UNIVERSITÀ DI PISA

Text Analytics Project

Sentiment analysis su Alexa's reviews: un piccolo e sbilanciato data-set.

Cristiano Ciaccio (627883, c.ciaccio@studenti.unipi.it), Silvia Cosmo (628299, s.cosmo@studenti.unipi.it), Alice Graziani (544557, a.graziani5@studenti.unipi.it), Naomi Esposito (562797, n.esposito3@studenti.unipi.it)

ANNO ACCADEMICO 2022/2023

Introduzione

Il nostro progetto ha lo scopo di approfondire vari approcci alla **sentiment analysis** e, in particolare, la gestione della mancanza di dati e lo sbilanciamento di classi. Più specificamente, il nostro obbiettivo è la classificazione binaria del sentimento utilizzando il data-set **Amazon Alexa Reviews** tramite differenti classificatori, architetture neurali, e risorse lessicali, con un approccio sia supervisionato sia non supervisionato. Il data-set consiste in circa 3000 recensioni, ognuna annotata con un valore di sentimento, **1** se positiva (92% delle recensioni totali) o **0** se negativa (8%)

Particolare attenzione è stata posta al preprocessing del testo in modo da gestire la specificità del nostro compito; diverse tecniche di **over-sampling** e **under-sampling** sono state testate per bilanciare il data-set (come generazione di recensioni artificiali, **SMOTE**, **Tomek's Links**). Inoltre, vari approcci sono stati utilizzati per ottenere delle rappresentazioni vettoriali dei token o delle recensioni: **Word2vec** embedding sia pre-addestrati che non, modelli **TF-IDF Count**, fine-tuning di **BERT**. Alternativamente, per l'addestramento non supervisionato, gli embedding sono stati usati per ottenere dei centroidi del sentimento positivo e negativo, permettendo di calcolare un punteggio di similarità e, dunque, di sentimento.

In conclusione, quando il processo di addestramento è supervisionato, i dati testuali pre-processati vengono passati a diversi classificatori e architetture neurali: **Support Vector Machine** (**SVM**), **Feedforward neural network** (i.e., multilayer perceptron) and **Convolutional neural network** (**CNN**). Tutti i metodi sono stati valutati con le metriche di valutazione standard e paragonati fra di loro con lo scopo di trovare conclusioni e informazioni rilevanti per questo complesso compito.

Indice

Capitolo 1

Data processing

Come accennato nell'introduzione, è stata prestata molta attenzione alla parte di pre-elaborazione in quanto l'analisi del sentimento può essere impegnativa a causa dell'impatto della struttura sintattica sul significato di una frase. Inoltre, i compiti di classificazione in generale sono influenzati dalla qualità, dal bilanciamento e dalla quantità dei dati.

Con un totale di 3150 recensioni e una distribuzione delle classi del 92%-8%, il nostro data-set è, dunque, composto da 2893 recensioni positive (se con tre o più stelle), con un numero medio di caratteri di 11, e appena 257 recensioni negative (se con una o due stelle) e una media di 19 caratteri. Non solo il set di dati è piccolo e sbilanciato, ma contiene anche molte recensioni duplicate, una condizione che, se non gestita, può portare ad avere gli stessi dati nel train e nel test-set. Per evitare ciò e non pesare eccessivamente alcune feature, abbiamo deciso di eliminare tutte le recensioni duplicate, ben 849. Inoltre, poiché la classificazione è binaria, abbiamo rimosso le recensioni con tre stelle a causa della loro neutralità (rimangono 2196 recensioni).

1.1 Elaborazione dei dati testuali

Per meglio valorizzare le feature nei nostri dati, la parte di pre-processamento è stata implementata tenendo conto della specificità del nostro data-set: essendovi una parte di recensioni in spagnolo, queste sono state tradotte tramite le API di Google Translate; inoltre, è stato creato un elenco personalizzato di stop-word contenente token che fanno riferimento ad altri produttori, versioni del prodotto (" 1st", "2nd", ecc.) e vari fornitori di servizi (ad es. servizi di streaming e intrattenimento). La pipeline consiste in: rimuovere numeri e punteggiatura, tokenizzare il testo usando l'**NLTK** tokenizer, rendere minuscoli tutti caratteri, gestire gli avverbi di negazione ("not", "no", "non" ecc.), lemmatizzare usando il **WordNet** lemmatizer e l'**NLTK PoS tagger**, rimuovere token più corti di tre caratteri.

La funzione che abbiamo definito per il pre-processamento dei dati testuali può inoltre, se specificato negli argomenti, tokenizzare il testo annotando ogni parola con la sua **Part of Speech**:

```
sentence = ["It's like other competitors, same funcionalities."]
x, y = tokenize_list_of_text(sentence, pos_tagging=True, token_len=0)
print(x)

# outputs

["it_PRP 's_VBZ like_IN other_JJ competitor_NNS same_JJ funcionalities_NNS"]
```

In questa frase il token "like" è annotato come una preposizione, infatti non porta il significato emotivo tipico, invece, del verbo. Questo permette di ottenere un vocabolario più disambiguato e completo, ma non sempre ha portato a risultati migliori.

Infine, vengono estratte le collocazioni usando **Gensim Phrases**. La classe implementa l'estrazione di espressioni composte da più parole utilizzando varie funzioni e soglie: abbiamo optato per la **Normalized Pointwise Mutual Information** con una soglia del 60%.

1.1.1 Gestione delle negazioni

Nella sentiment analysis, la negazione gioca un ruolo importante in quanto inverte il significato della parola che precede. Per gestire ciò, abbiamo utilizzato un metodo che converte un'espressione negativa in un nuovo token che esprime l'inverso della parola negata[1]: ogni volta che viene trovato un token preceduto da una negazione, vengono estratti da **WordNet** tutti i synset disponibili e per ognuno di essi, sfruttando le relazioni semantiche del synset, si ricercano eventuali antonimi. Tra questi abbiamo scelto quello con il più alto grado di dissomiglianza (1 - **Wu-Palmer Similarity**) dalla parola negata. Il seguente codice mostra il comportamento della funzione:

```
sentence1 = nltk.tokenize.word_tokenize("I'm not happy about this product")
sentence2 = nltk.tokenize.word_tokenize("The audio is not strong, i don't recommend")
print(negation_handler(sentence1))
print(negation_handler(sentence2))
```

[1]Vedi `https://github.com/UtkarshRedd/Negation_handling`. La repository contiene il codice base del metodo, vengono apportate piccole modifiche al nostro codice.

```
5
6 # outputs:
7
8 ['I', "'m", 'unhappy', 'about', 'this', 'product']
9 ['The', 'audio', 'is', 'weak', ',', 'i', 'do', 'not_recommend'
    ]
```

Quando gli antonimi non vengono trovati, l'enunciato negato viene sostituito con una stringa (es. "not_recommend") come mostrato nel codice sopra, questo ci permette di avere una rappresentazione univoca e specifica dell'espressione negativa.

1.2 Bilanciamento

Il bilanciamento dei dati, durante l'addestramento di un classificatore, è una tecnica utilizzata per prevenire lo sviluppo di bias verso la classe maggioritaria. Nel nostro caso, ogni classificatore supervisionato, quando i dati sono sbilanciati, ottiene sempre un'elevata **precision** e una bassa **recall** per la classe negativa, risultando in una valutazione scadente poiché, a causa del bias nella distribuzione delle classi,una recensione per essere classificata come negativa deve essere fortemente composta da feature negative. Per il nostro specifico compito, si è puntato ad avere una buona copertura della classe negativa, massimizzando la **recall**, rendendo il bilanciamento necessario per ottenere tale risultato. La popolazione, quindi, viene ridimensionata a una distribuzione del 67%-33% per mantenere un bias minore, risultando in 412 recensioni positive e 206 recensioni negative.

1.2.1 Data augmentation e Tomek's Links

Si è optato per la generazione di dati sintetici per la classe negativa in modo da aumentare il numero di feature rappresentative per questa categoria. Poichè non sempre si è lavorato con delle rappresentazioni vettoriali geometricamente significative (modelli **Count TF-IDF** o token IDs), si è optato per la generazione di recensioni artificiali piuttosto che la creazione di data-points sintetici (i.e., **SMOTE**). In aiuto di questa tecnica è stato addestrato un modello **Word2Vec** (vector_size = 100, window = 10) su un data-set di circa 20000 recensioni di prodotti Alexa estratto da Amazon.

La metodologia usata sfrutta **WordNet** e **Word2Vec** per estrarre sinonimi con cui generare le nuove recensioni artificiali: inizialmente viene scelto un campione di recensioni negative random, circa 1/2; per ogni recensione e per ogni token, si estraggono con **WordNet** i synsets disponibili; per ognuno di questi si estraggono eventuali sinonimi ai quali viene associato un punteggio calcolato come la similarità (si usa la funzione **.similarity()** di **Word2Vec**) fra il sinonimo e il token originale; trovato il punteggio massimo e se questo supera una certa soglia (0.20), allora alla

recensione artificiale verrà aggiunto il sinonimo, altrimenti il token originale. Un esempio (a sinistra la recensione originale, a destra l'artificiale):

```
['joke'] ---> ['laugh']
['never', 'work'] ---> ['never', 'run']
['speaker', 'pretty', 'terrible', 'home', 'good', 'product']
--->
['speaker', 'fairly', 'awful', 'house', 'well', 'product']
```

L'altro approccio utilizzato, in questo caso per fare under-sampling, sono i **Tomek's Links**. Questi definiscono dei punti nello spazio vettoriale che sono vicini fra loro appartenendo però a classi diverse, si tratta, dunque, di recensioni ambigue. In tale caso, questi punti vengono eliminati.

Entrambe queste tecniche di over-sampling e under-sampling vengono utilizzate esclusivamente sul train-set (329 positive e 165 negative) che, in seguito alle modifiche sopracitate, diviene composto da 317 recensioni positive e 229 negative.

Capitolo 2

Classificazione

In questa fase abbiamo provveduto ad effettuare l'addestramento supervisionato tramite i modelli **SVM** (Support Vector Machine) e **MNB** (Multinomial Naive Bayes). Effettuato il pre-processing sulle recensioni, seguendo gli step indicati nella sezione precedente, abbiamo utilizzato due pipeline diverse per poter preparare i dati avvalendoci delle classi e dei metodi della libreria **sk-learn**:

1. Tramite la classe **CountVectorizer** è stata ottenuta una document-term matrix popolata dalla frequenza assoluta dei token (frequenza minima due); avendo estratto sia monogrammi che bigrammi si ottengono 1224 feature (equivalente alla dimensione del vocabolario). Si procede con la feature selection tramite **SelectKBest**, sulla base del Chi2 (misura di dipendenza tra le variabili) sono state tenute le 800 parole più importanti; infine, si calcolano i punteggi **TF-IDF**.

2. L'altra metodologia prvede di calcolare la media ponderata degli embedding **Word2Vec** (addestrati sul nostro dataset, skip-gram, window_size = 10, 100 dimensioni) pesati con i valori **TF-IDF** ottenendo, così, un vettore per ogni recensione.

2.1 Modello Count-KBest-TF-IDF

Per quanto riguarda il Multinomial Naive Bayes, sono stati ottenuti i seguenti risultati:

	precision	**recall**	**f1-score**	support
negative	0.90	0.46	0.61	41
positive	0.98	0.79	0.87	83
accuracy			0.81	124
macro avg	0.85	0.72	0.74	124

Lo sbilanciamento del data-set influisce molto nelle previsioni effettuate dal classificatore; come si può notare dalla bassa **recall** della classe negativa: il modello identifica con maggior semplicità gli esempi positivi da quelli negativi. Nonostante ciò, la **precision** di entrambe la classi è molto alta.

Per l'**SVM**, invece, è stato utilizzata una **Grid-Search** al fine di selezionare gli iperparametri che massimizzassero l'**f1-macro**. L'addestramento è stato eseguito tramite la **Stratified KFold Cross-validation** con 5 fold.

Gli iperparametri selezionati per implementare **SVM** sono:

- Kernel, che determina la funzione usata per trasformare i dati in input in uno spazio ad alto numero di dimensioni;
- C, che controlla il trade off tra aderenza ai dati di addestramento e generalizzazione del modello; In altre parole, C controlla la quantità di penalizzazione assegnata a eventuali errori di classificazione nel processo di addestramento del modello;
- Gamma, parametro che controlla l'influenza di ogni esempio del training;
- Class weight, parametro di regolazione che viene utilizzato per gestire le classi sbilanciate.

Seguono i risultati ottenuti rispettivamente senza e con PosTagging:

	precision	**recall**	**f1-score**	support
negative	0.74	0.68	0.71	41
positive	0.85	0.88	0.86	83
accuracy			0.81	124
macro avg	0.79	0.78	0.79	124

	precision	**recall**	**f1-score**	support
negative	0.71	0.59	0.64	41
positive	0.81	0.88	0.84	83
accuracy			0.78	124
macro avg	0.76	0.73	0.74	124

Si nota come l'**SVM** sia in grado di meglio discernere il sentimento, con una **recall** decisamente più alta per entrambe le classi a discapito, però, della **precision**.

2.2 Modello Word2Vec & Tf-Idf

Si è provveduto a combinare l'informazione semantica contenuta negli embedding di **Word2Vec** con il coefficiente di importanza espresso da **TF-IDF**; inoltre, si è optato per una rappresentazione vettoriale dell'intera recensione secondo la seguente formula, dove E rappresenta l'embedding e w il coefficiente **TF-IDF**:

$$\mathbf{V}_{rev} = \frac{1}{n} \sum_{1}^{n} E_i * w_i$$

Dopo aver utilizzato il metodo Tomek's Links per rimuovere le recensioni ambigue e la **Grid-search**, questi sono i risultati:

	precision	**recall**	**f1-score**	support
negative	0.77	0.73	0.75	41
positive	0.87	0.89	0.88	83
accuracy			0.84	124
macro avg	0.82	0.81	0.82	124

I risultati mostrano un miglior bilanciamento fra **precision** e **recall** per entrambe le classi, con un**f1-score** macro di 0.82, 3 punti percentuali sopra la metodologia precedentemente discussa.

Capitolo 3

Reti Neurali

Questa sezione descrive le diverse architetture neurali e confronta i risultati ottenuti per individuare quale metodologia sia la più solida con riferimento al nostro task. Sono state utilizzate tre architetture diverse: un **Multi-layer Perceptron** addestrato sulle recensioni vettorizzate e pesate con **TF-IDF**, un **Convolutional Neural Network** e, infine, fine-tuning di **BERT** seguito da due dense-layer.

Il data-set viene sempre diviso in 80% train e 20% test, del train il 20% è usato come validation-set.

3.1 Multi-layer Perceptron

Per questa metodologia i dati sono stati preparati nel seguente modo: vengono calcolati i pesi **TF-IDF** e vengono estratti gli embedding con un modello **Word2Vec** addestrato sul nostro data-set (skip-gram, dimensione 100 e finestra 10); infine, si ottiene un vettore per ogni recensione (**V**rev) tramite la sommatoria degli embedding (E) ponderata per il rispettivo peso **TF-IDF** (w):

$$\mathbf{V}_{rev} = \frac{1}{n} \sum_{1}^{n} E_i * w_i$$

Una volta applicata questa trasformazione sia al train che al test-set, i dati vengono passati a una semplice rete **Feed Forward** strutturata come segue:

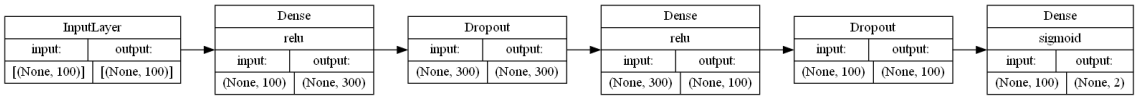

Figura 3.1: Architettura del Multi-layer Perceptron utilizzata

La rete viene addestrata con una **KFold Cross Validation** (5 splits) per 100 epoche (interviene un meccanismo di early stopping con pazienza 25) e con una batch size di 32. Seguono i risultati:

	precision	**recall**	**f1-score**	support
negative	0.76	0.78	0.77	41
positive	0.89	0.88	0.88	83
accuracy			0.85	124
macro avg	0.83	0.83	0.83	124

La valutazione mostra un buona capacità da parte della rete di classificare le recensioni con un **f1-score macro** di 0.83 e un buon bilanciamento fra **precision** e **recall** per entrambe le classi.

3.2 Convolutional Neural Network

Le architetture di tipo **Convolutional** sono in grado di apprendere rappresentazioni e generalizzazioni di alto livello grazie all'utilizzo della convoluzione e dei **Kernel**. Nonostante trovino un grande utilizzo nella computer vision, negli ultimi anni sono state adottate nel language processing con ottimi risultati.

Due sono gli approcci che abbiamo adottato: il primo prevede di vettorizzare le recensioni con gli id associati ad ogni token e, tramite il layer **Embedding** di

Keras, addestrare i vettori risultanti specificamente al task; il secondo, invece, parte dagli embedding pre-addestrati (cfr 1.2.1) per, ugualmente, fare fine-tuning specifico al task. La prima metodologia ha ottenuto approssimativamente lo stesso risultato del **Multi-layer Perceptron**, per cui si discuterà esclusivamente della seconda.

Per comodità di visualizzazione l'architettura è stata divisa in due parti:

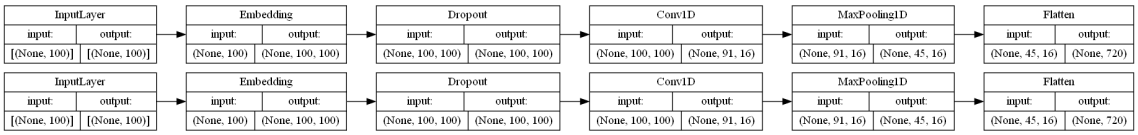

Figura 3.2: Architettura CNN

La prima riga mostra il layer Embedding che si connette allo strato Convolutional a 1 dimensione, vengono usati 16 filtri con una kernel_size di 10 (come la window usata in **Word2Vec**), segue il **Max Pooling** con una size di 2 e un **Flatten** layer che riporta i dati nella corretta dimensionalità; la seconda riga, invece, mostra un semplice **Multi-layer Perceptron**. Seguono i risultati:

	precision	**recall**	**f1-score**	support
negative	0.82	0.80	0.81	41
positive	0.90	0.92	0.91	83
accuracy			0.88	124
macro avg	0.86	0.86	0.86	124

Come si evince dai risultati, questa architettura ha una valutazione migliore rispetto alle precedenti e ai classificatori descritti nel capitolo 2. Sia **precision** che **recall** sono ben bilanciate per entrambe le classi, con l'**f1-score** macro che è aumentato di 3 punti percentuali rispetto al **Multi-layer Perceptron**.

3.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) è un modello di comprensione del linguaggio naturale che utilizza l'architettura Transformer per comprendere le parole in un contesto specifico. A differenza dei modelli unidirezionali, i quali leggono i dati in input in successione, BERT utilizza un meccanismo di attenzione basato su Transformer per decidere quali parti del testo siano più salienti nel comprendere il contesto, considerando in modo bidirezionale sia le parole presenti a sinistra che a destra di una data parola.

I dati sono pre-processati secondo la pipeline descritta nel capitolo precente, tuttavia, al fine di poter eseguire il modello, è stato necessario effettuare ulteriori elaborazioni. La tokenizzazione delle frasi è stata compiuta attraverso **AutoTokenizer** con metodo "encode plus", il quale restituisce un dizionario contenente **ids**

ed **attention masks**, ovvero l'input del nostro modello BERT. Esso, in aggiunta, richiede che le frasi in input abbiano tutte la medesima lunghezza, perciò si è provveduto ad impostare una lunghezza massima, effettuando il padding per adattare tutte le frasi a tale misura. Il valore di tale parametro è stato selezionato osservando la lunghezza di tutte le recensioni ed optando per il massimo (134): non disponendo di un dataset molto grande, tale decisione ha fatto sì che non si perdessero alcune informazioni, evitando quindi di troncare alcune frasi, senza che venga superata la soglia massima di token (512) imposta dal modello. Sono stati, inoltre, aggiunti token speciali [CLS] e [SEP], la cui presenza permette al modello di comprendere la struttura del testo e di elaborarlo in modo più preciso. In un secondo momento gli id e le attention masks sono stati trasformati in tensori.

La costruzione di una rete neurale è stata effettuata con le librerie **TensorFlow** e **Keras** ed utilizzando il modello pre-addestrato BERT come encoder. I tensori vengono utilizzati come input per l'encoder ed il suo output viene successivamente elaborato dalla rete neurale. Il modello viene compilato con una funzione di perdita **binary cross-entropy** e un ottimizzatore **Adam**. Durante l'addestramento è stato possibile osservare che tanto la funzione di perdita quanto l'accuracy migliorino durante il training e durante la validazione. I valori di loss e accuracy durante la validazione si presentano leggermente più bassi che durante il training. Concludiamo che il modello non stia memorizzando i dati di training, e sia in grado di generalizzare su nuovi dati.

	precision	**recall**	**f1-score**	support
negative	0.83	0.93	0.87	41
positive	0.96	0.90	0.93	83
accuracy			0.91	124
macro avg	0.89	0.92	0.90	124

I risultati finali mostrano una buona capacità della rete si svolgere il task della classificazione, con un generale miglioramento delle metriche di valutazione.

Capitolo 4

Approcci non supervisionati

Il capitolo introduce due tecniche per la **Sentiment Analysis** non supervisionata: la prima utilizza delle risorse linguistiche, dei lessici nei quali ad ogni token viene

associato un valore emozionale; la seconda si avvale di embedding **Word2Vec** preaddestrati (cfr. cap. 1.2.1) per generare un lessico emozionale che meglio si adatta al nostro contesto linguistico. Poichè si tratta di apprendimento non supervisionato il data-set non è stato diviso in train e test e non è stato fatto bilanciamento.

4.1 Lessici emozionali

La **weak labeling** è un concetto che si riferisce alla pratica di fornire etichette considerate come "deboli" poiché potrebbero essere sbagliate o imprecise, rispetto a una "etichettatura forte" dove le etichette sono definite in modo preciso e completo (etichettatura manuale). Queste tecniche possono essere utilizzate in situazioni in cui è difficile o costoso ottenere etichette complete o precise: è un modo utile per risolvere il problema dell'etichettatura dei dati, fondamentale per il successo dell'apprendimento automatico. Le metodologie utilizzate sono **NRC Lexicon, NRC-VAD, VADER e TextBlob**.

4.1.1 Metodi utilizzati

NRC Lexicon è uno strumento sviluppato dal National Research Council of Canada che offre una valutazione delle emozioni associate a parole specifiche. Il lexicon include una lista di 14.000 parole con un punteggio assegnato per otto emozioni diverse: felicità, tristezza, paura, sorpresa, disgusto, rabbia, fastidio e neutralità. Questi punteggi vengono ottenuti da una valutazione umana delle emozioni associate a ciascuna parola. Altro strumento sviluppato dal NRC è **NRC − VAD Lexicon**(Valence, Arousal e Dominance) che include un elenco di oltre 20.000 parole inglesi e i loro punteggi di valenza, eccitazione e dominanza. Per una data parola e una dimensione (V/A/D), i punteggi vanno da 0 (V/A/D più basso) a 1 (V/A/D più alto). Si tratta di uno dei lessici notevolmente più grandi di qualsiasi altro lessico VAD esistente. **VADER** (Valence Aware Dictionary and sEntiment Reasoner) è un modello di sentiment analysis che viene utilizzato per valutare l'opinione o l'emozione espressa in un testo. È stato specificamente progettato per l'analisi del sentimento su testi scritti in lingua inglese, si tratta di un modello basato su un dizionario che tiene conto delle connotazioni positive, negative o neutrali associate a ogni parola: analizza il testo e calcola un punteggio che indica la valenza (positiva, negativa o neutrale) del testo. VADER è molto preciso perché tiene conto di molte informazioni contestuali, compresa la punteggiatura, per questo sarà effettuata una prova anche con il testo contenente la punteggiatura. **Textblob** fornisce una semplice API per immergersi in attività comuni di elaborazione del linguaggio naturale (NLP) come **PoS Tagging**, estrazione di frasi nominali, analisi del sentiment, classificazione, traduzione e altro ancora. Si tratta di una tecnica di analisi dei sentimenti che restituisce due proprietà per una determinata frase di input: - la polarità che è compresa tra [-1,1], -1 indica un sentimento negativo e +1 indica

sentimenti positivi. - la soggettività che è compres a tra [0,1]: - le frasi soggettive generalmente si riferiscono a opinioni, emozioni o giudizi.

4.1.2 Risultati

Le metodologie sono state applicate a tutte le recensioni all'interno del dataset. Al fine di comprendere meglio i risultati ottenuti, abbiamo confrontato l'etichettatura triviale del dataset, con le etichettature ottenute dalle metodologie usate: il risultato è riportato come **ER-class**, il quale indica il numero di predizioni etichettate in modo errato. Come è possibile vedere nella tabella sotto, Vader ha ottenuto risultati leggermente migliori se in input inseriamo le recensioni con annessa punteggiatura (nella tabella VADER 2). NRCLex, invece, ha prodotto il risultato meno apprezzato, probabilmente penalizzato anche dal numero ristretto di recensioni; mentre, la versione NRC-VAD è risultata essere di gran lunga più performante. Tra i metodi utilizzati, quelli che hanno prodotto un risultato migliore risultano essere TextBlob e NRC-VAD, i quali hanno ottenuto rispettivamente un errore nella predizione di 185 e 451 elementi.

	Positive		**Negative**			
	Precision	Recall	Precision	Recall	ER Class	Accuracy
NRC	0.99	0.06	0.10	1.00	1863	0.15
NRC-VAD	0.96	0.80	0.27	0.72	451	0.79
VADER	0.97	0.92	0.23	0.78	594	0.72
VADER 2	0.97	0.73	0.24	0.81	580	0.74
TextBlob	0.93	0.98	0.59	0.33	185	0.92

4.2 Generazione di lessici emozionali con Word2Vec

Un primo approccio è stato quello di esplorare lo spazio vettoriale tramite varie tecniche di clustering (**K-Means** e **Hierarchical Clustering**), i risultati, però, non hanno mostrato gruppi di data-points legati al sentimento positivo e negativo (se non per un cluster di parole che riguardano i malfunzionamenti del prodotto). Piuttosto, avendo notato uno spike nel **Silhouette Score** con un k = 2, l'analisi ci ha permesso di individuare un cluster di recensioni in lingua spagnola che abbiamo provveduto a tradurre tramite le API di Google Translate; inoltre, è stato possibile realizzare, tramite **TSNE**, uno rappresentazione 3d dello spazio vettoriale che è possibile navigare nel file **w2vNonSup.ipynb**.

Il nostro approccio si ispira alla metodologia descritta nel capitolo 25 di "Speech and Language Processing" di Daniel Jurafsky & James H. Martin, specicamente nella sezione 25.4 "**Semi-supervised Induction of Affect Lexicons**". Inizialmente, vengono definite due liste di token, una di parole positive e una di negative; queste hanno la funzione di centroidi, ovvero punti di estrema rappresentatività del sentimento. Calcolando per ogni lista la media dei vettori, come riportato nelle seguenti

formule, si ottengono due punti fittizi nello spazio vettoriale, uno rappresentativo della positività e uno della negatività:

$$\mathbf{V}^{+} = \frac{1}{n}\sum_{1}^{n} E({w_i}^{+})$$

$$\mathbf{V}^{-} = \frac{1}{m}\sum_{1}^{m} E({w_i}^{-})$$

Viene ora calcolato l'asse semantico che descriva la "direzione" vettoriale della positività. Ciò può essere derivabile semplicemente dalla sottrazione del polo positivo con quello negativo:

$$\mathbf{V}_{axis} = \mathbf{V}^{+} - \mathbf{V}^{-}$$

Infine, si calcola un punteggio dato dalla similarità (**Cosine Similarity**) fra la rapprentazione vettoriale di ogni token nel nostro data-set e l'asse semantico sopra calcolato. Seguono i 10 token con punteggio più alto per ogni classe:

word	**score**	**word**	**score**
happy	0.520724	horrible	-0.447070
fantastic	0.500865	awful	-0.437519
great	0.455940	afterwards	-0.391189
pleased	0.439680	unresponsive	-0.388564
awesome	0.436962	hate	-0.342750
enjoy	0.406554	seriously	-0.330945
nice	0.402090	microphone	-0.328395
love	0.384281	headache	-0.324527
lyric	0.354649	bad	-0.322235
worthwhile	0.337414	questionable	-0.320123

Per la classificazione di una recensione, vengono sommati gli score di ogni token (solo se questo supera una soglia di attivazione di 0.03): quando il punteggio ottenuto è maggiore di 0 la recensione viene considerata positiva, viceversa negativa. Poichè si tratta di classificazione non supervisionata, questa viene fatta sull'intero data-set senza bilanciamento. Seguono i risultati:

	precision	**recall**	**f1-score**	support
negative	0.41	0.72	0.52	206.00
positive	0.97	0.89	0.93	1990.00
accuracy			0.88	
macro avg	0.69	0.80	0.72	2196.00

Considerando il grande sbilanciamento fra le due classi, conviene guardare alle valutazioni singolarmente: le recensioni positive vengono classificate correttamente,

con **precision** e **recall** alte e bilanciate; diversamente, le negative vengono in buona parte individuate, il 72% del totale, mentre la **precision** è al 41%. Nonostante ciò, poichè riteniamo più importante massimizzare la **recall** della classe negativa, il risultato è interessante. Inoltre, questa metodologia porta con sé dei vantaggi, in quanto, per via della semplicità del metodo, ha un buon livello di explainability. Ad esempio volendo predire il sentimento della frase: "Awful experience, not recommend. The sound quality is bad and it's difficult to set up", è possibile ispezionare i risultati:

```
#total number of types extracted is: 8
#awful : -0.437518946085277
#experience : -0.0017466939248616737
#not_recommend : -0.10725275237774456
#sound : 0.08728140535524974
#quality : 0.058605065462636136
#bad : -0.3222348857971328
#difficult : -0.0026027480483543073
#set : 0.12453535259016155
[0] ---> negative
```

Capitolo 5

Conclusioni e risultati

Nella discussione sono state applicate varie metodologie per la **Sentiment Analysis** su un data-set di piccole dimensioni, sbilanciato e di cattiva qualità. Nonostante questo, le diverse tecniche utilizzate sono state in grado di "addolcire" il data-set e sono state fatte riflessioni sugli effetti del bilanciamento e sulle conseguenze dei classificatori e architetture utilizzati. Infatti, ognuna di queste ha prodotto risultati variegati, talvolta massimizzando **recall** piuttosto che **precision** e viceversa. Focalizzandoci sull'armonia delle metriche di valutazione, l'**f1-score** macro è sicuramente quella che meglio descrive la performance di un classificatore. Se invece si volesse guardare alla maggior copertura sulla classe negativa (tralasciando un eventuale alto numero di falsi positivi), allora la **recall** sarebbe di certo la metrica da osservare. Come si potrà notare dalla tabella che riporta le diverse valutazioni, i classificatori sono in ordine crescente di **f1-macro** e **accuracy** a dimostrare che le diverse metodologie hanno sempre prodotto un miglioramento rispetto alle precedenti. Segue la tabella per l'apprendimento non supervisionato:

	Positive		**Negative**			
	Precision	Recall	Precision	Recall	F1-macro	Accuracy
count+tfidf MNB	0.98	0.79	0.90	0.46	0.74	0.81
count+tfidf SVM	0.85	0.88	0.74	0.68	0.79	0.81
w2v*tfidf SVM	0.87	0.89	0.77	0.73	0.82	0.84
w2v*tfidf multilayer	0.89	0.88	0.76	0.78	0.83	0.85
w2vPt + CNN	0.90	0.92	0.82	0.80	0.86	0.88
BERT	0.96	0.90	0.83	0.93	0.90	0.91

Bibliografia

[1] Daniel Jurafsky & James H. Martin. Speech and language processing. `https://web.stanford.edu/~jurafsky/slp3/`. Last visit: 02.12.2023.

[2] Saif Mohammad. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 1:174–184, 2018.

[3] Saif M. Mohammad and Peter D. Turney. Nrc emotion lexicon. *National Research Council*, 2:234, 2013.

[4] Utkarsh Redd. Negation_handling. `https://github.com/UtkarshRedd/Negation_handling`. Last visit: 02.12.2023.