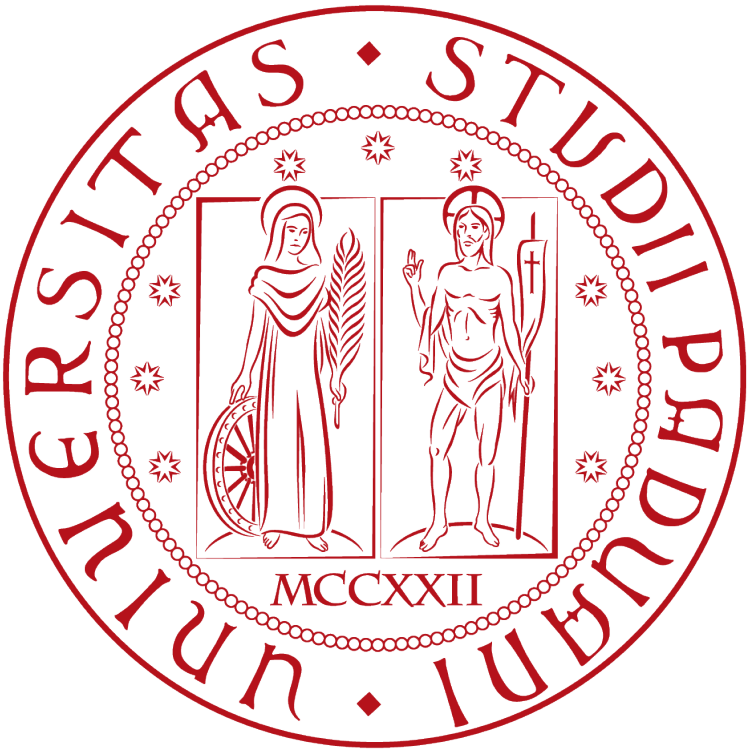

Relazione progetto di Intelligenza Artificiale

Informazioni sul progetto

Redatto	Francesco Corti - 1142525 Giovanni Sorice - 1144558
Referente	Giovanni Sorice - 1144558 giovanni.sorice@studenti.unipd.it

Link al sito

http://tecweb1819.studenti.math.unipd.it/gsorice/

Descrizione

Documento riportante le informazioni relative al progetto di intelligenza artificiale.

Indice

1 Introduzione

1.1 Scopo dell'analisi

Questo progetto consiste nell'applicazione e confronto di due metodi apprendimento supervisionato, per sottolinearne le potenzialità e i problemi.

1.2 Il problema dello spam

Il problema dello spam affligge da molti anni tutte le persone che dispongano di una casella di posta elettronica oppure di uno smartphone.
In passato è stato constato come l'eliminazione manuale dei messaggi spam, data la quantità di messaggi inviati, presentasse costi di tempo insostenibili.
Questo ha portato ad uno sviluppo di tecniche algoritmiche che permettessero di classificare automaticamente un messaggio ricevuto, come spam o ham.
Si è però scoperto che un approccio di tipo *offline learning*, presentava dei problemi.
Gli spammer, persone o bot che spediscono messaggi spam, riuscivano a modificare i messaggi in modo da renderli classificati come ham dai sistemi anti-spam presenti.
Questo era possibile in quanto gli algoritmi non evolvevano nel tempo, cambiando la struttura del messaggio di spam questo veniva erroneamente identificato come un messaggio non spam.
Si è quindi passati a un approccio di tipo online che si è visto essere quello più ottimale.
Gli algoritmi in questo modo non smettono di imparare una volta terminato l'input dei dati ma evolvono nel corso del tempo imparando a classificare nuove tipologie di messaggi come spam.
Abbiamo scelto questo problema dato che si adatta molto bene all'applicazione di algoritmi supervisionati.

2 Panoramica

Molte aziende e molti esperti del settore hanno studiato il problema dello spam, ciò ha portato ad algoritmi sempre più affidabili ed efficienti.
Gli algoritmi presi in considerazione all'interno del nostro progetto sono stati:

- Logistic Regression;
- Neural Network (con e senza Dropout).

Maggiori info: Dropout: A Simple Way to Prevent Neural Networks fromOverfitting.
Lo sviluppo di tali algoritmi è stato svolto all'interno di Google Colab.

*****Scrivere obiettivi********* Gli obiettivi principali che ci siamo prefissati trattano come la fase di progettazione e sviluppo doveva avvenire (cioè in modo corretto, efficace ed efficiente) e dei confronti in termini di velocità di training, precision e recall. *****Scrivere Fasi e tempi di lavoro********** Abbiamo deciso di dividere il progetto in 7 fasi:

- Scelta e studio dell'ambito
- Scelta e studio del problema
- Scelta e studio degli algoritmi
- Progettazione
- Implementazione
- Tracciamento dei risultati
- Confronto tra gli algoritmi

alle quali abbiamo dedicato n1,n2,...,n7 ore/lavoro persona rispettivamente. Il progetto ha avuto inizio gg/mm/aaaa ed è stato terminato gg/mm/aaaa. ****Scrivere Persone coinvolte nel lavoro e loro ruolo******

3 Metodologie utilizzate

3.1 Studio del problema

Lo **spam** è l'invio anche verso indirizzi generici, non verificati o sconosciuti, di messaggi ripetuti ad alta frequenza o a carattere di monotematicità tale da renderli indesiderati. Il problema dello spam nasce quando, con l'avvento di tecnologie informatiche, questa tipologia di messaggi ha iniziato ad invadere le caselle di posta elettronica e gli smartphone di aziende e semplici utilizzatori. Ciò ha portato molte persone ad interessarsi al tema dello spam e di come riuscire a contrastarlo in modo efficace.
In prima battuta, ci sono state molte discussioni riguardanti la tipologia del problema. Le principali controversie furono tra chi sosteneva fosse un problema di *classificazione* e tra chi, invece, sosteneva fosse un problema di *modellazione*. Alcune correnti di pensiero continuano a sostenere che la metodologia migliore da applicare in questo caso sia la modellazione, ma la maggior parte della comunità degli studiosi afferma con convinzione che la tecnica migliore sia quella della classificazione. Noi concordiamo con quest'ultimi, vero che il bisogno iniziale di dati già classificati è uno svantaggio, ma nell'era dei big data, è ormai semplice trovare dei set di dati consistenti e pronti all'uso. Inoltre, c'è da sottolineare la maggiore flessibilità della tecnica che è portata maggiormente ad imparare dai propri errori.
Detto ciò, si può capire perché tra i tanti tipi di approcci al problema abbiamo deciso di utilizzare la *Logistic Regression* e le *Neural Networks*, vediamoli ora in modo più approfondito.

3.2 Neural networks

Le Neural networks, o reti neurali, sono ispirate alle reti di neuroni biologici. Solitamente, sono sistemi che imparano dagli esempi e che inizialmente sono configurati in modo randomizzato.
Il concetto alla base è il singolo neurone artificiale, che poi viene rappresentato in collezioni connesse con altri neuroni, come nelle reti biologiche.

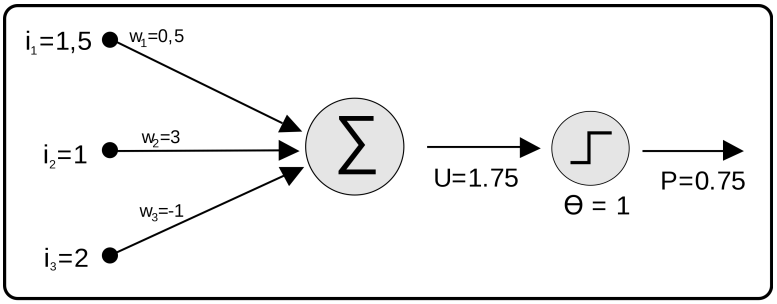

Figura 1: Esempio di neurone artificiale

Queste collezioni vengono comunemente chiamati layer e le connessioni tra ogni nodo vengono chiamati edge.
Ad ogni neurone e ad ogni edge viene assegnato un peso, il quale, in concreto, ha la funzione di "aggiustamento" del tasso di apprendimento influenzando la il valore dei nodi o delle connessioni.

3.3 Logistic Regression

La Logistic Regression è un modello statistico che si basa sull'utilizzo della Logistic Function per modellare una variabile binaria dipendente.

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{1}$$

Il fulcro della Logistic Regression sta nell'assegnare un valore che va da 0 a 1 della probabilità che un dato elemento sia o meno del tipo A, ovviamente la somma delle probabilità di appartenere o del non appartenere alla categoria A deve essere esattamente 1. Sottolineiamo quindi che, come dal nome, è una forma di regressione matematica.
Interessante la possibilità di avere un grado maggiore o minore di tolleranza, e quindi una sorta di scelta, spostando in alto o in basso la soglia minima per essere considerati di un certo tipo. Questo ci consente maggiore flessibilità in caso di incertezza del nostro algoritmo o maggiore rigidezza in caso di estrema precisione.

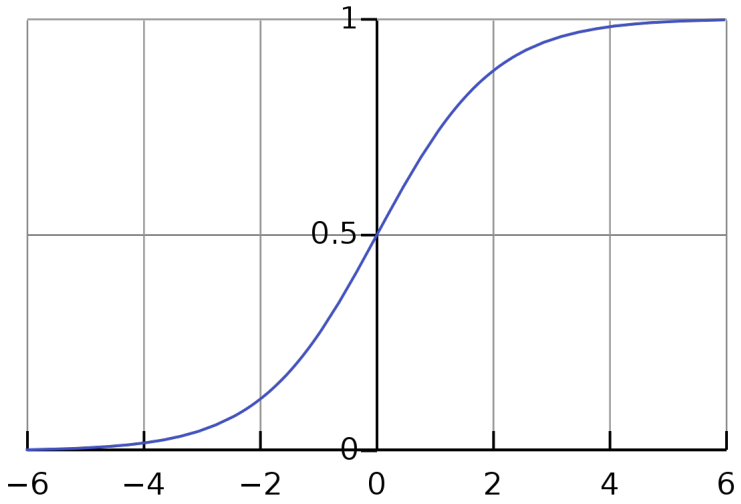

Figura 2: La logistic function standard

4 Realizzazione

È stato scelto di suddividere il dataset in 80 e 20 percento. Dove 80% dei dati è stato destinato al training set e il rimanente 20% è stato utilizzato per il test

set.
Il dataset che è stato utilizzato è quello fornito dalla piattaforma Kaggle, è disponibile al seguente link *Spam Collection Dataset*.
Abbiamo utilizzato il metodo *train_test_split* presente nella libreria scikit-learn.

4.1 Reti neurali

Il primo approccio che abbiamo scelto di utilizzare è stato quello delle reti neurali. Per farlo ci siamo appoggiati alla libreria *TensorFlow* la quale, dalla versione 2.0.0, integra *Keras*.
Maggiori informazioni riguardanti l'integrazione di Keras sono presenti al seguente link *tf.keras*.
Questo ci ha permesso di:

1. Utilizzare TensorFlow(*tf*) come ecosistema;
2. Definire la rete tramite la libreria *tf.keras*;

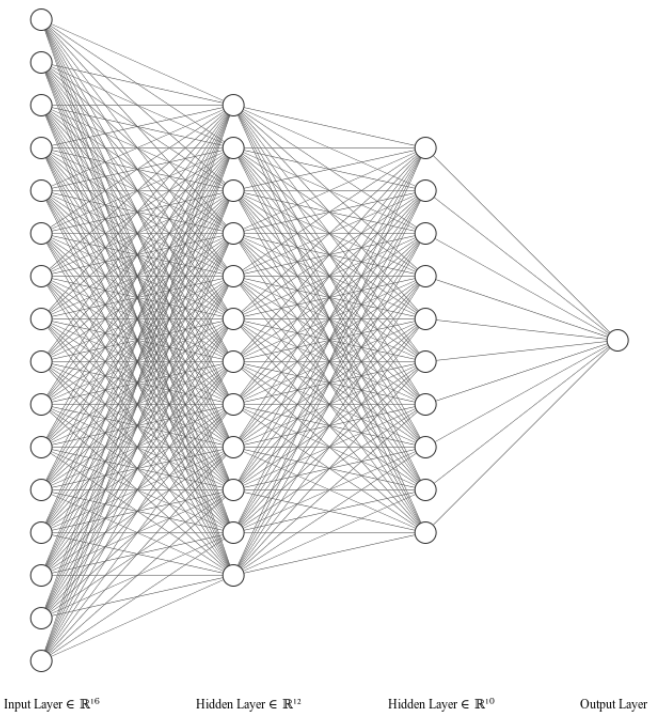

Figura 3: Esempio di rete neurale *sequential*

4.1.1 Struttura

È stata scelto di utilizzare il tipo di rete *Sequential*.
Abbiamo creato due reti, con tre layer di tipo *Dense*. Le funzione di attivazione scelte sono le seguenti:

1. *Relu* per i nodi interni;
2. *Sigmoid* per l'output della rete.

La prima rete presenta la seguente struttura:

```
model = Sequential()
model.add(Dense(512, activation = 'relu')
model.add(Dense(256, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
```

La seconda rete presenta la seguente struttura:

```
model = Sequential()
model.add(Dense(512, activation = 'relu')
model.add(Dropout(0.2))
model.add(Dense(256, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation = 'sigmoid'))
```

4.1.2 Preparazione del dataset

Per preparare il dataset abbiamo utilizzato il Text Preprocessing presente nelle API di Keras. Questo permette di vettorizzare il corpus del testo, convertendo ogni parola presente in una sequenza di interi.
In particolare abbiamo scelto di assegnare al parametro *num_words* il valore di 1000. Questo stabilisce il numero massimo di parole da mantenere, in base alla loro frequenza.
Vengono quindi mantenute solo le parole *num_words-1* più comuni.

```
tokenizer = Tokenizer(num_words = num_max)
```

Dopo aver definito il nostro Tokenizer abbiamo utilizzato i metodi:

1. *fit_on_texts* per aggiornare il vocabolario interno basato sulla frequenza delle parole e crearne l'indice;
2. *texts_to_matrix* per convertire il testo in un numpy array di forma: `(len(texts), num_words)`.

Maggiori info sui metodi di Keras preprocessing sono presenti al seguente link Tokenizer Preprocessing.

4.1.3 Configurazione

Tramite il metodo *compile()* presente in Keras è possibile stabilire:

1. **Loss functions**: *binary crossentropy* nel nostro caso, maggiori info al seguente link *loss functions*;
2. **Optimizer**: l'ottimizzatore che verrà utilizzato per l'aggiustamento dei pesi e per minimizzare la loss function. Nel nostro caso *Adam*.
3. **Metrics**: lista di metriche che verranno valutate dal modello durante la fase di training e testing. Nel nostro caso è stata scelta la metrica di *binary accuracy*.

Perchè relu, perchè adam, sistema dropout (TEST SENZA dropout)

Riferimenti

https://it.wikipedia.org/wiki/Spam

https://www.matchilling.com/comparison-of-machine-learning-methods-in-email-spam-detection/

https://en.wikipedia.org/wiki/Artificial_neural_network