Università degli studi di
Milano-Bicocca

TEXT MINING AND SEARCH

Final Project

Amazon Reviews Classification:
On the Role of Text Representation in Neural Network Architectures

Authors:
Ivan Mera - 783086 - i.merafranco@campus.unimib.it

February 15, 2021

Abstract

L'obiettivo generale di questo lavoro è quello di classificare recensioni su prodotti lasciate dai clienti di Amazon. La classificazione è di tipo *multi-class*. Ogni recensione può appartenere ad un'unica macro categoria di prodotti.
L'obiettivo più specifico, che ha determinato lo sviluppo della pipeline, è quello di valutare l'impatto che hanno sulla performance due tecniche di *text representation* quali *TF-IDF* e *word embeddings*.
Sono state implementate due architetture di rete neurale: la prima con soli strati densi aveva come input la matrice TF-IDF. La seconda con strati CNN e GRU aveva la matrice embedding.
Da questo lavoro si è potuto concludere che la differenza in termini di accuracy è stata minima. La tecnica che ha raggiunto il più alto livello di accuracy è stata la TF-IDF, con un valore pari a **0.8685** mentre con i word embeddings si è ottenuto un valore di accuracy pari a **0.8622**.

1 Introduction

Questo progetto nasce con l'obiettivo di analizzare qual è l'impatto che tecniche di *text representation* possono avere sulla performance di una rete neurale.
In una pipeline di un sistema di NLP, il preprocessing insieme alla rappresentazione del testo, hanno un impatto considerevole nel risultato finale.
Per questo lavoro dunque, ci si è chiesto se i vari metodi di text representation vadano a influenzare in maniera significativa il risultato nel caso di una *Multi-Class Classification*. Bisogna tenere presente che si è cercato di massimizzare la performance tenendo in considerazione la capacità computazionale che si aveva a disposizione.
I dati di input sono stati ricavati dalle recensioni di clienti Amazon. Si è deciso di analizzare un subset del dataset iniziale prendendo solo quattro macro categorie quali: "Industrial_and_Scientific", "Luxury_Beauty", "Musical_Instruments" e "Appliances".
Si è deciso quindi di implementare due tipologie di text representation: *TF-IDF* e *word embeddings*. Per ogni tecnica di text representation è stato implementata un'architettura di rete neurale diversa

2 Datasets

Il dataset utilizzato, come specificato in precedenza, è basato su recensioni di prodotti venduti su Amazon America del 2018. Si tratta di una raccolta di recensioni, in lingua inglese, su prodotti venduti dal colosso dell'ecommerce. Per motivi computazionali si è deciso di selezionare quattro macro categorie quali: "Industrial_and_Scientific", "Luxury_Beauty", "Musical_Instruments" e "Appliances".
I dati all'origine erano in formato *json* perciò sono stati trasformati in *dataframe* di Pandas attraverso due funzioni. Ogni dataset è composto dalle seguenti variabili:

- **reviewerID**: ID dell'utente che ha lasciato la recensione.
- **asin**: ID del prodotto.
- **reviewerName**: Nome del utente.
- **verified**: Verifica da parte di Amazon sull'effettivo acquisto.
- **reviewText**: Testo della recensione.
- **overall**: Valutazione del prodotto.
- **reviewTime**: Data di pubblicazione della recensione in formato MM/DD/AAAA.
- **summary**: Sintesi della recensione.
- **unixReviewTime**: Data di pubblicazione della recensione in formato UNIX.
- **style**: Dizionario contenente la descrizione del prodotto.
- **vote**: Numero di persone che hanno ritenuto utile la recensione.
- **image**: Link a eventuali immagine da parte dell'utente sul prodotto.

2.1 EDA and Feature Engineering

I quattro dataset così caricati avevano insieme 4448268 osservazioni. Una volta rimossi i missing values dalla variabile *reviewText* si arriva a 4445705 osservazioni.
Successivamente si è calcolato il numero di recensioni per prodotto. Si è osservato che alcuni prodotti presentavano solo una recensione mentre per altri si andava oltre 14000. Si è pertanto deciso di tenere un'unica recensione per prodotto in modo da poter allenare in meno tempo le reti neurali. La recensione che si è tenuto è stata quella con la lunghezza maggiore in modo che sia quella più significativa. Quest'operazione è stata svolta in fase di preprocessing del testo e prima di applicare la *lemmatization*. Si è svolta in fase di preprocessing in modo da avere un testo *pulito* con le sole parole significative e prima della lemmatization in quanto quest'ultima richiede molta potenza computazionale per essere eseguita.
Per quanto riguarda le variabili invece si è deciso di tenere solo quella che riporta la recensione ed è stata create una variabile target ad-hoc per ogni macro categoria. I valori che può assumere la variabile target vanno da 0 a 3.
La dimensione finale del dataset è di 2 variabili e 320309 osservazioni.
Per questo lavoro si è scelto di utilizzare il GloVe che è un embedding preaddestrato. In questo sistema la creazione dei vettori avviene considerando le relazioni nella *Term-Context Matrix*, in pratica viene calcolato quante volte una parola è presente in un contesto. Parole simili hanno una rappresentazione numerica simile. Il file scelto contiene sei miliardi di parole uniche e la lunghezza di ogni vettore è di 200 feature, risultando migliore rispetto alla versione di 100 feature. La scelta di questa tipologia di embedding si è basata su altri lavori precedentemente svolti in cui risultava essere la tipologia di embedding migliore.

2.2 Preprocessing

Il primo step nella preparazione del testo è stato effettuare una *noise removal* e *text normalization*, in particolare sono stati rimossi i seguenti componenti e caratteristiche e applicate le seguenti tecniche:

- **upper case**
- **link web**

- **contrazioni**
- **emoji**
- **caratteri speciali**
- **spazi bianchi**
- **stop words**
- **lemmatization**

La prima trasformazione è stata quella di avere tutto il testo in formato *lowercase*. Il dataset a disposizione non è di piccole dimensioni ma questa tecnica potrebbe risultare utile per evitare problemi di *sparsity*. Inoltre in questo modo vengono rimosse correttamente le *stop words* in quanto per esempio *Not* non è inclusa nella lista ma viene considerata solo la forma senza *capitalize*.
I link web sono una componente del testo che rientra nella categoria noise, per tale motivo è stata rimossa in quanto non è utile al fine della classificazione.
La terza componente di testo rimossa sono state le contrazioni, la scelta di rimuovere le contrazioni è stata dettata dal fatto che nella lingua inglese esse sono presenti in maniera consistente. Il testo è stato dunque riportato alla sua versione estesa.
Anche gli emoji in questo caso sono stati rimossi in quanto considerati noise. Preme ricordare che comunque potrebbero essere trasformati in testo e valutare il loro impatto.
La penultima componente rimossa sono stati i caratteri speciali, tra essi troviamo per esempio i trattini e gli apici singoli. Questa è stata una scelta di progetto che in letteratura è molto dibattuta quando si parla di *tokenization*. Per esempio se nel testo c'è la seguente frase *Filand's capital* il risultato finale in questo caso sarebbe *finland s capital*. La *s* sarà successivamente rimossa tra le *stop words*.
L'ultima componente di testo rimossa sono stati gli spazzi bianchi in quanto poco utili al task.
Una volta ottenuto un testo normalizzato e senza noise si sono rimosse le stop word, ovvero sia quelle parole che non aggiungono significato ad una frase.
Nell'ultima fase è stato effettuata la *lemmatization*, tecnica che riduce ad un

lemma le parole. Questa tecnica è stata preferita allo *stemming* in quanto da una parte portava alle stesse performance, dall'altra era meno dispendiosa in termini computazionali. La funzione di *lemmatization* includeva una parte di POS tagging in modo da ottenere il lemma corretto in base alla categoria lessicale di ogni parola nel contesto nel quale è usata.
Il risultato così ottenuto è stato utilizzato per implementare le due tecniche di text representation.

3 Text Representation Techniques

Una volta completata la fase di preprocessing si è proceduto ad applicare le techiche di rappresentazione del testo per le recensioni. Esistono diverse tecniche che si possono utilizzare. Per questo progetto sono state scelte: **term-document matrix** e **Word Embeddings**. La prima prevede la creazione di una matrice in cui ogni riga rappresenta un documento e ogni colonna una parola contenuta nel dizionario a cui è associato un peso. Le principali funzioni di *weighting*, per calcolare il peso, sono:

- *Binary: 1 se la parola è contenuta nel documento, 0 altrimenti*
- *Raw frequency: frequenza della parola nel documento*
- *TF-IDF: combinazione di term frequency e inverse document frequency*

La seconda tecnica prevede la conversione di vettori sparsi, che rappresentano una parola o un documento, in vettori densi di numeri reali. Matematicamente si tratta di modelli che proiettano uno spazio a molte dimensioni per parola in uno spazio vettoriale a meno dimensioni. Esistono due tipologie principali du *word embedding*: *count-based models*, che effettuano una *dimensionality reduction* sulla *term-context matrix* e *predictive models* che sfruttano reti neurali. In questo progetto si sono utilizzate come *text representation* una *term-document matrix* con funzione di weighting *TF-IDF* e le *word embedding* di tipo count-based, in particolare il modello *GloVe*

3.1 TF-IDF Matrix

TF-IDF matrix è una rappresentazione testuale che consiste in una matrice *M term-document* dove ogni riga rappresenta un documento, ogni colonna una parola e l'elemento m_{ij} indica il peso TF-IDF associato alla parola j nel

documento *i*. La funzione di peso TF-IDF, riportata nell'equazione (1) ,è la combinazione di due metriche: *term frequency* (TF) e *inverse document frequency* (IDF). La *term frequency* calcola la frequenza del termine t in ogni documento d, la *inverse document frequency* invece è una misura inversa dell'informatività del termine t, nello specifico è il logaritmo del rapporto tra il totale di documenti nel corpus e il numero di documenti in cui compare il termine t

$$TF - IDF_{t,d} = \frac{tf_{t,d}}{max_{t_i \in d}} \cdot log(\frac{N}{df_t}) \tag{1}$$

La TF-IDF matrix è stata implementata mediante una classe di *sklearn* con i seguenti paramentri:

- max_features = 10.000
- ngram_range = 1,1

Il primo parametro permette di considerare le n parole per commento ordinate in base alla *term frequency*. Il secondo definisce la tipologia di n-grammi utilizzati che in questo caso risultano essere solo unigrammi. Sebbene questa tipologia di funzione permetta di valutare in maniera efficace l'importanza di una parola all'interno di un corpus resta comunque il problema della sparsità della matrice. Per risolvere tale problema si è deciso di applicare i *word embedding*, più precisamente il *GloVe*.

3.2 GloVe

I GloVe embeddings sono una tipologia di word embedding che codificano la *co-occurence probability ratio* tra due parole come la differenza tra due vettori. GloVe usa la funzione (2) come peso, la quale minimizza la differenza tra il prodotto scalare dei vettori di due parole e il logaritmo del loro numero di co-occorrenze. Nel GloVe, per ottenere la rappresentazione vettoriale delle parole viene applicato un algoritmo non supervisionato. In questo contesto parole simili hanno una rappresentazione vettoriale simile.

$$J = \sum_{i,j=1}^{V} f(X_{ij})(\omega_i^T \acute{\omega_j} + b_i + \acute{b_j} - logX_i) \tag{2}$$

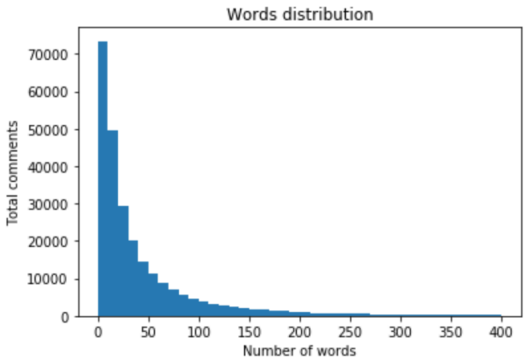

Figure 1: Distribuzione del numero di parole

È stata creata una *embedding matrix* dove ogni riga rappresenta una parola a cui è associata un vettore di numeri reali. La dimensione vettoriale che risultava più performante è stata quella con 200 features. Il numero di parole presenti in ogni commento è stato invece impostato a 200. Per i commenti che non avessero questa lunghezza viene applicato uno zero-padding mentre quelli con una lunghezza superiore vengono troncati. Questo valore è stato scelto dopo aver osservato la distribuzione del numero di parole per commento. Si è deciso un valore elevato in modo tale da essere più conservativi. Nella figura 1 è riportata la distribuzione del numero di parole.

Nella creazione della *embedding matrix*, utilizzata dall'embedding layer, si è deciso di tenere le 100.000 parole più comuni dalle 108877 parole uniche iniziali. Un'altra osservazione da fare è che per le parole non presenti all'interno del GloVe si è scelto di impostare valori random presi da una distribuzione normale con media e deviazione standard calcolate sui valori del GloVe stesso. Questo approccio aumentava la performance dei modelli rispetto ad impostare zero.

4 The Methodological Approach

Una volta ottenute due diverse matrici, uno per ogni tecnica di text representation, sono state implementate due architetture di rete neurale. La suddivisone dei dati è stata la seguente: Train 80% e Test 20%. Il Train a sua volta è stato diviso in 90% Train e 10% Validation. Di seguito vengono descritte più in dettaglio le due architetture:

4.1 Dense

1. Due strati densi con 2048 neuroni con funzione di attivazione elu.
2. Strato denso con 1024 neuroni con funzione di attivazione elu.
3. Strato denso con 512 neuroni con funzione di attivazione elu.
4. Strato denso finale con 4 neuroni con funzione di attivazione *softmax* in modo da avere che la somma delle probabilità classi sia 1. La *loss function* invece è la *sparse categorical crossentropy*.

Il modello è stato allenato in 4 epoche. Come ottimizzatore invece è stato scelto l'adam con parametri di default.
Questa rete neurale utilizza come input la matrice TF-IDF.

4.2 CNN + GRU

La seconda rete è composta da uno strato di CNN e uno di GRU in modo da catturare anche la relazione tra le parole. È stato scelto il GRU anziché l'LSTM in quanto il primo performava meglio e aveva migliori prestazioni computazionali.

1. Embedding con 200 features
2. Strato GRU con numero di unità pari a 80 e restituzione dell'ultimo output dalla sequenza di output
3. Convoluzione 1D con 64 filtri, dimensione del kernel pari a 3, stride pari a 1, inizializzazione dei parametri normale con funzione di attivazione *elu*. Successivamente vengono applicati un GlobalAveragePooling1D e un GlobalMaxPooling1D che a loro volta vengono concatenati.
4. Strato denso finale con 4 neuroni con funzione di attivazione *softmax* in modo da avere che la somma delle probabilità classi sia 1. La *loss function* invece è la *sparse categorical crossentropy*.

Il modello è stato allenato in 4 epoche. Come ottimizzatore invece è stato scelto l'adam con parametri di default.
Questa rete neurale utilizza come input la matrice degli embeddings.

5 Results and Evaluation

Di seguito nella tabella 1 vengono presentati i risultati ottenuti dai due modelli in fase di test:

Text representation	Loss	Accuracy
TF-IDF	0.3784	0.86854
Word Embeddings	0.3790	0.8622

Table 1: Performance by TR

Per valutare la performance dei modelli si è deciso di considerare l'accuracy come *measure* in quanto i dati sono ben bilanciati tra le quattro classi. Come si può osservare dai risultati non si evidenziano differenze significative sebbene il modello che utilizza la TF-IDF matrix risulti leggermente migliore. I risultati nei due casi sono buoni, si potrebbe ottenere un'accuracy più alta effettuando alcune modifiche progettuali.
Bisogna fare una considerazione sull'architettura sviluppata per ogni modello. Il modello TF-IDF ha una struttura meno complessa rispetto all'embedding che fa uso di *layers* complessi come CNN e GRU, ciononostante la performance ha retto il confronto. Questa struttura più legegra è stata scelta in base alla *shape* della TF-IDF matrix.
Uno dei motivi che potrebbe aver influito è che il modello TF-IDF fa uso di tutte le parole presenti nelle reviews, mentre il modello embedding usa solo fino a 200 parole per commento.
Il modello embedding contiene molto più *noisy* in confronto al TF-IDF. La rappresentazione vettoriale delle parole tramite embedding è più complessa e si porta dietro informazioni nascoste. In questo caso la maggior parte di questa informazione non è necessaria e potrebbe creare solo falsi patterns nel modello.
All'inizio di questo lavoro ci si aspettava che il modello che utilizza la matrice TF-IDF performasse meglio in quanto viene ritenuta una tecnica molto robusta e così è stato anche se la differenza è minima.
Durante la fase di training si è visto come dopo la seconda epoca i modelli iniziassero ad andare in leggero overfitting. Di seguito nelle figure 2 e 3 i grafici a dimostrazione:

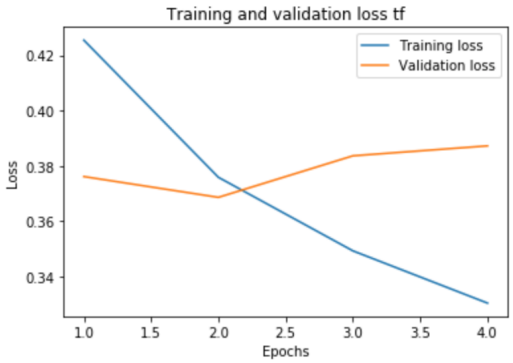

Figure 2: Loss for TF-IDF

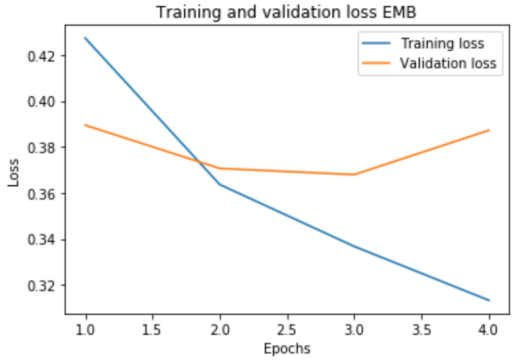

Figure 3: Loss for Word Embedding

6 Discussion And Conclusions

In conclusione possiamo dire che i due modelli eseguono correttamente il task richiesto e fornendo altre recensioni si riesca a predire con una buona probabilità la macro categoria.Per quanto riguarda il secondo obiettivo non si evidenziano significative differenze in termini di performance, ma considerando la differenza nelle architetture implementate bisognerebbe uniformare le stesse per avere una valutazione più equa.
Come miglioramenti si potrebbe avere a disposizione una capacità computazionale maggiore in modo da avere più dati per il training. Da questo punto di vista non sarebbe necessario avere un unico commento per prodotto. Anche per quanto riguarda la parte di modeling si può considerare di applicare tecniche di ottimizzazione all'avanguardia come per esempio quella Bayesiana.
Si può affermare dunque che queste due tecniche di rappresentazione del testo sono sicuramente tra le migliori, ma bisognerebbe effettuare altre prove per avere un'idea più chiara, anche in base al contesto di riferimento.

References

https://www.researchgate.net/publication/326425709_Text_Mining_Use_of_TF-IDF_to_Examine_the_Relevance_of_Words_to_Documents https://www.aclweb.org/anthology/W18-5406.pdf
https://webthesis.biblio.polito.it/12395/1/tesi.pdf