Documentazione tecnica

Che cos'è Reviù?

Reviù è un nome, una versione italianizzata della parola inglese **"Review"** dal significato di **"Recensione"**.

È anche molto più di questo. È a tutti gli effetti un assistente personale pensato per aiutare i possessori di siti specializzati in E-commerce a fare una precisa analisi delle recensioni dei propri prodotti da parte degli acquirenti, ma non solo. Tutti possono usare Reviù. È semplice, è veloce e può aiutarvi a capire i sentimenti delle persone a voi più care.

Ma effettivamente **cos'è**?

Reviù è un modello di intelligenza artificiale sviluppato con il famoso linguaggio di programmazione **"Python",** utilizzando specifiche tecniche di deep learning e più specificatamente creando una rete neurale ricorrente chiamata **LSTM(Long short-term memory).** Tutto è stato possibile grazie all'utilizzo di alcune delle più famose librerie open source scritte in Python, come :

- Pythorch, e alcune delle sue librerie minori:
    - torchtext
    - torchaudio
    - torchvision
- Pandas
- Spacy

IL TIPO DI MODELLO

Il modello da noi creato si può descrivere molto semplicemente con la definizione di **“Sentiment Analysis model”**, ovvero un sistema di intelligenza artificiale che da un input di tipo testuale deve riuscire a produrre un output di tipo numerico che riporta una valutazione del sentimento percepito nell’input, questa valutazione viene chiamata **predizione,** se nell’input viene passata una recensione o un qualsiasi scritto contenente delle emozioni positive o negative verso un determinato oggetto o una determinata persona o altro ancora , il modello deve essere in grado di riconoscere queste emozioni e dare un risultato conforme alla valutazione, nel nostro caso :

predizione di tipo negativo : valori che vanno da 0 a 0.50.
predizione di tipo positivo : valori che vanno da 0.50 a 1.

IL TIPO DI APPROCCIO

Perché una rete neurale?

La scelta della rete neurale è semplice da spiegare, in sé è uno dei pochi metodi per svolgere efficacemente un lavoro del genere, altri metodi meno conosciuti come **SVM, Naive Bayes o Random Forest**, potrebbero essere la soluzione in alcuni specifici casi ma data la possibilità di avere un numero di dati e risorse computazionali elevato è indicato l’utilizzo di una rete neurale, che se programmata bene riesce a gestire molti più dati e a dare risultati migliori.

Perché proprio una LSTM?

Tra la scelta delle diverse reti neurali, ci è subito venuto in mente l'utilizzo di una **RNN(Recurrent Neural Network)**, questo è dato dalla loro efficienza nel gestire dati sequenziali, come il linguaggio naturale. Tra le varie scelte di **RNN** la migliore è proprio la **LSTM**, capace di gestire **lunghe sequenze di dati** e di mantenere informazioni rilevanti nel tempo. Sfrutta Un'architettura particolare, costituita da una serie di unità LSTM, ognuna delle quali ha un'interfaccia simile a una cella di memoria. Queste unità sono collegate in una struttura sequenziale, in cui l'output di una unità LSTM viene passato come input alla successiva. Le unità LSTM lavorano insieme per imparare i pattern all'interno della sequenza di dati. Ogni unità LSTM ha diversi componenti fondamentali che contribuiscono al suo funzionamento:

Cell State (Stato della cella):
È la memoria a lungo termine della LSTM. La cella può memorizzare informazioni rilevanti per lunghe sequenze di dati attraverso il tempo.

Hidden State (Stato nascosto):
È il "pensiero attuale" della LSTM. Si basa sull'input corrente, sull'output precedente e sullo stato della cella per produrre un nuovo stato nascosto. Questo stato contiene informazioni rilevanti per l'output corrente e viene passato alla successiva unità LSTM.

È basata su un sistema di porte (Gates). Le LSTM utilizzano tre tipi di porte per regolare il flusso di informazioni:

Porta di dimenticanza (Forget Gate):
Decide quali informazioni rimuovere o dimenticare dalla cella di memoria.
Porta di input (Input Gate):
Decide quali nuove informazioni aggiungere alla cella di memoria.
Porta di output (Output Gate):
Decide quali informazioni inviare all'output basandosi sullo stato della cella.

Gestione del dataset

Il dataset è forse il più famoso e il più utilizzato in questi contesti: **“IMDB movie review dataset”**. Come suggerisce il nome è un dataset contenente circa **50.000** recensioni di film in **INGLESE,**(importante da sottolineare visto il nostro fallimento nel trovare un dataset anche molto più piccolo, in lingua italiana). All’interno dello script Python che descrive il modello da noi creato, il dataset viene gestito nel modo più documentato e sperimentato possibile, sono tecniche conosciutissime all’interno del mondo delle reti neurali. Effettivamente accade che:

Il dataset viene scaricato con la chiamata su un terminale di uno specifico comando che trovate nel file **ReadMe** della repository github di **Reviù**, di cui comunque lasciamo il link alla fine di questa relazione.

Questo Dataset che viene scaricato non è quello originale, ma una versione **“sgrezzata”**, che assolutamente non è da definire perfetta, ma appunto compie una pulizia approssimativa iniziale dei dati.

Viene successivamente definito un **tokenizer**, un elemento che ha come funzione quella di suddividere un testo in unità più piccole, come parole, quello di default proposto da **torchtext** semplicemente divide grazie agli spazi presenti nel testo, ma , fatte alcune prove, abbiamo riscontrato non essere completamente affidabile per cui abbiamo deciso di utilizzare Spacy, una libreria open source per l'elaborazione del linguaggio naturale, che offre un suo personale tokenizer che abbiamo riscontrato essere più efficace anche se non perfetto.

Il dataset viene quindi suddiviso in tre parti, famosissime quando si parla di questo tipo di operazioni :

1. train_data.
2. valid_data .
3. test_data.

Questi vengono usati rispettivamente per:

1. Allenare il modello .
2. Validare come il modello sia migliorabile con parole a lui sconosciute durante l'allenamento.
3. Testare ad allenamento concluso che il modello sia efficace con altre parole a lui sconosciute.

Viene infine creato un vocabolario dalla parte di dati riservata al training, ovvero una lista di tutte le parole riscontrate, riportate una sola volta.

La classe LSTM

La classe **LSTM** che si può trovare nel file **LSTM_MODEL** del nostro progetto, non è altro che la definizione di valori e comportamenti che il modello deve assumere durante il training, quindi tutte le manovre di preparazione del testo come **l'embedding** o la funzione forward (quello che le unità **LSTM** devono svolgere singolarmente per poi andare avanti ovvero passare alla prossima unità).

IL FILE TRAINING.PY

Questo è il file di allenamento del modello, nel file sono presenti funzioni per il calcolo dell'accuratezza durante il training e soprattutto le implementazioni di **Back Propagation**, ovvero il processo di calcolo del **gradiente** della funzione di perdita rispetto ai pesi del modello, in modo da poter aggiornare i pesi utilizzando un algoritmo di ottimizzazione come l'algoritmo di discesa del gradiente stocastica (SGD) o **Adam**(da noi utilizzato).

IL FILE APP.PY

All'interno di questo file c'è essenzialmente il caricamento unico iniziale del modello salvato alla fine del training, una funzione per predirre la probabilità e un'altra serie di comandi che non sono importanti per il modello, ma lo sono invece **per...**

IL SITO WEB

Questo ovviamente non è parte del modello, è invece un modo pensato da noi per verificare il funzionamento del modello stesso. All'interno della **repository GitHub** è quindi presente un sito web fittizio, hostato in locale, che rappresenta un sito di E-commerce, e allo stesso tempo è la dimostrazione di come potrebbe essere utile un'applicazione come Reviù. Effettivamente sono due cose diverse, Reviù è il modello, l'applicazione, il **BackEnd**, invece **MVBL-SHOP** è il modo grafico per far capire la potenza di Reviù a persone che non sono esperte di intelligenza artificiale o a chi, nel momento in cui Reviù diventasse pubblico, fosse interessato al suo utilizzo.

Il file index.html fa partire una pagina di login in cui due diversi account esistono:

- Account Admin : account del creatore o possessore del sito che è interessato a capire che recensioni stanno arrivando ai suoi prodotti e che valutazione hanno, questa ovviamente prodotta da Reviù.
    - credenziali di accesso: admin, admin.

- Account utente : account del semplice visitatore del sito che vuole lasciare una recensione del prodotto da lui acquistato, possibilmente lunga ed esaustiva.
    - credenziali di accesso: utente, utente.

In queste due rispettive pagine successive appaiono due situazioni simili , entrambe presentano una scelta di **4 specialissimi prodotti.**

Ovviamente per l'admin si aprirà una sorta di dashboard per controllare le ultime recensioni arrivate, e invece per l'utente si presenterà una scheda di recensione da compilare.

Scrivendo la propria personale recensione nella pagina delle recensioni per gli utenti, nella rispettiva pagina di dashboard per gli admin, verrà visualizzata in tempo reale : il contenuto della recensione e la predizione di Reviù.

Come effettuiamo gli scambi di informazioni?

Per quanto riguarda la comunicazione tra sito web e applicazione in Python, abbiamo usato l'ormai sempre più famoso framework **Flask.** Specificando funzioni di POST e di GET all'interno del file app.py. L'applicazione Flask si trova sulla porta **5000**.

Come fa ad essere così bello?

La base della pagina di login e tutte le animazioni di tilt delle immagini non sono state sviluppate da noi, sono state prese da una repository pubblica il cui link di riferimento trovate all'interno del file **ReadMe** della nostra **Repository,** che anch'essa vi linkiamo qua sotto.

RINGRAZIAMENTI

Il team MVBL ringrazia profondamente di questa opportunità tutta l'azienda SEEWEB e ovviamente il magnifico staff da noi conosciuto. Un ringraziamento speciale è da riservare al Sig.Marco Cristofanilli, persona di una gentilezza e umiltà immane, che durante i mesi di lavoro si è sempre reso disponibile a tutti i nostri capricci e le nostre pretese.

Un progetto pensato,
sviluppato,
e reso possibile da

Viselli Marco
Belli Leonardo