Selecting Interesting Articles Using Their Similarity Based Only on Positive Examples

Jiří Hroza and Jan Žižka

Faculty of Informatics, Department of Information Technologies,
Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
{xhroza1, zizka}@informatics.muni.cz

Abstract. The task of automated searching for interesting text documents frequently suffers from a very poor balance among documents representing both positive and negative examples or from one completely missing class. This paper suggests the *ranking approach* based on the k-NN algorithm adapted for determining the *similarity degree* of new documents just to the representative *positive* collection. From the viewpoint of the precision-recall relation, a user can decide in advance how many and how similar articles should be released through a filter.

1 Introduction

When selecting from unstructured natural language text documents, a pragmatic trouble can aggravate the design of a filter: many users collect articles that represent (almost) only the interesting ones, and the required *relevant negative examples* for training an algorithm are missing. Typically physicians, having only positive examples of articles, need to automatically single out very specific medical documents within a narrow expert area—yet, containing too many articles around very similar topics [1]; here is the inspiration for the described research. The problem with synthetical filling in the missing examples is that *arbitrary* text documents different from the positive ones cannot be generally used: how to define effectively the dissimilarity? This paper describes the *ranking approach* based on the k-NN (k-nearest neighbors) algorithm adapted for determining the similarity of articles to the representative *positive* examples. For the comparison, outcomes of the SVM (*support vector machines*) algorithm are also shown.

2 Text Documents and Their Preprocessing

To test performance of the one-class k-NN and SVM, one of the standard benchmarks 20Newsgroups dataset was used[1]. Then, the one-class k-NN was also applied to a specific set of real expert medical documents[2] from MEDLINE [1].

[1] http://www.ai.mit.edu/~jrennie/20Newsgroups/
[2] http://www.fi.muni.cz/~xhroza1/datasets/glall/

Porter's algorithm [4] was applied to obtain a stem of each word. The dictionary was created as a set of all distinct words in the exemplary articles (*bag of words*), and 100 of the most common English words plus words occurring less than three times were removed. Each document was encoded into a *feature vector* where every position in the vector corresponded to one word in the dictionary (the number at the position was a *relative frequency* of the word). For SVM, the *binary representation* [a word is/is not (1/0) at a given position] and other parameter settings were used as recommended in [2].

3 Applied Document-Filtering Algorithms

One-Class Ranking by k-NN: The k-nearest neighbors algorithm (k-NN) [3] has a simple training phase based just on storing of training text document examples. During the classification phase, the algorithm finds the k most similar training examples for an unclassified article. Then the article's class is the most common class of the k found training examples weighted by their similarity (the cosine measure, i.e., the document similarity obtained by the vector-representation comparison). For the one-class problem, this paper suggests a modified, *one-class* k-NN ranking version. The training set consists only from instances of available interesting articles. The ranking phase computes similarities to the k nearest neighbors. When the similarities (playing a role of weights) of all new unclassified articles are known, the documents are sorted according to sums of these values as shown in Table 1.

Table 1. The ranking algorithm used by the modified k-NN

1. Represent m new unclassified documents as vectors using a *bag of words* from the training phase.
2. For each new vector $\mathbf{u_i}$, compute its cosine similarity measure $s(\mathbf{u_i}, \mathbf{v_j})$ to all training vectors $\mathbf{v_j},\ i = 1, \ldots, n,\ j = 1, \ldots, m$:

$$s(\mathbf{u_i}, \mathbf{v_j}) = \frac{\mathbf{u_i}^T \mathbf{v_j}}{\|\mathbf{u_i}\| \ \|\mathbf{v_j}\|}, \tag{1}$$

3. For each $\mathbf{u_i}$, select its k nearest neighbors $\mathbf{v_j},\ j = 1, \ldots, k$, where a higher similarity s means a closer distance. Using the k highest similarities s_{kNN}, compute the resulting $\mathbf{u_i}$'s value $w(\mathbf{u_i})$ used for setting up its ranking position:

$$w(\mathbf{u_i}) = \sum_{j=1}^{k} s_{kNN}(\mathbf{u_i}, \mathbf{v_j}) \tag{2}$$

4. According to the w's obtained in the previous step, create the rank of all investigated text documents: higher w's mean higher positions in the rank.
5. Within the acquired rank, classify the first r vectors as *positive* ones and release them through the filter as *interesting articles*, where r is a user's parameter.

Comparative Filtering by SVM: The one-class SVM enables to learn a concept of classification into a relevant/irrelevant class while it learns only from relevant instances; the linear kernel for the one-class SVM does not seem to be very sensitive to the choice of parameters [2]. The BSVM[3] 2.6 implementation with linear kernel and default parameters was employed. To determine the difference between training SVM by one class and—if available—two classes, the same two-class SVM software was used.

4 Experiments and Their Results

For each of 20 newsgroups, experiments were carried out. Each of the experiments consisted of the 10-fold cross-validation of a dataset created from one newsgroup as a positive class and the rest of 19 newsgroups as a negative class to establish a situation similar to a real user's one.[4] The evaluation uses *precision, recall,* and the F_1 measure [5].

5-NN Ranking: The ranking algorithm was used to rearrange filtered examples. Experiments revealed that comparisons with 5 nearest training articles provided the best results. Figure 1 shows the precision of the 5-NN when browsing through the rearranged examples from the most relevant ones to the less relevant ones. Table 2 compares results with the SVM methods. It is necessary to emphasize that a real user typically can exploit only a very small part of suggested documents, so there is a high chance to find them at the top of the rank with the 70%–80% likelihood.

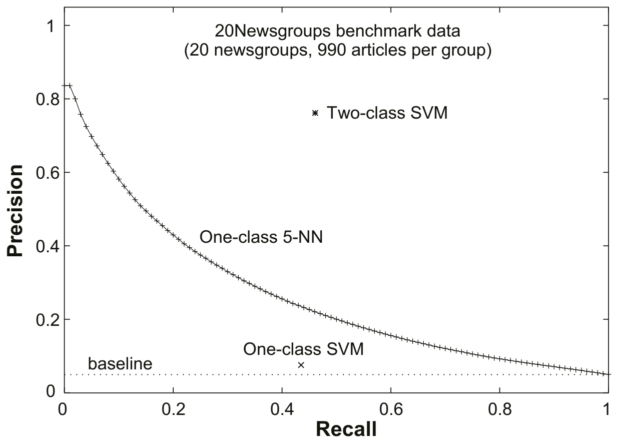

Fig. 1. 5-NN ranking, one- and two-class SVM

[3] http://www.csie.ntu.edu.tw/∼cjlin/bsvm/

[4] in the one-class and ranking cases, the splits were the same as in the two-class training, except that in the former cases the negative training examples were unused for building a classifier/ranker

Table 2. Results of the algorithms' success for the benchmark data

Algorithm	Precision	Recall	F_1 measure
2-class SVM	76.2%	46,1%	52.5%
1-class SVM	7.6%	43.5%	12.9%
1-class 5-NN	29.8%	34.0%	31.8%
baseline	5.0%	100%	9.5%

One- and Two-Class SVM: The one-class SVM results were poor, see Table 2. Predictably, the two-class SVM achieved very good results; however, it *does require* instances from both classes.

MEDLINE Documents: After verifying functionality of the one-class k-NN, it was successfully applied to the medical data: $F_1 = 0.73$, however, the best results were obtained for 1-NN and cannot be directly compared with the benchmark outcomes because of different document distribution and a higher baseline. In the tested cases, the initial problem with filtering articles using only positive examples quite acceptably handles the suggested one-class k-NN.

5 Conclusions

The ranking k-nearest neighbor algorithm trained only from the available positive examples was able to correctly arrange new text articles. Such an arrangement allows acquiring a reasonable portion of interesting documents with a higher precision, up to 70%–80%. The one-class SVM had difficult problems—its results were only slightly above the F_1 baseline; the two-class SVM algorithm cannot be used when only one class is available, otherwise it would be superior.

Acknowledgments

This research was partly supported by the Grant *Human-Computer Interaction, Dialog Systems and Assistive Technologies, MSM*-143300003.

References

1. Hroza, J., Žižka, J., and Bourek, A.: Filtering Very Similar Text Documents: A Case Study. In: Gelbukh, A. (Ed.): Proc. of the Fifth Intl. Conf. on Intelligent Text Processing and Computational Linguistics CICLing-2004, February 15-21, 2004, Seoul, South Korea, Springer Verlag, 2004, LNCS 2945, pp. 511-520
2. Manevitz, L. R. and Yousef, M.: One-Class SVMs for Document Classification. In: Journal of Machine Learning Research 2 (2001), December 2001, pp. 139-154
3. Mitchell, T. M. (1997): Machine Learning. McGraw Hill, 1997
4. Porter, M. F.: An Algorithm For Suffix Stripping. Program 14(3), 1980, pp. 130-137
5. Van Rijsbergen, C. J.: Information Retrieval, 2nd Edition. Department of Computer Science, University of Glasgow, 1979