UNIVERSITÀ DI PISA

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

Studio e sviluppo di REST API per Sentiment Analysis

Relatori:

Prof. Mario Cimino

Prof.ssa Gigliola Vaglini

Dott. Giacomo Giorgi

Dott. Giacomo Iadarola

Candidato:

Diego Del Castello

Anno Accademico 2019/2020

Indice

Sommario

Negli ultimi anni i social media hanno invaso la nostra quotidianità, diventando per gli utenti il principale mezzo per scambiarsi opinioni, emozioni, idee e critiche riguardo ai più disparati oggetti di interesse. Queste opinioni cumulativamente, costituiscono una mole di dati impressionante, la cui analisi si presta a molteplici applicazioni che spaziano dai mercati azionari alle preferenze del consumatore, dalla politica alle scienze mediche.

I tweets, ed in generale ogni forma di comunicazione social, non solo rappresentano un efficace indicatore dei sentimenti della popolazione, ma sono anche un importante mezzo di condizionamento. Per questo una analisi approfondita effettuata sui social network, può aiutarci sia a comprendere gli effetti che eventi recenti hanno avuto sulla popolazione, sia può darci dei suggerimenti per cercare di prevedere accadimenti futuri.

Il primo scopo del progetto di tesi, era proprio quello di eseguire una sentiment analysis, con rilevamento della polarità, su una particolare classe di tweet. Il secondo invece, era di rendere questo processo più pratico ed adattabile agli ricerche di un utente, mediante lo sviluppo di delle API che consentano di sfruttare la rete addestrata per ottenere la popolarità di un tweet, un utente, o una keyword. Per affrontare le varie problematiche sono stati usati metodi di machine learning e deep learning quali la Convolutional Neural Network basata su Long Short-Term Memory, per addestrare un modello che ha raggiunto come valori accuracy 79%.

Capitolo 1

Introduzione

Negli ultimi anni l'informatica ha riscontrato un progresso esponenziale sia da un punto di vista software che hardware. Ciò ha dato la possibilità di riprendere lo sviluppo di un campo che aveva preso vita già negli anni '50, l'intelligenza artificiale, che però era stata un po' trascurata per una mancanza di risorse che rendeva difficile anche la risoluzione di problemi non eccessivamente complicati. Oggi però è tornata ad essere argomento di studio e di ricerca, trovando applicazione in diversi settori.

Inoltre l'ultimo anno è stato caratterizzato dalla terribile espansione del virus covid-19, fino a raggiungere persino il livello di pandemia globale. Questo evento ha avuto forti ripercussioni su ogni essere umano. Solo in Italia, più di 50 mila morti [1], calo a picco del tasso di occupazione, istruzione limitata dalle norme di sicurezza all'alternativa delle lezioni online. I tweet circolanti nell'anno corrente (2020), costituiscono un'ottima testimonianza delle reazioni e degli umori della popolazione a questo tragico evento. Per questo motivo può essere importante applicare i nuovi sviluppi dell'intelligenza artificiale ad un'analisi del sentimento verso uno degli eventi più nefasti che l'umanità si sia trovata ad affrontare nell'ultimo secolo [15], in modo da aiutarci sia a comprenderne a fondo gli effetti che ha avuto sulla popolazione, sia a cercare di prevedere simili accadimenti futuri.

Nel seguente capitolo vengono presentati sia gli obiettivi della tesi e i contributi che essa può fornire, sia la struttura dell'intera documentazione.

1.1 Contributi

La seguente documentazione riassume il risultato finale del mio percorso di tesi per il Corso di Laurea in Ingegneria Informatica. I punti chiave sono stati dapprima una fase di studio, ricerca e apprendimento dei concetti base dell'intelligenza artificiale, seguita da un richiamo e sviluppo delle capacità di programmazione in linguaggio Python. L'obiettivo era l'ampliamento delle mie conoscenze nel campo dell'intelligneza artificile e successivamente l'implementazione di alcune API utili per lo studio del Sentiment Analysis. Tali strumenti potranno essere messi a disposizione di determinati utenti che vogliano eseguire un'analisi del sentimento approfondita, in particolar modo a coloro che vogliano effettuarla relativamente ai sentimenti delle persone durante il periodo della diffusione del covid-19.

1.2 Struttura della tesi

La tesi è strutturata come segue:

- Il Capitolo 2 espone da un punto di vista teorico i concetti di background, ossia Intelligenza Artificiale (AI) e REST API.
- Il Capitolo 3 fornisce le indicazioni che sono state seguite per eseguire la sentiment analysis e le specifiche richieste per le API implementate.
- Il Capitolo 4 è dedicato all'analisi dettagliata dell'implementazione delle API e del classificatore.
- Il Capitolo 5 illustra alcuni esperimenti effettuati e i loro risultati.
- Il Capitolo 6 riporta le conclusioni del lavoro svolto.

Capitolo 2

Concetti di Background

Nel seguente capitolo vengono fornite le nozioni base che riguardano l'intelligenza artificiale, le reti neurali, il machine learning, il deep learning, l'analisi del linguaggio naturale, l'analisi del sentimento e le API secondo il modello REST.

2.1 Introduzione all'Intelligenza Artificiale (AI)

Per avere una comprensione diretta del significato di AI, è bene partire da cosa si intende per intelligenza umana. L'intelligenza umana spesso viene definita come la capacità di ragionare e risolvere problemi, imparando velocemente dalla propria esperienza [3]. Proprio quest'ultimo concetto, relativo all'apprendimento, è quello più importante e caratteristico dell'intelligenza. In sostanza i punti chiave che contraddistinguono l'intelligenza umana sono due: apprendimento in seguito all'esperienza, infatti basandosi su esperienze passate, l'essere umano è capace di combinare le informazioni in suo possesso e di riadattarle al fine di prendere decisioni funzionali alla situazione che sta affrontando; secondo, la capacità che consente di prendere decisioni rapidamente e prevedere azioni future.
D'altronde, nel mondo artificiale, anche una normale calcolatrice ha capacità di computazione e di risolvere problemi. Essa però non rappresenta di certo un esempio di AI, proprio perché non è presente alcun processo di apprendimento insito. Lo scopo dell'AI dunque, è quello di ricreare queste capacità dell'intelligenza umana e di combinarle con la capacità di calcolo dei sistemi informatici.

I concetti che contraddistinguono il processo attraverso il quale un essere umano prende decisioni sono:

- Apprendimento (Learning): Sbagliando si impara, si crea apprendimento cercando di minimizzare progressivamente gli errori.
- Ragionamento (Reasoning): Creare inferenze appropriate alle situazioni.
- Problem Solving: Una ricerca sistematica attraverso un range di possibili azioni per raggiungere un obiettivo predefinito.

2.2 Reti neurali

La base da cui partire per la riproduzione artificiale di questo processo è la comprensione biologica del sistema nervoso umano.

2.2.1 Reti neurali biologiche

Generalmente una rete neurale è costituita da:

- neuroni (soma): ricevono e processano le informazioni. In caso di potenziale di azione in ingresso superiore ad una soglia generano a loro volta degli impulsi in grado di propagarsi nella rete.
- neurotrasmettitori: responsabili degli impulsi nervosi.
- dendriti: principale via di comunicazione in ingresso (molteplici per ogni neurone).
- assoni: principale via di comunicazione in uscita.
- sinapsi: siti per il passaggio delle informazioni tra neuroni. Facilitano la trasmissione dell'impulso tra neuroni.

Il cervello umano rappresenta l'unità centrale del sistema nervoso, il cui compito è elaborare informazioni. Al fine di compiere tali operazioni, le reti biologiche si servono di un numero imponente di semplici elementi computazionali (neuroni) fittamente interconnessi e inclini

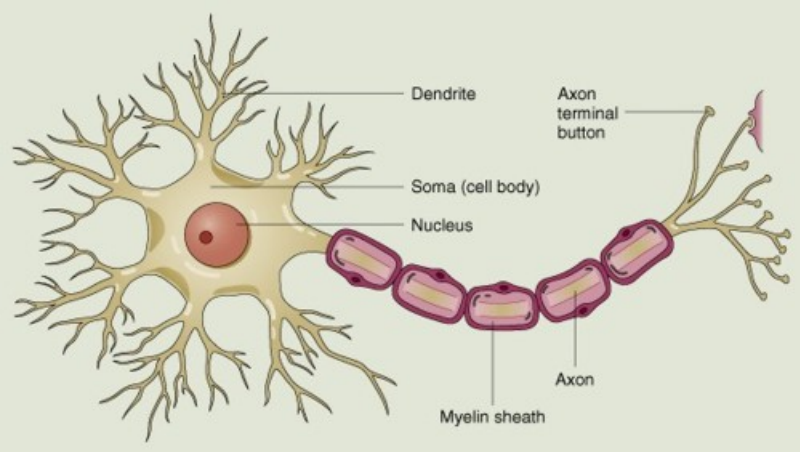

Figura 2.1: *Struttura di un neurone, immagine presa da http://www.gerard-siena.it/*

a variare la loro configurazione in risposta agli stimoli esterni. Un singolo neurone riceve simultaneamente segnali da diverse sinapsi. Il neurone misura il potenziale di azione di tali segnali stabilendo se è stata raggiunta la soglia di attivazione per generare un impulso nervoso ed in caso affermativo invia l'impulso al neurone successivo [8]. Il numero di sinapsi tra un neurone e il successivo può aumentare e diminuire a seconda degli stimoli che riceve, Più stimoli riceve, più connessioni sinaptiche avvengono. Un esempio di neurone è illustrato in figura 2.1.

Questo addattamento implica un apprendimento ed i modelli artificiali cercano di replicare questo meccanismo fondamentale della biologia.

2.2.2 Reti neurali artificiali

Le componenti di una rete neurale artificiale sono le seguenti:

- Strato di input: composto dai dati in ingresso.
- Strato/i nascosto/i: composto da una serie di strati di neuroni con il compito di apprendere determinate caratteristiche dei dati in ingresso al fine di risolvere il problema.
- Strato di uscita: composto dalle soluzioni del problema.
- Pesi dei neuroni: distribuiti in ogni strato, rappresentano la forza (legame) tra i neuroni della rete (forza delle sinapsi).

- Funzione di attivazione del neurone: somma pesata degli stimoli in ingresso al neurone.

Similmente a quanto avviene per le reti neurali biologiche, i suddetti neuroni ricevono in ingresso degli stimoli e li elaborano.
Lo strato di input si occupa di fornire i dati in ingresso ai neuroni, eventualmente adeguandoli alle richieste dei neuroni.
Lo strato nascosto è quello che effettivamente si occupa dell'elaborazione dei dati, e può essere composto da più livelli di neuroni.
La funzione di attivazione può essere molto sofisticata ma in un caso semplice si può pensare che i singoli ingressi vengano moltiplicati per un opportuno valore detto peso, il risultato delle moltiplicazioni viene sommato e se la somma supera una certa soglia il neurone si attiva attivando la sua uscita.
Il peso indica l'efficacia sinaptica della linea di ingresso e serve a quantificarne l'importanza, un ingresso molto importante avrà un peso elevato, mentre un ingresso poco utile all'elaborazione avrà un peso inferiore. Si può pensare che se due neuroni comunicano fra loro utilizzando maggiormente alcune connessioni allora tali connessioni avranno un peso maggiore [4].
Infine i dati vengono raccolti dallo strato di uscita della rete che elabora le soluzioni del problema, e le trasmette al livello successivo.

2.3 Machine Learning

Con il termine Machine Learning (ML) si intende una sottocategoria dell'AI costituita da un insieme di algoritmi statistici, che è stata così definita dal prof. Tom M. Mitchell della Carnagie Mellon University: "Il Machine Learning è lo studio di algoritmi informatici che migliorano automaticamente attraverso l'esperienza" [9].
Quando si parla di ML, spesso si pensa solo ad applicazioni in campi super-specifici, in settori di ricerca della scienza e della medicina, dell'ingegneria spaziale o di altri rami non comunemente compresi dalle persone comuni. Si tratta di un errore molto comune visto che l'apprendimento automatico presenta, invece, moltissime applicazioni di uso quotidiano. Naturalmente, per quotidiano si intende comunque un utilizzo legato alla tecnologia: un'applicazione classica di ML, ad esempio, è quella del riconoscimento vocale di cui sono dotati molti smartphone

e che permettono di attivare comandi tramite la propria voce. Ancora, molto comuni sono gli strumenti intelligenti che fanno uso di riconoscimento vocale per le diverse applicazioni di domotica, e che imparano nuovi vocaboli o modi di dire seguendo i comandi vocali che vengono impartiti. Un altro utilizzo dell'apprendimento automatico legato al comune utilizzo dei computer e della rete, ad esempio, è quello che permette alle aziende di realizzare pubblicità traccianti. Questo significa che, a seconda dell'utente di internet, vengono effettuate proposte pubblicitarie strettamente collegate agli interessi dell'utente stesso, le cui necessità e gusti vengono riconosciuti tramite l'analisi delle ricerche maggiormente effettuate in rete.

2.3.1 Machine Learning tradizionale

Il ML utilizza un algoritmo che il sistema adatta, solo dopo aver ricevuto un feedback umano. Un presupposto per l'utilizzo della tecnologia è l'esistenza di dati strutturati. Il sistema viene prima alimentato con dati strutturati e categorizzati e quindi capisce come classificare i nuovi dati a seconda del tipo. In base alla classificazione, il sistema esegue poi le attività programmate. Ad esempio è in grado di riconoscere un cane o un gatto in una foto e di spostare i file nelle cartelle corrispondenti. Dopo una fase iniziale di applicazione, l'algoritmo è ottimizzato dal feedback umano, che indica al sistema le classificazioni errate e le categorizzazioni corrette. Come verrà spiegato poi, nel paragrafo 2.3.2, esiste una sottocategoria del ML, più approfondita, il Deep Learning (DL), in cui i dati strutturati non sono necessari [11]. Nel DL, il sistema stesso identifica nei dati le caratteristiche distintive adeguate, senza la necessità di una categorizzazione dall'esterno. L'addestramento da parte di uno sviluppatore non è necessario. È il sistema stesso a controllare se le classificazioni cambiano a causa di un nuovo input o se ne vanno introdotte di nuove. In compenso però, il ML funziona già con un database controllabile, mentre il DL richiede molti più dati, è più complessa da implementare e richiede più risorse informatiche, risultando quindi molto più costosa del ML.

2.3.2 Deep Learning

La traduzione strettamente letterale di questo termine, è *apprendimento approfondito*. Ed è proprio questo il centro del suo significato, perché il Deep Learning, sottocategoria del Machine

Learning, non fa altro che creare modelli di apprendimento su più livelli [6]. Immaginiamo di esporre una nozione, la apprendiamo e subito dopo ne esponiamo un'altra. Il nostro cervello raccoglie l'input della prima e la elabora insieme alla seconda, trasformandola e astraendola sempre di più. Scientificamente, è corretto definire l'azione del deep learning come l'apprendimento di dati che non sono forniti dall'uomo, ma sono appresi grazie all'utilizzo di algoritmi di calcolo statistico. Questi algoritmi hanno uno scopo: comprendere il funzionamento del cervello umano e come riesca ad interpretare le immagini e linguaggio. L'apprendimento così realizzato ha la forma di una piramide: i concetti più alti sono appresi a partire dai livelli più bassi. Il deep learning ha compiuto passi da gigante, ottenendo risultati che, fino a qualche decennio fa, erano pura utopia. Tale successo è dovuto alle numerose conquiste in campo informatico, relative soprattutto alla sfera dell'hardware. Abbiamo visto come sia importante portare il calcolatore a fare esperienza su un quantitativo sempre maggiore di dati sensibili e come, fino a poco tempo fa, il tempo per ottenere tale addestramento fosse abbastanza elevato. Oggi, grazie all'introduzione delle GPUs, ovvero nuove unità che concorrono all'elaborazione dati, questo processo è diventato migliaia di volte più veloce [16]. Un altro importante aiuto è derivato dalla facilità di trovare numerose collezioni di dati (dataset) fondamentali per allenare il sistema. Il deep learning fa una cosa fondamentale: ci regala la rappresentazione dei dati, ma lo fa a livello gerarchico e soprattutto a livelli diversi tra loro, riuscendo a elaborarli e a trasformarli. Questa trasformazione è stupefacente perché ci consente di assistere ad una macchina che riesce a classificare i dati in entrata (input) e quelli in uscita (output), evidenziando quelli importanti ai fini della risoluzione del problema e scartando quelli che non servono. La rivoluzione apportata dal deep learning è tutta nella capacità, simile a quella umana, di elaborare i dati, le proprie conoscenze a livelli che non sono affatto lineari. Grazie a questa facoltà, la macchine apprende e perfeziona funzionalità sempre più complesse.

Recurrent Neural Network Le RNNs (Recurrent Neural Networks), sono un tipo di reti neurali profonde che sono state impiegate con successo nella risoluzione di problemi complessi, come ad esempio il riconoscimento della grafia [2] ed il riconoscimento vocale.
Le reti ricorrenti, rispetto alle Convolutional Neural Network e alle altre reti di tipo feed-forward, prevedono anche collegamenti verso il precedente livello di neuroni. I modelli più

comuni e diffusi, come ad esempio LSTM, prevedono anche collegamenti verso lo stesso livello. A ogni step della sequenza (ad esempio a ogni istante temporale t) il livello riceve oltre all'input al tempo t anche il suo output dello step precedente, al tempo t-1. Questo consente alla rete di basare le sue decisioni sulla storia passata (effetto memoria) ovvero su tutti gli elementi di una sequenza e sulla loro posizione reciproca.
In applicazioni che richiedono che l'input e/o l'output possano essere sequenze, per avere prestazioni ottimali è necessario che vengano considerate le relazioni esistenti tra gli elementi della sequenza. Un esempio di problema many to many, dove sia l'input che l'output sono delle sequenze, è la traduzione: l'input è una frase in inglese e l'output la sua traduzione in italiano.
Nelle reti RNNs tradizionali esiste un problema, definito vanishing gradient, che porta alla dissoluzione della memoria a lungo termine. Per ovviare a questo problema sono stati introdotte le reti LSTM (Long Short-Term Memory) [12], una tipologia di reti RNN che presentano due stati di memoria, uno a breve e uno a lungo termine.

2.4 Natural Language Processing

Attualmente, la quantità di dati testuali in circolazione sul web è immensa e, consideratone il volume, è necessario un certo grado di forza bruta computazionale per poterla maneggiare. L'AI risponde con efficacia a questa esigenza, essendo in grado di svolgere gran parte del lavoro di decifrazione con ottimi risultati, ad esempio riuscendo ad analizzare in tempi non limitanti anche milioni di tweet [10]. Il Natural Language Processing è una sottodisciplina del machine learning che si occupa dell'analisi del linguaggio naturale in forma testuale, dove per linguaggio naturale si intende quello umano, per distinguerlo da altri linguaggi detti artificiali. In particolare l'AI, viene utilizzata per scomporre il contenuto testuale in parti componenti (nomi, verbi, parole di sentiment ecc.), in modo da poter agevolare la macchina nella comprensione del sentiment dell'autore. Per molte attività di analisi del testo, ma in particolare per quelle su larga scala dove il testo richiede un'analisi relativamente standardizzata, il machine learning può rivelarsi un approccio eccellente perché in grado di costruire modelli analitici in modo autonomo.

Il processo di analisi del linguaggio naturale presenta notevoli difficoltà che si spiegano considerando le caratteristiche presenti nel linguaggio stesso:

- Ambiguità: La stessa parola o frase può prestarsi a interpretazioni diverse.
- Sottintesi: Spesso vengono omesse delle parole che si pensa siano deducibili dal contesto.
- Flessibilità: Per trasmettere lo stesso messaggio possono essere usati diversi modi di dire.
- Dinamicità: Data ad esempio dalla creazione di nuove parole.

La comprensione dei testi necessita la comprensione dei concetti ad essi associati, e quindi una conoscenza estesa della realtà e una grande capacità di manipolarla. Lo scopo del NLP è proprio quello di calcolare e rappresentare grandi quantità di informazione testuale per produrre intuizioni ed analizzare l'uso del linguaggio sia su scala sincronica (studia i fenomeni linguistici contemporanei tra loro, in un determinato stadio della loro storia comune, indipendentemente dalla loro evoluzione nel tempo) che diacronica (prende in considerazione le strutture e gli elementi linguistici nella loro evoluzione attraverso il tempo).
Il text mining è una tecnica di analisi testuale, definibile come "Il processo o la pratica di esaminare grandi raccolte di risorse scritte per generare nuove informazioni" [5], con l'obiettivo di predisporre l'informazione testuale "non strutturata" in informazione testuale "strutturata" al fine di eseguirne l'analisi. Per superare le difficoltà menzionate prima, il text mining sfrutta proprio le potenzialità degli algoritmi di NLP. Le operazioni effettuate possono essere sintetizzate nei seguenti step:

- Collezione dati: Prevede la raccolta e la selezione dei documenti che si ritiene opportuno analizzare.
- Pre-processing del testo: Ogni parola viene ridotta alla sua forma radice; vengono eliminate parole ritenute poco significative perché utilizzate spesso all'interno delle frasi (ad esempio le congiunzioni), e non forniscono informazioni utili a classificare il testo; viene suddiviso il testo in token, è relativamente semplice per lingue che adoperano gli spazi per delimitare le parole, molto complessa per lingue a sistema ortografico continuo.

- Applicazione tecniche di text mining: In questa fase i dati del testo (nomi, aggettivi, verbi, concetti, parole chiave...) sono estratti tramite le tecniche di estrazione del testo. Vengono sfruttate le enormi quantità di dati testuali per permettere ad algoritmi di NLP di trovare, identificare ed estrarre informazioni rilevanti di vario genere.

2.4.1 Sentiment Analysis

Un importante campo dell'elaborazione del linguaggio naturale è la sentiment analysis, che si occupa di identificare e classificare informazioni soggettive nei dati di testo. Potrebbe trattarsi di un'opinione, un giudizio o un sentimento su un particolare argomento o caratteristica di un prodotto. Essa è utilizzata in molteplici settori: dalla politica ai mercati azionari, dal marketing alla comunicazione, dall'ambito sportivo a quello delle scienze mediche e naturali, dall'analisi dei social media alla valutazione delle preferenze del consumatore.
Sono possibili diversi approcci. Un primo esempio è quello dell'analisi lessicale basata sulla presenza di parole influenti non ambigue come contento, triste, impaurito, annoiato, a cui vengono assegnate arbitrariamente una probabile affinità a emozioni particolari [14]. Un'alternativa sono le analisi basate su ML, approfondite nel paragrafo 3.1, ma sono possibili anche soluzioni ibride.
Il tipo più comune di analisi del sentimento è chiamato "rilevamento della polarità" e implica la classificazione di un'affermazione come "positiva", "negativa" o "neutra".

2.4.2 Word Embedding

Un problema fondamentale nelle applicazioni di Natural Language Processing riguarda la conversione degli elementi testuali in numeri, in modo da renderli elaborabili da un calcolatore. Una soluzione molto utilizzata per risolvere questo problema consiste nel tradurre una parola in un vettore di numeri reali. Con word embedding indichiamo una serie di tecniche nell'ambito nel Natural Language Processing volte ad associare a tutte le parole di un vocabolario un vettore numerico. Lo scopo del vettore associato ad una determinata parola è di evidenziarne il contesto: gli elementi del vettore rappresentano la vicinanza semantica della parola stessa rispetto alle altre presenti nel vocabolario. Per cui, per conoscere la vicinanza semantica tra

due parole, basta analizzare la differenza vettoriale tra i vettori associati alle due parole.
E' possibile addestrare una rete neurale, tramite modelli statistici e linguistici, a generare dei vettori da associare ad ogni parola, ottimizzati cercando di ridurre la differenza tra i valori predetti e quelli reali. L'introduzione nel 2018 di nuovi metodi di word embedding, ha apportato notevoli miglioramenti alla capacità dei modelli di NLP di modellare il linguaggio naturale [7].
GloVe è un algoritmo di apprendimento non supervisionato, facente parte della classe di modelli word embedding, utilizzato per ottenere rappresentazioni vettoriali per parole. La costituizione dei vettori viene eseguita basandosi su statistiche che rilevano le co-occorrenze parola-parola. Nello spazio vettoriale generato le sottostrutture lineari tra i vettori rappresentanti le parole, approssimano efficientemente la semantica delle parole stesse.

2.5 REST API

Un'interfaccia di programmazione delle applicazioni (API) è un set di definizioni e protocolli per la compilazione e l'integrazione di software applicativi. Può essere considerata come un contratto tra un fornitore di informazioni e l'utente destinatario di tali dati: l'API stabilisce il contenuto richiesto dal consumatore (la chiamata) e il contenuto richiesto dal produttore (la risposta). Ne è un esempio l'API di un servizio meteorologico, in cui l'utente invia una richiesta contenente un codice postale al quale il produttore fornisce una risposta in due parti, dove la prima indica la temperatura massima e la seconda la minima.
Se, in altre parole, si desidera interagire con un computer o un sistema per recuperare informazioni o eseguire una funzione, un'API facilita la comunicazione con il sistema che può così comprendere e soddisfare la richiesta.L'API funge quindi da elemento di intermediazione tra gli utenti o i clienti e le risorse che questi intendono ottenere. È anche un mezzo con il quale un'organizzazione può condividere risorse e informazioni assicurando al contempo sicurezza e controllo, poiché stabilisce i criteri di accesso. L'utilizzo dell'API inoltre non impone all'utente di conoscere le specifiche con cui le risorse vengono recuperate o la loro provenienza [13].
REST è un insieme di principi architetturali, non un protocollo né uno standard. Chi sviluppa API può implementare i principi REST in diversi modi. Quando una richiesta viene inviata tramite un'API RESTful, questa trasferisce al richiedente uno stato rappresentativo della risor-

sa. L'informazione, o rappresentazione, viene consegnata in uno dei diversi formati tramite HTTP: JSON (Javascript Object Notation), HTML, XLT o testo semplice. Il formato JSON è uno dei più diffusi, perché indipendente dal linguaggio e facilmente leggibile da persone e macchine.
Affinché un'API sia considerata RESTful, deve rispettare i criteri indicati di seguito.

- Un'architettura client-server composta da client, server e risorse, con richieste gestite tramite HTTP.
- Una comunicazione client-server stateless, che quindi non prevede la memorizzazione delle informazioni del client tra le richieste; ogni richiesta è distinta e non connessa.
- Dati memorizzabili nella cache che ottimizzano le interazioni client-server.
- Un'interfaccia uniforme per i componenti, in modo che le informazioni vengano trasferite in una forma standard. Ciò impone che:
  - le risorse richieste siano identificabili e separate dalle rappresentazioni inviate al client;
  - le risorse possano essere manipolate dal client tramite la rappresentazione che ricevono poiché questa contiene le informazioni sufficienti alla manipolazione;
  - i messaggi autodescrittivi restituiti a un client contengano le informazioni necessarie per descrivere come il client deve elaborare l'informazione;
  - le informazioni siano ipermediali, ovvero accedendo alla risorsa il client deve poter individuare, attraverso hyperlink, tutte le altre azioni disponibili al momento.
- Un sistema su più livelli che organizza ogni tipo di server (ad esempio quelli responsabili della sicurezza, del bilanciamento del carico, ecc.) che si occupa di recuperare le informazioni richieste in gerarchie, invisibile al client.
- Codice on demand (facoltativo): la capacità di inviare codice eseguibile dal server al client quando richiesto, estendendo le funzioni del client.

Sebbene l'API REST debba essere conforme a questi criteri, il suo impiego è comunque considerato più semplice rispetto a quello di un protocollo prescrittivo come SOAP (Simple Object Access Protocol), che presenta requisiti specifici come la messaggistica XML e la conformità integrata di sicurezza e transazioni, che lo rendono più lento e pesante.
Al contrario, REST è un insieme di linee guida applicabili quando necessario, il che rende le API REST più rapide e leggere, ottimali quindi per l'Internet of Things (IoT) e lo sviluppo di app mobili.

Capitolo 3

Metodologia

Nel seguente capitolo viene esposta la metodologia utilizzata per eseguire la sentiment analysis e le specifiche richieste alle API.

3.1 Sentiment analysis

Per eseguire la sentiment analysis è stato utilizzato un modello di tipo RNN. Come detto nel paragrafo 2.4, la sentiment analysis, è proprio uno dei problemi su cui gli algoritmi RNN ottengono risultati migliori. L'addestramento della rete neurale è stato effettuato sfruttando un dataset, opportunatamente etichettato, in cui fosse presente una associazione testo-sentimento. Nello specifico, è stato utilizzato il dataset sentiment140 messo a disposizione dalla piattaforma kaggle.[1]
La rete neurale è stata cosi strutturata:

- Embedding Layer: Primo livello della struttura, crea un Embedding vector per ogni input.
- Conv1D: Un layer di tipo CNN composto da 64 neuroni.
- LSTM: Layer di tipo LSTM composto da 64 unità neuronali.

[1] https://www.kaggle.com/kazanova/sentiment140

- Dense: Due strati con 512 neuroni a testa, più lo strato finale, quello di output, con un solo livello a cui corrispondono le due etichette positivo e negativo.

Nel progetto di tesi, il dataset è stato suddiviso in training set (costituito dall'80% dei dati complessivi) e in test set (costituito dal restante 20%). Il training set è stato utilizzato per addestrare la rete LSTM. Successivamente è stato eseguito anche il testing sul test set.

3.2 Specifiche delle API

Si vogliono realizzare delle API per un servizio web sul sentiment analysis di tweet in modo che possa essere integrato dinamicamente su piattaforme o analisi statistiche più ampie. Ognuna dovrà essere composta da URL, porta, path, nome, ed essere disponibile all'indirizzo <URL:porta>/path/nome .

Si vuole implementare tali API secondo lo standard REST, inserendo un header appropriato, ad esempio *Content-Type:application/json* in modo da mandare una richiesta curl come riportato nel Listing 3.1:

Listing 3.1: Chiamata API

```
curl -X GET -d "keyword=cybesecurity" '<URL:port>/path/sentimentKeyword'
```

Il server risponderà poi inviando un body composto dai campi riportati in tabella 3.1:

STATUS	Nome identificativo dello stato di uscita
CODE	Codice numerico identificativo
MESSAGE	Un messaggio di uscita di facile interpretazione per l'utente
DATA	Dati aggiuntivi

Tabella 3.1: Format del body di risposta inviato dal server

Ad ogni situazione di errore è associato un codice diverso.

3.3 Lista dei possibili codici di risposta

Nella Tabella 3.2 viene riportato uno schema di tutte le possibli situazioni di errore, in forma tabellare, con il codice d'errore, il nome associato allo stato ed una breve descrizione del singolo caso.

CODICE	STATO	DESCRIZIONE
200	OK	
400	MISSING INPUT PARAMETER	Utilizzato per la mancanza di uno o più parametri al momento della richiesta.
401	INCORRECT INPUT PARAMETER	Utilizzato per un errore di formattazione, come ad esempio
402	NOT FOUND	Utilizzato come avviene comunemente con il codice 404 del protocollo HTTP, ossia quando un'informazione cercata non esiste.
444	GENERAL ERROR	Utilizzato per qualsiasi altra forma di errore.

Tabella 3.2: Codici risposta

3.3.1 Lista delle API richieste

In questo paragrafo vengono elencate le API richieste dalle specifiche di progetto indicando per ciascuna: il nome, il tipo, una descrizione funzionale, il modo corretto di eseguire la richiesta e i parametri richiesti, mentre per l'osservazione anche di un output standard della risposta nel caso in cui non ci siano errori, si rimanda ai listing A.1, A.2 e A.3, in appendice A

Sentiment Text

Tipo: GET

Descrizione: L'utente invia uno o più tweet e riceve in risposta il sentimento rilevato dalla rete.

Richiesta: Il modo corretto di eseguire la richiesta è riportato nel Listing 3.2

Listing 3.2: Esempio di comando input per la Sentiment Text

```
curl -X GET -d {"text": "tweet1 textual information", "id":
    1050118621198921728}, {"text": "tweet2 textual information","id":
    1050118621198921729} <MISSING_URL>:<MISSING_PORT>/<MISSING_PATH>/
    sentimentText
```

Parametri: lista di coppie text, id indicanti rispettivamente il testo di un tweet e il suo ID all'interno del dataset. Entrambi sono di tipo stringa. L'ID è facoltativo.

Sentiment Keyword

Tipo: GET

Descrizione: L'utente invia una keyword e riceve in risposta il sentimento rilevato dalla rete sui tweet in cui essa è contenuta.

Richiesta: Il modo corretto di eseguire la richiesta è riportato nel Listing 3.3

Listing 3.3: Esempio di comando input per la Sentiment Keyword

```
curl -X GET -d {"keyword": "example_keyword"} <MISSING_URL>:
<MISSING_PORT>/<MISSING_PATH>/sentimentKeyword
```

Parametri: keyword, non facoltativo, di tipo stringa.

Sentiment User

Tipo: GET

Descrizione: L'utente invia un user id e riceve in risposta il sentimento rilevato dalla rete sui tweet pubblicati da quell'utente.

Richiesta: Il modo corretto di eseguire la richiesta è riportato nel Listing 3.4

Listing 3.4: Esempio di comando input per la Sentiment User

```
curl -X GET -d {"user_id": "example_user"} <MISSING_URL>:
<MISSING_PORT>/<MISSING_PATH>/sentimentUser
```

Parametri: user_id, non facoltativo, di tipo stringa.

Capitolo 4

Implementazione

Per il lavoro svolto è stato scelto di addestrare un modello di Natural Language Processing e di sviluppare delle API che producessero i loro risultati utilizzando tale modello. In questo capitolo verranno illustrati i passaggi caratterizzanti dell'implementazione riportando anche il codice in appendice nei listing **??**.
Tanto per cominciare si è partiti suddividendo l'intero lavoro in due script: *TSA_trainig.py* e *TSA_testing_and_API.py*. Nel primo è implementato il classificatore, con tutte le funzioni necessarie all'addestramento e test della rete. Il secondo modulo, invece, contiene una funzione che carica il modello utilizzato per testare la rete e le tre API richieste, già esaminate nel paragrafo 3.2.

4.1 Implementazione Classificatore

Il file *TSA_trainig.py* contiene l'implementazione del modello di addestramento della rete. Dopo una iniziale importazione delle librerie si parte subito seguendo la procedura esposta nel paragrafo2.4: collezione dati, pre-processing del testo, applicazione di tecniche di text mining. Per collezionare i dati viene dichiarata la variabile df (Data Frame), che conterrà tutto il dataset, mantenendo però solo le colonne sentiment e text. Le altre vengono scartate perché non necessarie ai nostri scopi.
Si procede dunque alla fase di pre-processing del testo e dunque alla riduzione alla radice di ogni parola e eliminazione di articoli, congiunzioni e altre parole poco significativo. Per sem-

plificare queste operazioni è stata utilizzata la libreria NLTK (Native Language Tool Kit) di python. E' stata definita una funzione, preprocess, che è stata usata per ripulire i campi di testo di tutti i tweet del dataset.

In seguito, dopo aver suddiviso i dati in un training set (80% dei dati) e in un test set (20%), viene effettuata anche la tokenizzazione. La funzione train_test_split, effettua questa scissione, salvando il training set nella variabile train_data e il test set nella variabile test_data. Utilizzando la funzione Tokenizer, della libreria keras, è stato creato un oggetto tokenizer che crea un indice per mappare ogni parola nel corpus data. L'oggetto tokenizer potrà quindi essere utilizzato per convertire qualsiasi parola in un numero (l'indice corrispondente). Viene infine salvato il modello in un file tokenizer.pickle tramite la funzione pickle.dump. Tale file servirà poi, in fase di deploying, per tokenizzare con lo stesso vocabolario i tweets da analizzare.

E' inoltre necessario, affinché la procedura sia completa, che i dati in input al nostro modello abbiano tutti lo stesso formato. Considerando che il campo testo dei tweet può essere di dimensione variabile, le sequenze di indici in cui sono stati trasformati i tweet saranno di dimensioni diverse. Viene perciò utilizzata la funzione pad_sequences della libreria keras che rende ciascuna sequenza di una lunghezza costante MAX SEQUENCE LENGTH.

Viene poi utilizzato il transfer learning, scaricando Glove, per evitare di dover addestrare il word embedding. Viene associata ad ogni parola presente nel dizionario ottenuto con il tokenizer, il vettore corrispondente alla parola stessa nel dizionario di GloVe, creando una matrice embedding. I vettori costituenti la embedding matrix sono i pesi del primo layer della nostra rete. Seguiranno altri tre livelli di neuroni che completano l'architettura del modello: Conv1D Layer, LSTM e Dense, composto da molti strati completamente connessi.

Come accaduto per il tokenizer, il modello e i pesi della rete vengono salvati affinché siano poi riutilizzabili in fase di deploying per fare la prediction dei tweets da analizzare.

A questo punto comincia l'addestramento vero e proprio, con un numero di epoche fissato a duecento.

Infine il modello produce un punteggio di previsione compreso tra 0 e 1 per ogni messaggio del dataset considerato. E' stato impostato 0.5 come valore di soglia: se il punteggio è superiore, il messaggio viene etichettato come positivo, altrimenti come negativo.

4.2 Implementazione API

Il file *TSA_testing_and_API.py* contiene l'implementazione delle API. Anche in questo caso si parte da una preliminare, consueta, importazione delle librerie. Subito dopo, il primo passo è la definizione di una funzione, predict_tweet, che esegue la predizione del sentimento ad ogni testo ricevuto. Essa carica il tokenizer salvato e definito con i dati del training set, il quale viene utilizzato per effettuare la conversione del testo di input. Subito dopo effettua il caricamento anche del modello della rete e dei pesi addestrati. Infine viene effettuata la predizione del sentimento.

Segue poi la definizione di tre handler utilizzati per la gestione degli errori già discussi nel paragrafo 3.3 e successivamente, l'implementazione delle tre API.
La prima API, chiamata sentimentText, dopo un preliminare controllo sulla presenza e formato dei parametri, utilizza la predict_tweet per ottenere la positività dei testi ricevuti in input. Poiché però, per ogni testo si vuole ottenere come risposta non solo la positività, o meglio, la probabilità che un testo sia positivo, ma anche la negatività, ossia la probabilità che un testo sia negativo, e poiché nel nostro studio sono presenti solo due classi, quest'ultima può essere calcolata come 1-p dove p indica la positività. A questo punto per concludere, sia la positività che la negatività vengono inseriti all'interno di un campo data, passato poi all'interno del messaggio di risposta in formato JSON.

La seconda API, chiamata sentimentKeyword, dopo un preliminare controllo sulla presenza e formato dei parametri, va a ricercare all'interno del dataset, quali sono i tweet da analizzare. Per farlo controlla per ogni tweet del dataset se contiene la keyword passata all'interno della richiesta. I tweet risultanti vengono passati alla funzione predict_tweet. Dei risultati viene fatta una media, calcolata la complementare ed insieme vengono inserite all'interno di un dictionary data, che viene restituito all'interno del messaggio di risposta in formato JSON.

La terza API, chiamata sentimentUser, dopo un preliminare controllo sulla presenza e formato dei parametri, va a ricercare all'interno del dataset, quali sono i tweet da analizzare. Per farlo controlla per ogni tweet del dataset se è stato scritto dall'utente passato all'interno della

richiesta. Dopodiché il comportamento è del tutto analogo alla sentimentKeyword. I tweet risultanti vengono passati alla funzione predict_tweet; i risultati vengono riassunti in un'unica variabile, risultato della media di tutti i valori ottenuti; viene calcolata la complementare ed insieme vengono inserite all'interno di un dictionary data, che viene restituito all'interno del messaggio di risposta in formato JSON.

Capitolo 5

Esperimenti

In questa sezione viene descritto come sono stati costruiti gli esperimenti, a partire da una descrizione dettagliata del dataset utilizzato, seguita da alcune considerazioni sui trainig set e test set, e i risultati ottenuti di accuracy e loss.

5.1 Struttura del dataset

Il dataset su cui si è basato il lavoro di addestramento e testing della rete è il sentiment140 messo a disposizione dalla piattaforma kaggle. Questo vede un totale di 1.600.000 tweet, ripartiti perfettamente a metà tra positivi e negativi.
Per ogni tweet, oltre che il testo e il sentimento ad esso associato, sono riportati: l'utente da cui è stato pubblicato; l'ID numerico associato; il timestamp della data di pubblicazione; un flag di query non importante ai nostri scopi.
L'intero dataset è stato suddiviso in un training set, composto da 1.280.000 tweet (80% del totale), e test set, realizzato utilizzando i restanti 320.000. I tweet che compongono il test set non sono stati utilizzati per addestrare il classificatore, per evitare situazioni di overfitting. Ricordiamo che la distribuzione dei tweet tra positivi e negativi in sentiment140 è perfettamente bilanciata, tuttavia la funzione train_test_split, utilizzata per suddividere sentiment140, effettua un rimescolamento del dataset prima di suddividerlo per cui, dei 320.000 tweet, 160542 sono positivi e 159458 sono negativi.
Per valutare l'accuratezza raggiunta dal classificatore sono stati utilizzati i parametri classici,

	PRECISION	RECALL	F1-SCORE	ACCURACY	SUPPORT
POSITIVE	0.80	0.75	0.77	-	160542
NEGATIVE	0.76	0.81	0.79	-	159458
MACRO AVG	0.78	0.78	0.78	0.78	320000
WEIGHTED AVG	0.78	0.78	0.78	0.78	32000

Tabella 5.1: Classification report

ovvero accuracy e loss. In più, conoscendo il sentiment reale dei tweet, è stato possibile utilizzare i parametri generati dalla funzione classification report.

5.1.1 Metriche utilizzate

In tabella 5.1 è riportato il classification report, nel quale sono delineati anche, sia per i positivi che per i negativi, anche i valori di precision, recall e f1-score.
Questi si basano sull'analisi di True Positive (TP), True Negative (TN), False Positive (FP) e False Negative (FN). Con TP e TN vengono indicati tutti quei valori positivi e negativi, predetti correttamente dalla rete. I FP sono i valori negativi, che però sono stati predetti dalla rete come positivi. I FN sono infine i valori positivi predetti come negativi.
La precision è il rapporto tra i TP e il totale delle predizioni positive. Ciò vuol dire che sarà tanto più elevata quanto più sarà ridotto il numero dei FP.
La recall è il rapporto tra i TP e il totale dei dati effettivamente positivi (quindi TP + FN). Ciò può dare una diversa indicazione su quante volte abbiamo predetto correttamente un dato.
Infine f1-score è una media ponderata tra i valori di precision e recall. F1-score fornisce un dato ancora più accurato dell'accuracy, soprattutto se le classi della rete sono distribuite in maniera irregolare. Non è il nostro caso, ma comunuque per completezza riportiamo anche questo dato.

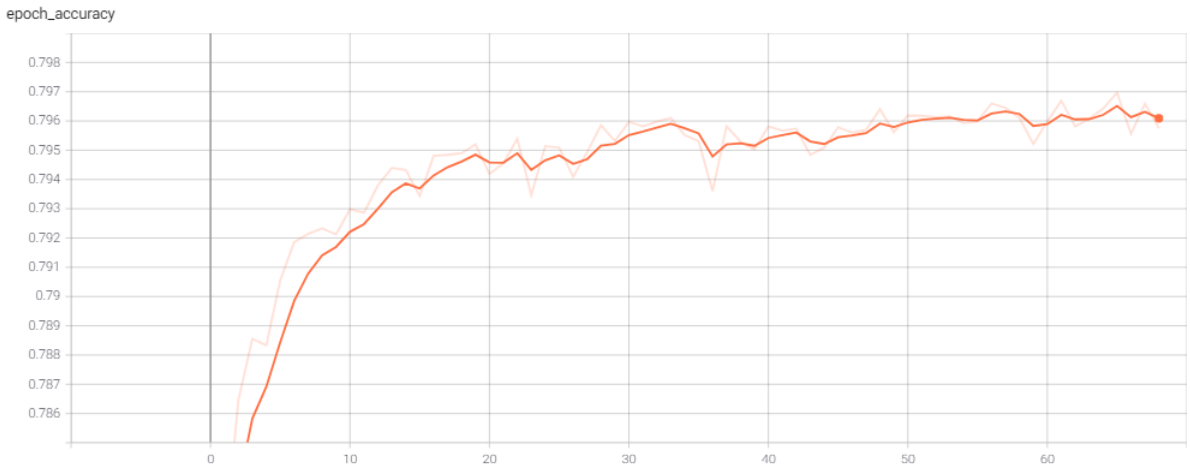

Figura 5.1: *Evoluzione dell'accuracy attraverso le varie epoche*

5.1.2 Risultati

Dal classification report di tabella 5.1 si possono osservare anche i risultati ottenuti che vedono, in un calcolo su 68 epoche, un accuracy di 0,78, e una precision, recall e f1-score, che raggiungono tutte un valor medio di 0,78. In figura 5.1, in cui è riportata su grafico l'evoluzione dell'accuracy nel corso di 68 epoche, si può notare di come questa cresca di epoca in epoca, andando oltre il valore di 0,78. Ciò vuol dire che il modello che abbiamo creato funziona quasi nell'80% dei casi.

Nel nostro caso abbiamo una precision che raggiunge un valor medio del 78%. Dal grafico in figura 5.2, in cui è riportata su grafico l'evoluzione della loss nel corso di 7 epoche, si può notare di come questa diminuisca di epoca in epoca, arrivando ad oscillare tra i valori di 0,438 e 0,436.

Figura 5.2: *Evoluzione della loss attraverso le varie epoche*

Capitolo 6

Conclusioni

E' stato sviluppato un framework con lo scopo di valutare il sentiment dei tweets, in particolare per monitorare l'opinione social riguardo tematiche sensibili. Il framework è composto da due moduli che si occupano di addestrare una rete per un modello di Sentiment Analysis e caricare delle API su un server per consentire ad altri utenti di avere un'agevolazione nell'analisi dei tweet. Si tenga conto che la classificazione del sentiment di un testo, è un'attività umana molto complessa e la cui astrazione presenta enormi difficoltà per un calcolatore. Anche le persone spesso presentano delle difficoltà nell'eseguire questa operazione. Ad esempio è difficile riconoscere l'ironia: un tweet ironico con lo scopo di criticare l'oggetto del messaggio, potrebbe erroneamente essere interpretato come positivo se decontestualizzato, mentre invece il sentiment reale dell'autore è negativo. A tal proposito uno sviluppo possibile per migliorare il classificatore, potrebbe essere quello di inserire 3 neuroni sul livello di output piuttosto che uno solo, consentendo di sviluppare tre differenti classi: positivo, neutro e negativo. La fascia intermedia, ossia quella neutrale, probabilmente è più indicata per rappresentare il sentiment dei tweet con un punteggio di sentiment vicino al valore di soglia. Inoltre nel dataset dei tweet collezionati, sono presenti dei tweet in lingue diverse dall'inglese, su cui i processi di embedding e tokenizzazione utilizzati non hanno effetto. Eliminando questi tweet potrebbe aumentare la precisione ottenuta. In futuro il classificatore potrebbe essere utilizzato per effettuare un'analisi più approfondita sui dati raccolti, sviluppando dei tool che consentano una suddivisione dei tweet per oggetto d'interesse e magari una visualizzazione grafica del sentiment sui sottogruppi generati, in modo da facilitare anche la compilazione delle le richieste

API da fare al server.

Appendice A

Appendice

Listing A.1: Sentiment Text - Risposta in caso di assenza di errori

```
Header: Content-Type:application/json
Body: {
        'status': 'STATUS.OK'
        'code': 200
        'data': [
                    {
                        "id": 1050118621198921728,
                        "positive": 0.9,
                        "negative": 0.1
                    }, {
                        "id": 1050118621198921729',
                        "positive": 0.2,
                        "negative": 0.8
                    }
                ]
        }
```

Listing A.2: Sentiment Keyword - Risposta in caso di assenza di errori

```
Header: Content-Type:application/json
Body: {
        'status': 'STATUS.OK'
        'code': 200
```

```
        'data': {
                "keyword": "example_keyword",
                "positive": 0.9,
                "negative": 0.1
                }
        }
```

Listing A.3: Sentiment User - Risposta in caso di assenza di errori

```
Header: Content-Type: application/json
Body: {
        'status': 'STATUS.OK'
        'code': 200
        'data': {
                "user_id": "example_user",
                "positive": 0.9,
                "negative": 0.1
                }
        }
```

Bibliografia

[1] Ministero della Salute. Covid-19-situazione nel mondo, 2020.

[2] Patrick Doetsch, Michal Kozielski, and Hermann Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 279–284. IEEE, 2014.

[3] Linda S Gottfredson. g theory: How recurring variation in human intelligence and the complexity of everyday tasks create social structure and the democratic dilemma. 2018.

[4] Daniel Graupe. *Principles of artificial neural networks*, volume 7. World Scientific, 2013.

[5] Anne Kao and Steve R Poteet. *Natural language processing and text mining*. Springer Science & Business Media, 2007.

[6] John D Kelleher. *Deep learning*. MIT press, 2019.

[7] Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14, 2016.

[8] Irwin B Levitan, Irwin B Levitan, Leonard K Kaczmarek, et al. *The neuron: cell and molecular biology*. Oxford University Press, USA, 2002.

[9] Tom Michael Mitchell. *The discipline of machine learning*, volume 9. Carnegie Mellon University, School of Computer Science, Machine Learning ..., 2006.

[10] Daniel E O'Leary. Artificial intelligence and big data. *IEEE intelligent systems*, 28(2):96–99, 2013.

[11] Nikolaos G Paterakis, Elena Mocanu, Madeleine Gibescu, Bart Stappers, and Walter van Alst. Deep learning versus traditional machine learning methods for aggregated energy demand prediction. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2017.

[12] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[13] Vijay Surwase. Rest api modeling languages-a developer's perspective. *Int. J. Sci. Technol. Eng*, 2(10):634–637, 2016.

[14] Harsh Thakkar and Dhiren Patel. Approaches for sentiment analysis on twitter: A state-of-art study. *arXiv preprint arXiv:1512.01043*, 2015.

[15] MARCO TUTINO and MAURO PAOLONI. *L'Italia ai tempi del Covid-19 Tomo II.* CEDAM, 2020.

[16] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.

Ringraziamenti

Voglio ringraziare i ragazzi Giacomo Iadarola e Giacomo Giorgi, dottorandi presso l'IIT del CNR di Pisa, e i professori Cimino e Vaglini, per avermi fornito le conoscenze necessarie allo sviluppo dell'elaborato e per aver reso possibile l'intero lavoro di tesi. Al di là di quello che può essere come tesi, come progetto e come specifiche per coloro che si occuperanno della manutenzione e del miglioramento del prodotto, per me è pur sempre iniziato tutto come una tesi, scelta per apprendere, arricchire le mie conoscenze, nonché concludere in bellezza questa parte del mio percorso di studi, e arrivato alla fine mi sento di dire che è stata totalmente un'esperienza positiva sotto tutti gli aspetti.