Supervised Information Retrieval

PROBLEM NO. 6
proposed by Neometrics/Accenture

VI UCM Modelling Week, Madrid June 11-15 2012

Instructor:
José Miguel García-Santesmases

Students:
Inmaculada Flores García
Ángel Manuel González Rueda
Javier Palacios Bermejo
Juan Sampedro Ruiz
Patricia Vicente Merino

CONTENTS

1. PROBLEM DESCRIPTION

Our topic, Supervision Retrieval, is developed in parallel with the Text Mining field. In particular, the problem we dealt with was Sentiment Analysis. This can be formulated as:

"Given several documents extracted from the Internet concerning certain brand or product, would it be possible to know the author's opinion expressed without actually reading them?"

In order to answer to this question, we were given a couple of data sets:

- Document set: consisting of the list of the documents downloaded from some web page in addition to their polarity target assigned by someone called "an expert". This target takes values in the set {positive, negative, neutral} meaning the sentiment expressed by the author according to this expert.
- Document_Term set: gathering the previous documents together with the frequency of the terms or words which appear on each of them, and an information importance measure, the tf-idf.

Our aim was to build at least one model capable of classifying new documents according to their polarity. We would like to categorize them as positive, negative or neutral in an automatic way.

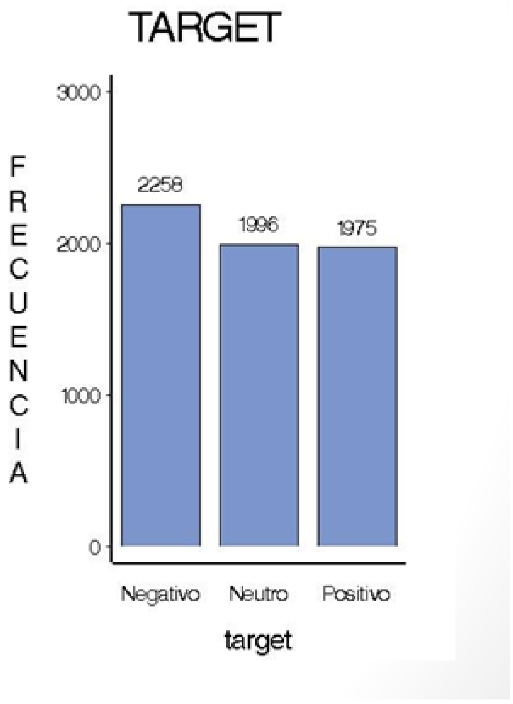
The purpose of this study is included into what is called Reputational Risk Analysis. It takes into account the probability of creating a negative public opinion on a service, offered by banks, telecommunication, food, assurance companies… The most frequent causes are credibility loss and the fact of not being present. Consequently there is a migration either of funds or clients to competitors companies on the market.

2. MATHEMATICAL TREATMENT

Once we have described the problem being treated, the information we count on and the objective of our work, it is the moment to start with the mathematics. We all agreed in following the SEMMA methodology (Sample, Explore, Manipulate, Model and Assess) which is quite popular in Data Mining.

Despite the order showed above, we decided to start with the data exploration due to the fact that we didn't really know which kind of information and what dimensions of data we were going to work with. The result at this phase of the study we obtained was:

- Huge size of the data sets, 6229 documents and 32638 terms which will play the role of our variables.
- A uniform polarity distribution. We can see on the following histogram that there is a balanced quantity of documents on each sentiment class:



We were indeed surprised by this graph in the sense that it is not usual to get this similar proportion of opinions about any product, brand or sector. After talking to the people from Neometrics we were said this set of documents have actually been manipulated before.

After the exploration we were then able to carry out the data sampling. We divided the original *document set* into two subsets: the *training sample* taking the 70% of the documents which will allow us to train the model; and the *validation sample*, gathering the 30% left, which will give us an idea of the precision we can expect. It is important to say that we have used the same validation and training data set in all our models.

Next step was the manipulation of the data. As we have mentioned earlier, we were dealing with a big amount of data, and it is well-known in the data mining field that this circumstance is fairly tricky. It is necessary to develop a deep analysis of the importance of all the data we were given and try to get to a level of information load suitable to work with. The decision we took was to reduce the dimension of this data. In first place, we consider filtering terms following two different, and compatible, criteria we will explain on the next chapter. After this, we also made use of the principal components algorithms reaching another reduction of variables. In parallel, we did also explore a couple of methods with no dimension reduction requirements.

Not having started with the modelization we introduce here the models we have suggested and tested:

- Frequency Model
- Frequency model with term filter
- KNN- k Nearest Neighbors
- SVM-Supported Vector Machine
- Neural Network
- Document-to-document similarity

The first two models were created by us from the beginning. We developed a very intuitive and simple (in terms of hypothesis) model based on the count of word occurrences in documents of each class. We use the filter on the second one.

The next three models listed above were considered taking advantage of methods already implemented in software available. Every of them will be explained on next chapters.

Finally, the last model was an alternative approach, attempting to measure the distance among two unrelated documents by using a multi-hop path in a shortest path fashion.

3. FREQUENCY MODEL

The frequency model is based on the creation of a dictionary for each term as a function of their frequencies. These frequencies are calculated from the target of every document: positive, negative or neutral.

After crossing both initial tables, and eliminating the variable text, we get a target linked to each word of every document. This word may have associated some classifications.

After use a "proc freq" we get the total events of every class, the total number, and, after a change of variable, we create proportions as number events word in a class over the total occurs. We introduce these proportions in the model.

Every word is classified as the class that counts with the highest proportions.

Repeating inversely the same process with the documents, each one will be associated with the claim with highest number of words of that specific class.

Then, we crossed the validation table with "Document-term", we repeat the analysis with our dictionary and the result is 74% of success, the best of all result, against a 26% misclassified.

4. FILTERED FREQUENCY MODEL

We need to delete some terms, noise, using a filter. First of all, we eliminate the words that appear only once or twice in the set of documents, then, we eliminated the words that because of their characteristic cannot be well classified following two conditions:

- At least one class of the term must have a proportion of 40%, to avoid proportions of 33% in every class for a word.
- For the model learn to distinguish between positive and negative, we eliminated terms with close appearance in positive and negative.

After creating the filter and passed the frequency model, we have a worst result, 71% well classified, but a lower cost in computational time.

5. K-Nearest Neighbors

Once we have develop a model of our own, we would also be interested in trying out some of the methods SAS has to offer to see if we can improve the already good results we have obtained.

The first idea that comes to mind is to try the well-known K-Nearest Neighbors algorithm -from now on KNN.

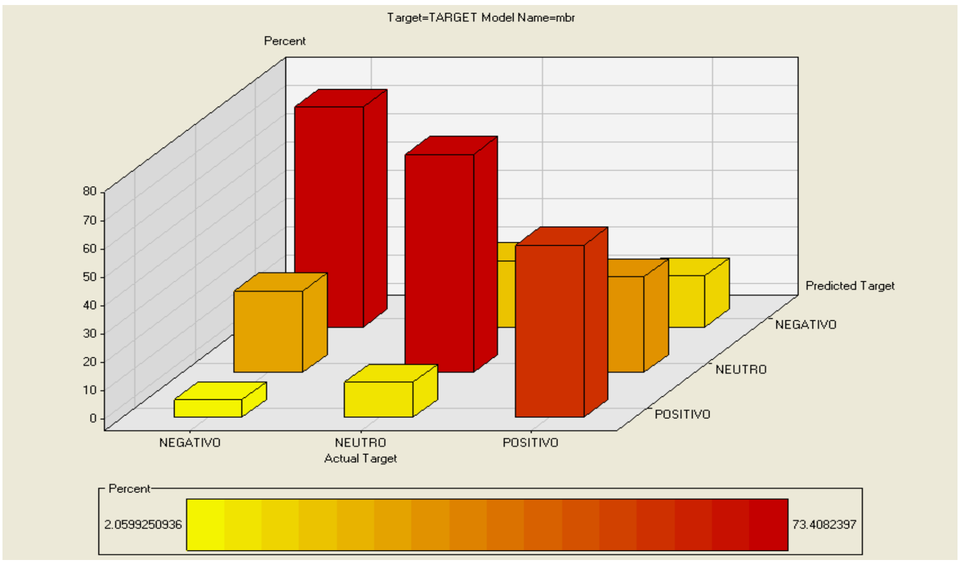
KNN is a method for classifying objects based on closest training examples. The algorithm classifies each object by a majority vote of its neighbors. Each document will then be assigned to the class most common among its k nearest neighbors.

Since we will be working with the euclidean distance and because of the size of the data we are dealing with, we would first like to reduce the dimension of the problem while at the same time make the point cloud round. We may achieve this by applying principal components decomposition, but first we need to get rid of some of the terms, so that the decomposition does not take too long. We therefore decide to work with the filtered data set from the frequency model.

After computing the principal components we take the 200 best of them and train a KNN model with the SAS Enterprise Miner node 'Memory-Based Reasoning'.

One of the heuristics to choose the value of the parameter k suggests setting k to the square root of the number of documents -in this case: 66.

The next graphic shows the confusion matrix of the KNN method trained by SAS with k equals 66. We have obtained a 32.27% of misclassification error in the validation sample. We may consider this result poor, since we already have methods that provide us with better results.



6. Neural Network

Another method we may consider using is a neural network model.

Neural Networks are often criticized because they do not offer a clear interpretation of the results they provide. However, this is the kind of problem where we are not as concerned for the model to be easy to understand as we would usually be. In this sort of problem, we would be willing to sacrifice interpretation in order to get a result as good as possible.

Nevertheless, the results do not seem so promising. We may use the filtered data set to train a neural network model with one hidden layer consisting of three neurons (the default neural network SAS Enterprise Miner offers). The misclassification error reaches the 34% and the result is once again very poor.

We leave to the reader the task of training some other neural networks to see if the results improve.

7. SVM

Another tool that we have used to classify the documents was support vector machines (SVM) which is a useful technique for data classification.

Given a training set of document-term pairs (x_i, y_i), $i = 1, \dots, l$ where $x_i \in R^n$ and $y_i \in \{-1,1\}$ that represent two category attribute for the documents, SVM requires the solution of the following optimization problem:

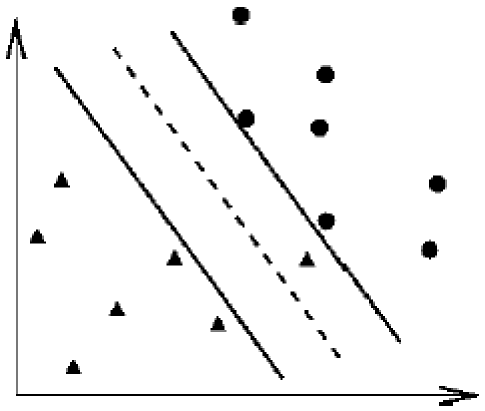
$$\begin{aligned} \text{Min} \quad & \tfrac{1}{2} w^t w + C \textstyle\sum_{i=1}^{l} \xi_i \\ \text{Subject to} \quad & y_i(w^t \emptyset(x_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

where $C > 0$ is the penalty parameter of the error term and ξ_i is the loss function, usually taken one of the forms below:

$$\xi_i = \max(1 - y_i w^t x_i, 0)$$

$$\xi_i = \max(1 - y_i w^t x_i, 0)^2$$

Using this method, the training vector x_i are mapped into a higher dimensional space by the function $\emptyset$, and SVM finds a separating hyperplane with the maximal margin in this higher dimensional space.



Furthermore, K(x_i, x_j)=$\emptyset(x_i)^T\emptyset(x_j)$ is called the kernel function. Some examples of basic kernels are:

- Linear: the simplest kernel that one can use.

$$K(x_i, x_j) = x_i^T x_j.$$

- RBF (radial basic function) kernel: is a reasonable first choice. This kernel nonlinearly maps samples into a higher dimensional space so it, unlike the linear kernel, can handle the case when the relation between class documents and terms is nonlinear.

$$K(x_i, x_j) = \exp(\gamma||x_i - x_j||^2)\ , \gamma > 0$$

where γ is a kernel parameter.

When the number of terms is large, one may not need to map data to a higher dimensional space. That is, the nonlinear mapping does not improve the performance, and using the linear kernel is good enough (see references).

Linear classification has become one of the most promising learning techniques for large sparse data with a huge number of documents and terms, and from the available software we have chosen to use liblinear, which is an open source library that implement SVM with linear kernel, written on C++ and implementing a Python interface.

The first step to use liblinear library is to write our data into a test file using its specific format. Each document must fill a single line with the following format, using only numeric values:

Target of the document   index:value … index:value

As we can see, first it must appear the target of the document, and we will denote positive documents by 1, neutral documents by 0 and negative documents by -1. Next we need to write the values from the document-term matrix into an sparse format, with the index of the terms that appears in the instance and the associated document-term matrix values. There are three possibilities to choose these values

- Term counts: this value indicates the frequency of the words in each document. We didn´t use it in our study.
- TF-IDF (*term frequency–inverse document frequency).*
- Term presence (binary) that indicates whether a term appears in a document.

We have three document classes and the linear kernel only allows binary classifications, so we need some schema to perform the classification. We propose three schemas, one for simple binary classification, and two for the ternary classification we are interested in:

1. Class reassign: reassign neutral documents either to positive or negative classes.
2. Two-phase classification:
   - First grouping neutral and positives.
   - Second reclassified the extended positives using a proper subset of the initial training dataset.
3. Three binary-class prediction built within liblinear.

Now we are going to show the results that we have obtain using liblinear library from Python with the previous schemas.

Class reassign:

As document-term matrix elements we have tried with tf-idf and binary, with raw (meaning unmodified values) and normalized values, and the best classification was achieved with raw binary values. The data transformation and initial tests with the library did show us that there were four documents in our dataset with no term, which were deleted from the corpus.

We obtained 85% success grouping neutral and negative, and 87% grouping neutral and positives, with confusion matrix for the validation set below. We present predicted values by rows vs. real values by columns.

Predicted	Real target	
	Neutral+Neg.	Pos.
Neutral+Neg.	63.65 %	5.51%
Pos.	9.05 %	21.79%

Precision Pos. : 87.56 %

Recall: 79.80 %

Predicted	Real target	
	Neg.	Neutral+Pos.
Neg.	28.48 %	6.64 %
Neutral+Pos.	6.64 %	58.24%

Precision Neg. : 89.77 %

Recall: 81.10 %

Three categories classification:

When comparing the two schemas for ternary classification, we see that the overall precision for liblinear building implementation is 77%, a bit higher than the 74% achieved with two-phase classification. This improvement comes for greatest precision for the class of negative document but two phases model is better for positive and neutral documents, giving the best results if we just consider the worst case classification. Below we show the confusion matrix for the validation set of the two methods:

Liblinear

Predicted	Real target		
	Neg.	Neutral	Pos.
Neg.	30.51 %	5.51 %	2.89 %
Neutral	3.00 %	23.18 %	4.66 %
Pos.	1.61 %	5.35 %	23.29 %

Precision Neg : 86.89 %

Precision Neutral: 68.08 %

Precision Pos : 75.52 %

Two-Phase

Predicted	Real target		
	Neg.	Neutral	Pos.
Neg.	28.48 %	4.44 %	2.19 %
Neutral	3.85 %	24.04 %	4.98 %
Pos.	2.78 %	5.57 %	23.66 %

Precision Neg: 81.10 %

Precision Neutral : 70.60 %

Precision Pos: 76.74 %

8. DOCUMENT-DOCUMENT DISTANCE

The obvious way to know the polarity of a document is to search similar documents and assign their polarity to the document under study. This requires a way to determine when two documents are close one to another, which in fact means to define a distance in our corpus, which in fact is an underlying issue in many of the available methods. For this purpose we will consider only the presence of terms within a document, initially discarding the number of occurrences and the tf-idf value. The distance will be derived from the number of terms shared by both documents, relative to the number of terms that appear on them, so we define it as

$$d_{ij}{}^2 = \frac{all\ terms}{common\ terms}$$

Defining distance in this manner gives a taste about the upper bound of the real distance, particularly in the case of two documents with no common terms, where the definition above leads to an undetermined (∞) distance. This restricted definition could serve for classification purposes alone, but as we are actually dealing with bounds, we tried to refine a bit, using network shortest path algorithm to give a better determination of the distance among documents with just a few terms in common causing an overestimated distance, but maybe connected one to another by a chain of close documents.

Although initial implementation of the method pointed to a non-connected corpus, this was quite an artifact due to a programming bug, partly caused by the existence of six documents with no neighbors. Revised implementations allowed us to define a distance for every pair of documents. So, we should start with a very sparse distance matrix, and after multiple shortest-path iterations end up producing a completely dense distance matrix, that should be suitable for many classification algorithms. In order to test the power of the method outlined here, we decide to use a K-neighbors method, easily implemented also in term of shortest path algorithm.

The first quality measurement was just an average of the number of neighbors with the right polarity for our validation dataset, and showed a not very promising 43% average, which is actually not too far from the $^1/_3$ that we could expect from pure randomness. A bit more work with the results allowed us to construct the confusion matrix, which confirms the very poor results, giving a global accuracy around 57%. The only hope for this method is that despite of its low accuracy, the classification achieved is close to the optimum, because if we inspect the confusion matrix for the auto-classification of the training dataset, the total accuracy is just 58%, so really close to the values achieved for the validation dataset, as the fraction of neighbors with right polarity did show in the preliminary results.

Training auto-classification

	Real target		
Predicted	Neg.	Neutral	Pos.
Neg.	21.58 %	7.19 %	4.62 %
Neutral	9.33 %	16.52 %	7.81 %
Pos.	5.81 %	7.42 %	19.70 %

Validation dataset

	Real target		
Predicted	Neg.	Neutral	Pos.
Neg.	20.80 %	7.74 %	4.57 %
Neutral	9.13 %	18.16 %	8.49 %
Pos.	5.16 %	8.11 %	17.84 %

Some results not showed here lead to the conclusion that the use of shortest path algorithms does not noticeably affect the method accuracy. Although some of the evaluated distances get reduce by the algorithm iterations, these reductions do rarely occurs among the nearest documents. Although the iterations do still allow building dense distance matrixes with no missing values that could be used with more sophisticated classification methods, the low performance achieved by these initial implementations did prevent us to actually get a fully populated matrix.

9. DIFFICULTIES

The main difficulties that arise during the elaboration of this work were originated by technical problems related to the amount of data and the available computer and software resources. The local installation of our main tool (SAS) did lack on procedures to handle sparse data, and some of the procedures were just not capable to manage the large matrixes that we get forced to use as input.

But the main difficulty for this problem, that we didn't actually hit due to the limited time, was related to the fact that we are processing human language, which is very difficult to effectively summarize with the three polarity levels that we are using, and is also quite sensitive to factors very hard to move into numeric values, as could be the presence of sarcasm. A self-explanatory example of this problem is easily picked up from documents misclassified as positive while being clearly negative for every human reader.

"Sanguijuelas Dios te ama. Nike te anima a hacerlo. Metro está mejorando para ti. En #BANCO#, lo que cuenta eres tu. El Corte Inglés es especialista en ti. Tu novio te quiere más que a nadie. La Comunidad de Madrid te escucha. En Bodybell son cómplices de tu belleza. Facebook te ayuda a comunicarte y compartir. Cuántas sanguijuelas para esa única necesidad que corre por tus venas: la necesidad de sentirte importante. Más importante que el resto. Publicado por Timita"

10. CONCLUSIONS& FURTHER WORK

Although there were much more available methods than those that we were able to explore, we did find that our very first attempt was quite competitive even when compared with models that in principle are much more elaborated. Moreover, even the simple filtering performed on the dataset did decrease a bit the performance of the classification, despite the fact that it was designed to clear out some of the noise.

Only the use of the more elaborated support vector machines technique did produce an important increase on the classification accuracy, although the increase was much less spectacular when the classification was attempted against the full ternary target, because SVM is designed for binary classification and the modifications that we did attempt had a strong impact in the overall accuracy.

Given our initial results, the most promising path to improve our classification ability is a deeper exploration of the SVM techniques, maybe with non-linear kernels. Other possibility that we started to work in was the construction of new terms, as for example grouping terms that usually appear together. And, if we are able to compensate the weaknesses of some methods by applying others, probably a combination of multiple methods is the best path for classification improvements.

REFERENCES

"A Practical Guide to Support Vector Classification", Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin.

"Introducción a la Minería de Datos", Hernández Orallo, J., Ramírez Quintana, M.J.,Ferri Ramírez, C.

"Data Mining, Concepts and Techniques", Han J., Kamber M.

*http://en.wikipedia.org/wiki/tf*idf*

Native python implementation of shortest-path Dijkstra's algorithm (*http://code.activestate.com/recipes/119466/*)

LIBLINEAR -- A Library for Large Linear Classification (http:// *www.csie.ntu.edu.tw/~cjlin/liblinear/*)