The Selection of Electronic Text Documents Supported by Only Positive Examples

Jan Žižka[1], Jiří Hroza[2], Bruno Pouliquen[3], Camelia Ignat[3] & Ralf Steinberger[3]

[1] CBA, Masaryk University, Kamenice 126/3, 62500 Brno, Czech Republic

[2] FI, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic

[3] European Commission – Joint Research Centre, T.P. 267, 21020 Ispra (VA), Italy

Abstract

The European Commission has a freely accessible news monitoring system called the *Europe Media Monitor NewsBrief* (http://press.jrc.it/), which is available for all twenty official languages of the European Union, plus some more languages. Among other things, *NewsBrief* categorizes articles through routing procedures and it alerts users interested in a large variety of different subject domains automatically. In the effort to improve the multilingual categorization and relevance ranking functionality for some complex interest profiles, for which *only positive examples* are currently available, we implemented a modified k-NN (*k-nearest neighbors*) algorithm and empirically detected parameters and parameter settings that produce good results for rather different subject areas (news on the *EU-Constitution, on Iraq,* and on *Terrorism*). Experiments on this real-life data yielded very satisfying results : a precision of over 90% for a recall of up to 70%. These results were then compared to others achieved with one-class SVM and with SVM that was trained on both positive and artificially generated negative example sets. Efforts are currently underway to incorporate this new functionality within *NewsBrief* and to make it available to the users.

Keywords : machine learning, text categorization, relevance ranking, k-nearest neighbors, support vector machines, positive examples, feature selection, feature optimization, domain-independence.

1. Introduction

Most large organizations monitor the media for information about specific subject areas. In the past, news clipping services, i.e. the retrieval and selection of relevant news articles, was done manually. Today, users can purchase already categorized news from news providers such as Reuters and Lexis-Nexis, or they can receive them freely from GoogleNews (http://news.google.com/) or the European Commission's *Europe Media Monitor* (Best et al. 2002 ; Steinberger et al. 2005). This saves a lot of manual work for the gathering of the news, but the news monitoring task is still rather time-consuming due to the large amount of news articles available and due to non-optimal selection procedures which often introduce errors. GoogleNews allows Boolean search expressions. NewsBrief additionally allows manually assigned positive and negative weights for search words, which need to sum up to a certain user-defined threshold before the article is forwarded to the users. This works rather well for well-defined searches involving, for instance, an organization or person name. For more complex and fuzzy searches, however, wrong hits (low precision) are frequent, especially when users aim at a good recall by adding many search words. For instance, when searching for articles about 'Political Unrest,' or 'Illegal activities in the area of the former Soviet Union,'

Boolean expressions and simple weighting are not powerful enough. Furthermore, writing and tuning such searches can become rather time-consuming and the subject domain specialists may not be well-equipped to carry out this tuning task.

In order to help users in this challenge, we want to provide them with a learning system that allows them to refine their query by giving relevance feedback. In our ideal scenario, users will be able to create a new interest profile from scratch by providing the system with positive and negative examples, so that Support Vector Machines, SVM, (Cristianini, N. and Shawe-Taylor, J., 2000 ; Cristianini, N., and Schölkopf, B. (2002)) or Bayesian classifiers (Mitchell, 1997) can be used. However, most users achieve reasonable results with the current means to formulate queries. Furthermore, large numbers of examples might be needed in order to produce a good classifier for a fuzzy interest profile such as 'Illegal activities in the area of the former Soviet Union.' We have therefore focused on refining the current interest profiles by adding *relevance judgments* (either binary judgments or continuous relevance-ranking) as a second step after the first filtering using Boolean expressions and term weighting.

A major bottleneck in our effort is the fact that *NewsBrief* users are usually extremely busy so that it is difficult to get them to give enough feedback to have both training and testing data. On the other hand, there is an abundance of previously manually selected articles for various subject domains (the so-called *Panorama* data) that can be used to train and optimize such categorizers. We therefore used these collections of manually selected user-relevant news articles, i.e. *collections of only positive examples*. Having these collections was at the basis of our development effort. The overall goal then was to identify a binary categorization or continuous relevance ranking algorithm, as well as a set of features and parameter settings that produce good results for a large variety of subject domains. It was thus a precondition that no individual tuning would take place for new subject areas, but that *the same features and parameter settings would have to be used for all subject domains*. A priority of our work was furthermore on achieving a relatively *high precision* in order to spare users from having to look through many wrong hits. Only in a second step, users should be confronted with those news articles that potentially talk about their subject of choice, but which are less likely to be relevant.

In order to find a good solution to our task where the condition was to train the system using positive examples only, we carried out experiments using the bag-of-words representation and the k-nearest neighbors approach (k-NN), which allows relevance ranking of new documents by comparing them to the positive training documents. However, we also ran a less exhaustive set of experiments using support vector machines (SVM), which typically require both positive and negative examples, but which we trained by using a random selection of other articles as negative examples. We also tested the empirical usefulness of different feature selections and representations on three different test sets : (a) usage or non-usage of stop words ; (b) usage or non-usage of a stemmer ; (c) binary feature values (presence vs. non-presence of a word) ; (d) feature values using the absolute frequency of words in the article or of a (e) weighted frequency.

The following sections give an overview of the k-NN algorithm (Sect. 2) and describe the test data sets (Sect. 3), the feature representations (Sect. 4) and the test results (Sect. 5). In Section 6, we draw conclusions.

2. The k-NN Ranking Algorithm

A general overview of machine-learning (ML) approaches to document classification and ranking can be found in Sebastiani (2002). The *k*-nearest neighbors (*k*-NN) algorithm is a method that is also often used for text categorization. Its principle for multi-class problems is very simple : *k*-NN requires just a trivial training phase – i.e., storing a suitable representation of labeled training examples. During the classification process, when a new, unlabeled example occurs, *k*-NN computes the distance (i.e., similarity) to all labeled examples. Using the *k* nearest labeled examples, the most frequent class is chosen for the unlabeled instance.

The *k*-NN algorithm cannot be applied directly to one-class problems. However, there is a possibility to employ the *ranking* process : unlabeled news texts are ranked according to their similarity to the training samples. When the distances of unlabeled news from *k* nearest positive examples are computed, the resulting values, see Eq. (2), can be used for sorting the classified examples : nearer unlabeled instances take positions ahead of the ones that are further away. Then, users decide what is the 'true' similarity, how many news they are willing to accept, what degree of *precision* is acceptable, and what *recall* is still satisfactory (Van Rijsbergen, 1979). According to priorities assigned to the parameters of the *k*-NN Ranking algorithm (*k*, *precision*, and *ranking*), users can tune the outcome to their needs.

Some other algorithms can also be adapted to learning from only one class of examples, e.g., (Manevitz and Yousef, 2001) dealing with support vector machines (SVM). However, when only one training class is available, less information can result in less good results of the classification process, as was shown in (Hroza and Žižka, 2005a). In this case, new unlabeled news items need to specify their similarity to the available positive training samples. Then, using the similarity degree, unlabeled documents can be divided into two categories : interesting ones (similar to a certain degree) and uninteresting ones (dissimilar, below a given degree). For such an approach, the k-nearest neighbors algorithm (Duda et al, 2001 ; Hroza and Žižka, 2005b) can be used.

The news texts are represented as a *bags of words*, where each distinct word creates its respective dimension, i.e. the occurrences of words correspond to coordinates of the multidimensional vectors. Using the cosine measure, see Eq. (1), which reflects the similarity of vectors representing individual news items, the positive documents accumulate in front of the negative ones. Based on the word contents without considering any word meanings or mutual relations, the method assigns higher priorities to news items that are more similar to the positive samples because they have more identical words (features). When the users investigate such news according to their *rank* of similarity, they have a much higher chance to obtain a proportionate number of truly relevant documents. The *k*-NN ranking is based on two phases : training and ranking.

2.1. The training phase

1. After the appropriate preprocessing of the data (e.g., stemming or lemmatization, exclusion of stop words, etc.), all positive training documents can be represented as *bags of words : n* labeled news articles are now multidimensional vectors $\mathbf{v}_j$, $j = 1, …, n$, with coordinates given by occurrences (binary, frequentative, or other ones) of individual words.

2. Then, the entire set of currently available positive training vectors can be stored as a representation for the relevant documents, against which new documents can be compared in the future.

2.2. The ranking phase

1. Represent m new unlabeled news texts in the same way as in the training phase.

2. For each unlabeled vector $\mathbf{u}_i$, compute its cosine (or other) similarity measure $s(\mathbf{u}_i, \mathbf{v}_j)$ to all labeled training vectors $\mathbf{v}_j$, $i = 1, \ldots, n, j = 1, \ldots, m$:

$$s(\mathbf{u}_i, \mathbf{v}_j) = \frac{\mathbf{u}_i^T \mathbf{v}_j}{\|\mathbf{u}_i\|\|\mathbf{v}_j\|} \ ,$$

(1)

where $\mathbf{u}_i^T$ is transpose of $\mathbf{u}_i$. Two identical vectors have the similarity $s = 1$, while two completely different vectors not having any words in common have the similarity $s = 0$.

3. For each $\mathbf{u}_i$, select its k nearest neighbors $\mathbf{v}_j$, $j = 1, \ldots, k$. Then, using the first k highest similarities $s_{kNN}(\mathbf{u}_i, \mathbf{v}_j)$ from the previous computation of all $s(\mathbf{u}_i, \mathbf{v}_j)$, compute the resulting $\mathbf{u}_i$'s value $w(\mathbf{u}_i)$ that is used for setting up the ranking position :

$$w(\mathbf{u}_i) = \sum_{j=1}^{k} s_{kNN}(\mathbf{u}_i, \mathbf{v}_j) \ .$$

(2)

4. According to the w's, create the rank of all investigated news items : higher w's mean higher positions (i.e., more relevant news) in the rank.

5. Within the acquired rank, label the first r vectors as definitely *positive* ones and propose them to the users as potentially *relevant* news. The value r depends on the user's preferences for *precision* and/or *recall* because higher *precision* usually entails lower *recall* and vice versa.

3. Training and Testing Data

The main aim was to develop a system that would learn user relevance judgments by automatically analyzing a selection of news texts that had been identified as being relevant in the past. This system would then have the task of selecting the potentially relevant documents out of a continuous incoming flow of unclassified electronic news texts. In our case, collaboration from the typically very busy users could not be expected so that the system would have to be trained on the basis of pre-existing user-relevant documents that had been selected over a long period in the past. This collection consisted of positive examples only, i.e. users with different user profiles had each collected their own documents of relevance (the so-called EMM *Panorama* data). No negative examples were available. This means that the system would either have to learn from positive samples only, or that we would have to artificially generate a collection of negative examples by making a random selection of articles from the other domains. We mainly experimented with positive samples only, using k-NN, but we also ran some promising experiments with both positive and negative examples, using SVM.

The system was supposed to be domain-independent, as users with a varied range of interests would eventually use it without getting any further help. Providing relevance *ranking* in addition to the binary relevance judgment would have been considered a positive feature, but was not an absolute requirement. In order to guarantee a domain-independent system, we would experiment with documents from three different domains and choose the method that performs best across the domains. The training and testing data was the following :

- *EU-Constitution* news. The topic is the EU Constitution : 96 positive examples, 18,074 words altogether, on average 188 words per article ; 96 negative examples, 22,237 words, on average 232 words per article.

- *Iraq* news. The topic is the contemporary situation in Iraq : 554 positive examples, 119,400 words, on average 216 words per article ; 554 negative examples, 131,890 words, on average 238 words per article.

- *Terrorism* news. The topic is the current terrorism problem : 117 positive examples, 29,235 words, on average 250 words per article ; 117 negative examples, 24,793 words, on average 212 words per article.

The negative samples were thus a random selection of other 'Panorama' news, i.e. news that were different from the manually selected relevant ones. The ratio between relevant and irrelevant news items was more or less one-to-one. When studying the ranking capabilities of the modified *k*-NN algorithm, the baseline for all three topics was thus 0.5.

4. Data Preprocessing

The first part of the experiments used a simple preprocessing based on stemming. To obtain a stem of each word, the commonly used Porter's stemming algorithm (Porter, 1980) for suffix stripping was applied. This step is often used for English texts and its major advantage is that it decreases the number of dimensions without losing much information. In a second go, the same experiments were carried out again on the same data, but without stemming, in order to find how much it affects the results. The main problem is that Porter's stemming was developed for English, so that other European languages should use respective word normalizations. However, stemming methods are not available for all languages that contribute to electronic news monitored by institutions of the European Commission. Similarly, the *lemmatization* process could also be used to decrease various forms of individual words but the situation is the same : for some languages, appropriate linguistic tools are either not available or do not work out on the same level.

After this stage, a common dictionary was created as a bag of all preprocessed words from the original training examples. In some of the experiments, removing a set of *stop words* was tested as well, to determine the usefulness of this procedure. Our stop word list consisted of 621 words that were either semantically meaningless (function words such as prepositions and determiners) or words that are extremely frequent in our news corpus (e.g. *today*, *Reuters*, *reported*, etc.). As the next step, all the appropriate news were encoded into vectors of numbers using the dictionary. Every position in the vector corresponded to a certain word in the dictionary. Three types of feature values were employed. The first straightforward method used the binary tag '1' in a position of the vector if the corresponding word was present in a document ; otherwise it was set to '0'. The second method used the word frequency in a text document to provide additional information to the *k*-NN algorithm. Essentially, these frequencies played a role of word weights. The third method employed the so-called *TF.IDF* weighting (*Term Frequency–Inverted Document Frequency* ; see Sebastiani, 2002), where frequencies of words are weighted by considering their frequencies in the rest of the document collection. Instead of using the absolute frequency of words, TF.IDF thus assigns higher weights to those words that appear significantly more frequently in the given document than they appear on average. The purpose of these experiments was to find out which of the various feature selection and representation mechanisms would produce the best results.

5. Outcome of the Experiments

In the experiments, we tested two algorithms, as described in section 2 : The first one was *k*-NN, trained only on positive samples ; the second one was SVM, which was used as a representative text classifier to have a comparison for the results obtained by *k*-NN Ranking. The SVM classifier was trained either by using only positive samples or by using both positive and negative samples, to show possible differences between the most common and promising approach (if at all possible) and the situation when it cannot be applied due to the lack of training counter-examples. Each algorithm was evaluated using 10-fold cross-validation. The overall result was given by the average and represented the pessimistic estimate of the expected classification error, because only 90% of training examples were put to use in every partition of the training set while the remaining 10% served as testing instances. In the final application, all samples would be used to train and use the algorithms.

Evaluation of the *k*-NN algorithm's performance was done in the following way : After ranking and subsequent sorting of the news, there should be relevant documents at the first positions, and irrelevant ones at the last positions. In the graphs, the effectiveness of algorithms is presented using the *precision* dependence on *recall*. The following figures, which demonstrate selected important results of the experiments, use these symbols : C = the *EU-Constitution* data, I = the *Iraq* data, T = the *Terrorism* data, 1/0 = binary word encoding, frq = frequency encoding of words, tfidf = *TF.IDF* encoding, SW = using stop-words, NW = not using stop-words, LM = stemming, and NL = no stemming).
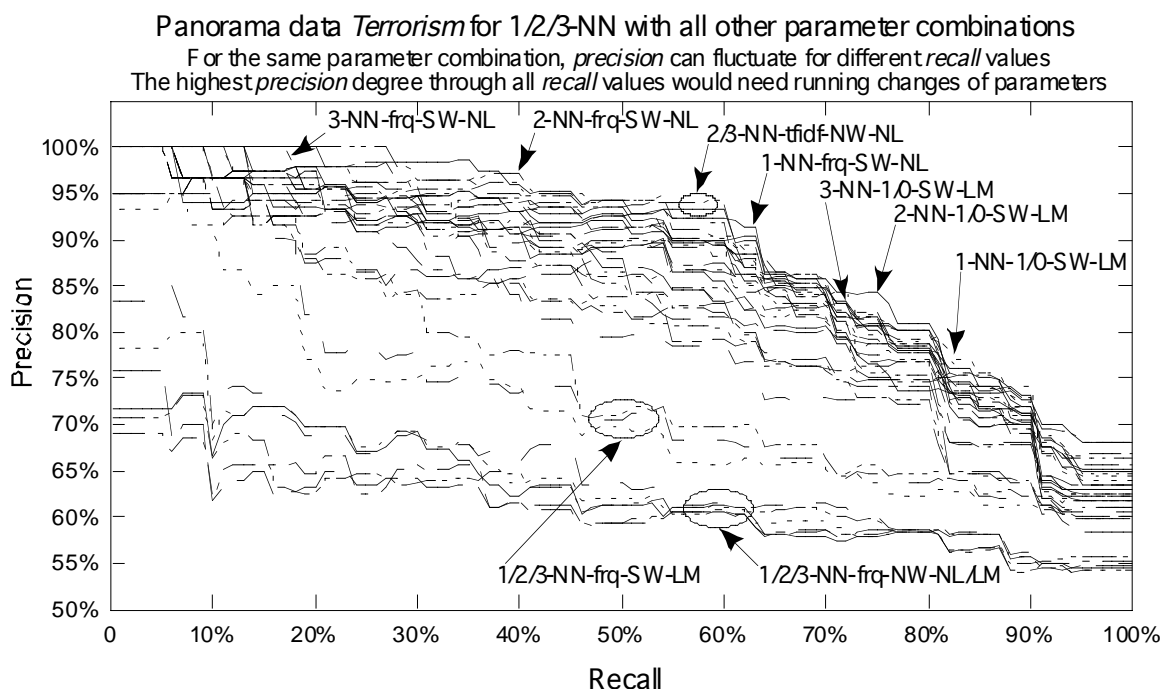

Figure 1. *Outcome of the Terrorism data for all tested combinations of the k-NN Ranking algorithm parameters (k=1,2,3, word-encoding 1/0 or frequency or TF.IDF, using/not using stop words SW/NW, stemming/not stemming LM/NL) : Only some combinations of parameters are acceptable for the requested high precision. The arrows point to several curves to show examples of parameter combinations that provide different or similar results. Obviously, it is not easy to find an optimal parameter combination for this data. The situation is similar with the EU-Constitution and Iraq data.*

Figure 1 illustrates that, even for the single domain 'Terrorism', it is not straightforward to find optimal parameter settings across different recall levels. It is thus difficult to find the common high-precision parameters for all data types. However, the results with the parameters k=2, frequency encoding, usage of stop-words, and no stemming seem to be among the best. The Figure also shows that some parameter settings clearly perform less well so that we can discard them for further experiments. Among several parameter settings that perform similarly well, it is advisable to select the ones that are computationally less demanding. From a computational complexity point of view, it is for example preferable to use a smaller number for 'k' (fewer nearest neighbors), to use the binary or frequency encoding instead of *TF.IDF*, and to use word full forms instead of carrying out stemming or lemmatization.
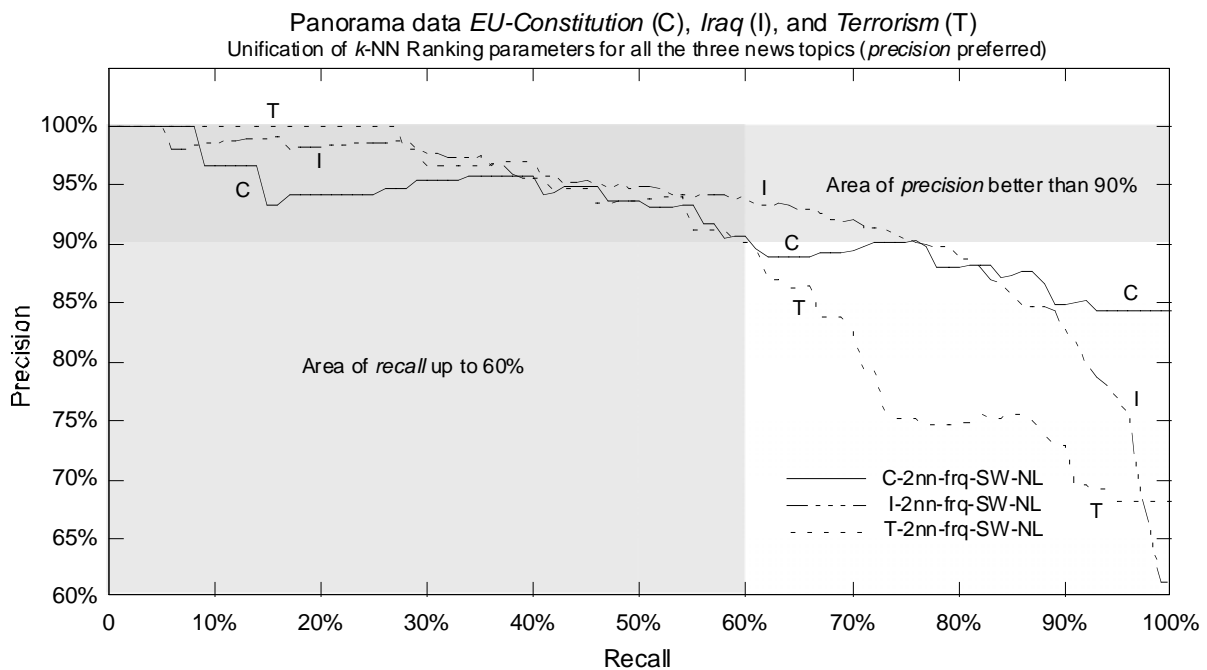

Figure 2. *Outcome of k-NN with the parameter trade-off for all the three topics EU-Constitution, Iraq, and Terrorism. The k-NN Ranking parameters are k = 2 (2-NN), frequency encoding frq, stop-words SW, and no stemming NL. Precision has the highest priority here.*

Figure 2 shows the precision-recall dependence for the well-performing parameter settings k=2, frequency encoding, usage of stop-words, and no stemming. Assuming that we need rather high precision (>90%), we see in Figure 2 that, for all three domains, the recall values are above 60%. It does, of course, depend on user preferences whether they wish to get higher precision and lower recall or vice versa. However, for large data volumes, higher precision would probably be preferred as the users may get less – but more relevant – news that could afterwards be reasonably processed.

Figure 3 shows only the best results for all three topics. Obviously, the precision of some data decreases faster, especially for the *Iraq* and *Terrorism* data sets, while for the *EU-Constitution* data set the results are very good up to recall ≤ 70%. Up to recall ≤ 60%, almost all data here provides precision better than 90% for various parameter combinations. The one-class training process successively used *k* = 1, 2, and 3 nearest-neighbors with all combinations of other parameters. Higher *k*'s did not improve the results. It is not quite

univocal which values of parameters should be chosen to obtain generally the best results for any data, but the parameter settings chosen for display in Figure 2 indeed seem to be a good all-purpose compromise solution.
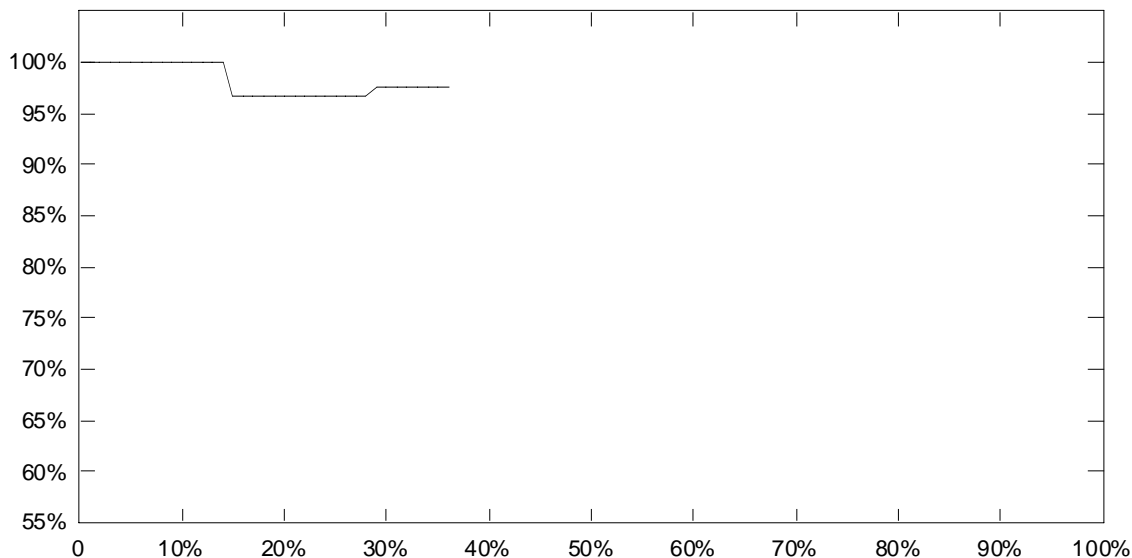

Figure 3. *This graph depicts results for the best selected results of the k-NN Ranking algorithm for k=1,2,3. As k-NN is driven by data, there are not always the same parameters for getting the best results for the EU-Constitution, Iraq, and Terrorism data. In such a case, a suitable trade-off solution has to be used.*

In Figure 3, we also display the results for the experiments using SVM with both positive and negative examples (the three dots in the top-right corner). Surprisingly, SVM here clearly outperforms k-NN even though the negative examples are a random selection of articles from the other two domains, as explained in Section 3, instead of well-chosen negative examples close to the border. SVM with artificially generated negative examples thus outperforms k-NN for our data sets. Its functionality differs, of course, as SVM only classifies documents as being or not being relevant, whereas k-NN provides a degree of relevance, which may be useful when presenting new data to users in a ranked way.

When *trained on only positive* samples, however, SVM's efficiency was quite poor. The best results obtained by the RBF-kernel function, $SVM_{RBF,}$ were slightly below 60% of *precision.* Other kernel functions (linear, polynomial, and sigmoidal) often even gave *precision* results below the baseline, between 45% and 58%, and therefore these results are not shown here. The main reason for this SVM failure probably lies in the fact that the one-class SVM algorithm relies on outliers and noise near the instance-space origin to separate relevant examples from irrelevant ones. The 'Panorama' data sets were compact, with very few outliers or low noise during the training phase. The vectors had relatively few zero entries, therefore they were far from the origin making the separation very difficult. It is interesting that the *Iraq* data gave the best results for the two-class SVM_{RBF} while for *k*-NN it was the most difficult data. The reason was that the *Iraq* data had the largest number (554) of positive and negative training samples which supports the two-class training process well.

Surprisingly, *k*-NN provided the best results for the *EU-Constitution* data, even if the number of (positive) training samples (96) was the lowest. A closer look at the *EU-Constitution* data revealed that the training samples were of very good quality, with very few outliers or noise, focusing just on one very specific topic, while the *Iraq* and *Terrorism* data was more heterogeneous, dealing with broader issues. Therefore, the ranking process gave a more continuous ordered scale for the latter two data groups. For the former data group, there was much more considerable transition between relevant and irrelevant news, so the *recall* values were more stable for the requested high *precision*.

6. Conclusions

We tested different Machine Learning approaches to produce relevance judgments for news data covering three different domains where the pre-condition was that only positive examples were available for training. We tested k-NN and one-class SVM. We also tested classical SVM where we artificially generated negative samples by making an arbitrary selection of articles from the other two domains, respectively. For the document and class vector representation, we furthermore experimented with different feature selection and representation mechanisms. More precisely, we compared results achieved with and without using stop word lists. We tested the usefulness of applying the Porter stemmer to our English news data. Finally, we tested three different feature values : binary (a word is or is not present in the document or class) versus word/stem frequency versus TF.IDF feature values. The priority was to achieve a rather high precision rather than a high recall. Furthermore, we needed to identify all-purpose high-performance parameter settings that could be used by end users for various document domains and training set sizes without any further technical intervention of the scientists and system providers.

The experiments showed that, for the three different news test sets, k-NN produced very satisfactory results : for a *precision* of 90%, the achieved *recall* was in all cases higher than 60%. Although no single parameter settings could be identified as producing best results across all recall levels and domains, we found that a good performance was achieved by setting k to 2 (two nearest-neighbors), by using stop words, and by using word frequency rather than binary or TF.IDF values. Using a stemmer did not improve the results for our English language news documents. While we hope that our insights regarding parameter settings will also hold for languages other than English – users of the *Europe Media Monitor* have to deal with news articles in up to 32 languages – we cannot assume that stemming or lemmatization is not useful for languages that are morphologically more complex than English. The usefulness of lemmatization for other languages must be verified separately for each language.

One-class SVM (SVM trained on positive samples only) clearly performed less well than k-NN with the best parameter settings. However, we found that SVM trained with artificially generated negative samples produced surprisingly good results and even outperformed k-NN.

Whether k-NN or SVM with artificial negative examples should be used depends on the user preferences : k-NN produces *ranked* relevance judgments so that the news items can be ranked starting with the potentially most relevant ones and leaving those to the end that seem to be least relevant ; SVM, on the other hand, simply includes documents into one of the two classes *relevant* or *irrelevant*. With k-NN, users can furthermore decide whether they aim at higher recall or at higher precision and, depending on their preferences, more or less documents can be presented to them. Regarding the amount of training data used for k-NN, it

seems reasonable to assume that there should be at least 100 positive examples, although this clearly depends on how homogeneous the domain of interest is. However, a number much higher than 100 positive samples (e.g. thousands) may be computationally too heavy due to the nonlinear computational complexity of the k-NN algorithm.

It is now planned to produce an efficient and automated implementation of the relevance ranking system and to make it available to those users whose interest profiles are too complex for the currently used filtering system, based on Boolean expressions, term weights and thresholds.

References

Best Clive, van der Goot Erik, de Paola Monica, Garcia Teofilo & Horby David (2002). *Europe Media Monitor – EMM*. JRC Technical Note No. I.02.88. http://press.jrc.it/NewsBrief/

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press.

Cristianini, N., and Schölkopf, B. (2002). Support Vector Machines and Kernel Methods : The New Generation of Learning Machines. *AI Magazine, the Journal of the American Association for Artificial Intelligence*, 2002 : 31- 41.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification.* Second Edition, John Wiley & Sons.

Hroza, J., Žižka, J. (2005a). Mining Relevant Text Documents Using Ranking-Based *k*-NN Algorithms Trained by Only Positive Examples. *Proceedings of the Fourth Czech-Slovak Conference Knowledge-2005*, February 9-11, 2005, Stará Lesná, Slovak Republic : 29-40.

Hroza, J., Žižka, J. (2005b). Selecting Interesting Articles Using Their Similarity Based Only on Positive Examples. *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics CICLing-2005*, February 13-19, 2005, Mexico City, Mexico, Springer-Verlag, 2005, LNCS/LNAI 3406 : 608- 611.

Manevitz, L. R. and Yousef, M. (2001). One-Class SVMs for Document Classification. J*ournal of Machine Learning Research 2 (2001)*, December 2001 : 139-154.

Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill.

Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program 14* (3), 1980 : 130-137.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, Vol. 34, No. 1, March 2002 : 1-47.

Steinberger Ralf, Bruno Pouliquen, Camelia Ignat (2005). Navigating multilingual news collections using automatically extracted information. *Journal of Computing and Information Technology - CIT 13*, 2005, 4 : 257-264. ISSN: 1330-1136.

Van Rijsbergen, C. J. (1979). Information Retrieval, 2nd Edition. Department of Computer Science, University of Glasgow.