FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Department of Informatics Engineering

Master in Informatics Engineering

Pattern Recognition Techniques

2013/2014

Event Matching System(EMS)

Celso Rafael Clara Mendes
celsom@student.dei.uc.pt
2009109378

Igor Nelson Garrido Cruz
igorcruz@student.dei.uc.pt
2009111924

Contents

1 Introduction

In the present work we pretend to implement and analyze an Event Matching System (EMS).

With the introduction of new Information and Communication technologies and the proliferation of the internet access and social media, "big data" databases are everyday more common. However it is hard to extract meaning from these databases for intelligent usage, in other words it is hard to attain semantic information from the "big data". In this work we implement an EMT system that could operate in such systems in order to verify if two events are the same or not.

For that we used a real dataset consisting of 2090 pairs of events from the south-Asian country of Singapore labeled as matches/non-matches. For each event you are given the following characteristics: event identifier, event title, venue, start/end dates and times, address, latitude, longitude, categories and description.

We are going to study diverse methods of feature selection, different classifiers (Bayes, SVM, Fisher, k-NN), data sampling techniques and evaluation metrics to analyze machine learning technologies applies to EMS.

2 Pre-Processing and Feature Extraction

The event's data is not directly usable for Pattern Recognition as it is available in the dataset. The dataset is made of text, which are not directly usable by classifiers. In order to overcome this issue we used a python script which extract numerical data from the raw text.

Distance between characters - We implemented a count of occurrences of characters in each string. And then calculated the distance between each count doing the sum of the absolute values of the subtraction of the occurrences. This measure is then divided by the maximum sum of characters between both strings, approximating to 0 if both strings tend to be equal or 1 if the characters are completely different. If both strings are empty we manually set this distance to be zero.

Levenshtein's Distance - The Levenshtein distance between two words is the minimum number of single-character edits (insertion, deletion, substitution) required to change one word into the other. This distance was calculated with the Fuzzycomp Python's library.

Jaccard's Distance - The Jaccard similarity coefficient is a statistic used for comparing the similarity and diversity of the sample sets, in this case two strings. The Jaccard coefficient measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. This distance was calculated with the Fuzzycomp Python's library.

Jaro's characters - The Jaro distance metric is designed and best suited for short strings such as person names. The higher the Jaro distance for two strings is, the more similar the strings are (0 - no similarity, 1 - same string). This distance was calculated with the Fuzzycomp Python's library.

Title's Features:

All the punctuation was removed from the strings.
Multiple white-spaces were reduced to a single one.
Distance between characters;
Levenshtein Distance;
Jaccard Distance;
Jaro Distance.

Venue's Features:

All the punctuation was removed from the strings.
Multiple white-spaces were reduced to a single one.
If some venue is empty we use the character '-' to represent it.
Distance between characters;
Levenshtein Distance;
Jaccard Distance;
Jaro Distance.

Starting Time's Features:

All the punctuation was removed from the strings.
Times were converted to integers.
Subtraction between starting time of events was extracted.

Distance's Features:

To extract distances between events we used the euclidean distance.

Categories Features:

To extract features from the categories we used regular expressions to capture the text inside parenthesis and then we removed punctuation and invalid text.

After that we create a list of categories of each instance and we calculate the minimum Levenshtein's distance between all of the categories of each event.

Description Features:

To extract features from the description we remove punctuation, and we make a list of words.

Then we check how many of the words are common to both of the descriptions and divide by the maximum number of words, giving us a relative measure of much of the description is common to both events.

3 Feature Selection

We used multiple methods to implement feature selection. In this chapter we will present our studies and a make some comments on our thoughts.

Kruskall-Wallis

The Kruskall-Wallis assesses whether c independent samples are from the same population (or from populations with continuous distribution) and from the same median for the variable being tested. In order to choose the most discrimative features we should choose the ones with a higher variance given by Chi-Square.

We tested the features given by the default extraction and the features after applying Principal Component Analisys (PCA). Looking at the values, the raw data seems to be more discriminative.

Raw Data			PCA	
Feature Index	**Chi-sq**		**Feature Index**	**Chi-sq**
2	290.1813		3	283.1228
4	264.0813		7	193.9149
1	222.9254		12	64.1303
3	167.2643		9	54.353
12	14.6448		11	47.6021
6	7.0865		2	19.1699
5	2.2562		10	11.6461
10	0.6073		4	6.0131
9	0.1567		8	2.6159
8	0.1152		1	2.5375
7	0.0581		6	0.9131
11	0.0239		5	0.3355

Figure 1: Kruskall-Wallis test results for raw data and for features after PCA projection.

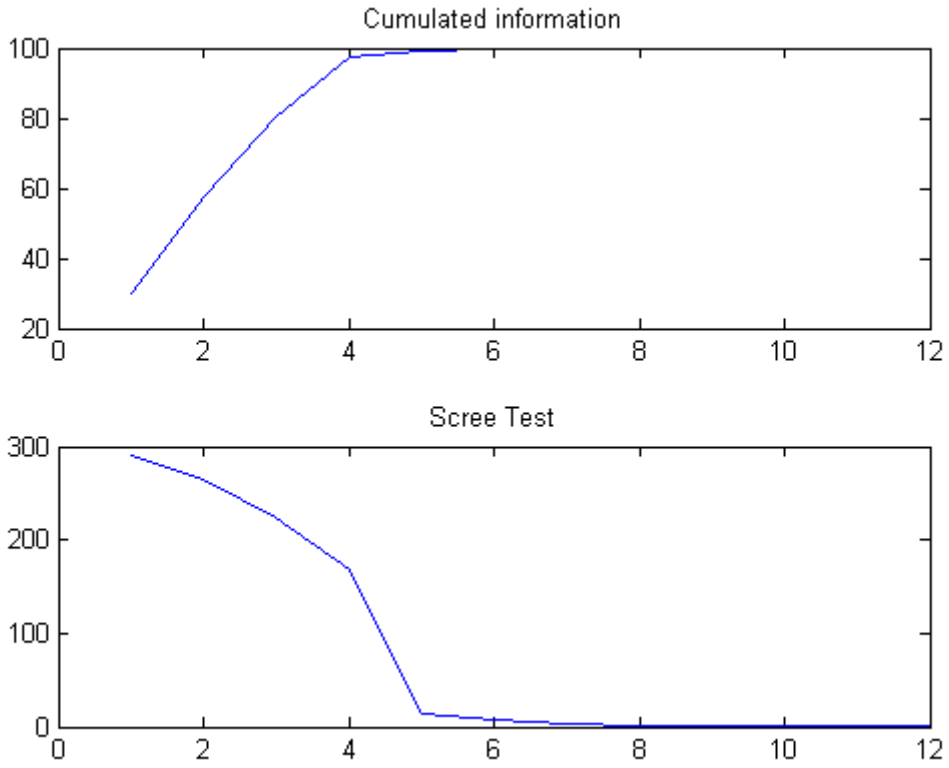

Figure 2: Feature tests for raw data

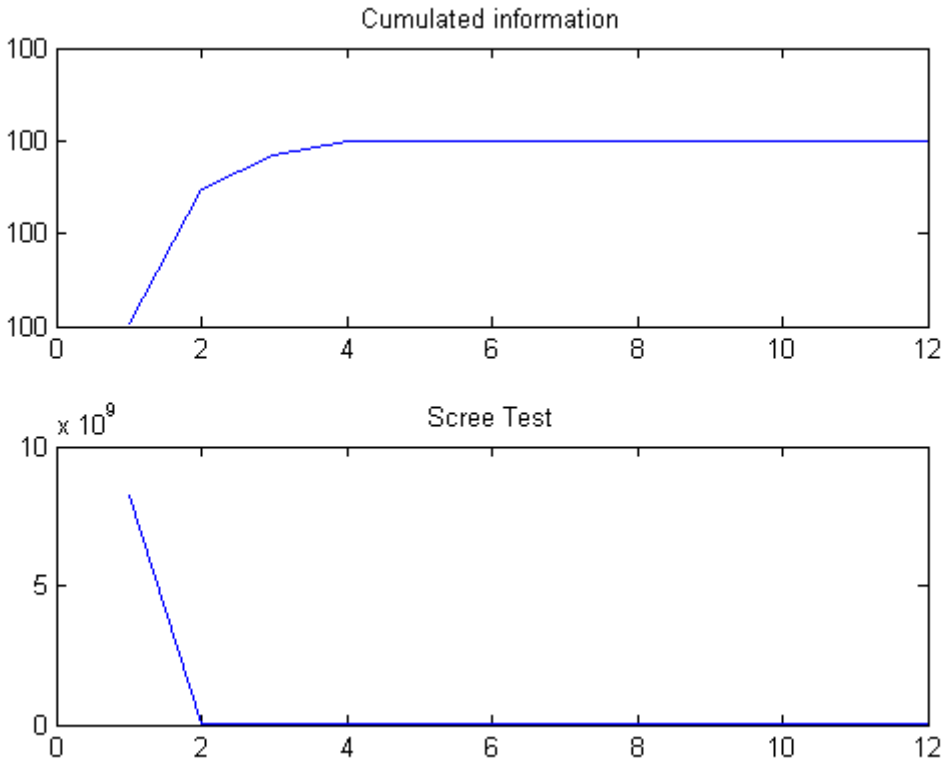

Figure 3: Feature tests for PCA data

As we can see in the upper plots ,using Kaiser Test, Scree Tests and Visual Threshold, without PCA the number of features that we should use to build the classifier should be around 5, since with less that that we lose too much information and with more than that we have too much dimensionality for the classifier to learn and generalize. With PCA the optimal number tends to be a little lower, decaying to 3 or 4 features.

4 Data Sampling

For data sampling we implemented a hold-out of 80% of the data for training and 20% data for testing. This method allows us to predict the behaviour of the classifier in an independent dataset.

5 Classifiers

The classifiers implemented for this work were the Bayes, K-NN, Fisher and SVM.

5.1 Bayes

Bayes classifiers are probabilistic classifiers based on the Bayes' theorem.

We used the Maximal Likelihood estimation of Gaussian mixture model for implementing the Bayes classifier. This classifier minimizes the Bayesian risk of loss. The input vectors X are classified into classes with the highest *a posterior* probability computed from given model.

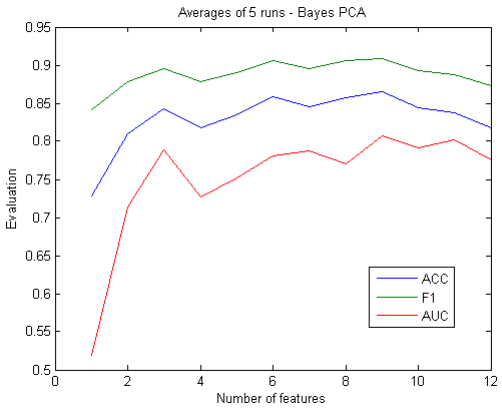

Figure 4: Bayes PCA

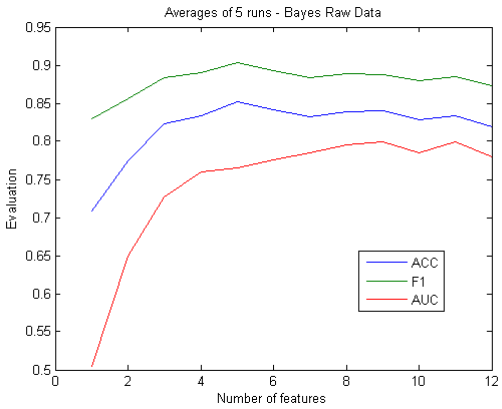

Figure 5: Bayes PCA RAW data

We made 5 runs of the Bayes classifier. We also tested all the amount of features that we extracted. The best results for raw data occurred for 5 features, giving us F1 slightly higher than 0,9 and accuracy around 0,85.

As we can see, using less than 5 features was not as good because the information present was not enough to correctly predict so many features. Adding more features is also not good because the classifier loses the capacity to generalize knowledge.

In the PCA the results were slightly lower than with the raw data. However the almost as accurate as the ones achieved with the raw data classifier.

5.2 Fisher

Fisher's linear discriminant is a classification method that projects high-dimensional data onto a line and performs classification in this one-dimensional space. The projection maximizes the distance between the means of the two classes while minimizing the variance within each class.

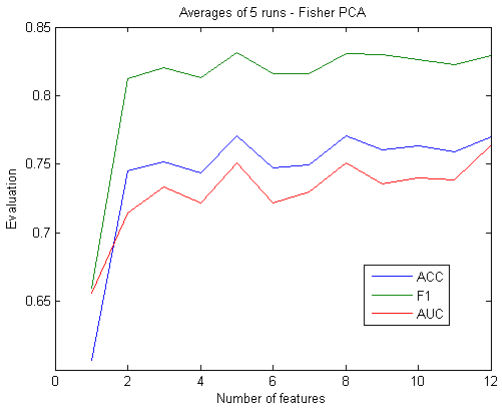

Figure 6: Fisher PCA

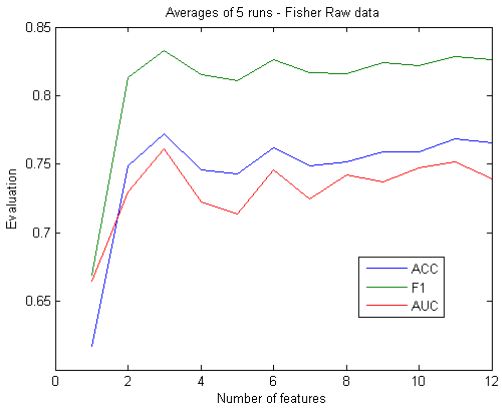

Figure 7: Fisher PCA RAW data

First we calculated the average of 5 runs of fisher, after applying to the dataset the PCA method.

By looking at the graphic we conclude which has the best number of features for this method. That was 5 features which we can get the F1 value around 0,84 with a accuracy of around 0,76 and the area under the curve was about 0,75.

After making an average of 5 runs of fisher, with raw data, we got the graphic with accuracy, area under curve and F1.

By analyses of the graphic, we can see the best values we got was with 3 features, giving F1 values within 0,8 and 0,85, with around 0,77 accuracy and AUC slightly over 0,75.

When we compare the two methods we can conclude that less of 5 features for PCA data or 3 features with raw data we got very bad results because we had limited information, more of that number of features the results undergo little change, because by increasing the amount of information also increases the complexity of the problem and can take the decision-making capacity.

5.3 k-NN

Knn is a simple classifier. It consists in choosing the class which is more present in the K samples near the pattern to be tested. By that we mean that analyzing the features dimensionality we get the K trained samples more similar to the one being tested and choose the class which is more present in that k samples.

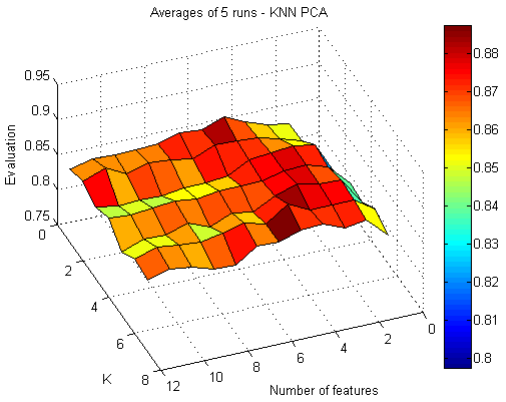

Figure 8: k-NN PCA

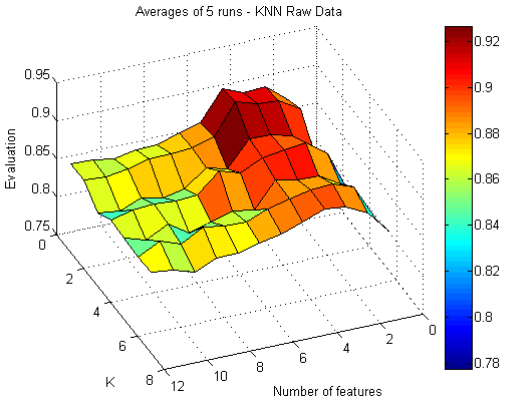

Figure 9: k-NN PCA RAW data

After making 5 runs for each number of features with raw data and for each value of K varying from 1 to 7, we easily verify that the best results of F1 score were achieved by K = 2 with 5 features.

Using PCA the results were not so good.

It's interesting how this simple classifier can achieve one of the best results described along this work.

5.4 SVM

Support Vector machines are machine learning algorithms that pretend to answer the question "What is the best separation hyper plane between the two classes?"

It calculates the hyper plane with the highest margin of separation between classes.

SVM algorithms base idea is to perform the empirical risk minimization, but not forgetting the structure. This allows such classifiers not to over fit like the classical neural networks since they are aware of the structure.

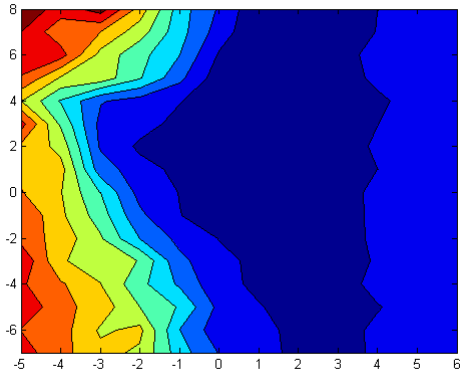

Figure 10: SVM

For SVM we used the rbf kernel.
We made the parameters vary along the window presented above. For each value of these parameters we built a new training set and testing set and tested SVM classifiers.

After 5 runs we calculate the averages of the errors and build the upper image in which we can see that the best parameters are near the center of it. We can't see by the colors, but the best values were 1,4.

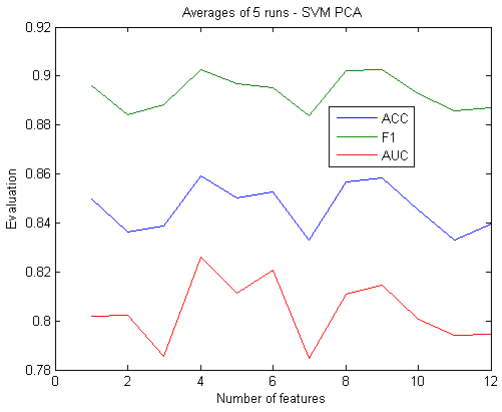

Figure 11: SVM PCA

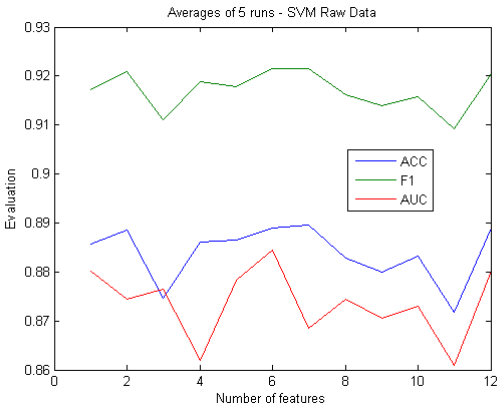

Figure 12: SVM Raw data

So after knowing that the best parameters for SVM were (1,4), we built that classifier with 80% training, 20% testing and got the results above for raw data and PCA features.

In the raw data plot we can see that the results achieved with the SVM classifier were the best results of this work. The F1 varied from 0,91 to 0,92 and the accuracy was in average 0,88 to 0,89. The optimal number of features was 6.

Using PCA the results were not as good as with the raw data and the optimal number of features was 4.

6 Conclusion

With this work we learned how to pick a real dataset and extract different features. After having the different features of the dataset we made the selection of the most relevant methods through the Kruskal-Walis, Scree Test and Kaiser Test. Then dividing our dataset into 80% for training and 20% for testing.

Finally we applied different classifiers to analyze our data, and we get better results with the SVM classifier using 6 features and raw data. Which we obtained the following results: F1 with values of 0,92, with accuracy around 0,88 and an area under the curve around 0,89.

On the other hand we have the k-NN classifier which in turn is a very simple classifier, also obtained very good results.