Progetto di Text Mining

Davide Banfi – 806539
Alessandro Pigato – 852482

14 GENNAIO 2021

Indice

1. Introduzione

Ispezione ed elaborazione del dataset

Il progetto prevede l'analisi delle recensioni contenute nel dataset *Amazon Fine Food Reviews*[1], avente 568.454 documenti e 10 variabili caratteristiche del profilo che ha effettuato la recensione (UserId, Id del prodotto recensito, nome del profilo, …).

È possibile suddividere il lavoro in tre macrosezioni.

1. Nella prima sezione si propongono le principali tecniche di preprocessamento dei dati testuali quali tokenizzazione, lemmatizzazione, POS tagging e rappresentazione dei documenti.
2. Nella seconda sezione viene effettuata l'analisi del sentimento dapprima con il modello VADER di NLTK e successivamente tramite un modello di machine learning.
3. Nell'ultima sezione si riporta il clustering dei documenti tramite l'algoritmo KMeans.

Essendo il dataset molto grande e non avendo a disposizione un'elevata potenza di calcolo, vengono considerate solamente 250.000 recensioni estratte in modo casuale.

Queste sono ulteriormente suddivise in due subsets in proporzione 70:30, rispettivamente il training set usato per addestrare il modello di machine learning e il test set su cui validarlo.

L'analisi e i metodi di NLP sono stati effettuati solamente sui 75.000 documenti del test dataset.

In realtà, non tutte le variabili sono utili, motivo per cui si mantengono solo le seguenti tre:

- **Score**: variabile intera che esprime il tasso di gradimento del prodotto.
  Originalmente i valori sono situati nell'intervallo [1, 5] ma, causa problemi di compatibilità con keras[2], vengono diminuiti tutti di 1.
- **Text**: variabile stringa contenente il testo della recensione.
- **Summary**: variabile stringa contenente un breve riassunto della recensione.

L'informazione complessiva apportata dalla singola recensione è stata ottenuta concatenando in una nuova variabile "*Text*" il testo contenuto nel sommario e nella recensione stessa. Tuttavia, la presenza di NaN in Summary mina la validità di questa operazione dato che la somma tra una stringa e un NaN restituisce NaN.

La soluzione proposta è stata rimpiazzare i valori non noti con una stopword arbitraria tale da non influire sul sentimento complessivo della frase. Di fatto, la stop word verrà rimossa proprio grazie alla funzione messa a disposizione di NLTK.

	Score	Text
0	3	Great For Dipping! My mother-in-law introduced...
1	3	Pleasant A very pleasant mild tea with just a ...
2	4	best tea ever I've searched in every grocery s...
3	3	Green Tea Powder I received my package of Gree...
4	4	Best thing that ever happened to my baby! :) M...

Si analizza, infine, la distribuzione di Score e si riscontra una forte class imbalance che, specialmente nella fase di addestramento della futura rete, potrebbe distorcere le previsioni e la conseguente performance.

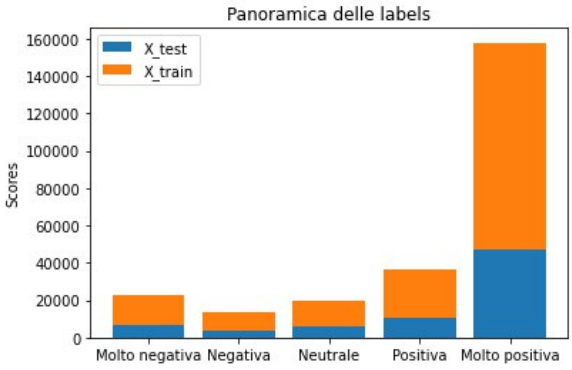
[1] Il dataset è reperibile al link Amazon Fine Food Reviews.

[2] L'indicizzazione di keras parte obbligatoriamente da 0.

Progetto di Text Mining

Davide Banfi – 806539
Alessandro Pigato – 852482

14 GENNAIO 2021



È stato ritenuto opportuno non correggere questo difetto non trattandosi puramente di un progetto di machine learning e anche per via del fatto che le dimensioni dei sets si sarebbero ridotte drasticamente.

2. NLP Preprocessing

Applicazione delle tecniche di preprocessing

Nel seguente capitolo vengono riportati i risultati derivanti dall'applicazione di metodi di processamento testuale.

Normalizzazione

Quando si lavora con documenti reali il testo conserva aspetti culturali del produttore e di conseguenza sarà molto eterogeneo. Una normalizzazione immediata viene eseguita sostituendo alle forme contratte della lingua inglese le relative forme estese.

	Score	Text
0	3	Great For Dipping! My mother-in-law introduced...
1	3	Pleasant A very pleasant mild tea with just a ...
2	4	best tea ever I have searched in every grocery...
3	3	Green Tea Powder I received my package of Gree...
4	4	Best thing that ever happened to my baby! :) M...

[3] Termini per lo più fino a 3 caratteri come mgf, khr, mge, esq, gsa, yr, così come molti errori di ortografia.

[4] Essendo un lavoro focalizzato sulle parole, i numeri e le parole "miste" sono stati esclusi.

Nel corpo di testo sono state ritrovate anche molte parole che non hanno senso compiuto[3].

La funzione *normalize( )* sfrutta una lista manualmente sviluppata e le espressioni regolari per rimuovere tutti quei termini fino a 3 caratteri senza significato, riducendo così il *noise* nei dati. Dopodiché rimuove le parole contenenti almeno un numero[4], elimina i caratteri non alfanumerici, elimina le stringhe vuote e converte tutti i caratteri in minuscolo.

Text	Normalized
Great For Dipping! My mother-in-law introduced...	great for dipping my mother in law introduced ...
Pleasant A very pleasant mild tea with just a ...	pleasant a very pleasant mild tea with just a ...
best tea ever I have searched in every grocery...	best tea ever i have searched in every grocery...
Green Tea Powder I received my package of Gree...	green tea powder i received my package of gree...
Best thing that ever happened to my baby! :) M...	best thing that ever happened to my baby my on...

A questo punto sul testo normalizzato vengono attuate tutte le successive tecniche.

Tokenizzazione

La funzione *word_tokenize( )* di NLTK separa le parole producendo un buon risultato proprio grazie alla funzione di cui al paragrafo precedente.

Iterando su ogni lista di tokens è possibile ricavare la dimensione complessiva del vocabolario.

Normalized	Tokens
great for dipping my mother in law introduced ...	[great, for, dipping, my, mother, in, law, int...
pleasant a very pleasant mild tea with just a ...	[pleasant, a, very, pleasant, mild, tea, with,...
best tea ever i have searched in every grocery...	[best, tea, ever, i, have, searched, in, every...
green tea powder i received my package of gree...	[green, tea, powder, i, received, my, package,...
best thing that ever happened to my baby my on...	[best, thing, that, ever, happened, to, my, ba...

Stop words removal

La rimozione di tutte quelle parole che non apportano nessuna informazione utile al testo[5] è stata eseguita

[5] Sono articoli, preposizioni, avverbi, pronomi, ecc.

utilizzando la lista di stop words fornita di default da NLTK. Sono stati esclusi da tale lista gli avverbi di negazione e tutte le varianti[6].

	Filtered_tokens	Removed_stopwords
	[great, dipping, mother, law, introduced, sauc...	[for, my, in, me, to, this, and, i, have, it, ...
	[pleasant, pleasant, mild, tea, hint, bergamot...	[a, very, with, just, a, of, and, too, it, is,...
	[best, tea, ever, searched, every, grocery, st...	[i, have, in, for, this, and, it, any, more, i...
	[green, tea, powder, received, package, green,...	[i, my, of, with, and, it, was, as, in, the]

Giunti a questo punto, nella colonna Filtered_tokens sono contenute liste quanto più pulite di tutte le parole facenti parte del vocabolario del nostro corpo di testo.

Ricordiamo che gli errori di ortografia non sono stati corretti vista la mancanza di una conoscenza assoluta e che, sebbene le parole corte siano state filtrate dal nonsense, rimangono comunque presenti quelle di lunghezza maggiore; dunque, nei dati permane un certo grado di eterogeneità e di ambiguità.

Stemming

Iterando sugli elementi delle liste di parole filtrate di cui al punto precedente si ricava il vocabolario di tokens unici. Su questi sono stati applicati due algoritmi di stemming: il PorterStemmer e lo SnowballStemmer.

I risultati sembrano suggerire che lo SnowballStemmer abbia lavorato meglio rispetto al PorterStemmer.

	Token	PorterStem	SnowballStem
	citrus	citru	citrus
	exactly	exactli	exact
	concious	conciou	concious
	slightly	slightli	slight
	certainly	certainli	certain

Lemmatizzazione

La riduzione dei tokens ai rispettivi lemmi è stata eseguita usando la classe *WordNetLemmatizer( )*.

	Token	Lem
6	years	year
11	kids	kid
21	rolls	roll
26	starts	start
46	flavors	flavor

L'algoritmo di lemmatizzazione ha fatto un ottimo lavoro solo su una piccola porzione del vocabolario, trattasi per lo più di sostantivi.

Non si può dire la stessa cosa per i verbi e le altre parti del discorso: il motivo è che il lemmatizzatore WordNet ha bisogno di sapere a che parte del discorso corrisponde una determinata parola.

Se non lo si specifica, l'algoritmo tratta tutte le parole come sostantivi.

	Token	Lem
98	expecting	expecting
99	one	one
100	rich	rich
101	full	full
102	smoky	smoky

POS Tagging

È noto che per migliorare l'efficacia dell'algoritmo di lemmatizzazione è bene attribuire a ciascun token la relativa *part of speech tag*.

WordNet mette a disposizione il tagger *averaged perceptron* per attribuire diverse POS tags[7]. Nel nostro caso sono state considerate solamente quattro classi

[6] "not", "no", "nor" potrebbero cambiare completamente il senso della frase.

[7] La lista completa delle tags è reperibile al link POS tags.

grammaticali: verbo (V), aggettivo (J), nome (N) e avverbio (R).

	Token	Lem_without_POS	Lem_with_POS
3	growing	growing	grow
7	taken	taken	take
17	better	better	good
40	guaranteed	guaranteed	guarantee
53	thrilled	thrilled	thrill

L'introduzione dei tags ha permesso al lemmatizzatore di trovare molte più parole di cui precedentemente non era riuscito a trovarne il lemma. Ad esempio, è stato in grado di rilevare che "better" deriva da "good".

3. Text representation

Codifica e rappresentazione delle frasi

Il modello proposto inizialmente per la codifica e rappresentazione dei documenti è stato il classico modello Bag of Words, cioè un modello che si basa sostanzialmente sul conteggio dei tokens all'interno di ogni frase.

Il primo passo è stato costruire un dizionario dove alle parole venisse assegnato un indice di riferimento, e viceversa. La frequenza delle parole, ottenuta iterando su ogni token e su ogni recensione, è stata assegnata alla posizione avente indice pari a quello del token in esame.

Avendo a che fare con 75.000 recensioni e con quasi 50.000 vocaboli unici si è così definita una matrice sparsa di $3.75 * 10^9$ valori.

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Una codifica di questo tipo è fuori luogo quando si ha a che fare con un corpus di testo ingente come l'Amazon Fine Food Reviews perché richiederebbe troppa RAM per lo storage.

Si potrebbe pensare di ridurre la dimensionalità oppure, alternativa certamente più efficace, provare a implementare una matrice "a dimensione fissa" grazie al word embedding.

Un problema comune che sorge quando si codificano le frasi riguarda il numero di embeddings, nonché la lunghezza del vettore di embeddings.

Nel nostro caso ogni termine è stato mappato in un vettore di 50 embeddings in quanto il criterio di scelta prevedeva che ci si basasse sulla lunghezza media e sulla deviazione[8] delle liste di tokens.

Il modello di codifica selezionato è il Word2Vec. Il motivo dietro questa scelta risiede nel fatto che l'algoritmo della libreria gensim permette di catturare le relazioni semantiche tra tokens considerando anche il contesto (nel nostro caso 5 parole).

Dato che ora ogni token è rappresentato da un vettore di 50 floats, diventa possibile calcolare le (dis)similarità tra qualunque coppia di parole.

Ad esempio, la cosine similarity tra la parola "mamma" e "moglie" è pari a:

```
word2vec.wv.similarity('mom', 'wife')
```

```
0.7123397
```

È possibile trovare anche le 5 parole più simili a un dato input.

```
word2vec.wv.most_similar("cat", topn=5)
```

```
[('cats', 0.8724837899208069),
 ('kitties', 0.8100258111953735),
 ('dog', 0.8014036417007446),
 ('kitty', 0.7848073840141296),
 ('kitten', 0.7522522211074829)]
```

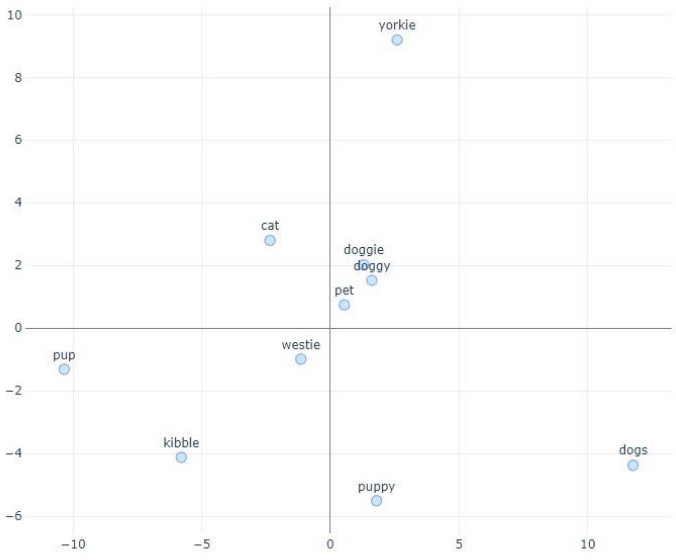
Avere parole codificate in array numerici consente di poterle rappresentare nell'embedding space. In realtà, allo stato corrente non è possibile rappresentarle dato che uno spazio a 50 dimensioni è poco pratico e sarebbe visivamente impossibile da interpretare.

[8] La media si aggira in un intorno di 40 con una deviazione standard più o meno regolare.

Si rende dunque necessario applicare una tecnica che riporti i vettori di 50 valori in vettori di al più 3 elementi. La tecnica implementata per ridurre la dimensionalità è stata la *principal component analysis*.

Dopodiché, una funzione *plot_embedding( )* è stata progettata ad hoc per permettere all'utente di inputare una parola arbitraria e poter scegliere sia il numero di dimensioni dello spazio in cui mappare sia il numero *n* di parole più simili[9] da rappresentare in esso.

L'immagine successiva raffigura nello spazio le 10 parole più simili a "dog".



4. Sentiment Analysis

Analisi del sentimento dietro a ogni recensione

L'analisi del sentimento è stata effettuata inizialmente tramite il modello VADER fornito da NLTK.

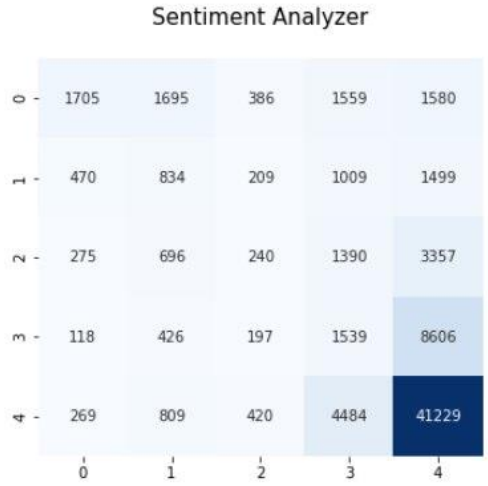
Dato che il modello in questione è molto sensibile alla punteggiatura si è deciso di non applicarlo sul testo normalizzato ma su quello originale.

Il valore compound restituito da *SentimentAnalyzer( )* viene impiegato per mappare la recensione nel sentimento secondo lo schema:

Compound	Score	Label
[-1, -0.70]	1	Molto negativa
(-0.60, -0.10]	2	Negativa
(-0.10, +0.10]	3	Neutrale
(+0.10, +0.70]	4	Positiva
(+0.70, +1]	5	Molto positiva

Confrontando manualmente alcune recensioni con l'etichetta così ottenuta è stato riscontrato che il modello è stato un po' troppo generoso sia con le recensioni positive sia con quelle negative.

Ciò è dovuto non solo al fatto che l'algoritmo VADER è molto semplice (mentre le recensioni sono ambigue), ma anche che è un modello progettato con focus sui testi prodotti nei social media. È nato, quindi, per enfatizzare le regole che catturano l'essenza del testo reperibile tipicamente sui social media, come le frasi brevi, le emoji, la ripetitività dei termini e l'uso abbondante della punteggiatura. Di seguito vengono riportati il report di classificazione[10] e la matrice di confusione[11] riguardo la performance conseguita dal classificatore di NLTK.



[9] Le parole simili dipendono molto dalla casualità con cui le recensioni sono state selezionate all'inizio.

[10] Per poter stendere il report di classificazione si deve considerare lo Score nel range originale [1, 5], non quello diminuito per compatibilità con keras.

[11] Sebbene le etichette siano riportate in [0, 4] nella figura, la matrice è equivalente al range [1, 5] shiftato di 1.

	precision	recall	f1-score	support
1	0.60	0.25	0.35	6925
2	0.19	0.21	0.20	4021
3	0.17	0.04	0.06	5958
4	0.15	0.14	0.15	10886
5	0.73	0.87	0.80	47211
accuracy			0.61	75001
macro avg	0.37	0.30	0.31	75001
weighted avg	0.56	0.61	0.57	75001

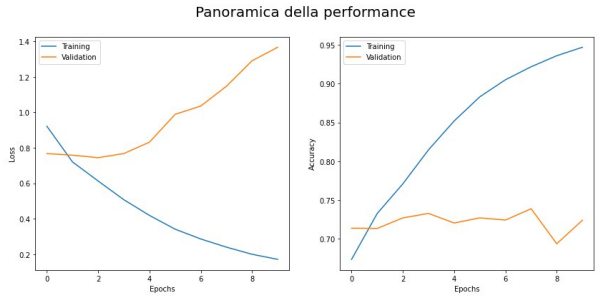
5. Classificazione con il machine learning

Machine learning per la previsione del sentimento

Si prova a risolvere i problemi del modello VADER impiegando una rete neurale. La rete è così composta:

- Uno strato iniziale per codificare le frasi in embeddings di 100 floats.
- Una convoluzione lineare[12] a cui segue uno strato di MaxPooling per ridurre la dimensione.
- Uno strato LSTM per accertare che anche la sequenza di parole abbia comunque un proprio grado di rilevanza.
- Nella rete si possono trovare anche strati di Dropout e parametri di regolarizzazione per disincentivare il modello a memorizzare i dati, piuttosto che ad apprenderne la struttura.

Il tutto è stato eseguito per 10 epoche e validato su un dataset pari al 30% del set di addestramento.



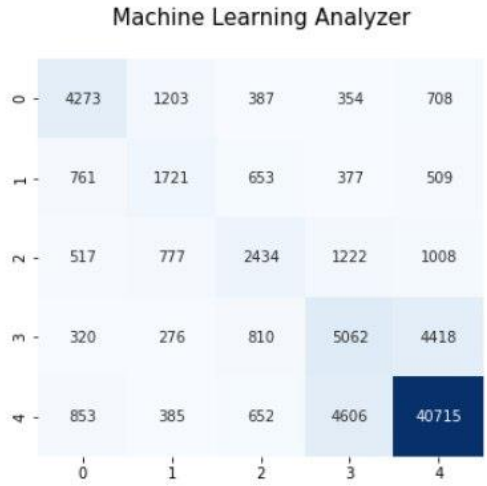
[12] L'utilizzo della convoluzione è incentivato dal fatto che nella sentiment analysis la sequenza non è importante tanto quanto il significato intrinseco della singola parola.

Dai grafici si può notare come il modello consegua comunque un overfitting non indifferente nonostante le numerose pratiche inserite per limitarlo.

Dietro le cause di questo comportamento potrebbero celarsi diversi fattori quali l'alta complessità delle recensioni, la presenza di noise non del tutto pulito e per ultimo la classificazione di un numero comunque significativo di classi.

Ciò nonostante, il modello resta in grado di classificare[13] le recensioni con una accuratezza pari a $73\% \pm 2\%$.

Si riportano il report di classificazione e la matrice di confusione.



	precision	recall	f1-score	support
1	0.64	0.62	0.63	6925
2	0.39	0.43	0.41	4021
3	0.49	0.41	0.45	5958
4	0.44	0.47	0.45	10886
5	0.86	0.86	0.86	47211
accuracy			0.72	75001
macro avg	0.56	0.56	0.56	75001
weighted avg	0.72	0.72	0.72	75001

Nella sua semplicità, il modello di machine learning ha conseguito una performance nettamente migliore (intorno al 10-15%) rispetto al modello VADER.

[13] A differenza di VADER, il training e il testing del modello sono stati eseguiti sulle frasi normalizzate affinché il disturbo dalla punteggiatura non alteri la performance.

Progetto di Text Mining

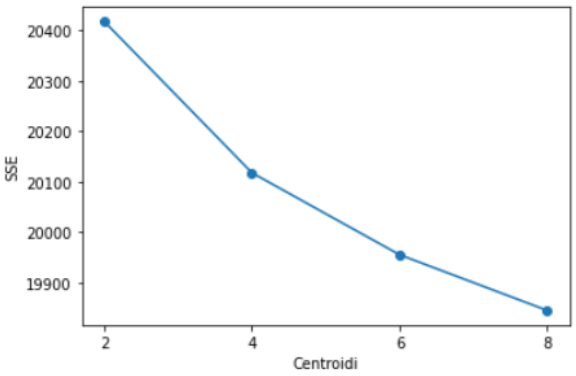
Davide Banfi – 806539
Alessandro Pigato – 852482

14 GENNAIO 2021

6. Text Clustering

Raggruppamento dei documenti per similarità

Nella terza sezione si effettua il clustering allo scopo di raggruppare le recensioni in base all'omogeneità. Si procede quindi a calcolare la matrice Tf-Idf per ogni documento grazie alla funzione *TfIdfVectorizer( )* del pacchetto sklearn.



Dato che il clustering è molto dispendioso in termini di risorse all'aumentare dei clusters *k* da formare, si sceglie di raggruppare le recensioni in 5 gruppi[14].

	Text	Score	Sentiment score	ML score	Cluster score
0	Fantastic little treats! Growing up on Oreos a...	4	5	4	4
1	morning thunder tea this is my favorite tea an...	5	4	5	5
2	Yummy Yummy Yummy!!! I originally received thi...	5	5	5	4
3	A must have on my carousel! I was not expectin...	5	5	5	4
4	worked for me This product has worked since th...	4	4	5	4

È lecito chiedersi se l'algoritmo KMeans abbia agito in base al sentimento delle recensioni, ma ciò sembra da escludersi in quanto il report di classificazione mostra un'accuratezza molto bassa ($13\% \pm 2\%$).

	precision	recall	f1-score	support
1	0.11	0.72	0.19	6925
2	0.04	0.08	0.05	4021
3	0.10	0.12	0.11	5958
4	0.17	0.15	0.16	10886
5	0.72	0.08	0.14	47211
accuracy			0.15	75001
macro avg	0.23	0.23	0.13	75001
weighted avg	0.50	0.15	0.14	75001

L'immagine sottostante raffigura 5000[15] documenti che sono stati posizionati nello spazio in base alla misura di similarità Tf-Idf. Le croci rosse identificano i centroidi e dal grafico è possibile individuare chiaramente una forma approssimativa degli insiemi.

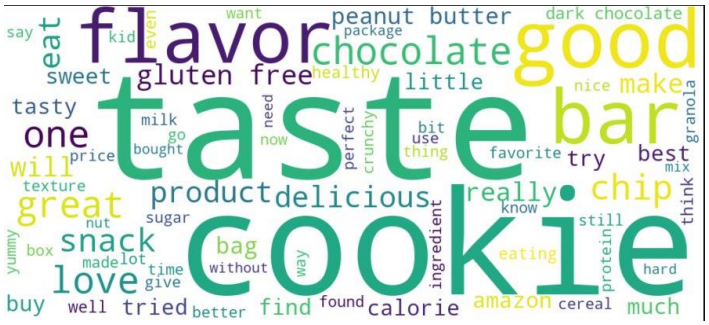
I clusters non sono esclusivi e questo aspetto è confermato anche dall'indice di Silhouette[16] con valori approssimabili a 0. In particolare, dal grafico si nota come tre clusters siano relativamente ben distinti mentre i due cluster nella parte alta condividano molti documenti.

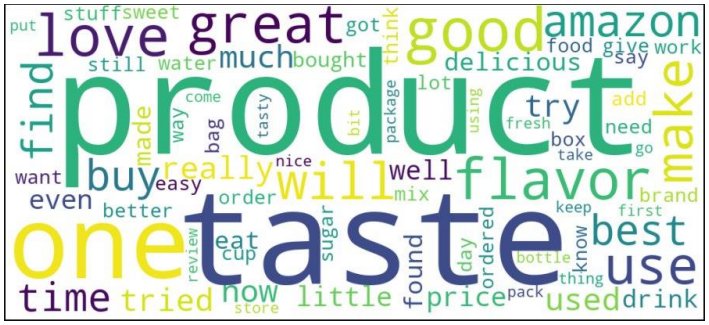
Questo è molto interessante perché se si considerasse il word cloud di ciascun cluster si nota come esistano tre nuvole molto diverse tra loro e altre due che invece sono molto simili.

[14] I cluster sono 5 come le classi della colonna Score, per cui successivamente si potrebbe verificare la qualità del clustering in base al sentimento.

[15] Purtroppo, per problemi di hardware si è stati costretti a rappresentarne solamente una parte.

[16] Valori vicini a 0 indicano clusters sovrapposti.





Sebbene alcune parole siano ricorrenti, ci sono altre parole che sono *cluster-specific*. Infatti, il primo word cloud sembra spaziare più nel campo semantico del pet food, il secondo nel campo semantico del caffè, il terzo in quello del tè, mentre per gli ultimi due è molto difficile indentificare la regola di distinzione.

7. Conclusioni

Riepilogo

Dopo aver pulito il più possibile le recensioni sono state impiegate tecniche di preprocessing che hanno fornito buoni risultati. Dopodiché è stata effettuata un'analisi del sentimento che ha evidenziato come la rete neurale performi nettamente meglio rispetto al modello VADER, sebbene questa non consegua una accuratezza accettabile per fini pratici. Il modello AI potrebbe essere ulteriormente migliorato bilanciando le classi, come già esposto nell'introduzione.

Probabilmente, anche avere documenti più puliti e senza troppi errori di ortografia contribuirebbe a migliorare le prestazioni non solo del modello di machine learning ma anche di tutte le altre tecniche applicate.

Per concludere, si può dire che la classificazione e il clustering restano comunque due strumenti utili alla comprensione del livello di (in)soddisfazione dei prodotti venduti da Amazon, e quindi sono validi per assistere le vendite e/o revisionare l'andamento del marketing.