamaños de word embeddings. Concretamente se han lanzado tres pruebas modificando el modelo del experimento inicial con vectores de talla 32, 128 y 256.

2.3 Opcional: Ajuste del *encoder-decoder*

Este segundo experimento consiste en ajustar el número de neuronas de las redes LSTM del *encoder-decoder*. De la misma forma que en experimento anterior, se han lanzado tres pruebas con 32, 128 y 256 neuronas.

2.4 Opcional: Nuevos algoritmos de aprendizaje, Adagrad y Adadelta

Finalmente, este último experimento consiste en probar con Adagrad y con Adadelta como algoritmos de aprendizaje. El experimento inicial hace uso del algoritmo de optimización de descenso por gradiente Adam.

Adam [2] o Adaptive Moment Estimation es un método para calcular *learning rates* adaptados a cada parámetro. Además de almacenar la media decreciente exponencialmente de los anteriores gradientes cuadrados como Adadelta, también almacena los gradientes anteriores.

Adagrad [1] consiste en un algoritmo de descenso por gradiente con la peculiaridad que adapta el *learning rate* a los parámetros que se están estimando. Esto se consigue realizando pequeños cambios para los parámetros asociados a las características que aparecen frecuentemente y grandes cambios para los parámetros asociados a las características de menor frecuencia. Concretamente, este algoritmo es útil al trabajar con datos dispersos.

Adadelta [4] por otra parte se puede ver como una extensión del algoritmo Adagrad. Adadelta evita reducir monótonamente el *learning rate*. Para ello, en vez de almacenar los gradientes anteriores, este algoritmo trabaja con una ventana de gradientes acumulados almacenando así únicamente la información en un determinado rango.

3 Resultados

En esta sección se presenta una comparativa de la medida BLEU obtenida en cada uno de los experimentos. En la tabla 3 se pueden observar los resultados obtenidos siguiendo las instrucciones básicas del boletín.

Exper.	**Enc-Dec size**	**Grad algorithm**	**BLEU**
1	64	Adam	94.47

Table 1: Resultados del experimento inicial.

Además de estos resultados, a continuación se pueden observar los resultados obtenidos con las modificaciones realizadas sobre la red original.

3.1 Ajuste de los *word embeddings*

En esta primera prueba, a partir del modelo usado en el modelo inicial, modificando el tamaño del vector se han obtenido los valores de BLEU mostrados en la tabla 3.1

Exper.	**Word embedding size**	**Grad algorithm**	**BLEU**
embeddingmod1	32	Adam	90.00
embeddingmod2	128	Adam	97.53
embeddingmod2	256	Adam	97.65

Table 2: Resultados de al ajustar el tamaño de los *word embeddings*.

3.2 Opcional: Ajuste del *encoder-decoder*

Haciendo uso del tamaño de vector de 128, se ha realizado el ajuste del número de neuronas del *encoder-decoder*. En la tabla 3.2 se observan los valores BLEU obtenidos tras variar el tamaño de las redes del *encoder-decoder*.

Exper.	**Encoder-Decoder size**	**Grad algorithm**	**BLEU**
enc-dec32	64	Adam	79.09
enc-dec128	128	Adam	97.66
enc-dec256	256	Adam	89.29

Table 3: Resultados al variar el tamaño de la red LSTM

Como podemos observar, en las variaciones realizadas, al disminuir el número de neuronas la precisión del modelo decae. Sin embargo al aumentarlo mejora, pero hasta cierto punto, puesto que al hacerla demasiado grande, debido al gran número de parámetros a estimar, la precisión vuelve a caer.

3.3 Opcional: Nuevos algoritmos de aprendizaje, Adagrad y Adadelta

Se han lanzado experimentos usando la red LSTM de 128 neuronas, sin embargo debido al gran número de parámetros a estimar tanto en Adagrad como en Adadelta, se han obtenido valores de BLEU peores. En la tabla 3.3 podemos observar los valores de BLEU conseguidos al variar el argoritmo de optimización de descenso por gradiente.

Exper.	Encoder-Decoder size	Grad algorithm	BLEU
gradmod1	128	Adagrad	88.41
gradmod2	128	Adadelta	84.49

Table 4: Resultados al variar el algoritmo de aprendizaje

4 Conclusiones

En esta práctica se ha hecho uso de las redes neuronales para abordar el problema de la traducción automática. Además se ha jugueteado con varios de los parámetros principales de esta tecnología. Se ha podido observar el impacto que tiene el tamaño de los word embedding, de las redes LSTM encoder-decoder y también el papel de los distintos algoritmos de optimización a la hora de aplicar descenso por gradiente. En conclusión, se ha podido aprender de forma aplicada gran parte de la teoría estudiada en la segunda parte del curso.

References

[1] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[4] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.