MEMORIA DE PRÁCTICAS DE LABORATORIO

Traducción Automática

Autora
Aitana Menárguez Box
Enero 2024

1. Introducción

El objetivo de esta memoria es documentar el trabajo realizado en las sesiones prácticas de la asignatura de TA. En concreto, se trata de experimentar con dos herramientas diferentes (**Moses** y **OpenNMT**) para el modelado de sistemas de traducción automática de textos del idioma inglés al español.

2. Trabajo realizado

A continuación se detallan las tareas realizadas para las diferentes herramientas mencionadas anteriormente:

2.1. Moses

El uso de Moses se ha hecho a través de Docker, a partir de la imagen *moses* disponible en [1]. Tras haber completado los pasos de instalación y pruebas indicadas en el boletín, las actividades realizadas han sido las siguientes:

Ejercicio 1 - Modelo sin ajuste de pesos

Se ha realizado la traducción con el modelo obtenido en el apartado 6 del boletín. Es decir, sin realizar un ajuste de los pesos del mismo con MERT o MIRA. Ha sido suficiente con utilizar el comando[1] a continuación:

```
$ opt/moses/bin/moses -threads 10 \
  -f /data/alignment/model/moses.ini \
  < /data/test/test.es \
  > /data/test/test.hyp
```

Ejercicio 2 - Nuevos valores de máximo de iteraciones del MERT

Ha sido suficiente con sustituir el valor asociado al argumento `--maximum-iterations` con el valor deseado. Se han probado los valores 5, 6, 7, 8, 9 y 10. El comando es el siguiente:

```
$ opt/moses/scripts/training/mert-moses.pl \
  data/dev/development.clean.es data/dev/development.clean.en \
  /opt/moses/bin/moses /data/alignment/model/moses.ini \
  -threads=10 --maximum-iterations=<num_iter> \
  --working-dir /data/mert<num_iter>it --mertdir /opt/moses/bin/ \
  --decoder-flags "-threads 10"
```

Ejercicio 3 - Distintos valores de n-gramas

Como en el caso anterior, basta con modificar el parámetro del comando con el valor deseado y después seguir con los pasos como se indica en el resto del boletín. Se han probado los valores de n-gramas 2, 5 y 6.

[1] Todos los comandos ejecutados en esta práctica se sobrentiende que han sido ejecutados en el *root* del contenedor de Docker.

```
$ opt/srilm/lm/bin/i686-m64/ngram-count -order <n> /
  -unk -interpolate -kndiscount -text /
  data/train/training.clean.en -lm /data/lm/ex3_n<n>.lm

$ opt/moses/scripts/training/train-model.perl \
  -mgiza -mgiza-cpus 1 --first-step 1 \
  -root-dir /data/ex3/alignment_n<n> \
  -corpus /data/train/training.clean -f es -e en \
  -alignment grow-diag-final-and -reordering msd-bidirectional-fe \
  -lm 0:3:/data/lm/ex3_n<n>.lm \
  -external-bin-dir opt/moses/mgiza/mgizapp/bin/

$ opt/moses/scripts/training/mert-moses.pl \
  data/dev/development.clean.es data/dev/development.clean.en \
  /opt/moses/bin/moses /data/ex3/alignment_n<n>/model/moses.ini \
  -threads=10 --maximum-iterations=5 \
  --working-dir /data/ex3/mert_n<n> --mertdir /opt/moses/bin/ \
  --decoder-flags "-threads 10"
```

Ejercicio 4 - Probar MIRA para el entrenamiento de los pesos

En este caso, como indica en [**moses'mira**], se debe añadir el argumento `--batch-mira` de la siguiente manera:

```
$ opt/moses/scripts/training/mert-moses.pl \
  /data/dev/development.clean.es data/dev/development.clean.en \
  /opt/moses/bin/moses /data/alignment/model/moses.ini \
  -threads=10 --maximum-iterations=5 --batch-mira \
  --working-dir /data/ex4/mert --mertdir /opt/moses/bin/ \
  --decoder-flags "-threads 10"
```

Ejercicio 5 - Diferentes técnicas de suavizado

A partir de la opción `-help` de `opt/srilm/lm/bin/i686-m64/ngram-count` se pueden comprobar qué suavizados están implementados para ser usados. En este caso se ha probado con: *unmodified Kneser-Ney* (-ukndiscount) y *Witten-Bell* (-wbdiscount), además de *modified Kneser-Ney* (que era el que venía por defecto en las instrucciones de la práctica).

```
$ opt/srilm/lm/bin/i686-m64/ngram-count -order 3 -unk -interpolate <smoothing> -
data/train/training.clean.en -lm /data/lm/ex5_ukn.lm

$ opt/moses/scripts/training/train-model.perl \
  -mgiza -mgiza-cpus 1 --first-step 1 \
  -root-dir /data/ex5/alignment_<smoothing> \
  -corpus /data/train/training.clean -f es -e en \
  -alignment grow-diag-final-and -reordering msd-bidirectional-fe \
  -lm 0:3:/data/lm/ex5_<smoothing>.lm \
  -external-bin-dir opt/moses/mgiza/mgizapp/bin/
```

```
$ opt/moses/scripts/training/mert-moses.pl \
  data/dev/development.clean.es data/dev/development.clean.en \
  /opt/moses/bin/moses /data/ex5/alignment_<smoothing>/model/moses.ini \
  --threads=10 --maximum-iterations=5 \
  --working-dir /data/ex5/mert_<smoothing> --mertdir /opt/moses/bin/ \
  --decoder-flags "-threads 10"
```

Ejercicio 6 - Prueba con Moses monótono

Para realizar esta tarea es necesario cambiar el `distortion-limit` del modelo a 0. Para ello, se utilizará `-distortion-limit 0` a la hora de entrenar el modelo así:

```
$ opt/moses/scripts/training/train-model.perl \
  -mgiza -mgiza-cpus 1 --first-step 1 \
  -root-dir /data/ex6/alignment \
  -corpus /data/train/training.clean -f es -e en \
  -alignment grow-diag-final-and -reordering msd-bidirectional-fe \
  -lm 0:3:/data/lm/turista.lm \
  -external-bin-dir opt/moses/mgiza/mgizapp/bin/ -distortion-limit 0

$ opt/moses/scripts/training/mert-moses.pl \
  data/dev/development.clean.es data/dev/development.clean.en \
  /opt/moses/bin/moses /data/ex6/alignment/model/moses.ini \
  --threads=10 --maximum-iterations=5 \
  --working-dir /data/ex6/mert --mertdir /opt/moses/bin/ \
  --decoder-flags "-threads 10"
```

2.2. OpenNMT

Después de haber realizado la instalación tal y como se indica en el *notebook* disponible en [**colab˙repo**], se han completado las siguientes actividades:

Ejercicio 1 - Diferentes tamaños de *word embeddings*

Para cambiar el tamaño de los *embeddings* es necesario cambiar el parámetro `word_vec_size` al valor deseado (en este caso 128 y 256), además de cambiar el tamaño del codificador y del decodificador a partir del parámetro `rnn_size`.

Ejercicio 2 - Variar capas del codificador y decodificador

Con tal de cambiar la cantidad de capas del codificador y/o decodificador, habrá que variar el valor `layers`. En este caso se ha probado con 4 y 6 capas.

Ejercicio 3 - Otros algoritmos de aprendizaje

Si se quiere modificar el algoritmo de optimización es necesario cambiar el parámetro `optim` por el valor del método deseado. En este caso se han probado `sgd`, `adagrad` y `adadelta`. Por otro lado, para cualquier algoritmo que no sea `adam` o `sgd`, será necesario

cambiar el `decay_method`; en este caso se ha puesto a `none`. Además, es recomendable para obtener mejores resultados cambiar el *learning rate* según el aprendizaje que se use. En este caso se ha utilizado (como se indica en la documentación [**onmt˙docu**]): para `sgd` 1, para `adagrad` 0.1 y para `adadelta` 1.

Ejercicio 4 - Redes recurrentes

Se han cambiado los valores de `decoder_type` y `encoder_type` a `rnn` y se han probado dos valores diferentes para `rnn_type`: `LSTM` y `GRU`

3. Resultados obtenidos

3.1. Moses

Todas las experimentaciones anteriores han sido realizadas a partir del modelo principal que se propone a lo largo del boletín. Es decir, no se han acumulado. El resultado de BLEU obtenido en el entrenamiento básico que propone la guía es de **91.84**. A continuación, se muestran los resultados obtenidos en todas las actividades con tal de analizar cómo han afectado las modificaciones del entrenamiento al rendimiento del modelo.

	Modelo	**BLEU**
Ejercicio 1	sin ajuste de pesos	88.42
Ejercicio 2	máx. iter. 6	**92.04**
	máx. iter. 7	91.98
	máx. iter. 8	91.58
	máx. iter. 9	91.62
	máx. iter. 10	91.99
Ejercicio 3	2-gramas	91.14
	5-gramas	**92.34**
	6-gramas	92.31
Ejercicio 4	pesos con MIRA	90.91
Ejercicio 5	unmodified Kneser-Ney	**91.92**
	Witten-Bell	91.90
Ejercicio 6	MOSES monótono	**92.07**

3.2. OpenNMT

El modelo inicial presentaba un BLEU de **95.4** al ser evaluado. En la siguiente tabla se muestran los resultados obtenidos tras la experimentación con los cambios sobre este modelo.

	Modelo	BLEU
Ejercicio 1	w.e. 128	**97.9**
	w.e. 256	**96.0**
Ejercicio 2	4 layers	95.5
	6 layers	94.4
Ejercicio 3	SGD	95.0
	Adagrad	56.9
	Adadelta	94.0
Ejercicio 4	LSTM	82.1
	GRU	58.7

4. Conclusiones

El mejor resultado obtenido ha sido un BLEU de **92.34**, para la primera parte de la práctica, con un modelo de lenguaje de 5-gramas con suavizado *Kneser-Ney* y con un MERT con 5 iteraciones de entrenamiento de pesos.
Por otro lado, en la segunda parte, el mejor resultado de BLEU ha sido **97.9**, obtenido con una red como la original (transformer para el encoder y el decoder) y un tamaño de *word embedding* de 128.

Estos datos muestran que los métodos de entrenamiento realizados en la segunda parte dan mejores valores de BLEU. Además, se aprecia que la calidad de los modelos de la segunda parte es muy sensible al cambio de los valores de sus parámetros. En cuanto a los experimentos realizados, los modelos neuronales requieren de más tiempo de entrenamiento y capacidad de cómputo, por lo que se han podido probar menos configuraciones que en el caso de los modelos estadísticos.

Referencias

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