Fast k-NN classifier for documents based on a graph structure

F. Artigas-Fuentes[1], R. Gil-García[1], J. Badía-Contelles[2], A. Pons-Porrata[1]

[1] Center of Pattern Recognition and Data Mining
Universidad de Oriente, Santiago de Cuba, Cuba
email:{artigas,gil,aurora}@csd.uo.edu.cu
[2] Departament de Llenguatges i Sistemes Informàtics
Universitat Jaume I, Castelló, Spain
email:badia@uji.es

Abstract. In this paper, a fast k nearest neighbors (k-NN) classifier for documents is presented. Documents are usually represented in a high-dimensional feature space, where terms appeared on it are treated as features and the weight of each term reflects its importance in the document. There are many approaches to find the vicinity of an object, but their performance drastically decreases as the number of dimensions grows. This problem prevents its application for documents. The proposed method is based on a graph index structure with a fast search algorithm. It's high selectivity permits to obtain a similar classification quality than exhaustive classifier, with a few number of computed distances. Our experimental results show that it is feasible the use of the proposed method in problems of very high dimensionality, such as Text Mining.

Key words: nearest neighbor classifier, fast nearest neighbor search, text documents

1 Introduction

Text classification is the task to assigning documents to one or more predefined classes. This task relies on the availability of an initial set of classified text documents under these classes (known as training data). This task falls at the crossroads of information retrieval, pattern recognition and data mining, that involves data sets which are very large. Moreover, the dimensionality of the text documents is usually large. Therefore, it is crucial to design algorithms which scale well with the dimension.

The k nearest neighbor (k-NN) classifier is a very simple and popular approach used in classification [1], but it has the problem of the exhaustive computation of distances to training objects. Several methods have been proposed in order to avoid this problem. One approach consist in the improvement of access methods with a varied set of index structures and fast search algorithms. But, in the most of cases, their performance drastically decrease as the number of

dimensions grows. This problem is known as "the curse of dimensionality" [2], and prevents its application for text documents.

On higher dimensions several exacts methods, like VA-File [2], VA+-File [3], IQ-tree [4] and more recently VQ-index [5], have been proposed. The main purpose of those methods is to overcome the I/O disk bottleneck, which is crucial in large databases. Those methods were tested over relative high dimensional spaces, with 32, 64, 200 and 500 dimensions. However the main purpose of our proposal is to work over spaces with several thousands of dimensions, like the case of text documents. In this case we can obtain over 30000 dimensions with a relative small set of text documents.

On the other hand, several fast search algorithms have been proposed, like the optimal nearest neighbor algorithm for data structures that are stored in main memory, by Arya et. al in [6], and others.

But, in the higher dimensions the most of those methods have a performance as bad as a linear scan, or even worse. Such a linear scan does not scale well when the set of objects to search is large or the relationship function (distance, similarity or dissimilarity function) is very hard to compute.

Different relaxations on the precision of the result have been proposed in order to obtain a computationally feasible solution in those cases. This is called inexact proximity searching, as opposed to the classical exact proximity searching. Inexact proximity searching is possible in many applications because the preprocessing of data already involve an approximation to reality, and therefore a second approximation at search time is acceptable. Examples of those methods can be found in [7, 8].

In [9] an approximated classifier was presented for mixed data. This method uses a tree index structure and a fast search algorithm. It obtain the information necessary to classify at the same time the searching is doing. Even when the accuracy of classification obtained for mixed data is high, when this method is applied to documents it decreases.

In this paper, we introduce a fast k-NN classifier for text documents based on a graph index structure with an approximate k nearest neighbors fast search algorithm. It's high selectivity and precision permits to obtain a similar classification quality than exhaustive classifier, with a few number of computed relationships. Our experimental results show that it is feasible the use of the proposed method in problems of very high dimensionality, such as Text Mining.

The rest of the paper is organized as follows: Our proposed classifier is presented in Section 2. The obtained experimental results over Reuters Corpus Version 1 (RCV1-v2) are presented in Section 3. Finally, in Section 4, we present some conclusions and future work.

2 Proposed classifier

In this section, an approximate fast k-NN classifier for documents is introduced. The classifier consists of two phases. In the first one, the graph structure, using training set T, is constructed. In the second phase, novel documents are classified.

2.1 Preprocessing phase

The main idea of this phase is to build an index structure based on a connected graph. This graph must complain with the following conditions:

- Each vertex correspond with a different and unique training document (represented by it features vector).
- Each edge represent a relationship value between two vertices calculated by a similarity, dissimilarity or distance function Ψ.
- Each vertex v has, as minimum, ϕ adjacent vertices, were ϕ is an integer value preset by the user. That vertices correspond with the ϕ nearest neighbors of v.
- Vertices are connected forming triangles of minimum area.
- A small number of vertices are fixed and used as possible entry points to structure during the search phase.

In [10] we introduce the algorithm to build the index structure, whereas in [11] some improvements in order to reduce the time cost to build it are presented.

It is important to describe how the triangles of graph are built. We presented two ways to do this task.

The simple way is to connect first the most Ψ-related pair of documents in the training set TS. This pair of documents are used to obtain the first two vertices and the first edge of the graph. Then, the rest of documents in TS are candidates to be connected to this pair in order to obtain the third vertex of the first triangle. To do that we calculate the media object of this pair of vertices. It is obtained by the sum of each dimension of original objects and inserted into the set of media objects MOS. Then, the most Ψ-related pair formed by a candidate and one member of MOS is computed, and the candidate of this pair is selected as the next vertex. Following a similar iterative process new media objects for the new edges are calculated and inserted into MOS, and the rest of documents becomes in candidates to be connected into the graph.

The second way reduces the number of candidates to be considered in each iteration for obtain the new vertex to be connected into the graph. This improvement reduces too the general time cost to generate the index structure. In this case, the documents in TS are previously sorted by its relationship values with the global centroid GC. It is calculated in the same way of media objects, but using all documents in TS.

Then, the sorted set is divided into a number of equal sized subsets. As result, a sequence of naturally ordered subsets, by its general and descendent relationship value with GC is obtained. Next, the first subset is selected as the current one (CS). In this way, in each iteration, we only consider as candidates to become in new vertices those ones that belong to CS. When all candidates have been connected, the next subset by order becomes CS, and begin the next iteration. This process is follow until all documents are connected into the graph.

To guaranty the ϕ condition, if any vertex results with a less number n of adjacent vertices, simply we connect that vertex with its other $\phi - n$ nearest

neighbors. The ϕ nearest neighbors of all documents are calculated and kept in a previous stage.

Finally, vertices belonging to the borders of the region defined by the graph are selected as entry points to the structure (See [10, 11] for details). We call those vertices as Entry Points Set (EPS). This is the main difference of our access method to others: our index structure has several but a few number of entry points, despite off other structures based on trees with a single root.

2.2 Classification phase

In this phase, given the index structure G builded previously, the classes associated with the documents in the training set, and a novel document d to be classified, our classifier finds the k nearest neighbors, according with Ψ, and assign to d the majority class of its.

The classification involves three main stages:

1. The k nearest neighbors of the novel document are searched.
2. The votes of classes are counted, using the vote rule.
3. Finally, the classification is done using the decision rule.

In the first main stage, the fast search algorithm proposed in [10] and improved in [11] is used. It has three steps, in the first one a proper entry point EP_d to G for the current search d is selected. This choice can be different for each d and it must be the most Ψ-related to d of the members of EPS. EP_d is calculate in an exhaustive way, and it becomes the current solution (NN).

In the second step of search the actual nearest neighbor vertex to d is found. This task is doing by traversing the index structure following the edges of graph, selecting as a new current solution the most Ψ-related adjacent vertex of NN to d if it is better than the current one. The process ends when there is no new NN. The problem is that not always is possible to obtain the actual nearest neighbor. In a few number of cases an approximated one is obtained.

In order to improve the quality of k-NN, a variation of the search algorithm and a prune rule were introduced in [11]. The variation consists in selecting the best results of three independent searches of the k-NN, using three different entry points to the index structure. This solution increases the number of comparisons computed during searches. To avoid this problem we used the pruning rule that increases the selectivity of the search algorithm, avoiding extra computations.

In the last step of search, if $k > 1$, the other $k-1$ neighbors of d are obtained. This task is doing using another algorithm described in [10]. It uses the same strategy of follow the edges of the graph, but using as initial point the actual (or approximated) NN calculated in the previous step. Besides, the algorithm keep in each iteration a vector with the current list of k nearest neighbors.

After finding k-NN, the votes of each class are counted and the majority class is assigned to d.

3 Experimental results

In this section, the results of applying the proposed fast approximate k-NN classifier are presented.

To perform our experiments, we use the well-known benchmark collection Reuters Corpus Version 1 (RCV1-v2) [12]. This collection has a set of documents represented as vectors. The feature vector for each document was produced from the concatenation of text in the <headline> and <text> XML elements. Text was reduced to lower case characters, after which tokenization, punctuation removal and stemming, stop word removal, term weighting, feature selection, and length normalization was applied. The LYRL2004 partition, with 23149 training, and 781265 testing vectors, was used.

Classes files of both training and test sets were modified to avoid overlapping among classes. The resulting sets belong only to four non overlapped classes: ECAT, CCAT, MCAT and GCAT. It was necessary because the other fast approximated classifier implemented (FC) to compare with our proposal (FGC) do not support class overlapping. A k-NN exhaustive classifier (EC) was implemented too, and was used as base line.

First, a 10% of training documents (692) documents were randomly chosen to build index structures for both FC and FGC classifiers, while maintaining the distribution of the class probabilities in the original training and test sets. The representation space obtained has 8731 dimensions.

The FC is a k-NN classifier [9] that uses an index structure based on a tree. Each node of the tree contains a certain number of elements selected using a grouping algorithm. In the original paper, the authors present and use a new clustering algorithm called *k-means*, but in our experiments we used both *k-means* and the well-known *c-means* [13].

Besides, FC requires additional parameters to build the tree. The minimal number of objects in a node was fixed to 20, the maximum number of clusters by level was fixed to 5, and the maximum number of iterations of the clustering algorithm was fixed to 10. The authors used three stops conditions to determine the leaf nodes. But, we use only the last two conditions based on non-homogeneous (noHomo) and homogeneous (Homo) nodes. In the case of our proposal, the value of ϕ was fixed to 50.

The low number of documents selected as training set was because the very high time cost of *k-means* when it is applied to grouping objects with an elevate number of dimensions.

The Ψ used was a distance function based on the well-known cosine similarity (1), because it is the most wide used to compare documents in text mining. This measurement reaches its minimum value at 0 and maximum at 1. All the classifiers uses the same vote rule.

$$\Psi(d_1, d_2) = \sqrt{1 - cos(d_1, d_2)} \tag{1}$$

were d_1 and d_2 are the documents to be compared.

All the algorithms were implemented using Python 2.5 over an Intel(R) Core(TM)2 Quad CPU, 2.50 GHz and 3GB of RAM with Linux Mandriva 2009 OS.

For the classification phase, 500 documents from the test set were randomly selected. The classification was carried out using the three classifiers: FC, FGC and EC. For FC, its authors offer two algorithms to search the approximates k-NN. But, we only show the results obtained using the KMSNLocal algorithm, because the results for the other one were very similar.

To compare the quality of classifiers, macro F1-measure was used. It is the average on F1 scores of all the topics. The F1 score (2) can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

$$F1 = 2.\frac{precision.recall}{precision + recall} \tag{2}$$

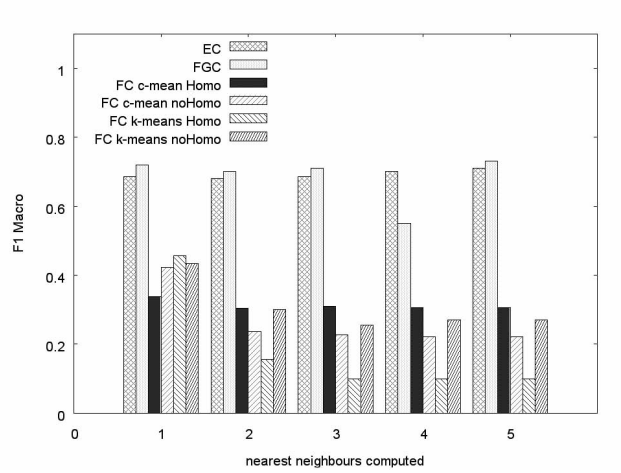

Fig. 1. A comparison of the quality of the implemented classifiers varying the number of nearest neighbors calculated.

The figure 1 shows the macro-average F1 values obtained with the classifiers varying the number of k-NN computed from 1 to 5. As you can see, our classifier was better than FC in all the cases. Our proposal results even better than the EC, although it uses an exact method to obtain the vicinity of document queries. This is an strange behavior.

In order to investigate what is the cause of this behavior, we implemented a 4th classifier βFGC, based on FGC. Despite of the original, in the voting phase, for βFGC only those nearest neighbors with a relationship value greater than a certain threshold β are taking into account. We do the same change to EC and obtain a new base line classifier (βEC).

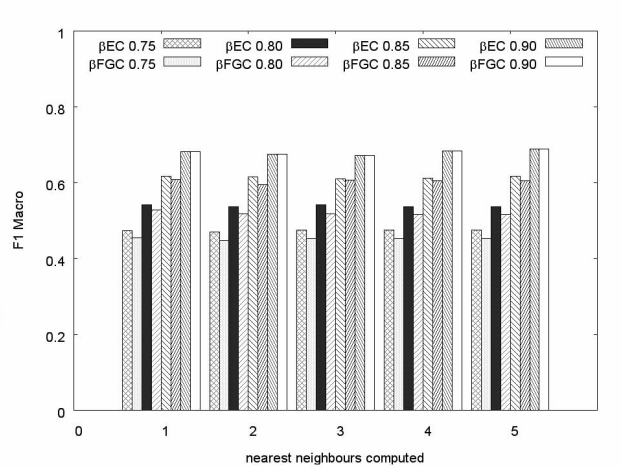

Fig. 2. A comparison between βEC and βFGC varying β parameter.

The figure 2 shows the quality of results when β has the values 0.75, 0.80, 0.85 and 0.90. For β values less than 0.90 the quality of βFGC are slightly worst than βEC. On the contrary, for β values from 0.90 it is slightly better. This results mean that the previous strange behavior was provoked by the elements of the k-NN more distanced to queries. When all k-NN are taking into account, in the most of approximate results, the last elements found by our search algorithm belong to the actual classes of queries despite of the elements found by EC.

Other aspect that we consider, in order to evaluate our proposal, was the time required to classify documents. The best improvement obtained as consequence to use our access method based on a graph was the dramatical reduction of the number of distances computed to obtain the k-NN against the 100% needed by the EC. Table 1 shows the time cost and the best quality in classification obtained by EC, FC and our proposal FGC.

Table 1. A summary of best results for each classifier

Parameter	EC	FC	FGC
Time cost(s)	60.06	1.20	14.53
F1-macro	0.71	0.45	0.73

As you can see, FC is very fast, but its results are not good for documents classification. On the other hand, our proposal obtains good results in a reasonable time.

4 Conclusions

In this work, an approximated fast k-NN classifier for text documents, a problem with a very high dimensionality, was proposed. In order to compare our method, different variants of a fast k-NN classifier were implemented using the same distance function for compare documents, and the same vote and decision rules to classification. An exhaustive classifier was used as base line. Based in our results, our proposal obtains high quality results, better than the state-of-the-art classifiers presented.

In comparison with the exhaustive classifier, our method obtain a drastically reduction on the time cost due to the use of an index structure based on a connected graph and a fast search algorithm, with very similar, even better, classification quality results.

References

1. Myles, J. P. and Hand, D. J.: The Multi-Class Metric Problem in Nearest Neighbor Discrimination Rule. Pattern Recognition 23, 1291-1297 (1990)
2. Schek, H., et. al: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. VLDB'98, New York, USA, 194-205 (1998)
3. Ferhatosmanoglu, H., et. al: High dimensional nearest neighbor searching. Information Systems, 31, 512-540 (2006)
4. Berchtold, S. et. al: Independent quantization: An index compression technique for high-dimensional data spaces. In Proc. 16th Int. Conf. on Data Engineering, San Diego, CA, 577-588 (2000)
5. Tuncel, E., et. al:VQ-Index: An Index Structure for Similarity Searching in Multimedia Databases. In 10th ACM International Conference on Multimedia 2002, Juan Les Pins, France, 543-552 (2002)
6. Arya, S. et. al: An optimal algorithm for approximate nearest neighbor searching. In 5th Ann. ACM-SIAM Symposium on Discrete Algorithms, 573-582 (1994)
7. E. Chávez, K. Figueroa and G. Navarro: Effective proximity retrieval by ordering permutation. TPAMI07. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(9), 1647-1658, (2008).
8. K. Figueroa and K. Fredriksson: Speeding up permutation based indexing with indexing. SISAP09. IEEE Computer Society, 107114, (2009).
9. S. Hernández-Rodríguez, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa: Fast Most Similar Neighbor Classifier for Mixed Data Based on a Tree Structure. L. Rueda, D. Mery and J. Kittlel (Eds.), CIARP 2007, LNCS 4756, 407-416, (2007).
10. F. Artigas-Fuentes, R. Gil-García,J.M. Badía- Contelles and A. Pons-Porrata: Vicinity calculation with graph in text mining. UCT,48, F. Genolet(Eds.), 1-10, (2008).
11. F. Artigas-Fuentes, R. Gil-García and J.M. Badía- Contelles: A High-dimensional Access Method for Approximated Similarity Search in Text Mining. Accepted in the ICPR 2010 Congress. 23 to 26 of August, Istanbul, Turkey, (2010)
12. D.L. Lewis, Y. Yang, T.G. Rose and F. Li. RCV1: A new benchmark collection for text categorization research. Journal of Machine Learning Research, 5:361-397, (2004).
13. J.Yu: General C-Means Clustering Model. IEEE Transactions on Pattern Analysis and Machine Intelligence,27(8), 1197-1211, (2005).