Natural Language Processing

In sentiment analysis, an accuracy of 80% is considered the to-be-expected value

Words are usually considered into groups of 2, 3 or even more words (bigrams, trigrams, n-grams) to account for the presence of senteces like "very funny" and "not very funny). Also, in a dataset there is a removal of stopwords (words without noticeable meaning)

If words are rapresented as one-hot vectors (every word is a vector of the size of the dictionary of words and only one bit is set to "1"), they are all equidistant. So we introduce **word vectors**, using the fascinating work of `word2vec`, a program that uses a neural network to predict relationship between words (example: "Athens is to Greece what Oslo is to...?" or "walking is to walked what swim is to...?). However, when using this model, we don't use one vector per word, but we use one vector per document which is the result of the average of the single word vectors. Unfortunately, this gives a worse performance than the one-hot encoded vectors.

recall: how many positives reviews detected divided by how many positive reviews are there.

recall: how many positives reviews correctly detected divided by how many positive reviews detected.

Ricordati che la precisione intorno ai 70% è ottima, in quanto ci sono tre bucket; se ce ne fossero solo due, allora dovresti avere 85%.

La stratiefied KFold cross validation è una tecnica che genera vari folds, ognungo con percentule più o meno uguale delle varie categorie di dato (cioè non c'è un fold con tutti commenti negativi).

La tecnica OneVsRestClassifier crea N classifier dove N è il numero di classi possibili dei dati. La tecnica OneVsOneClassifier crea N(N-1)/2 classifier, uno per ogni coppia di classi. Non si riscontrano differenze notevoli nella precisione del fit.

Bayes: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$.

Naive Bayes (si usano i log per evitare underflow e maggiore velocità) : $\hat{c} = \max_{c \in C} \left[logP(c) + \sum_i logP(w_i|c) \right]$

L'assunzione "naive" che si fa nel Naive Bayes è che tutti gli eventi w_i siano indipendenti, e quindi $P(w_0 \dots w_i|c) = P(w_0|c) \dots P(w_i|c)$. Inoltre, si assume che la posizione delle parole non conti; conta solo la loro frequenza!

La precision e la recall possono essere combinate con $F = \frac{(\beta^2+1)PR}{\beta^2 P+R}$. Se $\beta < 1$, si da più importanza al recall; altrimenti si da più importanza alla precision. Se $\beta = 1$, entrambe hanno uguale importanza: $F_1 = \frac{2PR}{P+R}$. Quello che F è in realtà è la media armonica di precision e recall.

I classificatori generativi creano un modello per ogni classe e vedono per un documento quale modello lo rappresenta meglio. I classificatori discriminativi vedono quali feature sono importanti per distinguere le varie classi direttamente dall'input.

I lexicons sono dei dizionari disponibili online con delle parole già classificate come positive o negative.

chi_squared test: $X^2 = \sum \frac{(obs-exp)^2}{exp}$. Un chi2 molto alto significa che l'ipotesi nulla è falsa. Nel nostro caso, l'ipotesi nulla è che una parola non è collegata con la frequenza con cui sta in una classe.

Algoritmi:

Linear multiple regression: cerco il minimo della residual sum of squares (RSS). Il gradiente è molto facile da calcolare, infatti RSS $= \sum_{i=1}^{N} (y_i - (mx_i + b))^2$, e la sua derivata è $\frac{\delta}{\delta m} = \sum_{i=1}^{N} -x_i(y_i - (mx_i + b))$

Stochastic gradient descent: Lo stochastic gradient descent è un normalissimo gradient descent solo che invece di usare tutti i training samples per aggiornare i parametri a ogni iterazione ne uso solo una parte (mini-batch). In pratica aggiorno i parametri a ogni mini-batch (che può essere composto anche da un solo sample)

Logistic Regression: Nella logistic regression uso la funzione sigmoide: $\sigma(x) = \frac{1}{1 + e^{-mx}}$, che mi restituisce valori compresi tra 0 e 1. Quindi faccio la cara vecchia regressione lineare sapendo che $\sigma'(x) = \sigma(x)(1 - \sigma(x))$. Ricordati che il risultato esprime una probabilità di appartenere a una certa classe (come le mail di spam), e tutti i label y sono o 0 o 1.

Activation Functions:

ReLU: $f(x) = max(0, x)$. Range: $(0, \infty)$. Approssimazione buona: *softplus*: $f(x) = log(1 + e^x)$, la cui derivata è la sigmoide $\frac{1}{1 + e^{-x}}$.

Logistic: $\sigma(x) = \frac{1}{1 + e^{-x}}$. Range: $(0, 1)$.
Derivata: $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

TanH: $f(x) = \frac{2}{1 + e^{-2x}} - 1$. Range: $(-1, 1)$. Derivata: $f(x) = 1 - f(x)^2$.

Softmax: $f(\underline{x}) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$ for $j \in \{1 \dots K\}$. Range: $(0, 1)$.

L'altro collaboratore si chiama Antonio Norelli, fa il quinto anno della magistrale, e fa il secondo anno di magistrale a fisica. E' uno studente della ssas. Il suo numero di matricola è 1612487. Il suo username su slack (e su youtube) è `noranta4`.

Appunti

Qui verranno listate varie prove fatte;

cv	Negative	Neutral	Positive	Media
3	.6706	.6536	.7271	.6837
10	.6886	.6638	.7487	.7003
20	.6851	.6666	.7475	.6997
40	.6883	.6683	.7494	.7020

Folds	Negative	Neutral	Positive	Media
2	.6540	.6391	.7154	.6695
3	.6689	.6517	.7183	.6797
4	.6744	.6591	.7334	.6890
5	.6782	.6552	.7386	.6907
6	.6782	.6591	.7395	.6923
7	.6805	.6522	.7414	.6914
8	.6864	.6649	.7443	.6985
9	.6812	.6608	.7401	.6940
10	.6854	.6611	.7410	.6958
11	.6863	.6587	.7406	.6952
12	.6867	.6644	.7540	.7017
13	.6865	.6654	.7508	.7009
14	.6865	.6657	.7472	.6998
15	.6844	.6661	.7449	.6985
16	.6912	.6683	.7467	.7021
17	.6833	.6649	.7477	.6986
18	.6896	.6673	.7506	.7025
19	.6876	.6663	.7428	.6989
20	.6908	.6675	.7500	.7028
21	.6894	.6708	.7435	.7012
22	.6847	.6681	.7489	.7006
23	.6865	.6673	.7462	.7000
24	.6837	.6665	.7495	.6999
25	.6883	.6665	.7504	.7017

C	NegativeP	NeutralP	PositiveP	MediaP	NegativeR	NeutralR	PositiveR	MediaR
.01	.6573	.6087	.7217	.6626	.5812	.7631	.6216	.6553
.02	.6623	.6225	.7313	.6720	.5953	.7728	.6281	.6654
.03	.6600	.6354	.7253	.6736	.6046	.7651	.6378	.6692
.04	.6726	.6396	.7249	.6790	.6127	.7639	.6475	.6747
.05	.6698	.6420	.7270	.6796	.6103	.7659	.6503	.6755
.06	.6597	.6499	.7335	.6810	.6257	.7615	.6443	.6771
.07	.6662	.6503	.7215	.6793	.6236	.7566	.6495	.6766
.08	.6692	.6570	.7344	.6869	.6358	.7631	.6511	.6833
.09	.6695	.6591	.7295	.6860	.6257	.7639	.6608	.6835
.1	.6636	.6574	.7242	.6817	.6188	.7679	.6511	.6793
.11	.6674	.6559	.7247	.6827	.6253	.7534	.6620	.6802
.12	.6673	.6530	.7308	.6837	.6265	.7534	.6624	.6808
.13	.6764	.6601	.7328	.6897	.6301	.7635	.6677	.6871
.14	.6675	.6524	.7270	.6823	.6337	.7558	.6479	.6791
.15	.6698	.6579	.7272	.6850	.6309	.7558	.6604	.6824
.16	.6736	.6617	.7358	.6904	.6301	.7667	.6665	.6878
.17	.6639	.6526	.7258	.6808	.6313	.7477	.6544	.6778
.18	.6754	.6594	.7377	.6909	.6350	.7603	.6689	.6880
.19	.6806	.6647	.7407	.6954	.6402	.7695	.6677	.6925
.2	.6760	.6528	.7261	.6850	.6293	.7550	.6624	.6822
.21	.6848	.6631	.7317	.6932	.6394	.7611	.6713	.6906
.22	.6745	.6588	.7288	.6874	.6269	.7587	.6689	.6848
.23	.6751	.6536	.7288	.6858	.6248	.7599	.6640	.6829
.24	.6626	.6630	.7318	.6858	.6265	.7590	.6657	.6837
.25	.6753	.6587	.7283	.6874	.6390	.7498	.6665	.6851
.26	.6675	.6570	.7309	.6851	.6245	.7570	.6661	.6825
.27	.6781	.6643	.7405	.6943	.6402	.7603	.6746	.6917
.28	.6772	.6661	.7266	.6900	.6366	.7497	.6774	.6879
.29	.6749	.6549	.7346	.6882	.6289	.7562	.6693	.6848

Beta	NegativeP	NeutralP	PositiveP	MediaP	NegativeR	NeutralR	PositiveR	MediaR
.5	.6623	.6611	.7292	.6842	.6394	.7437	.6628	.6820
.51	.6662	.6593	.7323	.6860	.6313	.7550	.6641	.6835
.52	.6682	.6578	.7324	.6862	.6422	.7461	.6620	.6835
.53	.6682	.6548	.7298	.6843	.6273	.7538	.6641	.6817
.54	.6612	.6528	.7191	.6777	.6265	.7469	.6527	.6754
.55	.6655	.6505	.7318	.6826	.6245	.7514	.6632	.6797
.56	.6626	.6536	.7315	.6826	.6325	.7473	.6596	.6798
.57	.6645	.6537	.7262	.6815	.6329	.7493	.6544	.6789
.58	.6724	.6595	.7275	.6865	.6329	.7591	.6592	.6837
.59	.6614	.6584	.7337	.6845	.6252	.7619	.6584	.6818
.6	.6768	.6582	.7314	.6888	.6277	.7615	.6693	.6862
.61	.6705	.6501	.7250	.6819	.6245	.7570	.6548	.6787
.62	.6654	.6589	.7245	.6830	.6341	.7522	.6560	.6808
.63	.6680	.6545	.7324	.6849	.6305	.7554	.6600	.6820
.64	.6631	.6545	.7357	.6844	.6220	.7619	.6604	.6814
.65	.6677	.6536	.7218	.6811	.6313	.7514	.6527	.6785
.66	.6668	.6573	.7351	.6864	.6293	.7627	.6564	.6828
.67	.6587	.6455	.7309	.6783	.6253	.7465	.6527	.6748
.68	.6692	.6577	.7320	.6863	.6321	.7599	.6584	.6835
.69	.6780	.6514	.7293	.6862	.6192	.7639	.6661	.6831
.7	.6684	.6529	.7299	.6837	.6261	.7582	.6584	.6809
.71	.6674	.6480	.7250	.6801	.6285	.7526	.6499	.6770
.72	.6670	.6466	.7266	.6801	.6204	.7566	.6535	.6769
.73	.6675	.6501	.7262	.6813	.6200	.7566	.6588	.6785
.74	.6672	.6482	.7231	.6795	.6200	.7566	.6519	.6762
.75	.6736	.6503	.7275	.6838	.6350	.7570	.6495	.6805
.76	.6658	.6495	.7291	.6814	.6196	.7566	.6596	.6786
.77	.6678	.6520	.7299	.6832	.6297	.7595	.6511	.6801
.78	.6733	.6522	.7299	.6851	.6192	.7611	.6661	.6821
.79	.6762	.6550	.7257	.6857	.6216	.7590	.6685	.6830
.8	.6740	.6560	.7353	.6884	.6337	.7667	.6540	.6848
.81	.6631	.6455	.7297	.6794	.6176	.7619	.6475	.6756
.82	.6721	.6502	.7318	.6847	.6228	.7655	.6552	.6812
.83	.6699	.6518	.7267	.6828	.6156	.7655	.6580	.6797
.84	.6724	.6534	.7270	.6843	.6321	.7570	.6552	.6814
.85	.6619	.6493	.7276	.6796	.6232	.7558	.6499	.6763
.86	.6683	.6434	.7275	.6798	.6119	.7586	.6576	.6760
.87	.6762	.6428	.7443	.6878	.6305	.7675	.6499	.6826
.88	.6779	.6479	.7276	.6844	.6180	.7683	.6564	.6809
.89	.6635	.6512	.7277	.6808	.6139	.7671	.6519	.6777
.9	.6814	.6582	.7297	.6898	.6325	.7700	.6580	.6868
.91	.6699	.6528	.7289	.6839	.6232	.7639	.6552	.6808
.92	.6741	.6510	.7202	.6818	.6204	.7594	.6572	.6790
.93	.6700	.6502	.7305	.6836	.6309	.7607	.6479	.6798
.94	.6714	.6466	.7324	.6835	.6249	.7594	.6548	.6797
.95	.6690	.6478	.7362	.6843	.6261	.7643	.6507	.6804
.96	.6712	.6445	.7255	.6804	.6151	.7582	.6576	.6770
.97	.6772	.6468	.7313	.6851	.6224	.7639	.6580	.6814
.98	.6731	.6431	.7333	.6831	.6184	.7667	.6503	.6785
.99	.6700	.6493	.7315	.6836	.6180	.7691	.6531	.6801

max n-grams	analyzer	binary	cv	average f1
4	char	true	3	.6869
6	char	true	3	.6990
8	char	true	3	.6991
8	char	true	12	.7104

max n-grams	analyzer	binary	cv	average f1
4	char	false	3	.6926
8	char	false	3	.7004
6	char	false	12	.7101

Prove fatte a mano:

Prendendo 100 commenti (quasi) a caso, classificandoli per conto mio ottengo 96/100 di correttezza.

Prendendo 50 commenti da quelli sbagliati, ottengo il 37/50 di correttezza.

Usando le letteree al posto delle parole, si arriva a guadagnare un punto e mezzo di percentuale di f1. (Tuttavia ancora da provare con binary a True)

Se si usano tanti n-gram, l'impatto del random è un pochino più basso.

Il Tfidf fa schifo, lascia perdere. Molto meglio avere le occorrenze binarie delle parole.

Idee

Giocare molto, **molto** con i possibili argomenti di countVectorizer. Ad esempio min_df e max_df potrebbero essere utili.
La cosa che ha detto Chierichetti: non si può avere un train data fatto bene; bisogna assegnare un punteggio ai commenti, non una categoria.
Devo proprio mettere shuffle=False in classifier_report()?
LE COSE NON SHUFFLATE FANNO CAGARE RISPETTO ALLE COSE SHUFFLATE.
Provare word vectors invece che bag-of-words
Provare Keras.

Cambiando parametri del machine learning cambia veramente poco. Molto di più il countVectorizer.
Il countVectorizer è molto lento; invece la logistic regression no. Perchè?
Qual è la parte dell'algortimo che usa multithreading e quale no?
Perchè fate un estimator fatto da voi invece che usare normalmente la logistic regression?
Domande:
I commenti che l'algoritmo trova sbagliati sono spesso e volentieri sbagliati. Perchè?
Spesso capita che le probabilità sono molto vicine tra loro e l'algoritmo sbaglia poco.
Le lettere in aggiunta potrebbero essere parole sbagliate.
Perchè i dati sono una sparse matrix?
Perchè la sparse matrix va scalata?
Non si potrebbero fare classificatori diversi per ogni sito?
Matrix correlation?? Glove dataset.
Perchè avete filtrato via gli hashtags?
Posso toccare la funzione build_dataset()? E Tagger?
Ho parlato con Flavio Chierichetti.
Ciao gabriele,
multilayer perception (rete neurale di base)
1 - Keras (tensorflow o teano) 2 - Videolezioni 3 - Ristruttura codice 4 - Scheda grafica (CUDA)
da testare selectKbest con chi2