Traducción estadística basada en modelos sintácticos-jerárquicos

Toolkit: Moses

Autor: Pascual Andrés Carrasco Gómez

Asignatura: Traducción automática (TA)

EJERCICIO BÁSICO

El corpus utilizado es un subconjunto del corpus europarl (europarl-v7.es-en) compuesto por un conjunto de entrenamiento (pares de frases inglés-español) de 50.000 frases y un conjunto de test (pares de frases inglés-español) de 1.000 frases. Se ha creado una estructura del Corpus con la finalidad de que los scripts que se han implementado sirvan para cualquier Corpus de entrada simplemente estructurando y renombrando los ficheros del corpus de la siguiente forma:

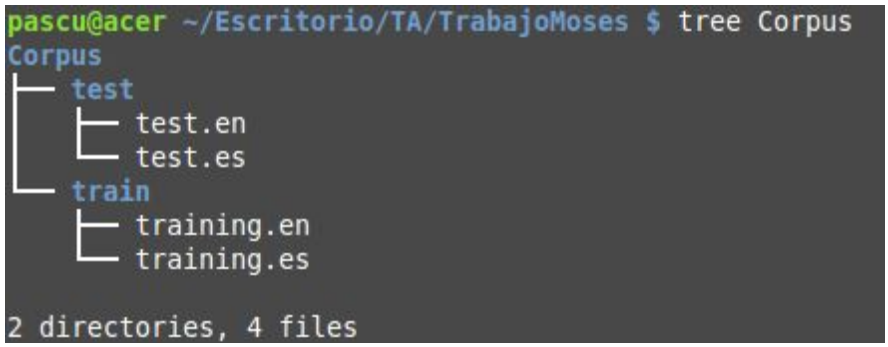

Imagen 1: Estructura del Corpus de entrada para el correcto funcionamiento de los scripts.

Se ha optado por dividir el corpus de entrenamiento en dos partes, una parte de entrenamiento (85%) para entrenar el modelo en Moses y una parte de desarrollo (15%) para entrenar los pesos del modelo log-lineal mediante MERT (utilizando 5 iteraciones).
Los scripts que se han utilizado para el ejercicio básico se describen a continuación:

- formatear_corpus.sh: Limpia y tokeniza el corpus.
- script_train.sh: Genera el traductor automatico basandose en modelos sintáctico-jerárquicos con moses.
- script_test.sh: Evalua el traductor automático obtenido mediante el script “script_train.sh” y devuelve el BLEU obtenido.

Los mejores parámetros obtenidos para generar el traductor de moses son los siguientes:

Configuración	Parámetros
Modelo de lenguaje	5-gramas Técnica de suavizado: backoff Método de descuento: kndiscount
Tabla de segmentos	Máxima longitud frase: 5
Mert	Iteraciones: 5

Los resultados obtenidos se muestran a continuación:

Sentido	BLEU
Inglés - Español	25.93

TRADUCCIÓN DEL JAPONÉS AL INGLÉS Y DEL INGLÉS AL JAPONÉS

Para realizar este ejercicio se ha utilizado el corpus de la tarea *Kyoto Free Translation Task (KFTT)* el cual está compuesto por un conjunto de pares de frases del japonés (ja) al inglés (en).

KFTT → http://www.phontron.com/kftt/

Corpus paralelo → http://www.phontron.com/kftt/download/kftt-data-1.0.tar.gz

El corpus ya se encuentra tokenizado, la estructura de ficheros en las que se proporciona el corpus se muestra en la siguiente ilustración:

Imagen 1: Estructura del corpus KFFT → /kftt-data-1.0/data/tok/

Para trabajar con nuestros scripts hemos de adaptar la estructura, para realizar la adaptación del corpus se ha implementado el script “preparar_corpus_kftt.sh”. Este script renombra los ficheros y los sitúa en la estructura adecuada:

./preparar_corpus_kftt.sh kftt-data-1.0/data/tok/

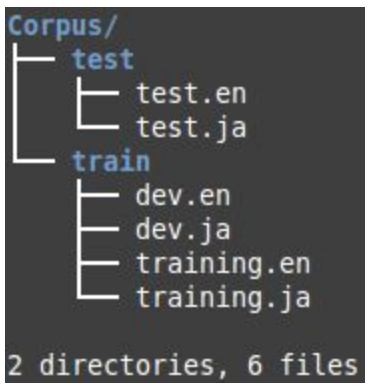

Imagen 2: Estructura generada por el script “preparar_corpus_kftt.sh”.

Nota: El training.* está compuesto por la unión de kyoto-train.* y kyoto-tune.*.

Una vez tenemos la estructura definida procedemos a limpiar el corpus con el script “limpiar_corpus.sh”. El corpus resultante tiene las siguientes características:

Conjunto de datos (japonés e inglés)	Número de frases (japonés e inglés)
Test	1081
Train	330992
Dev	1088

Se ha realizado la traducción en ambos sentidos, los scripts que se han utilizado para cada caso se encuentran en los directorios “en-ja” y “ja-en”, para poder ejecutar los scripts se tiene que incluir el directorio “Corpus” , generado en el paso anterior, en dichos directorios. Los directorios contienen dos scripts que se describen a continuación:

- script_train.sh: Genera el traductor automatico basandose en modelos sintáctico-jerárquicos con moses.
- script_test.sh: Evalua el traductor automático obtenido mediante el script “script_train.sh” y devuelve el BLEU obtenido.

Los parámetros para generar el traductor de moses son los siguientes:

Configuración	**Parámetros**
Modelo de lenguaje	5-gramas Técnica de suavizado: backoff Método de descuento: kndiscount
Tabla de segmentos	Máxima longitud *phrase*: 5
Mert	Iteraciones: 5

Al realizar varias pruebas hemos observado que al trabajar con un conjunto de entrenamiento tan elevado y al utilizar una tabla de segmentos de longitud máxima por *phrases* igual a 5 el coste espacial explota exponencialmente, sobrepasando los 6gb de RAM que dispone mi máquina (ordenador portátil personal) obligandonos a reiniciar el equipo.
Para solucionar el problema descrito hemos optado a realizar dos experimentaciones, la primera experimentación consiste en mantener la configuración descrita reduciendo el corpus de entrenamiento (train) a 50.000 frases ya que hemos observado con el ejercicio básico que la RAM de la máquina lo puede soportar utilizando un total de 3.7 GB de RAM.

Para reducir el corpus de entrenamiento train simplemente se han ejecutado las siguientes instrucciones bash:

```
cat Corpus/train/training.clean.tok.en | head -50000 > train.en
mv train.en Corpus/train/training.clean.tok.en
cat Corpus/train/training.clean.tok.ja | head -50000 > train.ja
mv train.ja Corpus/train/training.clean.tok.ja
```

Los resultados reduciendo el corpus de entrenamiento se muestran a continuación:

Sentido	**BLEU**
Inglés - Japonés	11.59
Japonés - Inglés	9.55

La segunda experimentación consiste en mantener el corpus de entrenamiento (train) y reducir el parámetro de longitud máxima por *phrases* a 3 para la obtención de la tabla de segmentos. El consumo de RAM para esta ejecución ha sido de 4.4 GB.

Nota: Se han realizado pruebas con longitud máxima por *phrases* igual a 4 y el coste espacial seguía siendo demasiado elevado consumiendo los 6 GB de RAM de la máquina y obligandonos a reiniciar el equipo.

Los resultados reduciendo a 3 la longitud máxima por *phrases* se muestran a continuación:

Sentido	**BLEU**
Inglés - Japonés	15.58
Japonés - Inglés	13.06

Observamos en las experimentaciones que hemos realizado que obtenemos un mejor traductor basado en modelos sintáctico-jerárquicos, tanto en sentido Inglés - Japonés como en sentido Japonés - Inglés, utilizando todo el corpus de entrenamiento (train) y reduciendo la longitud máxima por *phrases* a 3. Al trabajar con moses y modelos sintáctico-jerárquicos es importante tener en cuenta los recursos que disponemos para realizar la ejecución ya que el coste espacial explota exponencialmente como hemos podido comprobar en este trabajo.