Web Information Retrieval - Homework 3

Simone Conia, Marco Favorito, Shani Dana Guetta

June 17 2017

Contents

1 Spam Comments **1**
- 1.1 Dataset overview . . . 1
- 1.2 First tuning: a simple approach . . . 1
  - 1.2.1 Parameters . . . 1
  - 1.2.2 Training-Validation . . . 3
  - 1.2.3 Best configuration . . . 3
  - 1.2.4 Classification report . . . 5
  - 1.2.5 Confusion matrix . . . 5
  - 1.2.6 Normalized-Accuracy value . . . 5
  - 1.2.7 Matthews-Correlation-Coefficient value . . . 5
- 1.3 Second tuning: improving the classifier . . . 6
  - 1.3.1 Parameters . . . 6
  - 1.3.2 Best configuration . . . 6
  - 1.3.3 Classification report . . . 8
  - 1.3.4 Confusion matrix . . . 8
  - 1.3.5 Normalized-Accuracy value . . . 8
  - 1.3.6 Matthews-Correlation-Coefficient value . . . 8
- 1.4 General considerations and Conclusions . . . 9
  - 1.4.1 First tuning . . . 9
  - 1.4.2 Second tuning . . . 11

2 Opinions About Videos **15**
- 2.1 Dataset overview . . . 15
- 2.2 Vectorizer Parameters . . . 15
- 2.3 Training-Validation . . . 15
- 2.4 k-Nearest Neighbors . . . 16
  - 2.4.1 Classifier Parameters . . . 16
  - 2.4.2 Best configuration . . . 16
  - 2.4.3 Classification report . . . 18
  - 2.4.4 Confusion matrix . . . 18
  - 2.4.5 Normalized-Accuracy value . . . 18
  - 2.4.6 Matthews-Correlation-Coefficient value . . . 18

2.5 Multinomial Naïve Bayes 19
2.5.1 Classifier Parameters 19
2.5.2 Best configuration 19
2.5.3 Classification report 21
2.5.4 Confusion matrix 21
2.5.5 Normalized-Accuracy value 21
2.5.6 Matthews-Correlation-Coefficient value 21
2.6 Support Vector Machine 22
2.6.1 Classifier Parameters 22
2.6.2 Best configuration 22
2.6.3 Classification report 24
2.6.4 Confusion matrix 24
2.6.5 Normalized-Accuracy value 24
2.6.6 Matthews-Correlation-Coefficient value 24

1 Spam Comments

In this part of the homework it is requested to tune and train a **kNN** classifier for detecting spam comments associated to YouTube videos.

1.1 Dataset overview

	Training Set	Test Set
Spam	122	53
Ham	120	52
Total	**242**	**105**

1.2 First tuning: a simple approach

In this section we describe how we tuned the kNN classifier using the following pipeline:

- **TfidfVectorizer** (sklearn.feature_extraction.text.TfidfVectorizer)
- **KNeighborsClassifier** (sklearn.neighbors.KNeighborsClassifier)

1.2.1 Parameters

This section lists the parameters taken into account, and for each parameter, a brief description.

Vectorizer Parameters These are the parameters taken into consideration for the TfIdf Vectorizer.

Parameter	Description
analyzer	Whether the feature should be made of word (='word') or character n-grams (='char').
tokenizer	A callable for override the string tokenization.
ngram_range	$(\min_n, \max_n)$, where $\min_n$ and $\max_n$ are, respectively, the minimum and maximum number of consecutive token per feature.
stop_words	None, or a list of words that will be removed from the resulting tokens.
max_df	Threshold for document frequency in order to detect corpus-specific stop words. Can be expressed as absoulte count or as a proportion on the number of documents.
min_df	Cut-off, suited for filter out rare words in the corpus. Can be expressed as absoulte count or as a proportion on the number of documents.
binary	A boolean. If True, all non-zero term counts are set to 1.
use_idf	Enable inverse-document-frequency reweighting.
smooth_idf	Smooth idf weights by adding one to document frequencies, as if an extra document was seen containing every term in the collection exactly once. Prevents zero divisions.
sublinear_tf	Apply sublinear tf scaling, i.e. replace tf with $1 + \log(tf)$.
norm	Norm used to normalize term vectors, 'l1' or 'l2'. None for no normalization.

Classifier Parameters These are the parameters taken into consideration for the kNN classifier.

Parameter	Description
n_neighbors	Number of neighbors to use by default for k_neighbor queries.
weights	'uniform' or 'distance': with the former, all points in each neighborhood are weighted equally; with the latter, the points are weighted by the inverse of their distance.
p	Exponent to be applied to the minkowski distance. E.g. $p = 1$ for the Manhattan distance,

1.2.2 Training-Validation

In order to avoid overfitting, it is required to use a 10-Fold-Cross-Validation process in the training-validation phase. Also, due to the time-consuming nature of this task, it is required to use all available CPU cores.

This is quite simple if we use the GridSearchCV class provided by the sklearn package, as we only need to set the following parameters when initializing a GridSearchCV object:

Parameter	Value	Description
cv	10	Use 10-Fold-Cross-Validation
n_jobs	-1	Automatically detect the maximum number of logical CPU cores. Alternatively, assuming SMT[1], it could be set to $2 \times \#cpu\_cores$

1.2.3 Best configuration

This is the best configuration for the parameters taken into consideration.

Vectorizer This is the best configuration for the **TfIdf** Vectorizer.

Parameter	**Value**	**Range**
analyzer	'word'	[**'word'**, 'char']
tokenizer	Snowball Stemmer	[None, **Snowball Stemmer**, Porter Stemmer, Lancaster Stemmer]
ngram_range	(1,1)	[**(1,1)**, (1,2), (1,3), (1,4), (1,5), (2,2), (2,3), (2,4), (2,5), (3,3), (3,4), (3,5), (4,4), (4,5), (5,5)]
stop_words	nltk.corpus.stopwords	[None, **nltk.corpus.stopwords**]
max_df	0.3	[0.1, 0.2, **0.3**, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
min_df	14	[1, 2, 3, $\cdots$, **14**, $\cdots$, 19, 20]
binary	False	[True, **False**]
use_idf	True	[**True**, False]
smooth_idf	True	[**True**, False]
sublinear_tf	True	[**True**, False]
norm	'l1'	[None, **'l1'**, 'l2']

Classifier This is the best configuration for the **kNN** classifier.

Parameter	Value	Range
n_neighbors	7	[3, 4, 5, 6, **7**, 8, 9, 10, 11, 12, 13, 14]
weights	'uniform'	[**'uniform'**, 'distance']
p	2	[1, **2**]

1.2.4 Classification report

	Precision	Recall	f1-Score	Support
Ham	0.91	0.98	0.94	52
Spam	0.98	0.91	0.94	53
avg/total	0.95	0.94	0.94	105

1.2.5 Confusion matrix

	Predicted Spam	Predicted Ham
Spam	51	1
Ham	5	48

1.2.6 Normalized-Accuracy value

The Normalized-Accuracy value is **0.942857142857**.

1.2.7 Matthews-Correlation-Coefficient value

The Matthews-Correlation-Coefficient value is **0.888365750823**.

1.3 Second tuning: improving the classifier

In this section we describe how we improved the classifier by adding a dimensionality reduction model based on Singular Value Decomposition (SVD), i.e., we used the following pipeline:

- **TfidfVectorizer** (sklearn.feature_extraction.text.TfidfVectorizer)
- **TruncatedSVD** (sklearn.decomposition.TruncatedSVD)
- **KNeighborsClassifier** (sklearn.neighbors.KNeighborsClassifier)

1.3.1 Parameters

This section lists the parameters taken into account, and for each parameter, a brief description.

Vectorizer Parameters The same parameters as in Section 1.2.1 have been used.

Dimensionality Reducer Parameters These are the parameters taken into consideration for the SVD dimensionality reducer.

Parameter	Description
n_components	Desired dimensionality of output data.

Classifier Parameters The same parameters as in Section 1.2.1 have been used.

1.3.2 Best configuration

This is the best configuration for the parameters taken into consideration.

Vectorizer This is the best configuration for the **TfIdf** Vectorizer.

Parameter	Value	Range
analyzer	'char'	['word', **'char'**]
tokenizer	Snowball Stemmer	[None, **Snowball Stemmer**, Porter Stemmer, Lancaster Stemmer]
ngram_range	(1,6)	[(1,1), (1,2), (1,3), (1,4), (1,5), **(1,6)**, (1,7), (1,8)]
stop_words	nltk.corpus.stopwords	[None, **nltk.corpus.stopwords**]
max_df	1.0	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, **1.0**]
min_df	1	[**1**, 2, 3, 4, 5]
binary	False	[True, **False**]
use_idf	True	[**True**, False]
smooth_idf	True	[**True**, False]
sublinear_tf	True	[**True**, False]
norm	'l2'	[None, 'l1', **'l2'**]

Dimensionality Reducer This is the best configuration for the **SVD** dimensionality reducer.

Parameter	Value	Range
n_components	6	[1, $\cdots$, 6, $\cdots$, 10]

Classifier This is the best configuration for the **kNN** classifier.

Parameter	Value	Range
n_neighbors	3	[3, 4, 5, 6, **7**, 8, 9, 10]
weights	'distance'	['uniform', **'distance'**]
p	3	[1, 2, **3**]

1.3.3 Classification report

	Precision	Recall	f1-Score	Support
Ham	1.00	0.98	0.99	52
Spam	0.98	1.00	0.99	53
avg/total	0.99	0.99	0.99	105

1.3.4 Confusion matrix

	Predicted Spam	Predicted Ham
Spam	51	1
Ham	0	53

1.3.5 Normalized-Accuracy value

The Normalized-Accuracy value is **0.990476190476**.

1.3.6 Matthews-Correlation-Coefficient value

The Matthews-Correlation-Coefficient value is **0.981125291493**.

1.4 General considerations and Conclusions

In this section, we will discuss some results about the two tuning versions and the best configurations obtained.

1.4.1 First tuning

About the Vectorizer Section 1.2, the best representation of the input (i.e. sentences) is given by the `word` configuration of the vectorizer, which includes stemmed[2] unigrams, excluding stopwords. Moreover, it turned out that the optimized model has preferred a maximum document frequency (i.e. `max_df` less than or equal to 0.3 and a minimum document frequency (i.e.: `min_df`) greater than or equal to 14: this means that a generic term t_i has been stored in the learned vocabulary only if:

$$14 \leq |\{d : t_i \in d, d \in D\}| \leq |D| \times 0.3$$

.
Since `binary` is set to `False` and `use_idf` to `True`, we are in the bag-of-words assumption with document-frequency weighting. `smooth_idf` is set to `True`, which is a reasonable choice as it prevents division-by-zero. If it is set to `False`, during the cross-validation phase for a particular K-Fold iteration, it may give a zero score to the relative parameter combination, and this would strongly penalize that combination in the final evaluation. Also `sublinear_tf` is set to `True`, i.e. $1+\log(tf_{t,d})$, even though we are dealing with short sentences (and so not too high term-frequencies values or high variances in term-frequencies). It is worth to mention also the choice normalization metric of term vectors which turned out to be the L1 normalization, although usually L2 performs the best. Probably, this is due to the fact that the L1-norm gives equal penalty to all parameters, enforcing sparsity. In our experiments, the differences in accuracy is about 3% between L1 and L2 norm adoption.

We preferred not to use an external vocabulary: first of all, space dimensions would have grown a lot ($\sim$ 100k terms). Also, we would have to take care of Out-Of-Vocabulary words, e.g. "http", "facebook", "katy perry", variable-length words, misspelled words, and so on; indeed, some of these words resulted to be quite important in the classification task. In this case, as it often happens, let-the-data-talk-approach seemed to be a better choice.

About the Classifier We set the `algorithm` parameter set to `auto`, i.e. we let the classifier choose the best suited algorithm. Quite often the chosen algorithm is `brute`, which is ok since the number of samples is relatively small. The `uniform` neighbor weighting scheme, the exponent p of the Minkowski distance

[2]The various tested stemmers seem not to be so different to each other in terms of performance.

(in this case, an Euclidean metric, since $p = 2$) and the number of neighbors equal to 7 are strictly data-dependent parameters.
Here is the learned dictionary obtained using the best parameters:

thank	facebook	like
video	https	love
3	subscrib	pleas
check	www	guy
channel	http	amp
com	kati	song
perri		

The size of this dictionary is 19, which is a *very* small dictionary; this results directly from our narrow choice of allowed document-frequencies range. Notice that some words have been truncated by the stemmer.

The following are some examples of correctly-classified transformed sentences in their relative features. It can be done easily by the sklearn APIs `transform` and `inverse_transform`:

Original sentence	**Ham/Spam**	**Extracted features**
this is the best of the best video in world	0	['video']
free itunes gift card http shhort com a r x6j4gbrne	1	['http', 'com']
check out my covers i have a video coming out please subscribe'	1	['video', 'subscrib', 'pleas', 'check']
i love this song	0	['song', 'love']

From the table above, we can appreciate why it works. The presence of a website in the sentence string or a subscription request to other video channels or social network pages is well captured by, respectively, the words ["http", "com"] and words like ["subscrib", "pleas"], which are good features of a general spam comment. Instead, common words such as "video", "song" and "love" seem to characterize a non-spam comment, which is a reasonable choice (not always correct, obviously). Notice also the occurrence of the word "video" both in a spam and a ham comment, but in this case, it does not affect the correctness of the prediction.
An emblematic example of misclassified spam comment is "I love that you subscribed". Its transformation is ['subscrib', 'love'], which suggest us the weaknesses of this feature selection.

1.4.2 Second tuning

Differently from the first tuning, we applied another transformation in the pipeline: we employed SVD, a well-known dimensionality reduction technique. We chose a version suited for sparse data. The `sklearn` class that provides such implementation is `TruncatedSVD`. The `algorithm` parameter is set to `randomized` to use the randomized algorithm due to Halko (2009). In this case, we had to tune the number of components to be used to project our data in a better suited coordinate system, where data variances on each axis are maximized.

With respect to the first tuning, the best parameter configuration changes considerably. We will focus on each element of the pipeline.

About the Vectorizer This time, the search chosed a character-based vectorizer (i.e. `analyzer='char'`). This means that `tokenizer` and `stop_words` lose importance in terms of performance.

`ngram_range`, that this time refers to character instead of words, is larger than the previous case: $(1, 6)$, which means that all the sequences of characters with length from 1 up to 6 are taken into account. The parameter `min_df` is set to 1, quite different from 14, as in the previous case: in the current tuning, it seems worth to preserve every n-gram. The same for `max_df`, which is set to 1.0 (which is quite different from 1). As we will see, it makes the dimensionality very high, that will be effectively reduced by the dimensionality reducer. The norm is set reasonably to "l2", penalyzing the least important dimensions.

The parameters `binary`, `use_idf`, `smooth_idf` and `sublinear_tf` are set to the same values as before. The same arguments of the previous section hold, with the different meaning of "term": now it is not a word but a n-gram of characters.

About the Dimensionality Reducer The dimensionality reducer has been tuned only by taking into account the number of components the output should have, i.e. the number of "meta-features" that every tf-idf transformed instance has to have in the new coordinate system. In this new optimized coordinate system we will run the kNN algorithm. In this case, the optimization selected a number of principal components equal to 6. Moreover, the number of iteration in the randomized algorithm, configurable by `n_iter`, is set to a relatively high value (i.e. 50) in order to prevent variances of the performances.

About the Classifier Also for this element of the pipeline, the optimal configuration changes. The number of neighbor is set to 3 and, more importantly, the weighting scheme is `distance` (instead of `uniform`) and the exponent for the Minkowski metric is set to 3, which penalyzes more the lowest differences among vectors components.

The number of terms in the learned vocabulary, with the above described settings, is 42061. Some examples of ngrams are listed below:

”channe”	” n”	”ibe to”
”it m”	”ut my”	” subs”
”be pl”	” ch”	”t my c”
”hank”	”be me ”	”scri”
”y ch”	”e c”	”be a”
”plea”	”as”	”ubscr”
” to”	”scr”	” my”
”annel”	”ribe t”	”me pl”
”uys ”	”check ”	”nk”
”thank”	”uy”	”uys”
”out m”	”and ”	”hec”
”anne”	”e m”	”c”
”ibe ”	”se ”	”ibe an”
”bscrib”	”sub”	”hanne”

Actually, the above listed ngrams are ”special” ones: they are extracted from the highest values of the first component, let’s say the ”principal direction” of the data. This means that these ngrams are quite ”important” in the new coordinate system. This can be seen intuitively if we look at what they can mean: for example, some of them, like ”scri”, ”ubscr”, ”bscrib”, ”sub”, remember the ”subscribe to” expression which is very familiar in spam comments. The reader might have fun in finding other ”hidden words”, such as ”channel” or ”guys”.

The SVD operation allow us to reduce it effectively to a more reasonable number of 6. It is a well-known theoretical result that a PCA analysis performs a

sort of "pre-clustering" of the data, and helps a lot in the classification task, as the reader can check in the classification report in Section 1.3.3.

2 Opinions About Videos

In this part of the homework it is requested to tune 3 different classifiers to detect positive/negative comments about videos. In particular, the 3 classifiers are:

- **k-Nearest Neighbors**
- **Multinomial Naive Bayes**
- **Support Vector Machine**

2.1 Dataset overview

	Training Set	**Test Set**
Positive	308	308
Negative	249	250
Total	**557**	**558**

2.2 Vectorizer Parameters

All 3 classifiers have been trained using a TfIdfVectorizer. Please refer to Section 1.2.1 for a list of all the parameters taken into account.

2.3 Training-Validation

Please refer to Section 1.2.2 for a brief description on the training-validation process.

2.4 k-Nearest Neighbors

For this classifier we applied the same approach described in Section 1.

2.4.1 Classifier Parameters

These are the parameters taken into consideration for the kNN classifier.

Parameter	Description
n_neighbors	Number of neighbors to use by default for k_neighbor queries.
weights	'uniform' or 'distance': with the former, all points in each neighborhood are weighted equally; with the latter, the points are weighted by the inverse of their distance.
p	Exponent to be applied to the minkowski distance. E.g. $p = 1$ for the Manhattan distance,

2.4.2 Best configuration

This is the best configuration for the parameters taken into consideration.

Vectorizer This is the best configuration for the **TfIdf** Vectorizer.

Parameter	Value	Range
analyzer	'word'	[**'word'**, 'char']
tokenizer	None	[**None**, Snowball Stemmer, Porter Stemmer, Lancaster Stemmer]
ngram_range	(1,3)	[(1,1), (1,2), **(1,3)**, (1,4), (1,5), (2,2), (2,3), (2,4), (2,5), (3,3), (3,4), (3,5), (4,4), (4,5), (5,5)]
stop_words	nltk.corpus.stopwords	[None, **nltk.corpus.stopwords**]
max_df	0.2	[0.1, **0.2**, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
min_df	1	[**1**, 2, 3, $\cdots$, 10]
binary	True	[**True**, False]
use_idf	True	[**True**, False]
smooth_idf	False	[True, **False**]
sublinear_tf	True	[**True**, False]

Classifier This is the best configuration for the **kNN** classifier.

Parameter	Value	Range
n_neighbors	7	[3, 4, 5, 6, **7**, 8, 9, 10]
weights	'uniform'	[**'uniform'**, 'distance']
p	2	[1, **2**]

2.4.3 Classification report

	Precision	Recall	f1-Score	Support
Positive	0.91	0.95	0.93	308
Negative	0.93	0.88	0.91	250
avg/total	0.92	0.92	0.92	558

2.4.4 Confusion matrix

	Predicted Positive	Predicted Negative
Positive	292	16
Negative	30	220

2.4.5 Normalized-Accuracy value

The Normalized-Accuracy value is **0.917562724014**.

2.4.6 Matthews-Correlation-Coefficient value

The Matthews-Correlation-Coefficient value is **0.833525726001**.

2.5 Multinomial Naïve Bayes

2.5.1 Classifier Parameters

These are the parameters taken into consideration for the Multinomial Naïve Bayes classifier.

Parameter	Description
alpha	float: additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).
fit_prior	boolean: whether to learn class prior probabilities or not. If false, a uniform prior will be used.

2.5.2 Best configuration

This is the best configuration for the parameters taken into consideration.

Vectorizer This is the best configuration for the **TfIdf** Vectorizer.

Parameter	Value	Range
analyzer	'word'	['**word'**, 'char']
tokenizer	None	[**None**, Snowball Stemmer]
ngram_range	(1,2)	[(1,1), **(1,2)**, (1,3), (1,4), (1,5)]
stop_words	nltk.corpus.stopwords	[None, **nltk.corpus.stopwords**]
max_df	0.175	[0.1, 0.15, **0.175**, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
min_df	1	[**1**, 2, 3, $\cdots$, 10]
binary	False	[True, **False**]
use_idf	True	[**True**, False]
smooth_idf	True	[**True**, False]
sublinear_tf	False	[True, **False**]
norm	'l1'	[None, **'l1'**, 'l2']

Classifier This is the best configuration for the **MultinomialNB** classifier.

Parameter	Value	Range
alpha	0.78	[0.7, 0.75, **0.78**, 0.79, 0.8, 0.9, 1.0]
fit_prior	False	[True, **False**]

2.5.3 Classification report

	Precision	Recall	f1-Score	Support
Positive	0.95	0.98	0.96	308
Negative	0.97	0.94	0.95	250
avg/total	0.96	0.96	0.96	558

2.5.4 Confusion matrix

	Predicted Positive	Predicted Negative
Positive	301	7
Negative	16	234

2.5.5 Normalized-Accuracy value

The Normalized-Accuracy value is **0.958781362007**.

2.5.6 Matthews-Correlation-Coefficient value

The Matthews-Correlation-Coefficient value is **0.916869864917**.

2.6 Support Vector Machine

This time, we used an SVM model for implement the classifier. More precisely, we used `sklearn.svm.LinearSVC`.

2.6.1 Classifier Parameters

Parameter	Description
C	Penalty parameter C of the error term.
loss	Specifies the loss function. 'hinge' is the standard SVM loss (used e.g. by the SVC class) while 'squared_hinge' is the square of the hinge loss.

2.6.2 Best configuration

This is the best configuration for the parameters taken into consideration.

Vectorizer This is the best configuration for the **TfIdf** Vectorizer.

Parameter	Value	Range
analyzer	'word'	[**'word'**, 'char']
tokenizer	Snowball Stemmer	[None, **Snowball Stemmer**, Porter Stemmer, Lancaster Stemmer]
ngram_range	(1,2)	[(1,1), **(1,2)**, (1,3), (1,4), (1,5)]
stop_words	nltk.corpus.stopwords	[None, **nltk.corpus.stopwords**]
max_df	1.0	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, **1.0**]
min_df	2	[1, **2**, 3, 4, 5]
binary	False	[True, **False**]
use_idf	True	[**True**, False]
smooth_idf	True	[**True**, False]
sublinear_tf	True	[**True**, False]

Classifier This is the best configuration for the **LinearSVC** classifier.

Parameter	Value	Range
C	0.7	[0.125, 0.5, **0.7**, 1, 2, 4, 8]
loss	'squared_h	[**'squared_hinge'**, 'hinge']

2.6.3 Classification report

	Precision	Recall	f1-Score	Support
Positive	0.97	0.99	0.98	308
Negative	0.98	0.97	0.98	250
avg/total	0.98	0.98	0.98	558

2.6.4 Confusion matrix

	Predicted Positive	Predicted Negative
Positive	304	4
Negative	8	242

2.6.5 Normalized-Accuracy value

The Normalized-Accuracy value is **0.978494623656**.

2.6.6 Matthews-Correlation-Coefficient value

The Matthews-Correlation-Coefficient value is **0.956554655035**.