Università degli studi di
Milano-Bicocca

Text Mining and Search

Project

20 Newsgroups Classification

Autori:
Antonella Zaccaria - 848647
Marco Savino - 793516

Sommario

Il dataset "20 newsgroup" è una raccolta di circa 20.000 documenti di notizie, suddivise in 20 diverse categorie. E' stato originariamente raccolto da Ken Lang per il suo paper "Newsweeder: Learning to filter netnews". Tale raccolta è diventata un dataset spesso utilizzato per applicazioni di *Machine Learning*, come *Text Classification* e *Text Clustering*. E' interessante dunque capire, a partire semplicemente dal testo, quale sia la categoria di appartenenza di una notizia.
Per fare ciò, dopo una fase iniziale di *preprocessing* del testo, il dataset è stato splittato in *train set* e *test set* e successivimente sono state create le strutture (*tf* e *tf-idf*) da utilizzare negli algoritmi di *Machine Learning*. Le metriche di valutazione sono state applicate sia al *train set*, mediante la *Cross Validation*, sia al *test set* in modo tale da valutare la bontà del modello finale.

1 Introduzione

La classificazione di documenti testuali è uno dei tipici compiti dell'apprendimento automatico supervisionato (*Supervised Machine Learning*). Il dataset "20 Newsgroups" è spesso utilizzato per lo svolgimento di questa tipologia di *task*. E' interessante dunque capire, a partire semplicemente dal testo, quale sia la categoria di appartenenza di una notizia.

1.1 Obiettivo

L'obiettivo del progetto è quello di classificare correttamente le notizie sulla base del loro contenuto, assegnando a ciascuna la corretta categoria di appartenenza e valutare che la classificazione delle *news* sia stata svolta correttamente.
E' stato dunque scelto, per il progetto, un *task* di *Text Classification*.

2 Dataset

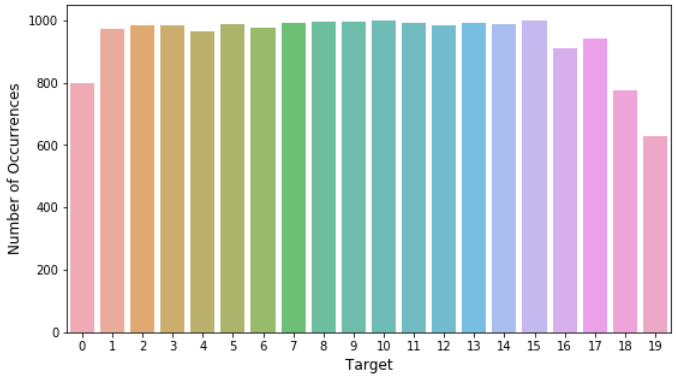
Il dataset "20 newsgroups", fornito direttamente dalla libreria *sklearn* di *Python*, è una raccolta di documenti suddivisi in 20 diverse categorie per lo più bilanciate:

- alt.atheism (0)

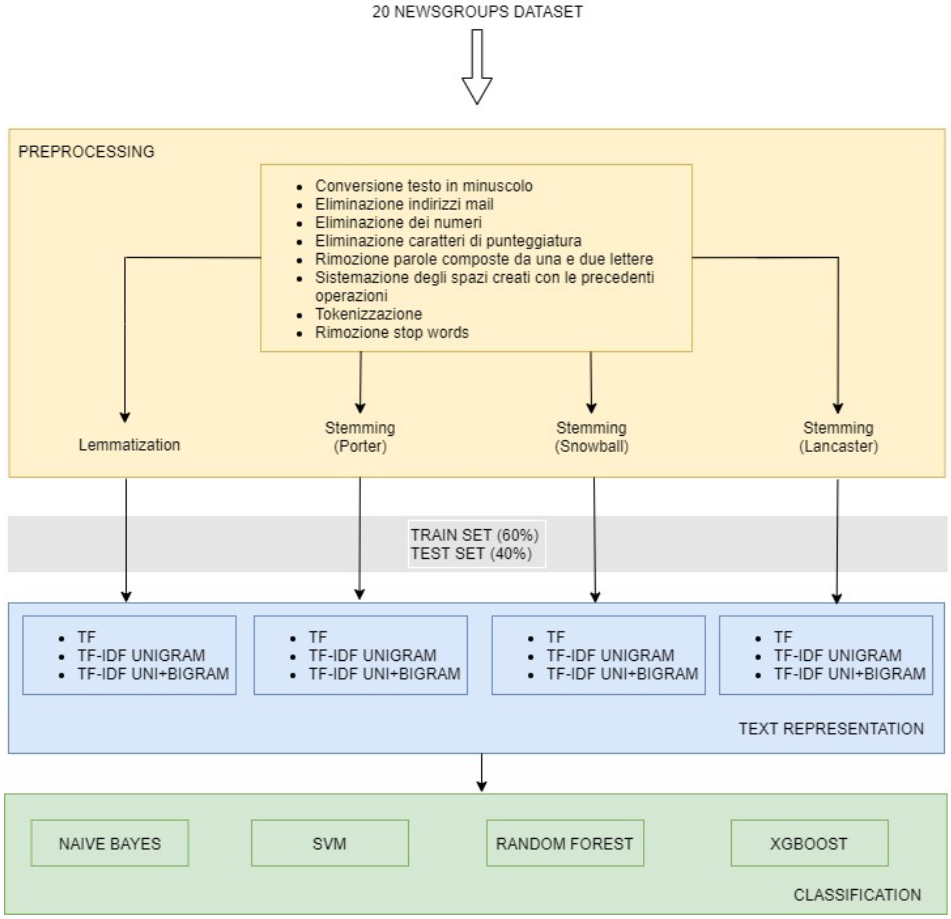
- comp.graphics (1)
- comp.os.ms-windows.misc (2)
- comp.sys.ibm.pc.hardware (3)
- comp.sys.mac.hardware (4)
- comp.windows.x (5)
- misc.forsale (6)
- rec.autos (7)
- rec.motorcycles (8)
- rec.sport.baseball (9)
- rec.sport.hockey (10)
- sci.crypt (11)
- sci.electronics (12)
- sci.med (13)
- sci.space (14)
- soc.religion.christian (15)
- talk.politics.guns (16)
- talk.politics.mideast (17)
- talk.politics.misc (18)
- talk.religion.misc (19)

Ogni record nel dataset è in realtà un file di testo in inglese, che presenta la seguente struttura: metadati - intestazione - testo del documento.

Il dataset è stato splittato in *training set* (60%) e *test set* (40%) per l'applicazione dei modelli di *Text Classification*.


Number of Occurrences
Target
0
200
400
600
800
1000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

3 Pipeline


20 NEWSGROUPS DATASET
PREPROCESSING
Conversione testo in minuscolo
Eliminazione indirizzi mail
Eliminazione dei numeri
Eliminazione caratteri di punteggiatura
Rimozione parole composte da una e due lettere
Sistemazione degli spazi creati con le precedenti operazioni
Tokenizzazione
Rimozione stop words
Lemmatization
Stemming (Porter)
Stemming (Snowball)
Stemming (Lancaster)
TRAIN SET (60%)
TEST SET (40%)
TF
TF-IDF UNIGRAM
TF-IDF UNI+BIGRAM
TEXT REPRESENTATION
NAIVE BAYES
SVM
RANDOM FOREST
XGBOOST
CLASSIFICATION

4 Text Preprocessing

La fase di *preprocessing* del testo è stata implementata servendosi principalmente della libreria *nltk* messa a disposizione da *Python*; sono state svolte le seguenti operazioni:

- conversione testo in minuscolo
- eliminazione indirizzi mail
- eliminazione dei numeri
- eliminazione caratteri di punteggiatura
- rimozione parole composte da una e due lettere
- sistemazione degli spazi creati con le precedenti operazioni
- tokenizzazione
- rimozione *stop words*
- *lemmatization*
- *stemming*: *Porter*, *Snowball* e *Lancaster*

5 Text Representation

La *Text Representation* è stata effettuata misurando i pesi associati a ciascun termine attraverso due euristiche:

- *Term Frequency* (TF): la *Term Frequency* $tf_{t,d}$ del termine t nel documento d è definita come il numero di volte che t si verifica in d
- *Term Frequency - Inverse Document Frequency* (TF-IDF): il peso tf-idf di un termine è il prodotto del suo peso tf e del suo peso idf.

La matrice tf-idf è stata creata per unigram e per unigram più bigram.

Il risultato di questa fase è la creazione di ventiquattro matrici (dodici per il *train set* e dodici per il *test set*), infatti le euristiche elencate sono state applicate separatamente ai dati lemmatizzati e stemmatizzati (Porter, Snowball e Lancaster).

6 Text Classification

Per la *Text Classification* sono stati testati quattro modelli su ogni rappresentazione precedentemente descritta, al fine di stabilire quale combinazione rappresentazione-modello fornisca la migliore classificazione. Nella valutazione dei risultati ottenuti è stata presa in considerazione l'*accuracy*, ossia la percentuale di classificazioni corrette.
L'accuratezza è stata valutata sia sul *train set*, attraverso la *Cross Validation*, sia sul *test set* (dati mai visti).

6.1 Multinomial Naive Bayes

Il *Multinomial Naive Bayes* è un classificatore *bayesiano* con un modello di probabilità sottostante che si basa sull'ipotesi di indipendenza delle *features*, ovvero assume che la presenza o l'assenza di una particolare *feature* in un documento testuale non sia correlata alla presenza o assenza di altre *features*.
Applicando il modello sulle rappresentazioni si ottengono i seguenti risultati:

	TRAIN (cross validation)		TEST	
	Accuracy	Time	Accuracy	Time
TF + LEM	0.86	383 ms	0.87	127 ms
TF + STEM (Porter)	0.85	351 ms	0.86	106 ms
TF + STEM (Snowball)	0.85	345 ms	0.86	113 ms
TF + STEM (Lancaster)	0.85	306 ms	0.85	111 ms
TF-IDF + LEM	0.86	252 ms	0.87	77.6 ms
TF-IDF + STEM (Porter)	0.85	215 ms	0.86	77.7 ms
TF-IDF + STEM (Snowball)	0.85	223 ms	0.86	77.3 ms
TF-IDF + STEM (Lancaster)	0.85	222 ms	0.86	76.6 ms
TF-IDF (bigram) + LEM	0.86	262 ms	0.87	101 ms
TF-IDF (bigram) + STEM (Porter)	0.86	267 ms	0.87	88.7 ms
TF-IDF (bigram) + STEM (Snowball)	0.86	261 ms	0.87	89.8 ms
TF-IDF (bigram) + STEM (Lancaster)	0.86	247 ms	0.86	88.2 ms

Il modello fitta bene su tutte le rappresentazioni.

6.2 Support Vector Machine

Il *Support Vector Machine* è un classificatore binario, ossia divide gli oggetti in due classi: ciò significa che determina se l'oggetto appartiene o meno alla classe. Per fare ciò, i valori sono mappati in un iperpiano; una funzione lineare è usata per determinare un confine che divide gli oggetti in due classi. Il limite si basa sulla distanza degli oggetti più vicini in entrambe le classi e tale distanza deve essere massimizzata. Gli oggetti più vicini sono chiamati "vettori di supporto" (*Support Vectors*).
Applicando il modello sulle rappresentazioni si ottengono i seguenti risultati:

	TRAIN (cross validation)		TEST	
	Accuracy	Time	Accuracy	Time
TF + LEM	0.75	9 min 10 s	0.78	3 min 29 s
TF + STEM (Porter)	0.75	8 min 52 s	0.78	3 min 28 s
TF + STEM (Snowball)	0.75	8 min 55 s	0.78	3 min 27 s
TF + STEM (Lancaster)	0.74	8 min 37 s	0.77	3 min 20 s
TF-IDF + LEM	0.89	12 min 40 s	0.90	4 min 35 s
TF-IDF + STEM (Porter)	0.89	12 min 40 s	0.90	4 min 34 s
TF-IDF + STEM (Snowball)	0.89	12 min 27 s	0.90	4 min 29 s
TF-IDF + STEM (Lancaster)	0.88	12 min 26 s	0.89	4 min 18 s
TF-IDF (bigram) + LEM	0.89	14 min 12 s	0.90	5 min 4 s
TF-IDF (bigram) + STEM (Porter)	0.89	14 min 34 s	0.90	5 min 10 s
TF-IDF (bigram) + STEM (Snowball)	0.89	14 min 5 s	0.90	4 min 59 s
TF-IDF (bigram) + STEM (Lancaster)	0.89	13 min 44 s	0.89	5 min

Il modello SVM fitta meglio sulle rappresentazioni TF-IDF.

6.3 Random Forest

Il *Random Forest* è un algoritmo di apprendimento supervisionato che calcola la media di più alberi decisionali in base a campioni casuali del dataset. Applicando tale modello alle rappresentazioni si ottengono i seguenti risultati:

	TRAIN (cross validation)		TEST	
	Accuracy	Time	Accuracy	Time
TF + LEM	0.83	2 min 46 s	0.84	32.6 s
TF + STEM (Porter)	0.82	2 min 25 s	0.84	30.6 s
TF + STEM (Snowball)	0.82	2 min 25 s	0.84	29.9 s
TF + STEM (Lancaster)	0.81	2 min 16 s	0.83	29.4 s
TF-IDF + LEM	0.82	1 min 31 s	0.83	23.8 s
TF-IDF + STEM (Porter)	0.81	1 min 33 s	0.83	23.7 s
TF-IDF + STEM (Snowball)	0.81	1 min 34 s	0.83	23.7 s
TF-IDF + STEM (Lancaster)	0.80	1 min 40 s	0.82	24.3 s
TF-IDF (bigram) + LEM	0.82	1 min 31 s	0.83	24.5 s
TF-IDF (bigram) + STEM (Porter)	0.81	1 min 31 s	0.83	24.8 s
TF-IDF (bigram) + STEM (Snowball)	0.82	1 min 31 s	0.83	24.4 s
TF-IDF (bigram) + STEM (Lancaster)	0.81	1 min 33 s	0.82	24.9 s

Il modello *Random Forest* fitta meglio sulla rappresentazione TF.

6.4 Xtreme Gradient Boosting

Gradient Boosting è una tecnica di *Machine Learning* per problemi di regressione e classificazione, che produce un modello di previsione sotto forma di un insieme di modelli previsionali deboli, tipicamente alberi decisionali. XGBoost è una delle implementazioni del concetto di *Gradient Boosting*, ma ciò che lo rende unico è che utilizza una formalizzazione del modello più regolarizzata per controllare l'*overfitting*, che conferisce prestazioni migliori. Applicando quest'ultimo modello sulle rappresentazioni si ottengono i seguenti risultati:

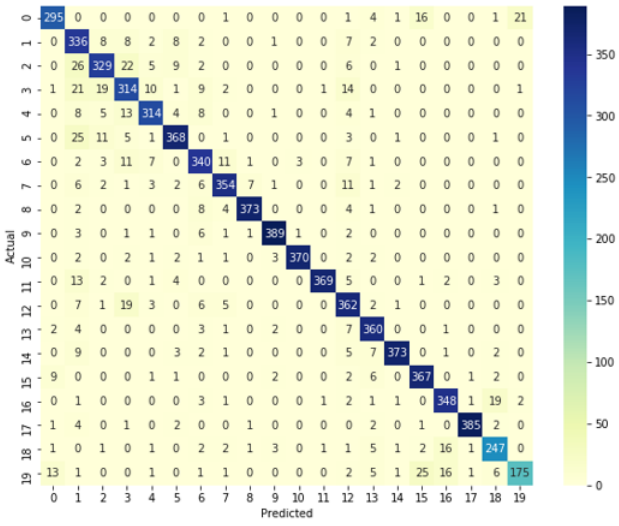
	TRAIN (cross validation)		TEST	
	Accuracy	Time	Accuracy	Time
TF + LEM	0.80	15 min 44 s	0.80	3 min 20 s
TF + STEM (Porter)	0.80	13 min 33 s	0.79	2 min 58 s
TF + STEM (Snowball)	0.79	13 min 24 s	0.79	2 min 56 s
TF + STEM (Lancaster)	0.78	12 min 10 s	0.79	2 min 40 s
TF-IDF + LEM	0.79	13 min 33 s	0.79	3 min 20 s
TF-IDF + STEM (Porter)	0.79	13 min 58 s	0.79	3 min 27 s
TF-IDF + STEM (Snowball)	0.79	14 min 14 s	0.79	3 min 30 s
TF-IDF + STEM (Lancaster)	0.78	14 min 25 s	0.78	3 min 32 s
TF-IDF (bigram) + LEM	0.80	15 min 32 s	0.80	3 min 51 s
TF-IDF (bigram) + STEM (Porter)	0.80	15 min 45 s	0.80	3 min 49 s
TF-IDF (bigram) + STEM (Snowball)	0.80	15 min 30 s	0.80	3 min 49 s
TF-IDF (bigram) + STEM (Lancaster)	0.79	15 min 42 s	0.79	3 min 53 s

Il modello XGBOOST fitta bene su tutte le rappresentazioni.

7 Conclusioni

Il modello di classificazione migliore, in termini di *recall*, *precision* ed *F1-score*, risulta essere *Support Vector Machine* nelle rappresentazioni tf-idf, di cui si prende come esempio di risultato la *confusion matrix* della rappresentazione tf-idf con unigram e bigram (*lemmatization*):

	precision	recall	f1-score	support
0	0.92	0.87	0.89	340
1	0.71	0.90	0.80	374
2	0.86	0.82	0.84	400
3	0.79	0.80	0.79	393
4	0.89	0.88	0.89	358
5	0.91	0.88	0.90	416
6	0.85	0.88	0.87	386
7	0.92	0.89	0.91	396
8	0.97	0.95	0.96	393
9	0.97	0.96	0.96	405
10	0.99	0.96	0.97	386
11	0.99	0.92	0.96	400
12	0.81	0.89	0.85	406
13	0.90	0.95	0.92	380
14	0.98	0.93	0.95	403
15	0.89	0.94	0.91	391
16	0.91	0.92	0.91	380
17	0.99	0.96	0.98	399
18	0.87	0.87	0.87	285
19	0.88	0.71	0.78	248
accuracy			0.90	7539
macro avg	0.90	0.89	0.90	7539
weighted avg	0.90	0.90	0.90	7539



Ci si aspetta dunque che, analizzando una qualsiasi *news*, essa venga classificata correttamente all'interno della categoria di appartenenza con un'elevata probabilità.