INDICE

1

OPERAZIONI PRELIMINARI

In questo capitolo verranno mostrati i passi necessari ad installare tutti i software e le librerie necessarie all'esecuzione del codice sviluppato per questa tesi. Il codice è scaricabile all'indirizzo `https://drive.google.com/open?id=0B_c80gg7c8oIQ2pibEJSc3VDQW8`. Durante la guida si assume che si disponga già di Python e si utilizzi un sistema Linux, nello specifico le procedure proposte sono state testate con Ubuntu Mate 16.04.
Scaricare, inoltre, il modello Word2Vec che è stato utilizzato, a questo indirizzo: `https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit`

1.1 INSTALLAZIONE LIBRERIE

1.1.1 *Installazione TensorFlow*

La seguente procedura si riferisce all'installazione della versione di TensorFlow senza accelerazione grafica. Verrano proposte due alternative modalità di installazione.

1.1.1.1 *Con PIP*

- Se pip non è gia presente o è presente in una versione minore di 8.1 (verificare con il comando pip3 -V o pip -V) eseguire il seguente comando:

  ```
  sudo apt-get install python-pip python-dev
  ```

- Installare TensorFlow con uno dei seguenti comandi a seconda della versione di Python che si è deciso di utilizzare:

  ```
  sudo pip3 install tensorflow
  ```

  oppure:

  ```
  sudo pip install tensorflow
  ```

- Per determinare se l'installazione è andata a buon fine digitare dall'interprete interattivo di Python (avviabile digitando python o python3 nel prompt dei comandi) le seguente istruzioni:

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

Se tutto è andato a buon fine comparirà la scritta "Hello, TensorFlow!", in caso contrario seguire le istruzioni per i problemi più comuni: `https://www.tensorflow.org/install/install_linux#CommonInstallationProblems`

1.1.1.2 *Con Virtual-Env*

- Se non sono già presenti, installare pip e virtualenv, tramite il comando:

```
sudo apt-get install python-pip python-dev
    python-virtualenv
```

- Creare un ambiente virtuale tramite il seguente comando, è possibile sostituire ~/tensorflow con qualsiasi altro nome, questa sarà la cartella dove verrà creato l'ambiente virtuale con tensorflow:

```
python3 -m venv -system-syte-packages ~/tensorflow
```

- Bisogna ora attivare l'ambiente virtualenv appena creato, utilizzando il seguente comando(se si è utilizzata una cartella diversa da ~/tensorflow , sostituitelo con il nome della directory scelta):

```
source ~/tensorflow/bin/activate
```

A questo punto la source del prompt dei comandi dovrebbe cambiare in:

```
(tensorflow)$
```

- A questo punto installare TensorFlow con uno dei seguenti comandi a seconda della versione di Python che si è deciso di utilizzare:

```
pip3 install --upgrade tensorflow
```

oppure:

```
pip install --upgrade tensorflow
```

- Per determinare se l'installazione è andata a buon fine digitare dall'interprete interattivo di Python (avviabile digitando python o python3 nel prompt dei comandi) le seguente istruzioni:

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
```

```
sess = tf.Session()
print(sess.run(hello))
```

Se tutto è andato a buon fine comparirà la scritta "Hello, TensorFlow!", in caso contrario seguire le istruzioni per i problemi più comuni: `https://www.tensorflow.org/install/install_linux#CommonInstallationProblems`

- Una volta terminata l'installazione, ogni qual volta si vuole utilizzare TensorFlow, bisognerà avviare il virtualenv associato:

```
source ~/tensorflow/bin/activate
```

- Per disattivare il virtual-env digitare:

```
deactivate
```

1.1.2 *Installazione librerie esterne*

- Salvare le seguenti righe di testo in un file chiamato 'requirement.txt':

```
nltk==3.2.3
gensim==2.2.0
numpy==1.11.0
```

- Installare le librerie python indicate nel file requirements.txt con il comando:

```
sudo pip install -r requirements.txt
```

1.1.3 *Installazione materiale per libreria DNC*

Sonnet è ora disponibile anche via PIP, il che dovrebbe rendere l'installazione molto più semplice. Una guida al suo utilizzo, nel caso fosse necessaria, la si può trovare all'indirizzo `https://deepmind.github.io/sonnet/#`

1.1.3.1 *Via PIP*

- Per l'installazione senza GPU-Support, eseguire il seguente comando:

```
pip install dm-sonnet
```

- Per l'installazione con GPU-Support, eseguire il seguente comando:

```
pip install dm-sonnet-gpu
```

1.1.3.2 *Installazione Bazel*

- Aggiungere l'URI di distribuzione di Bazel come sorgente di pacchetti:

```
echo "deb [arch=amd64]
    http://storage.googleapis.com/bazel-apt
    stable jdk1.8" | sudo tee
    /etc/apt/sources.list.d/bazel.list

curl https://bazel.build/bazel-release.pub.gpg |
    sudo apt-key add -
```

- Installare Bazel tramite il seguente comando:

```
sudo apt-get update && sudo apt-get install bazel
```

- Aggiornare Bazel:

```
sudo apt-get upgrade bazel
```

1.1.3.3 *Installazione Sonnet*

- Se si utilizza virtualEnv, attivarlo:

```
source ~/tensorflow/bin/activate
```

- Clonare il repository di Sonnet:

```
git clone --recursive
    https://github.com/deepmind/sonnet
```

- Eseguire la configurazione, lasciare le impostazione di default a meno di esigenze particolari:

```
cd sonnet/tensorflow
./configure
cd ../
```

- Ora lanciare lo script di installazione per creare un file wheel in una directory temporanea:

```
mkdir /tmp/sonnet

sudo bazel build --config=opt :install
    --genrule_strategy=standalone
    --spawn_strategy=standalone
    --copt="-D_GLIBCXX_USE_CXX11_ABI=0"

./bazel-bin/install /tmp/sonnet
```

- Installare attraverso pip utilizzando il file appena creato:

```
sudo pip install /tmp/sonnet/*.whle
```

- A questo punto uscire dalla cartella Sonnet e per determinare se l'installazione è andata a buon fine digitare dall'interprete interattivo di Python (avviabile digitando python o python3 nel prompt dei comandi) le seguente istruzioni:

```
import sonnet as snt
import tensorflow as tf
snt.resampler(tf.constant([0.]),
    tf.constant([0.]))
```

  L'output dovrebbe essere questo:

```
<tf.Tensor 'resampler/Resampler:0' shape=(1,)
    dtype=float32>
```

1.2 DOWNLOAD DEI DATASET

1.2.0.1 *Dataset Amazon*

I dataset utilizzati negli esperimenti relativi alle recensioni Amazon sono scaricabili ad i seguenti indirizzi:

- Books (B): `http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Books.json.gz`
- Electronics (E): `http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Electronics.json.gz`
- Clothing,Shoes and Jewelry (J): `http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Clothing_Shoes_and_Jewelry.json.gz`
- Movies and TV (M): `http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Movies_and_TV.json.gz`

1.2.0.2 *Stanford Sentiment Treebank*

Lo Stanford Sentiment Treebank è un dataset particolare, dove ogni recensione è organizzata sotto forma di albero dove ogni nodo è composto da una porzione di essa. Inoltre le valutazione sono espressi con punteggi da 0 ad 1. Si rende necessario quindi applicare alcuni accorgimenti al fine di poter utilizzare questo dataset al codice precedentemente creato. I dataset direttamente utilizzabili, da me composti, sono disponibili ai seguenti indirizzi:

- Dataset di training: `https://drive.google.com/file/d/0B_c80gg7c8oIOTF2OXdLUm55VGM/view?usp=sharing`

- Dataset di testing: `https://drive.google.com/open?id=0B_c80gg7c8oISkJsbUlMMkhOSUE`

2

AVVIO DI UN ESPERIMENTO

In questa sezione dell'appendice verranno descritte le procedure necessarie per lanciare un esperimento. Si presume siano stati già compiuti tutti i passi descritti nell'appendice 1.

2.1 PROCEDURE GENERALI

Se non lo si è già fatto scompattare l'archivio SentimentClassificationWithDNC, al suo interno troveremo vari file, quelli che sono utili a lanciare i vari esperimenti sono:

- SentimentClassificationCrossDomainOfficial.py
- SentimentClassificationOfficial.py
- SentimentClassificationStanford.py
- SentimentClassificationWithTitleOfficial.py
- TestOfficial.py
- TestWithTitleOfficial.py

La sintassi per lanciare un esperimento è la seguente:

```
python nome_file.py --configuration file_configurazione.json
```

A seconda dell'esperimento che si vuole lanciare si sostituisca il suo nome a nome_file.py. Il parametro —configuration serve invece per indicare al programma quale file di configurazione usare per quel determinato esperimento. Vediamo un esempio di file di configurazione, come quello fornito nel materiale:

```
{
"hidden_size": "128",
"memory_size": "64",
"word_size": "64",
"num_write_heads": "1",
"num_read_heads": "4",
"clip_value": "10",
"max_grad_norm": "10",
"batch_size":"60",
"learning_rate": "1e-2",
```

```
"final_learning_rate": "1e-3",
"optimizer_epsilon": "1e-10",
"num_training_iterations": "80040",
"num_testing_iterations": "20040",
"num_epochs": "5",
"report_interval": "20",
"checkpoint_dir" :
    "/home/pergolini/modelli_addestrati/80000/book",
"checkpoint_interval": "1334",
"word_dimension" : "300",
"max_lenght" : "150",
"dataset" :
    "/home/pergolini/ssd2/Reviews/reviews_Books.json",
"datasetDest" :
    "/home/pergolini/ssd2/Reviews/reviews_Electronics.json",
"w2v_model" : "/home/pergolini/ssd2/g2vec.bin",
"random" : "True",
"seed" : "19",
"num_classes": "2",
"gpu_fraction": "0.3"
}
```

Vediamo quindi in dettaglio cosa stanno a significare questi valori.

- hidden_size : è il numero di strati nascosti nel controller LSTM.
- memory_size : è il numero di locazioni di memoria da utilizzare nella DNC.
- word_size : è la dimensione delle singole locazioni di memoria.
- num_write_heads : è il numero di testine di scrittura.
- num_read_heads : è il numero di testine di lettura.
- clip_value: indica che gli output della rete potranno assumere valori compresi tra -clip_value e + clip_value.
- batch_size : è la dimensione del batch che si desidera usare.
- learning_rate: è il valore di learning rate iniziale
- final_learning_rate : è il valore finale del learning rate, il programma è fatto in modo cheil learning rate scali dal valore iniziale a quello finale nel corso delle epoche. Ovviamente se non si desidera variare il learning rate basta inserire lo stesso valore per quello iniziale e quello finale.
- num_training_operation: è il numero di recensioni che si vogliono utilizzare per la fase di training.
- num_testing_iterations: è il numero di recensioni che si vogliono utilizzare per la fase di testing.

- num_epoch: è il numero di epoche di cui si vuole comporre l'esperimento
- report_interval: indica ogni quanti batch riportare le informazioni relative all'addestramento o all test
- checkpoint_dir: è la cartella dove salvare o dove è contenuto,lo stato della rete.
- checkpoint_interval : indica ogni quanti batch salvare lo stato della rete, se posto a -1 significa che non verrà salvato alcuno stato.
- word_dimension : è la dimensionalità dei word embeddings, lasciarla a 300 se si utilizza il modello Word2Vec fornito.
- max_lenght : indica qual'è il numero massimo di parole consentito in una recensione per essere presa in esame.
- dataset : è il percorso al file dove si trova il dataset da considerare
- dataset_test : è il percorso al file dove si trova il dataset target in un esperimento cross-domain
- w2v_model : è il percorso al file dove sono contenuti i word embeddings pre-addestrati.
- random : può assumere i valori True e False, se posto a True significa che ogni recensione che viene scorsa nel dataset ha una probabilità su due di essere utilizzata.
- seed: è il seme di generazione per le operazioni che utilizzano la funzione random
- num_classes : è il numero di classi che si vogliono considerare, può assumere il valore 5 o 2.
- gpu_fraction : è la porzione di memoria della gpu che si desidera utilizzare.

2.2 ESPERIMENTI IN-DOMAIN

2.2.0.1 *Senza considerare il titolo delle recensioni*

Per lanciare un esperimento in-domain occorre :

- Specificare il dataset che si desidera utilizzare, aggiornando il campo 'dataset' del file di configurazione
- Aggiornare tutti quei campi del file di configurazione che si ritengano vadano cambiati.

- Lanciare l'esperimento (poniamo che il file di configurazione sia il file configuration.json):

```
python SentimentClassificationOfficial.py
    --configuration configuration.json
```

2.2.0.2 *Considerando il titolo delle recensioni*

Per lanciare un esperimento in-domain considerando il titolo delle recensioni occorre :

- Specificare il dataset che si desidera utilizzare, aggiornando il campo 'dataset' del file di configurazione
- Aggiornare tutti quei campi del file di configurazione che si ritengano vadano cambiati.
- Lanciare l'esperimento (poniamo che il file di configurazione sia il file configuration.json):

```
python SentimentClassificationWithTitleOfficial.py
    --configuration configuration.json
```

2.3 ESPERIMENTI CROSS-DOMAIN

Per lanciare un esperimento cross-domain occorre :

- Specificare i dataset che si desiderano utilizzare, aggiornando il campo 'dataset' (source domain) e 'dataset_dest' (target domain) del file di configurazione.
- Aggiornare tutti quei campi del file di configurazione che si ritengano vadano cambiati.
- Lanciare l'esperimento (poniamo che il file di configurazione sia il file configuration.json):

```
python SentimentClassificationCrossDomainOfficial.py
    --configuration configuration.json
```

2.4 ESPERIMENTI CON FINE-TUNING

Per eseguire un esperimento con fine tuning occorre:

- Specificare il dataset che si desidera utilizzare,che sarà quindi diverso da quello sorgente con cui è stato addestrato il modello su cui fare fine tuning, aggiornando il campo 'dataset' del file di configurazione.

- Nel campo num_training_operation porre il numero di recensioni con cui si vuole affinare il comportamento della rete, mentre in num_testing_operation il numero di recensioni da utilizzare per testarla.
- É fondamentale che nel campo checkpoint_dir vi sia la directory in cui è stato salvato il modello costruito sul dominio sorgente.
- Aggiornare tutti quei campi del file di configurazione che si ritengano vadano cambiati.
- Lanciare l'esperimento (poniamo che il file di configurazione sia il file configuration.json):

```
python SentimentClassificationOfficial.py
    --configuration configuration.json
```

2.5 TEST DI RETI GIÀ ADDESTRATE

Se si vuole semplicemente testare un certo modello di rete pre-addestrato, occorre:

- Specificare il dataset da cui prendere le recensioni di test, aggiornando il campo 'dataset' del file di configurazione.
- Nel campo num_testing_operation poorre il numero di recensioni da utilizzare per testarla.
- É fondamentale che nel campo checkpoint_dir vi sia la directory in cui è stato salvato il modello costruito sul dominio sorgente.
- Non è possibile cambiare nessun parametro relativo alla rete vera e propria, altrimenti il modello salvato non viene caricato.
- Lanciare l'esperimento (poniamo che il file di configurazione sia il file configuration.json):

```
python TestOfficial.py --configuration
    configuration.json
```

- Oppure per testare un modello addestrato comprendendo anche i titoli delle recensioni:

```
python TestWithTitleOfficial.py --configuration
    configuration.json
```

2.6 ESPERIMENTI DATASET STANFORD

Per lanciare un esperimento cross-domain occorre :

- Inserire nel campo 'dataset' il percorso al file StanfordSentencesNTest.json e nel campo dataset_dest il percorso al file StanfordSentencesNTest.json
- Aggiornare tutti quei campi del file di configurazione che si ritengano vadano cambiati.
- Lanciare l'esperimento (poniamo che il file di configurazione sia il file configuration.json):

```
python SentimentClassificationStanford.py
    --configuration configuration.json
```