CS54701 Homework3

Peng Li

April 18, 2016

1 Text categorization

1.1 For one-of classification, one document just belong to one we do calculation for each category when we evaluate the result of a classifier. In that case, if one document belong to class A but classifier assigned it to class B, this document is false positive for class B and false negative for class A. And this is the same for each wrongly assigned document. So, the total number of false positive decisions equals the total number of false negative decisions.

1.2 We can use the function of F1 measure to calculate the F1 value by precision and recall.

$$F_1 = \frac{2 \times p \times r}{p + r}$$

And we also know how to calculate p(precision) and r(recall)

$$p = \frac{tp}{tp + fp}$$

$$r = \frac{tp}{tp + fn}$$

So, we can get the function of F1 represented by tp(true positive), tn(true negative), fp(false positive) and fn(false negative) easily.

$$F_1 = \frac{2 \times tp}{2 \times tp + fp + fn}$$

So, p = r = F1. And the accuracy is showed below.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

Because of one document just belong one class, we can conclude that if one document is true positive for one class, it must be true negative for other classes.

2.1 For example, there is a word "Agoraphobia" which is rare and has no information about class "Computer Science". However, in our train set, there are several computer science documents and these documents use "Agoraphobia" as an example to explain something. So, classifier will assigns an document to class "Computer Science" if it contains the rare word "Agoraphobia" although this is wrong. So, if a document doesn't contain these rare word, classifier will assign it more precisely.

What's more, when we choose features we should remove stop words from our document. If we build our classifier based on the stop-word-contained training set, we will get a low efficient classifier and test documents will be assign to strange class. If we remove these stop words, our result will be better.

2.2 I think k-NN will give more weight on the presence word because k-NN algorithm uses unprocessed training set directly and use TF-IDF to weighting for documents representation. That means K-NN will use the information of word if a document contains this word. So, k-NN will weight more on presence word.

3 As we all know that there is a trade off between small number of neighbors(k) and large number of neighbors. And we can get higher micro-f1 value when we set k around 10. The situation is there is a special data set we can get best result when k is 20 and we didn't do a cross validation. When we set k around 10 we can't get a good result but we can get a better result if we set k equal 20. All in all, if we choose a inappropriate k value, the classifier will has a low efficient. If a appropriate k value is chose, classifier will be better.

4 This method is effective because when we do a text categorization we will look all document and it is possible for a classifier to put a spam email into a innocuous class because most of the content are innocuous paragraphs. What's more, some classifier will weight more on "special" zone such as first paragraph, last paragraph and the paragraph which contains most "common word" with subject. These spam email may put innocuous content in these field to avoid to be assigned into the spam class.

I think our classifier should assigns a document into several classes. Although a spam email maybe has a higher possibility for belonging to a innocuous class, if top-5 or top-10 contains spam class, we can claim that maybe this is a spam email and we should check it.

2 Text Clustering

1 The propose of text clustering is we want to make related documents get together. When we use vector space model, we will use some methods to measure the similarity between two documents, such as cosine similarity. For example,

there are two documents, doc A has word "automobile", and doc B use "car" and never has "automobile". Their similarity is high because most of their content are about same the topic. So, they will in the same cluster although one uses "automobile" and another uses "car".

2 First, we should create a matrix to store these documents. The word list is (hot, chocolate, cocoa, beans, ghana, africa, harvest, butter, truffles, sweet, sugar, cane, brazil, beet, cake, icing, black, forest)

Doc ID																		
1	1	1	1	1														
2			1		1	1												
3				1	1		1											
4			1					1										
5								1	1									
6		1								1								
7										1	1							
8											1	1	1					
9											1	1		1				
10															1	1		
11															1		1	1

Because of k equal 2, I want to use document 3 and document 8 for the seeds and use point1 and point2 denote the center of them respectively. I use cosine similarity to represent the distance between different document.

Step 1:
cosine(doc1, doc3) = 0.2886751
cosine(doc2, doc3) = 0.3333333
cosine(doc7, doc8) = 0.4082483
cosine(doc9, doc8) = 0.6666667
other cosine similarity are 0, so we assign them randomly. Cluster1 contains doc1, doc2, doc3, doc4, doc5
center1 is (0.2 0.2 0.6 0.4 0.4 0.2 0.2 0.4 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Cluster2 contains doc6, doc7, doc8, doc9, doc10, doc11
center2 is (0.0000000 0.1666667 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.5000000 0.5000000 0.3333333 0.1666667 0.1666667 0.3333333 0.1666667 0.1666667 0.1666667)

Step 2:
cosine(center1, a1) = 0.6864065
cosine(center1, a2) = 0.6793662
cosine(center1, a3) = 0.5661385
cosine(center1, a4) = 0.6933752
cosine(center1, a5) = 0.4160251
cosine(center1, a6) = 0.138675

cosine(center2, a1) = 0.08838835
cosine(center2, a6) = 0.5
cosine(center2, a7) = 0.75
cosine(center2, a8) = 0.6123724
cosine(center2, a9) = 0.6123724
cosine(center2, a10) = 0.6123724
cosine(center2, a11) = 0.4082483
So, Cluster1 contains doc1, doc2, doc3, doc4, doc5
center1 is (0.2 0.2 0.6 0.4 0.4 0.2 0.2 0.4 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)
Cluster2 contains doc6, doc7, doc8, doc9, doc10, doc11
center2 is (0.0000000 0.1666667 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.5000000 0.5000000 0.3333333 0.1666667 0.1666667 0.3333333 0.1666667 0.1666667 0.1666667).
The result doesn't change and totally two iteration.