**Web Platform per il training di modelli di NLP: DOCUMENTO TECNICO**

Introduzione:

Questo progetto rappresenta una Web Platform realizzata in Python che offre diversi servizi, tra cui la gestione di Intents, Entities e Training Phrases salvati come collezioni di documenti in un Database NoSQL, l’addestramento, il download, la cancellazione di tre modelli di NLP (Natural Language Processing), così come la visualizzazione dei risultati degli addestramenti e il loro testing.

Implementazione del server:

Il server è realizzato in Python utilizzando:

* Flask come WSGI Web App Framework, modulo Python che permette di sviluppare applicazioni web;
* Flask-PyMongo strettamente integrato con Flask e PyMongo; quest’ultimo rappresenta il Driver MongoDB ufficiale per Python che permette la gestione di Database NoSQL;
* Flask-SocketIO che fornisce alle applicazioni Flask l’accesso alle comunicazioni bidirezionali a bassa latenza tra il client e il server.

Per cui l’implementazione del server, che rappresenta il cuore dell’applicazione, è contenuta nel file **main.py**.

Addestramento dei modelli:

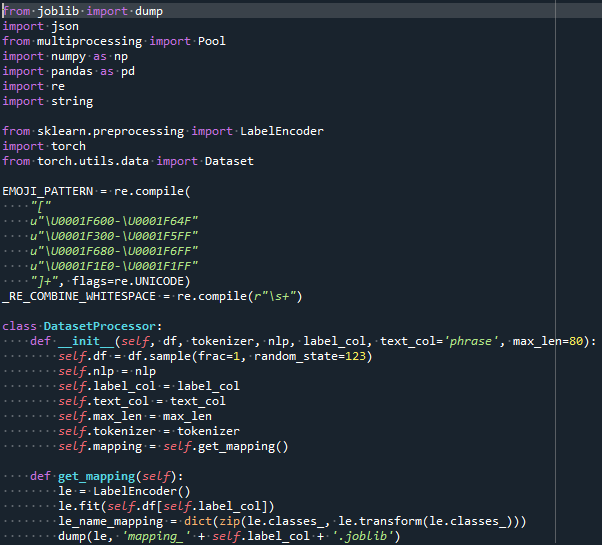
Il training di questi modelli è effettuato su thread paralleli (quindi è possibile avviare l’addestramento dei modelli in sequenza) in Python utilizzando principalmente:

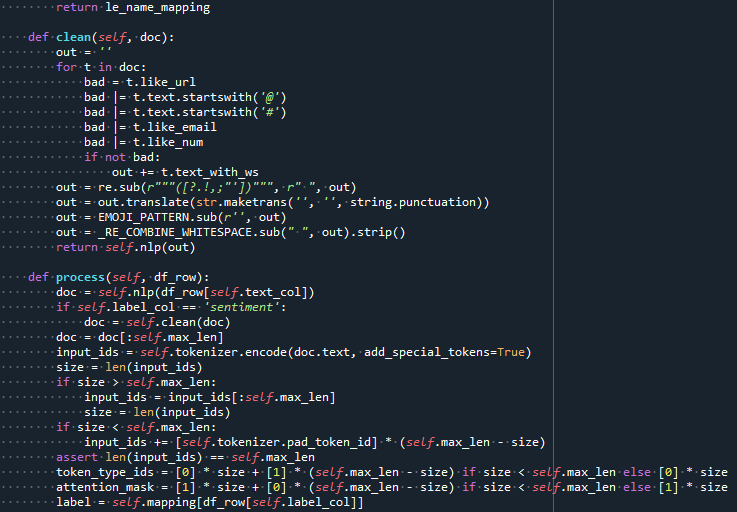
* spaCy, che rappresenta una libreria open-source progettata per lo sviluppo di applicazioni NLP;
* PyTorch, come pacchetto Python che fornisce Tensor Computation e Deep Neural Networks costruite su un sistema di autograd basato su nastro;
* Transformers che fornisce architetture di uso generale per NLU (Natural Language Understanding) e NLG (Natural Language Generation).

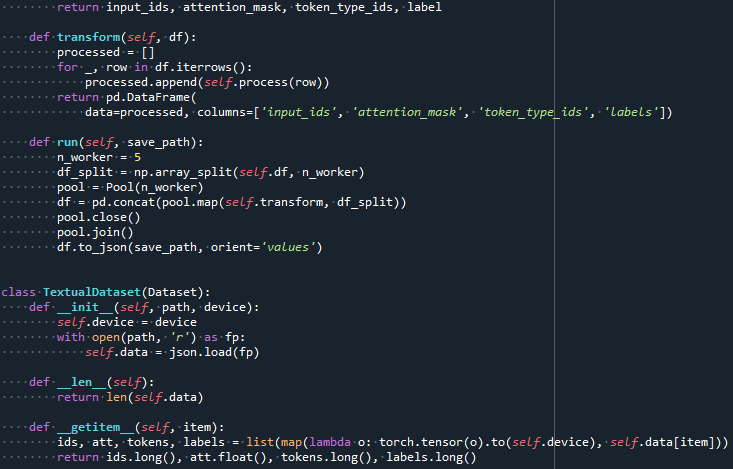
Intent Recognition e Sentiment Analysis:

L’addestramento dei modelli di IR e di SA avviene attraverso due classi e diversi file elencati di seguito:

**dataset.py:**

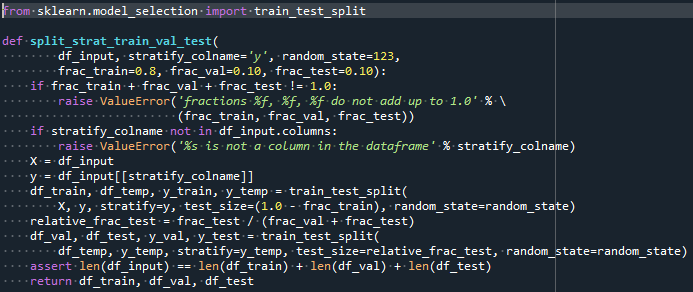






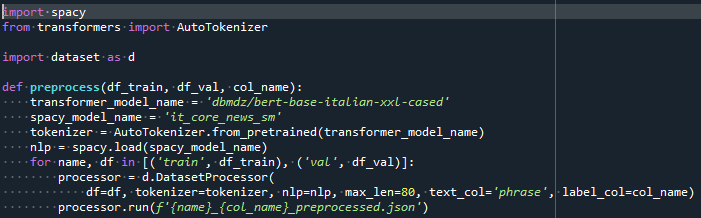
Nel file **dataset.py** ci sono le classi DatasetProcessor e TextualDataset che permettono il processing del dataset per l’addestramento dei modelli di IR (Intent Recognition) e di SA (Sentiment Analysis). Queste due classi vengono utilizzate principalmente in **preprocessing.py** e in **bert\_training.py**.

**splitting.py:**

****

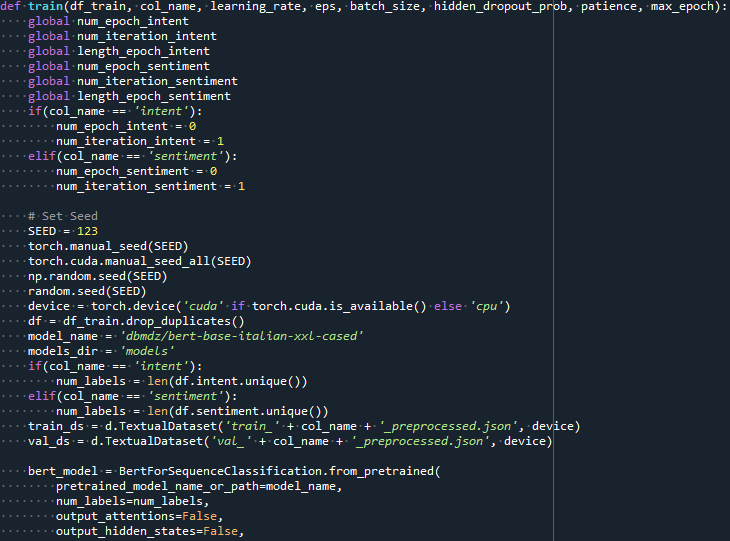
Nel file **splitting.py** avviene il processo dello splitting, cioè la divisione del dataset in frasi.

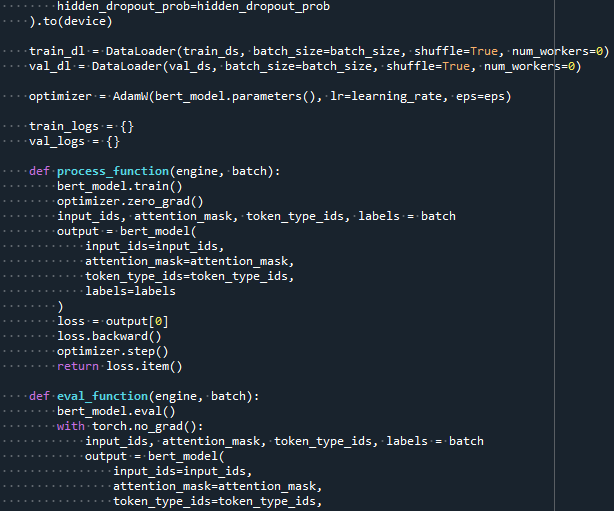
**preprocessing.py:**

****

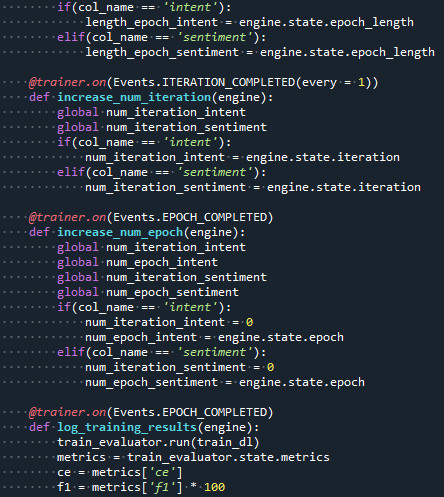
Nel file **preprocessing.py** avviene il preprocessamento del dataset, applicato trasversalmente sia al corpus sul quale è svolto l'addestramento, sia agli input da sottoporre al modello addestrato, il cui obiettivo è semplificare e trasformare il dato testuale a livello di singola parola o token*.*

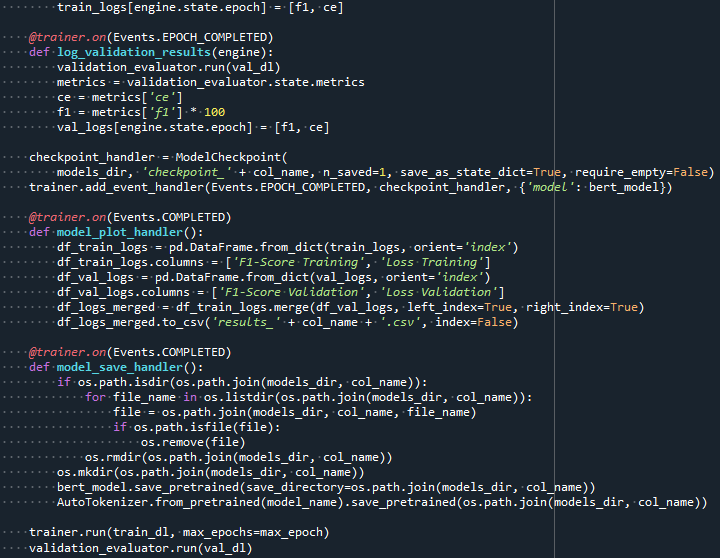
**bert\_training.py:**

****

****

****

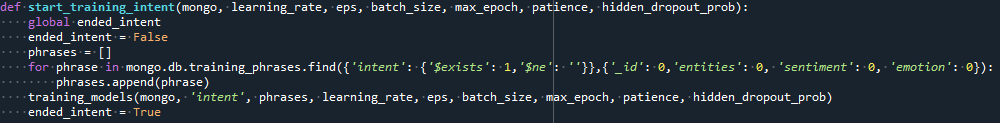
****

****

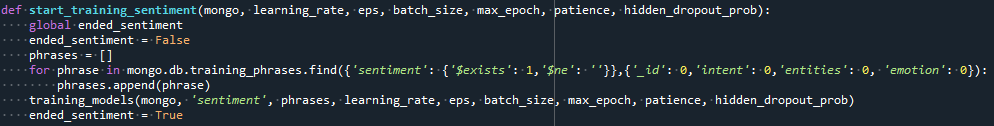
Nel file **bert\_training.py** la funzione principale è la funzione di train, che utilizza un algoritmo open source attualmente utilizzato da Google nel proprio motore di ricerca, cioè BERT(Bidirectional Encoder Representations from Transformers). BERT può comprendere parti di parlato come pronomi, fornendo un meccanismo per analizzare in maniera più efficace query scritte in linguaggio naturale.

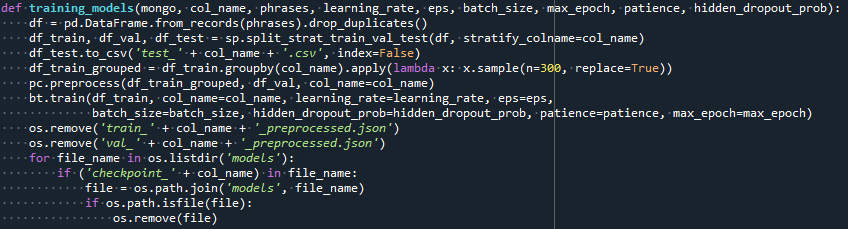
**training.py**

****

****

****

****

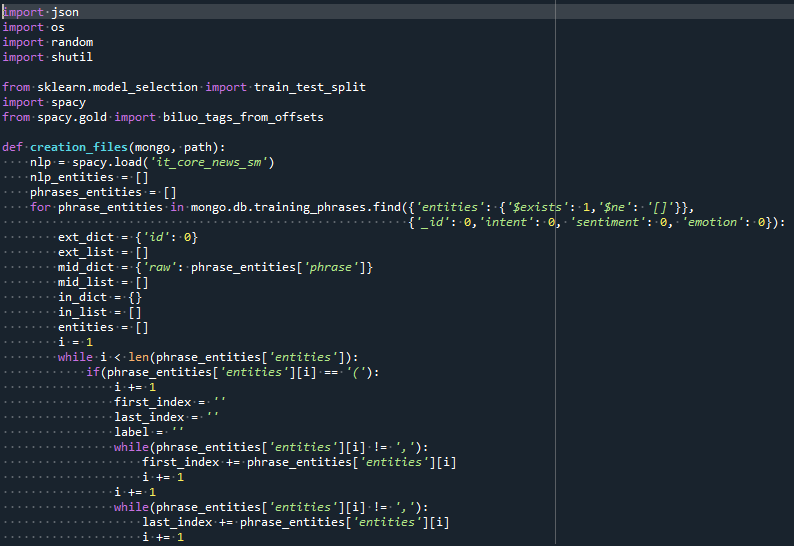
****

Nel file **training.py**, tra tutte le funzioni, ci sono le funzioni che iniziano l’addestramento del modello di IR o di SA su thread paralleli, cioè start\_training\_intent e start\_training\_sentiment. Entrambe richiamano la funzione training\_models, tra i cui parametri si ritrovano gli iperparametri utilizzati per l’addestramento del modello, cioè: learning rate, eps, batch size, max epoch, patience, hidden dropout. Essa si occupa dello splitting, del preprocessing e del bert training.

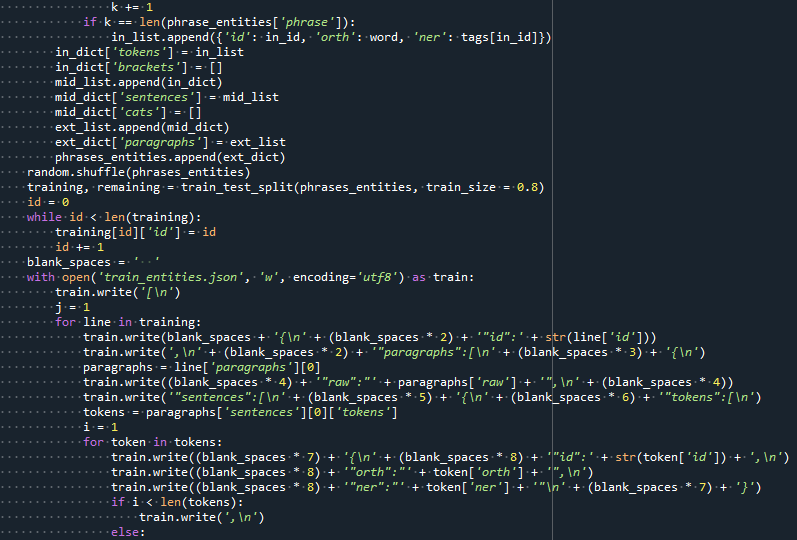
Entities Extraction:

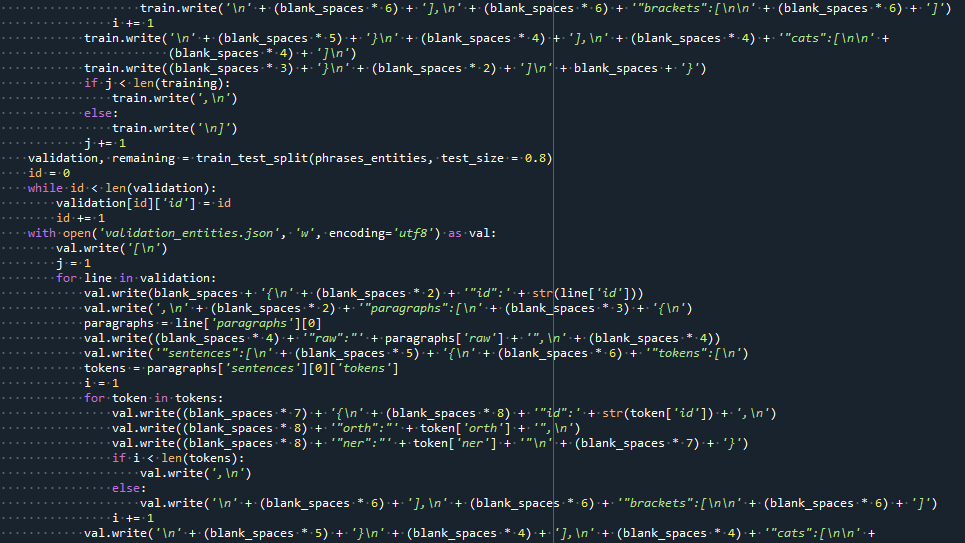
L’addestramento del modello di EE avviene attraverso due file elencati di seguito:

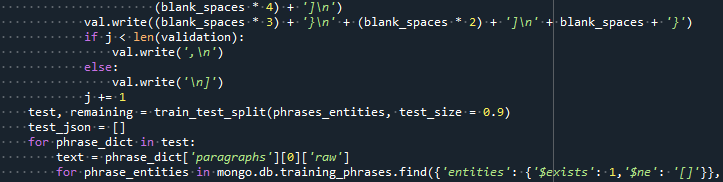
**preparation.py**



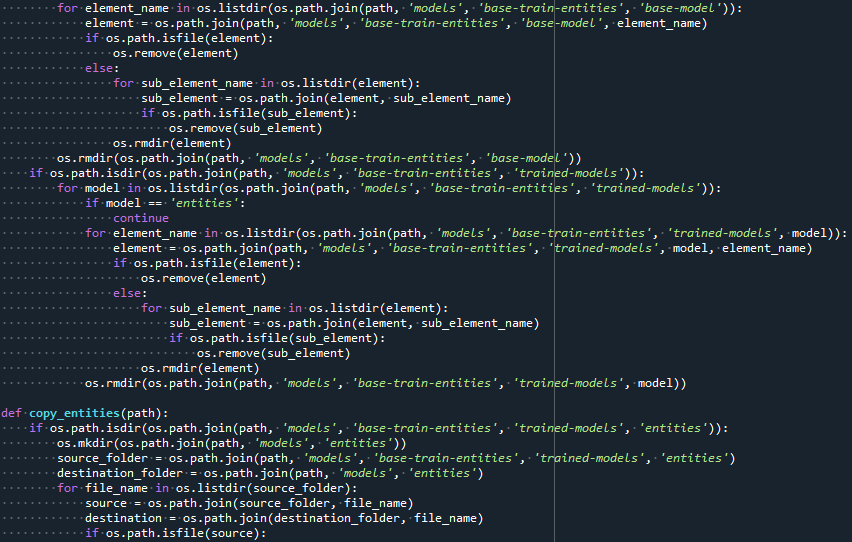


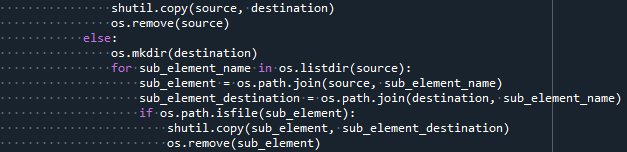


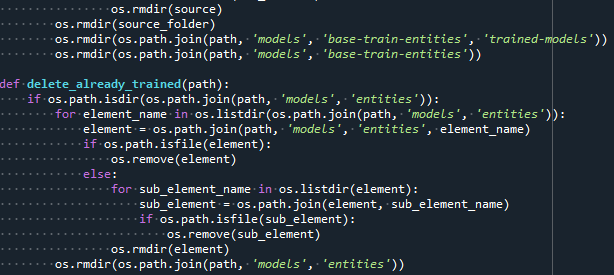






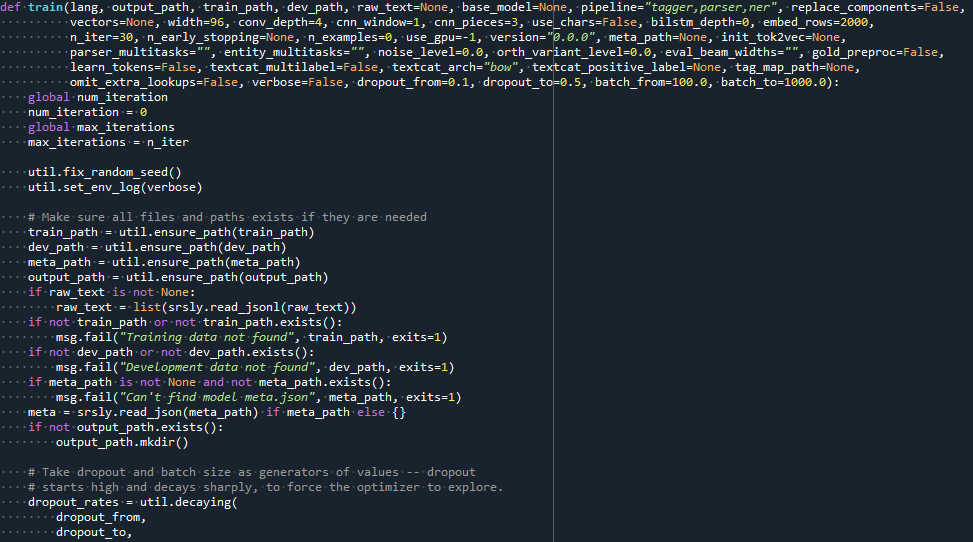






Nel file **preparation.py** ci sono le funzioni che si occupano di creare i file .json di training, validation e testing effettuando lo splitting del dataset, di cancellare i modelli addestrati lasciando il modello migliore chiamato entities, di copiare la cartella relativa al modello migliore all’interno della cartella models.

**training\_ee.py**

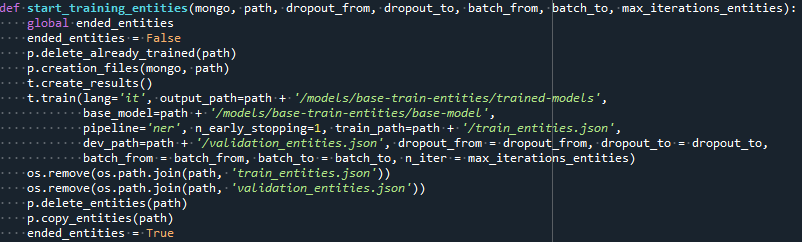


(immagine della funzione principale del training dell’Entities Extraction; si è deciso di inserire unicamente questa immagine per via della grandezza del file)

Nel file **training\_ee.py** è presente la funzione train, che permette di addestrare effettivamente il modello di EE. Essa si occupa di individuare e classificare le named entities menzionate in un dataset organizzato in categorie.

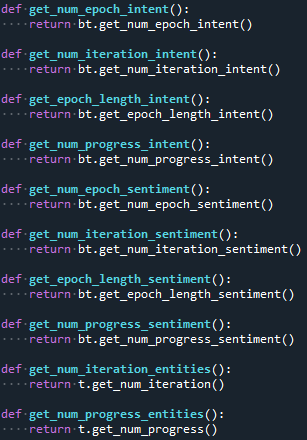
**training.py**





Nel file **training.py** c’è anche la funzione che inizia l’addestramento del modello di EE su un thread parallelo, cioè start\_training\_entities. Tra i parametri della funzione citata precedentemente si ritrovano gli iperparametri utilizzati per l’addestramento del modello, cioè: dropout from, dropout to, batch from, batch to, max iterations entities. Essa si occupa quindi di addestrare il modello utilizzando la funzione train presente nel file **training\_ee.py**.

Infine, nel file **training.py** ci sono anche funzioni get che permettono di visualizzare lo stato dell’addestramento in esecuzione nella pagina status relativa all’addestramento:



Testing dei modelli:

Le librerie rilevanti per il testing dei modelli sono spaCy, PyTorch e Transformers, descritte già precedentemente in questo documento.

Intent Recognition e Sentiment Analysis:

Il testing dei modelli di IR e di SA avviene attraverso una classe, cioè Recognition, e i file **inference.py** e **testing.py**:

**inference.py:**

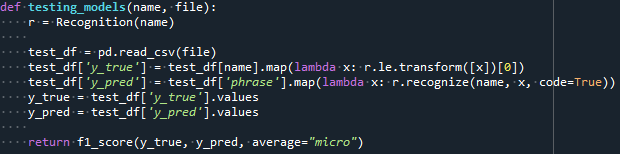


In **inference.py**, come accennato precedentemente, è presente la classe Recognition, che permette di effettuare la funzione di Natural Language Inference (NLI). Tale classe viene utilizzata per il testing dei modelli di IR e di SA nel file **testing.py**.

**testing.py:**





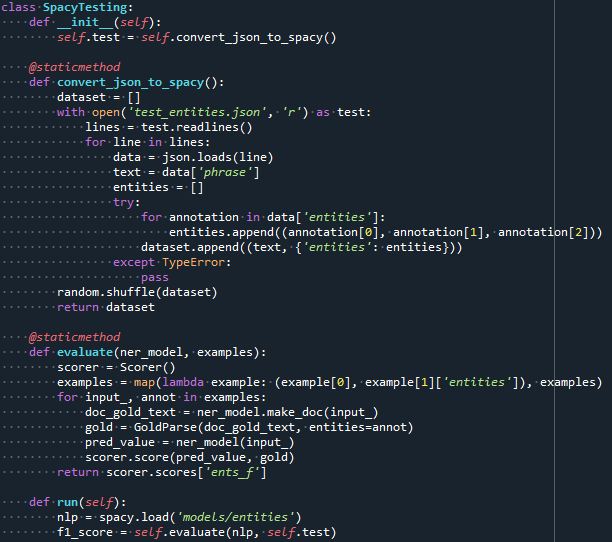


Nel file **testing.py** viene effettuato il testing del modello, rispettivamente modello di Intent Recognition (IR) tramite la funzione testing\_intent e modello di Sentiment Analysis (SA) tramite la funzione testing\_sentiment. Entrambe queste funzioni richiamano la funzione testing\_models, che ha come parametri name che rappresenta se il testing è per l’intent o per il sentiment, e file che rappresenta il file passato per il testing.

Entities Extraction:

Il testing del modello di EE avviene attraverso una classe, cioè SpacyTesting, e i file **inference.py** e **testing.py**:

**inference.py:**





In **inference.py** è presente la classe SpacyTesting, che, come la classe Recognition, permette di effettuare la funzione di Natural Language Inference (NLI). Tale classe viene utilizzata per il testing del modello di EE nel file **testing.py**.

**testing.py:**



Nel file **testing.py** viene effettuato il testing del modello di EE tramite la funzione testing\_entities. Questa funzione richiama la funzione run presente nel file **inference.py** definita nella classe SpacyTesting.