Report Progetto Deep Learning: Financial Sentiment Analysis

Stefanelli Francesco
Matricola 538549
fra.stefanelli3@stud.uniroma3.it

GitHub: financial-sentiment-analysis
Anno Accademico 2022/2023

In questa relazione è stato documentato il lavoro svolto durante lo sviluppo del progetto del corso Deep Learning. Il progetto consiste nell'utilizzare diverse tecniche e modelli di DL per effettuare Sentiment Analysis nel contento delle news finanziarie. La relazione di divide in tre sezioni: nella prima sezione sono riportati i dataset utilizzati per il caso di studio, nella seconda è riportata la fase di preparazione dei dati, nella terza l'implementazione dei modelli utilizzati e nella quarta si presentano i risultati ottenuti e le comparazioni effettuate.

Contents

1 Dataset

Nella seguente sezione sono illustrati ed analizzati i dataset impiegati per il task di Sentiment Analysis nell'ambito delle news finanziarie. Ciascun dataset sarà composto da: una serie di news headline annotate con il relativo sentiment, che indicherà quanto il titolo finanziario in questione è appetibile da parte di un possibile investitore.

1.1 FinancialPhraseBank

Il primo dataset utilizzato allo scopo è stato preso dalla piattaforma Kaggle[1]. Si compone di 4846 news headlines annotate con tre classi (sentiment) e le classi sono rispettivamente positiva (28.1% sul totale del dataset), negativa (12.5%) e neutrale (59.4%).

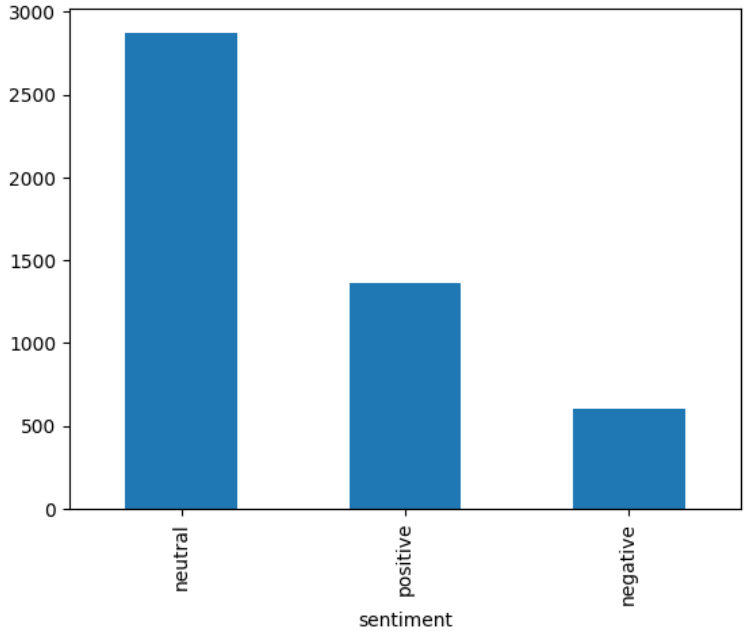

Figure 1: Sample occurencies bar plot per class

1.2 FinancialPhraseBank + FiQA

Il secondo dataset considerato per l'analisi è un estensione del precente, preso da Kaggle[2], che riporta in in totale 5322 news headline annotate nel medesimo modo del precedente dataset. Il dataset anche in questo caso si compone di tre classi e sono rispettivamente positiva (31.7% sul totale del dataset), negativa (14.7%) e neutrale (53.5%). Il bilanciamento delle classi, a meno di piccole variazioni percentuali, è lo stesse del precedente dataset.
Indicate in Figure 1.

[1] Kaggle dataset 1: FinancialPhraseBank
[2] Kaggle dataset 2: FinancialPhraseBank + FiQA

2 Data preparation

Per la fase di data preparation si è scelto di utilizzare un **Notebook Jupyter** insieme a **Python** utilizzando **Pandas**. Sono state applicate `dropna()` e `drop_duplicates()` pre-processing per cancellare le ennuple del dataset contenenti valori nulli e duplicati, la rimozione della punteggiatura e conversione a lowercase. Inoltre, non sono state rimosse le stopwords o effettuate ulteriori fasi di pre-processing dato che sono state ritenute caratteristiche importanti da mantenere per il task di Sentiment Analysis, riuscendo ad ottenere risultati migliori non modificando ulteriormente la frase originale. Ecco alcuni esempi di news headline dopo la data preparation:

- Positive news headline example : according to the company s updated strategy for the years 2009 2012 basware targets a long term net sales growth in the range of 20 40 with an operating profit margin of 10 20 of net sales
- Negative news headline example : a tinyurl link takes users to a scamming site promising that users can earn thousands of dollars by becoming a google nasdaq goog cash advertiser
- Neutral news headline example : technopolis plans to develop in stages an area of no less than 100 000 square meters in order to host companies working in computer technologies and telecommunications the statement said

3 Modelli e implementazioni

In questa sezione sono riportate le implementazioni dei vari modelli utilizzati che si basano su due diverse tecniche per la generazione di word embeddings. Entrambe le tecniche consistono in metodi che vanno ad analizzare il testo creando degli embeddings, ovvero delle rappresentazioni compatte delle parole all'interno di uno spazio vettoriale dove due parole simili sono rappresentate da vettori simili.

3.1 GloVe

In particolare la tecnica Global Vectors for Word Representation (GloVe) presentato in [4] si basa sulla tecnica skip-gram e andandone a modificare la loss function introducendone alcune varianti per ridurne la complessità computazionale.

3.1.1 Skip-Gram with Global Corpus Statistics

Si va a denotare con q_{ij} la probabilità condizionata $P(w_j|w_i)$ di una parola contestuale w_j data la parola target o centrale w_i nel modello skip-gram:

$$q_{ij} = \frac{\exp(\mathbf{u}_j \cdot \mathbf{v}_i)}{\sum_{k=1}^{V} \exp(\mathbf{u}_k \cdot \mathbf{v}_i)}$$

La formula rappresenta la probabilità condizionata q_{ij}, che indica la probabilità che la parola di contesto j si verifichi dato che la parola target i è presente. Nella formula, $\mathbf{u}_j$ rappresenta il vettore di contesto per la parola di contesto j, $\mathbf{v}_i$ rappresenta il vettore della parola target i, e $\mathbf{u}_k$ rappresenta il vettore di contesto per qualsiasi altra parola k nel corpus. La sommatoria nel denominatore si estende a tutte le parole nel vocabolario $V = \{0, 1, 2, \ldots, |V|\text{-}1\}$.

La parola w_i può presentarsi molte volte in un corpus. Tutte le parole contestuali che co-occorrono con wi creano un multiset, dove xij indica il conteggio del numero di volte che la parola wj co-occorre con wi nella stessa finestra di contesto nell'intero corpus. Utilizzando tali statistiche globali del corpus la loss function del modello skip-gram è:

$$-\sum_{i=1}^{V}\sum_{j=1}^{C} x_{ij} \log(q_{ij})$$

Inoltre, si denota con x_i il numero delle parole di contesto dove compare w_i come parola centrale. Si può definire come la probabilità condizionata $P_{ij} = \frac{x_{ij}}{x_i}$ di generare una parola di contesto w_j data una parola centrale w_i come:

$$-\sum_{i \in V} x_i \sum_{j \in V} p_{ij} \log q_{ij}$$

La sommatoria $-\sum_{j \in V} p_{ij} \log q_{ij}$ è la cross-entropy tra probabilità condizionata q_{ij} relativa alla predizione generata dal modello, e p_{ij} ottenuta analizzando le statistiche dell'intero corpus.

3.1.2 Modello GloVe

Per ridurre la complessità computazionale (soprattutto per generare qij) e per mitigare gli effetti generati dai termini che compaiono di rado nel corpus ma che possono assumere importanza elevata dalla cross entropy, il modello GloVe introduce alcune varianti.

Il modello GloVe (Global Vectors for Word Representation) introduce alcune varianti al fine di ridurre la complessità computazionale e mitigare gli effetti dei termini rari nel corpus ma che potrebbero assumere importanza elevata dalla cross-entropy. I cambiamenti nella loss function consistono in:

- Aggiunti due bias, b_i per le parole centrali e c_i per le parole contetsuali.
- Utilizza delle nuove assegnazioni per evitare di utilizzare la probabilità condizionata q_{ij} modificando la loss e introducendo gli ultimi due termini nel termine al quadrato.
- Si rimpiazza il peso di ogni termine della loss con una weight function $h_{x_{ij}}$ che varia nell'intervallo $[0, 1]$.

Ottenendo come nuova loss function:

$$\sum_{i=1}^{V} \sum_{j=1}^{V} h(x_{ij}) \left(\mathbf{u}_j^T \mathbf{v}_i + b_i + c_j - \log x_{ij} \right)^2$$

3.1.3 GloVe - CNN1d

L'implementazione con le Convolutional Neural Networks (CNNs) impiega GloVe per generare gli embeddings, una serie di strati conv1D seguiti da GlobalMaxPooling1D e infine alcuni layer MLP con una softmax sull'ultimo strato dense con 3 nodi pari al numero di classi in output. Il dettaglio è indicato in Table 1.

Layer	Maps	Kernel size	Activation	Dropout prob.
GloVe Embedding	-	-	-	-
Convolution1d	200	2x2	ReLU	-
GlobalMaxPooling1D	200	-	-	-
Convolution1d	200	3x3	ReLU	-
GlobalMaxPooling1D	200	-	-	-
Convolution1d	200	4x4	ReLU	-
GlobalMaxPooling1D	200	-	-	-
Convolution1d	200	5x5	ReLU	-
GlobalMaxPooling1D	200	-	-	-
Convolution1d	200	6x6	ReLU	-
GlobalMaxPooling1D	200	-	-	-
Concatenate	1000	-	-	-
Dropout	-	-	-	0.1
Fully Connected	128	-	ReLU	-
Dropout	-	-	-	0.2
Fully Connected	3	-	Softmax	-

Table 1: GloVe-CNN1d detailed model architecture

3.1.4 GloVe - LSTM

Anche l'implementazione con la Long short-term memory (LSTM) impiega GloVe per generare gli embeddings, successivamemte un layer ricorrente LSTM, un layer di flatten e infine alcuni layer MLP con una softmax sull'ultimo strato dense con 3 nodi pari al numero di classi in output. Nel dettaglio:

Layer	Nodes	Return sequences	Activation
GloVe Embedding	-	-	-
LSTM	256	True	TanH
Flatten	-	-	-
Fully Connected	3	-	Softmax

Table 2: GloVe-LSTM detailed model architecture

3.2 Ensemble Classifier for Financial Sentiment Analysis

3.2.1 Deep-FASP

La tecnica *Financial Aspect and Sentiment Prediction task with Deep neural networks (Deep-FASP)* proposta in [5] consiste in un ensemble classifier di modelli di DL. Gli autori questo approccio lo hanno utilizzato sia per il task sentiment analysis, sia per l'individuazione di aspetti target nelle news finanziarie. Nel seguente progetto l'analisi verrà limitata al primo task di sentiment analysis (sentiment classification).
In particolare Deep-FASP utilizza un ensemble di classificatori che comprende: CNNs, LSTMs e GRUs, dove l'output di tutte queste reti può essere comnbinata con un processo di voting che permette di ottenere poi, insieme ad una softmax come funzione di attivazione, la classificazione del sentiment in classi.
Questo approccio più nel dettaglio si compone di 5 steps (illustrata in Figure 2):

- **Input**: Ogni news headline è presa come sequenza di parole.
- **Word Embeddings**: Le parole vengono rappresentate in uno spazio latente di word embeddings. Utilizzando, ad esempio, GloVe.
- **DNNs**: La sequenza di word embeddings è usata come input alle CNNs, LSTMs e GRUs. Ogni singola rete avrà in output la propria sentiment classification.
- **Voting-Classification**: Infine, l'output delle singole reti viene combinato tramite un processo di voting e una MLP seguita da una softmax per ottenere la sentiment classification finale.

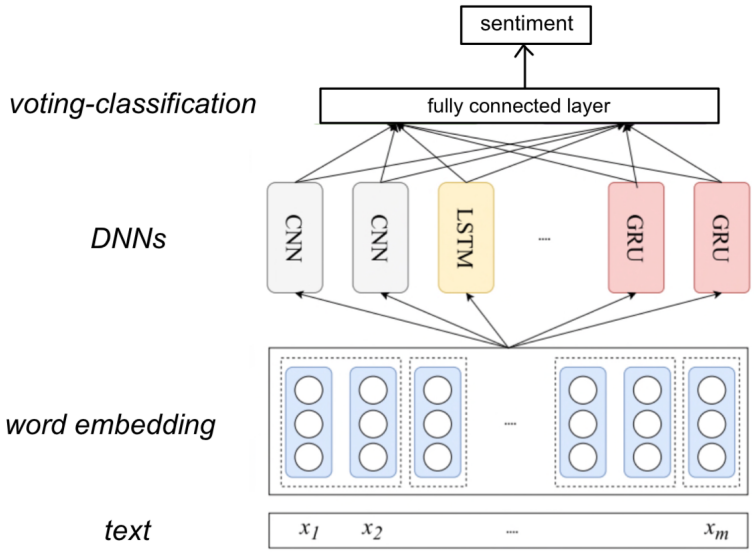

Figure 2: Overview of the voting approach Deep-FASP [5] for sentiment classification

3.2.2 Deep-FASP + SVR

La tecnica precente basata su ensemble è stata estesa, come nell'approccio proposto in [1], in modo tale da utilizzare un altro modello nell'insieme costituito per la classificazione del sentiment. Questa tecnica di ensemble considera oltre alle CNNs, LSTMs e GRUs anche le Support Vector Machines addrestrate tramite delle feature estratte dal modello tf-idf che consente di assegnare un'importanza ad una parola all'interno di un corpus di parole. Questa tecnica, come la precedente, utilizza uno strato fully connected per andare a realizzare l'ensemble voting. In dettaglio l'architettura è illustrata in Figure 3.

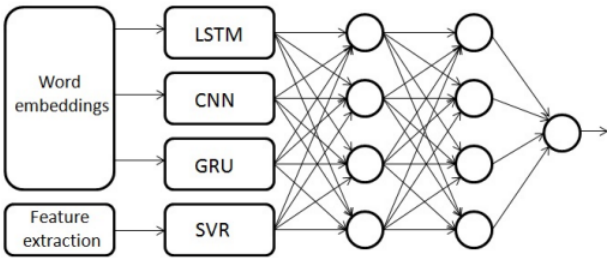

Figure 3: Overview of the extended ensemble [1] for sentiment classification

3.3 BERT

Il modello BERT proposto in [3] per la generazione degli word embeddings combina due approcci differenti:

- *Context-Indipendent*: data una parola, il vettore generato non dipenderà dal contesto attuale per cui termini polisemici o relazioni semantiche del linguaggio naturale saranno ignorate.
- *Context-sensitive*: si combinano le rappresentazioni intermedie per ottenere una rappresentazione che dipende dalla sequenza in input.

BERT rappresenta l'intero contesto mediante un approccio bidirezionale e si basa su un architettura encoder-decoder di trasformers pre-addestrati su auxiliary task. Dell'intera architettura transformers verrà utilizzata però solamente l'encoder dato che l'obiettivo di questo modello è quello di generare un language model. In particolare in input c'è un singolo testo o una coppia di testi, che permette di andare a definire due auxiliary task differenti su cui è stata pre-addestrata la rete sui cui poi sarà possibile effettuare fine-tuning in base al contesto specifico.
Un primo auxiliary task chiamato *Next Sentence Prediction* consiste nel prendere coppie di testi e far predire alla rete se la seconda frase in coppia è la frase successiva nel documento originale. Durante l'addestramento, il 50% degli input è una coppia in cui la seconda frase è la frase successiva nel documento originale, mentre nell'altro 50% viene scelta una frase random dal corpus come seconda frase. L'assunzione si basa sul fatto che la frase casuale sia slegata dalla prima frase.
Alla coppia di frasi si si aggiungono in fase di tokenizzazione due special token, rispettivamente *CLS* inserito all'inizio della frase e *SEP* inserito alla fine di ogni farse e poi successivamente viene sommato il sentence embedding che andrà a rappresentare se si la frase in questione è la frase A o la frase B. I due embedding così descritti verranno poi sommati ai positional embedding classici di un'architettura transformers, rappresentanti la posizione di un termine rispetto agli altri in una certa frase.
Per andare poi a predire se la seconda frase è effettivamente collegata alla prima l'intera sequenza in input viene passata all'interno del modello e tramite uno strato di classificazione binaria insieme ad una softmax come funzione di attivazione si ottiene la predizione.
Dal modello pre-addestrato su questo auxiliary task appena descritto si è andato poi ad effettuare fine-tuning per il task obiettivo del progetto, ovvero la Sentiment Analysis. Un ulteriore auxiliary task utile in altri contesti su cui gli autori di BERT hanno pre-addestrato il modello è il *Masked Language Modeling*, dove dato un corpus testuale il 15% dei tokens saranno selezionati in modo random per il task di predizione e al loro posto sarà presente un tag [mask].

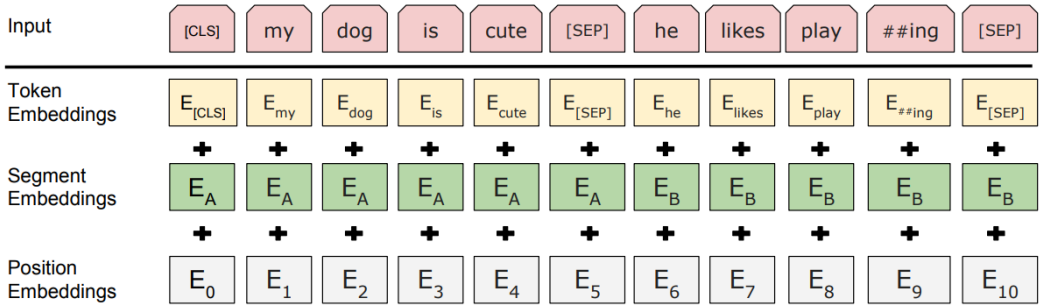

Figure 4: BERT input representation

Nel seguito verranno riportate le versioni pretrained e le implementazioni utilizzate per il caso d'uso.
Su entrambe le versioni di BERT l'implementazione della classe wrapper realizzata con Tensorflow è la seguente:

Table 3: BERT Wrapper detailed model architecture

Layer	Maps	Dropout prob.	Activation
Pretrained BERT layer	-	-	-
GlobalMaxPooling1D	768	-	-
Fully Connected	128	-	ReLU
Dropout	-	0.1	-
Fully Connected	32	-	ReLU
Fully Connected	3	-	Softmax

3.3.1 FinBERT

FinBERT [2] è una versione su cui è stato effettuato ulteriore pre-training della versione base di BERT. Gli autori di questa versione, FinBERT, hanno ulteriormente addestrato la rete su un corpus di dati finanziari per effettuare anche Sentiment Analysis, il dataset che hanno utilizzato gli autori è il Financial PhraseBank, lo stesso dataset utilizzato nella presente. Il modello pre-addestrato di FinBERT utilizzato nel training e nella valutazione del modello è quello offerto dalla libreria HuggingFace.[3]

3.3.2 DistilBERT

DistilBERT [6] è una versione più piccola, veloce e leggera di BERT base. Ha il 40% in meno di parametri rispetto a bert-base-uncased, funziona il 60% più velocemente, preservando al contempo oltre il 95% delle prestazioni di BERT misurate sul benchmark di comprensione del linguaggio GLUE. Il modello pre-addestrato di DistilBERT utilizzato nel training e nella valutazione del modello è stato sempre preso da HuggingFace.[4]

[3] Pretrained HuggingFace FinBERT from ProsusAI
[4] Pretrained HuggingFace distilbert-base-uncased

4 Risultati

In questa sezione sono riportati i risultati ottenuti in termini di metriche fondamentali, la pipeline seguita per il training, evalutaion dei modelli e i grafici a confronto dell'andamento dell'accuracy su train e validation set durante l'addestramento. Il dettaglio verrà approfondito nella sezione apposita.

4.1 Configurazione Software/Hardware

La configurazione con il quale sono stati effettuati i test è **Google Colab** utilizzando come tipo di runtime: Python3; Acceleratore hardware: GPU; Tipo di gpu: T4. Per l'implementazione, training dei modelli e valutazione dei modelli è stato utilizzato **Tensorflow** 2.12.0. La libreria utilizzata per i modelli preaddestrati di BERT è **Transformers** 4.29.2 di HuggingFace. Infine, **Sklearn** 1.2.2 per il preprocessing, train-test split e calcolo delle metriche.

4.2 Valutazione dei modelli

4.2.1 Metriche di valutazione

Per la valutazioni dei modelli sono state utilizzate le metriche di Precision (Eq. 1), Recall (Eq. 2), F1-Score (Eq. 3) e Accuracy (Eq. 4).

$$P = \frac{TP}{TP + FP} \tag{1}$$

$$R = \frac{TP}{TP + FN} \tag{2}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

4.2.2 Comparazione e Training dei modelli

Dopo aver effettuato il preprocessing dei dataset e la definizione dei modelli come da capitoli precenti la pipeline che si è andato a seguire (compresa anche di data preparation) è descritta in Figure 5.

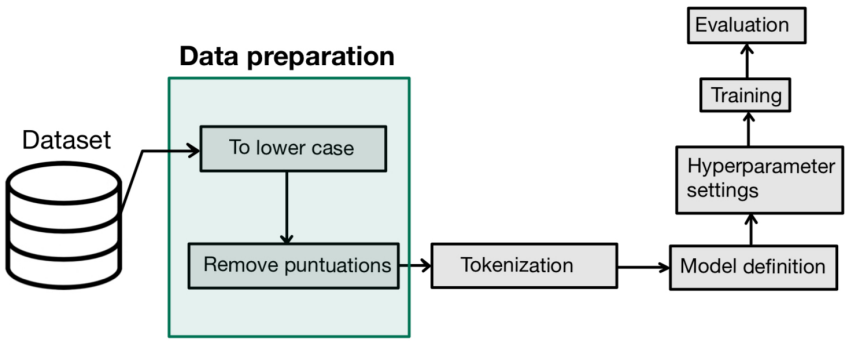

Figure 5: Implemented pipeline for Financial Sentiment Analysis

Inoltre la suddivisione in train-val-test set del dataset segue queste percentuali:

- Train set: 70%
- Validation set: 10% del train set
- Test set: 30%

Per il training di tutti e 4 i modelli impiegati nello studio del caso di studio sono state utilizzate due callbacks per cercare di evitare l'overfitting dei modelli durante il training, rispettivamente: early stopping e model checkpoint. Il model checkpoint è stato impostato tale per cui andrà a salvare la versione del modello con la validation accuracy più alta durante il training. Mentre, l'early stopping terminerà il training se la validation accuracy non è migliorata per un numero di volte pari al parametro di patience. L'optimizer utilizzato per tutti e 4 i modelli è Adam, dove per i modelli GloVe based sono stati utilizzati i parametri di default di Keras, mentre per le versioni di BERT il learning rate (lr) è pari a 5×10^{-5}, $\epsilon = 1 \times 10^{-8}$ e clipnorm = 1.0. Tutti gli altri iperparametri di training e le callbacks nello specifico sono indicate in Table 4.
Infine, in Table 5, 6 si possono vedere i valori, per entrambi i dataset considerati, di precision, recall e f1-score ottenuti in fase di validazione dei modelli. Tali valori sono stati presi tramite il metodo `classificationReport()` di Sklearn, riportando come valori quelli con macroAVG, ovvero mediati sul numero delle classi.

Methods	Opimizer	Loss	Metric	Batch Size	Epochs	ES Patience
GloVe+CNN	Adam	Sparse CCE	Accuracy	32	100	5
GloVe+LSTM	Adam	Sparse CCE	Accuracy	32	100	5
FinBERT	Adam	CCE	Accuracy	32	10	3
DistilBERT	Adam	CCE	Accuracy	32	10	3

Table 4: Hyperparameters and compiling details of used methods

Methods	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
GloVe+CNN	77.38	73.47	75.14	79.27
GloVe+LSTM	72.60	70.82	71.21	75.76
FinBERT	**84.52**	**83.22**	**83.82**	**85.54**
DistilBERT	84.49	81.14	82.66	84.44

Table 5: Performance comparization of implemented models on FinancialPhraseBank dataset

Methods	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
GloVe+CNN	63.76	60.84	61.28	73.22
GloVe+LSTM	61.18	57.51	57.77	69.16
FinBERT	**72.11**	**74.45**	**73.07**	**78.18**
DistilBERT	69.60	68.58	68.97	76.47

Table 6: Performance comparization of implemented models on FinancialPhraseBank+FiQA dataset

4.2.3 Grafici Training/Val Accuracy dei modelli su FinancialPhraseBank

Come nella precedente sottosezione anche qui sono riportati i grafici della training/val accuracy dei vari modelli sul dataset FinancialPhraseBank.
Rispettivamente del modello GloVe+CNN1d:

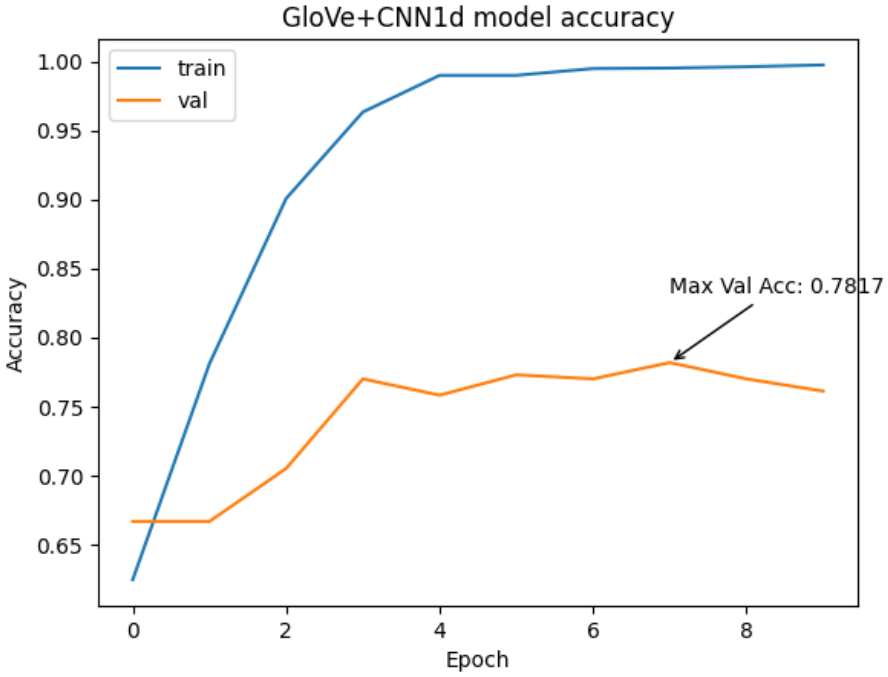

Figure 6: Plot of GloVe+CNN1d train/val accuracy

GloVe+LSTM:

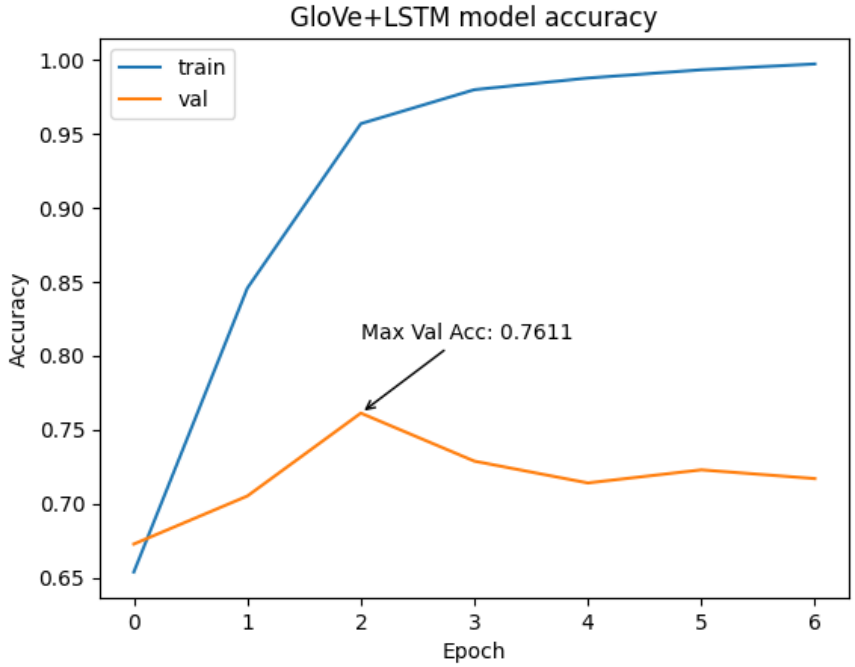

Figure 7: Plot of GloVe+LSTM train/val accuracy

finBERT:

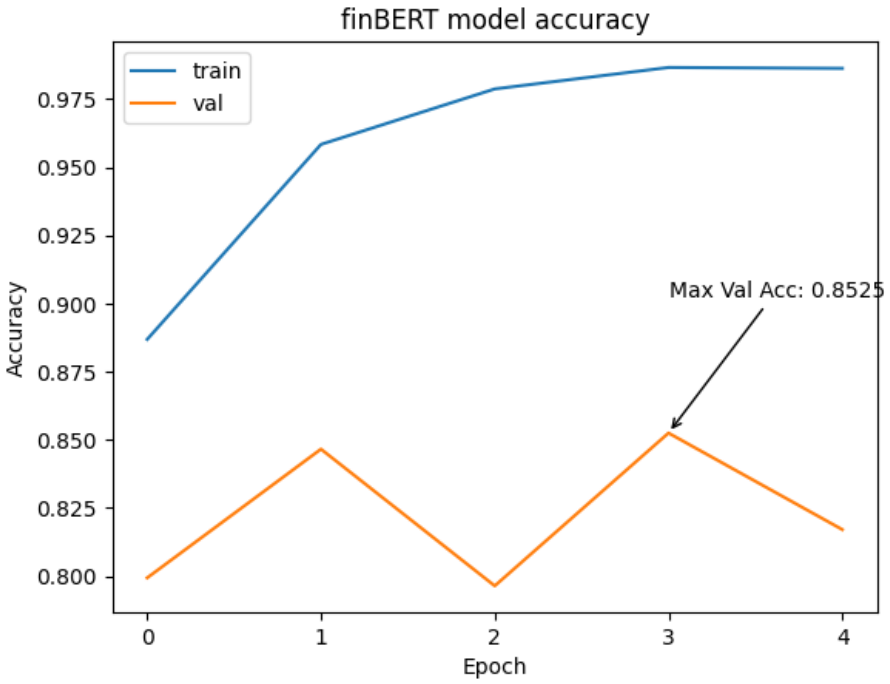

Figure 8: Plot of FinBERT train/val accuracy

distilBERT:

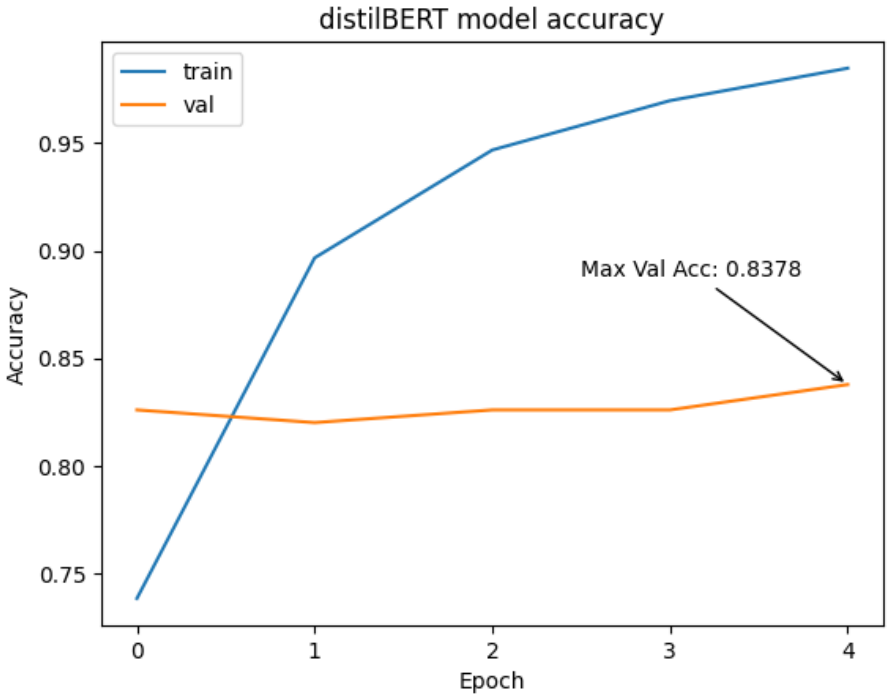

Figure 9: Plot of DistilBERT train/val accuracy

4.2.4 Grafici Training/Val Accuracy dei modelli su FinancialPhraseBank+FiQA

Come nella precedente sottosezione anche qui sono riportati i grafici della training/val accuracy dei vari modelli sul dataset FinancialPhraseBank+FiQA.
Rispettivamente del modello GloVe+CNN1d:

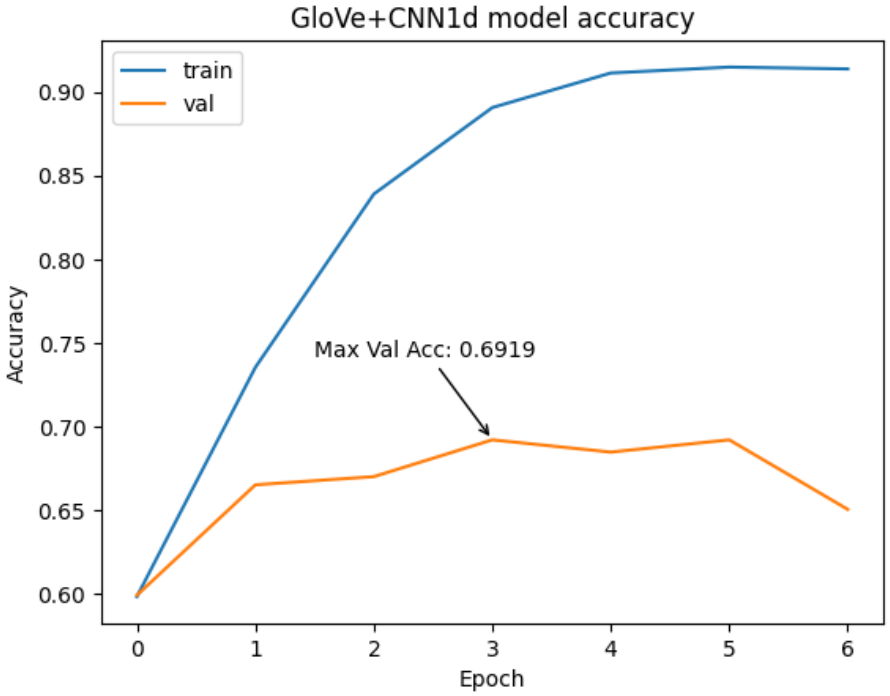

Figure 10: Plot of GloVe+CNN1d train/val accuracy

GloVe+LSTM:

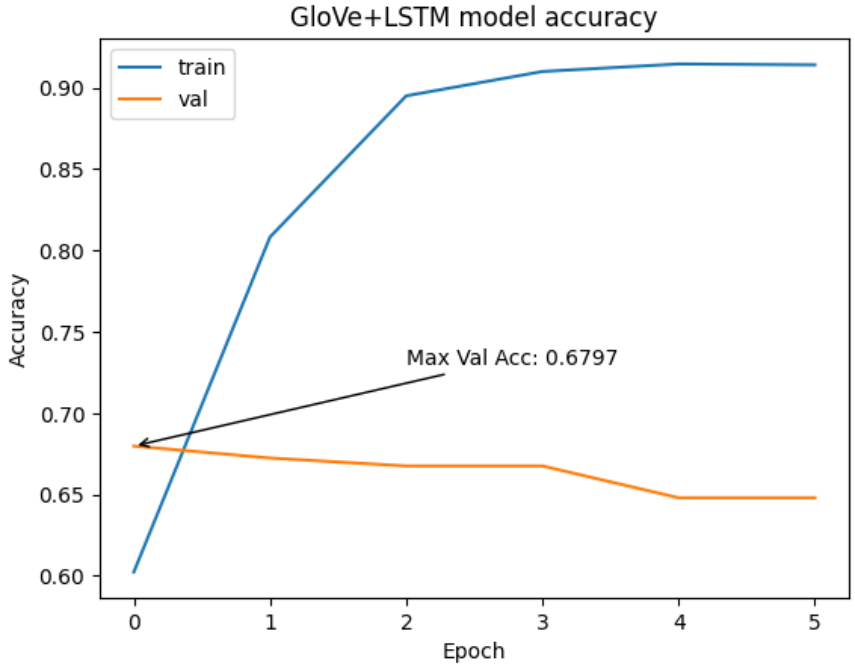

Figure 11: Plot of GloVe+LSTM train/val accuracy

finBERT:

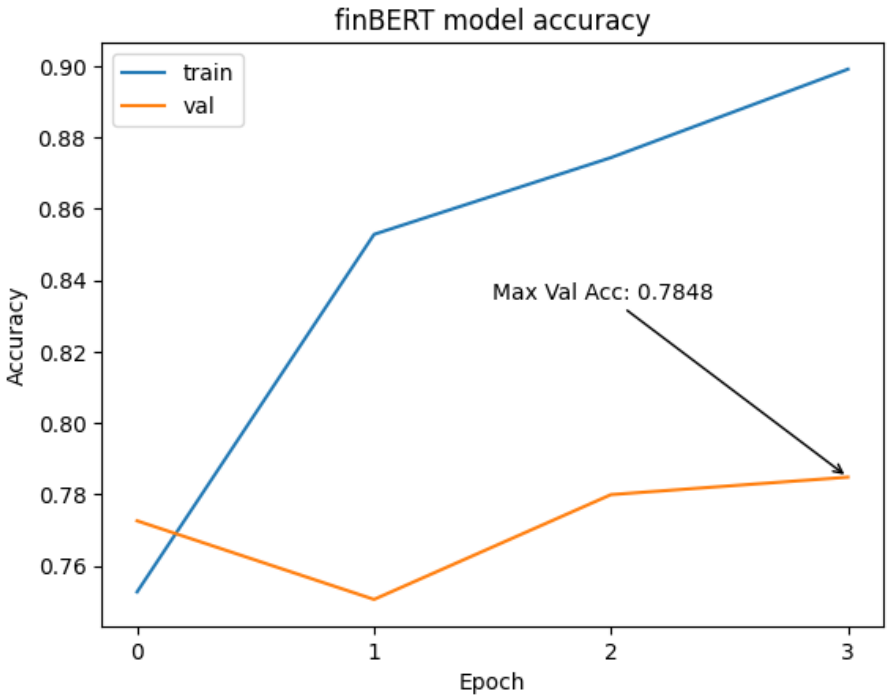

Figure 12: Plot of FinBERT train/val accuracy

distilBERT:

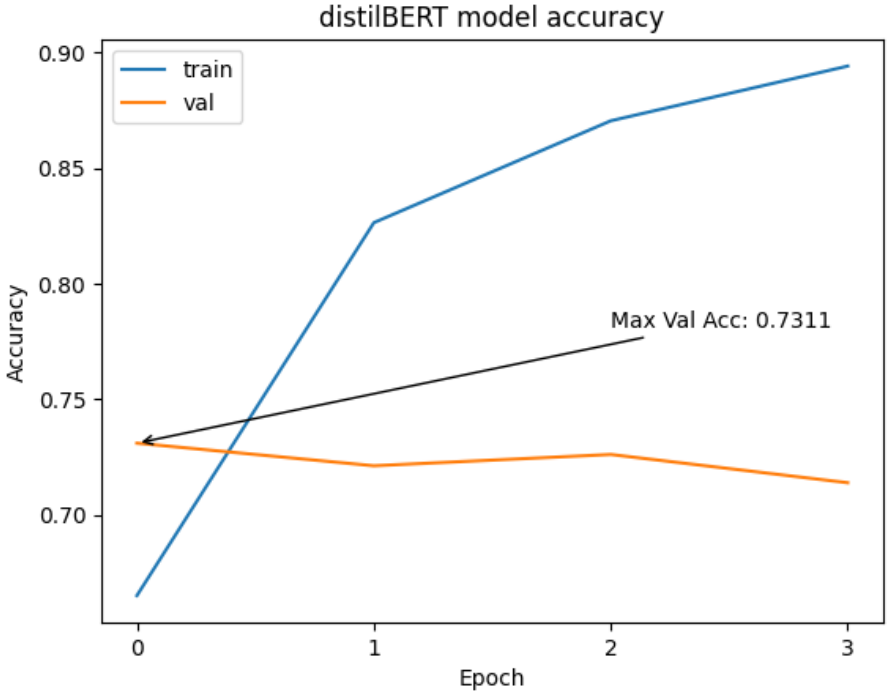

Figure 13: Plot of DistilBERT train/val accuracy

4.3 Output

Di seguito sono riportate le previsioni ottenute in output dai relativi modelli sulle headline prese in esempio in 2:

Table 7: Example prediction of GloVe+CNN

Headline	Prediction	True label
Positive news headline 2	neutral	positive
Negative news headline	neutral	negative
Neutral news headline	neutral	neutral

Table 8: Example prediction of GloVe+LSTM

Headline	Prediction	True label
Positive news headline 2	positive	positive
Negative news headline	negative	negative
Neutral news headline	neutral	neutral

Table 9: Example prediction of FinBERT

Headline	Prediction	True label
Positive news headline 2	positive	positive
Negative news headline	negative	negative
Neutral news headline	neutral	neutral

Table 10: Example prediction of DistilBERT

Headline	Prediction	True label
Positive news headline 2	positive	positive
Negative news headline	negative	negative
Neutral news headline	neutral	neutral

Riferimenti

[1] Md Akhtar et al. "A Multilayer Perceptron based Ensemble Technique for Fine-grained Financial Sentiment Analysis". In: Jan. 2017, pp. 540–546. DOI: 10.18653/v1/D17-1057.

[2] Dogu Araci. *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models*. 2019. arXiv: 1908.10063 [cs.CL].

[3] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global Vectors for Word Representation." In: *EMNLP*. Vol. 14. 2014, pp. 1532–1543.

[5] Guangyuan Piao and John G. Breslin. "Financial Aspect and Sentiment Predictions with Deep Neural Networks: An Ensemble Approach". In: *Companion Proceedings of the The Web Conference 2018*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 1973–1977. URL: https://doi.org/10.1145/3184558.3191829.

[6] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL].