Multiaspect Text Categorization Problem Solving: A Nearest Neighbours Classifier Based Approaches and Beyond

Submitted: 20th September 2015; accepted: 26th October 2015

Sławomir Zadrożny, Janusz Kacprzyk, Marek Gajewski

DOI: 10.14313/JAMRIS_4-2015/34

Abstract:
We deal with the problem of the multiaspect text categorization which calls for the classification of the documents with respect to two, in a sense, orthogonal sets of categories. We briefly define the problem, mainly referring to our previous work, and study the application of the k-nearest neighbours algorithm. We propose a new technique meant to enhance the effectiveness of this algorithm when applied to the problem in question. We show some experimental results confirming usefulness of the proposed approach.

Keywords: *text categorization, intelligent system, nearest neighbour classifiers, topic tracking and detection, fuzzy majority*

1. Introduction

An important feature desired for the intelligent systems is the capability to deal with textual information. Despite many efforts and success stories this area still poses many challenges to the research community. *Natural language processing* is an example of a domain where much has been achieved but the machines are still behind a human being and his capability to understand the text in its full meaning. Even the domain of *information retrieval*, setting for itself more modest goals with respect to textual information processing, calls for further research to address the tremendous growth of information to be processed as well as the ambition to assist a human user in tackling with more and more complex problems, so far reserved for a human being. In this paper, we study one of such problems, motivated by some real life applications, and try to propose and extend some well known techniques to deal with it.

Our starting point is the concept of the *multiaspect text categorization* (MTC) which we introduced earlier in a series of papers [9, 22, 23, 25]. The motivation is a real, practical problem of managing collections of documents for the purposes of an organization, notably a public institution which has to be carried out following formal regulations imposed by the state. A part of this problem fits the well-known concept of the *text categorization* (TC) [16] and thus relevant techniques and tools are readily applicable. Another part is, however, more challenging. Although it also can be interpreted as a TC problem, its characteristic makes it a more difficult task – first of all due to a limited number of training documents available but also due to the different motives underlying the grouping of documents.

We have studied the MCT problem in a number of papers, cited above, and proposed some solutions to it. Here we study the use of the k nearest neighbours classifier (k-nn) and propose a new algorithm inspired by this study. The starting point is the study of Yang et al. [20] which is concerning a similar problem of the *topic detection and tracking* (TDT) [1] and proposes some extensions to the basic k-nn algorithm in order to deal properly with the specificity of the problem at hand.

The structure of this paper is the following. The next section briefly introduces the MTC problem. In Section 3 we recall the work of Yang et al. on the use of the k-nn classifier for the purposes of the TDT problem solution and their extensions to the basic algorithm. In subsection 3.2 we present our algorithm inspired by the work of Yang et al. and combining somehow the paradigms of the nearest neighbour classifier and the profile based classifiers [16]. Section 4 shows the results of our computational experiments meant to compare discussed methods and Section 5 concludes and discusses some ideas for the further research.

2. MTC Problem Description

2.1. The Problem

The multiaspect text categorization problem (MTC) may be considered as a twofold standard multiclass single-label classification. Thus, a collection of documents is assumed:

$$D = \{d_1, \dots, d_n\} \tag{1}$$

These documents are, on one hand, assigned to the set of predefined *categories*

$$C = \{c_1, \dots, c_t\} \tag{2}$$

On the other hand, they are also assigned to the sequences of documents, referred to as *cases*, within their own categories. The cases, generally, are not predefined and are established based on the documents arriving to the classification system. We will assume that at the beginning there are some cases already formed. Some of them may be treated as *closed*, i.e., no new document should be assigned to them, and some of them are *on-going*, i.e., they are the candidates for the new documents to be assigned (classified) to. Each document d belongs to exactly one category and one case within this category.

The cases will be denoted as σ and their set as Σ:

$$\sigma_k = < d_{k_1}, \dots, d_{k_l} > \tag{3}$$

$$\Sigma = \{\sigma_1, \dots, \sigma_p\} \tag{4}$$

When a new document d^* arrives it has to be properly added to the collection D, i.e., d^* has to be classified to a proper category and assigned to a proper case within this category. We consider the task of the system as of the decision support type, i.e., a human user should be assisted by the system in choosing a proper category $c \in C$ and a proper a case $\sigma \in \Sigma$ for the document d^* but he or she is responsible for performing these actions. Several ways of assigning documents to categories/cases may be conceived; cf., e.g., [23, 25]. We follow here the line of conduct presented in the latter paper, i.e., of a two stage assignment: first to a category and then to a case. A set of categories is prespecified and each of them may be assumed to be represented by a sufficient number of documents in the collection D. Thus, the standard text categorization techniques may be employed [16]. On the other hand, the cases may be quite short and, moreover, emerge dynamically during the lifetime of document management system. Moreover, it should be assumed that the organization of the documents into categories is based on some top level thematic grouping. For example, in the structure of documents collection of a company one category may comprise documents concerning relations of the company with the public administration institutions, another category may gather documents related to the activity of this company's supervisory board, while still another category may concern all matters related to the human resources. On the other hand, documents are grouped into cases based on some business process they are related to, e.g., hiring a new employee. Thus, documents belonging to the cases within the same category are in general thematically similar and what should decide on assigning a new document to one of them is somehow different from the clue on document assignment to a category. One of the aspects which may be helpful in making the decision on assigning a document to a case is the fact that the documents are arranged within the case in a specific order. This order is based on the logic of the business process related to a given case and reflects the chronology of the running of this process. We assume that the documents arrive for the classification exactly in this order and thus this order may be exploited during the classification.

2.2. Related Work

The MTC problem has been formulated in our previous papers; cf., e.g., [22]. It belongs to the broad class of text categorization problems. Its most similar problem well-known in the literature is the Topic Detection and Tracking (TDT) [1] which may be very briefly described as follows. Topic detection and tracking concerns a stream of news on a set of topics. The basic task is to group together news *stories* on the same *topic*. A story in TDT corresponds to a document in our MTC problem definition while a topic is a counterpart of a case. Categories as such are not considered in the original formulation of the TDT problem although later on the concept of hierarchical TDT has been introduced [8] what brings the TDT and the MTC even closer.

Topics, similarly to cases, are not predefined and new topics have to be *detected* in the stream of stories and then *tracked*, i.e., all subsequent stories concerning this topic should be recognized and properly classified. A subtask of the *first story detection* is distinguished which consists in deciding if a newly arrived story belongs to one of earlier recognized topics or is starting a new topic.

Although the MTC and TDT problems share many points they are still different. In the former, categories and cases are considered while only topics are presumed in the latter (even in the hierarchical TDT, mentioned earlier, the relation between MTC's categories and cases is not reflected as the hierarchy of topics is there meant in the standard text categorization sense, i.e., the categories at different levels of a hierarchy are just themes considered at the different levels of abstraction and do not follow a different principle of grouping such as theme versus business process, as it is assumed for the MTC). Moreover, cases in MTC are sequences of documents while topics in TDT are just sets of stories. Again, even if stories in TDT are timestamped, their succession within a topic is not assumed to carry out any semantic information and the use of this temporal information to solve the tasks of the TDT, if any, is limited to reducing the influence of the older stories on the classification decision. Finally, the practical context is different: for TDT this is the news stories stream analysis while for MTC this is the business documents management. The reader is referred to our forthcoming paper [9] for a more in depth analysis of the relations between the TDT and MTC problems.

The solution approaches to the TDT problem belong to the mainstream information retrieval. Standard representation, most often in the framework of the vector space model, is assumed for the stories. The notion of the similarity/dissimilarity of stories represented as vectors in a multidimensional space is employed to detect and track topics. Often, various cluster analysis techniques are used to group stories, interpret the clusters as topics and represent them by the centroids of these clusters. A new story is compared against the centroids of particular topics to decide where it belongs. If there is no centroid similar enough then a new topic is established and the newly arrived document is assigned to it.

In our previous papers we have proposed a number of solutions to the MTC problem. We also most often adopt the vector space model as the starting point. The matching of a document and a case was computed as the weighted average of fuzzy subsethood degrees of the fuzzy set representing the document to a fuzzy set representing a given case and fuzzy set representing the category of this case, respectively. This way, the assignment of a document to a case for which the highest matching was obtained, which implied also the assignment to its category, was based on the combination of the matching of this document with respect to the case as well as to the whole category.

Then we propose to model the cases and proceed with the classification of the documents in the framework of the hidden Markov models and sequence mining [22], using the concepts of the computational intelligence [23], or employing the support vector machines [24]. We also pursued other paths, including semantic representation of documents, finding a parallel of the MTC with text segmentation, studying the asymmetry of similarity [13, 14], devising new cluster analysis techniques [11] or investigating the applicability of the concepts related to the coreference detection in data schemas [17].

In this paper we follow the line of research on combining some approaches related to the classification task and computational intelligence tools to propose a new approach and also study the applicability of the well known techniques to the problem of the the multiaspect text categorization.

3. The Techniques Employed

3.1. Basic k-nn Technique and Its Extensions to Topic Tracking

In this paper we study the use of the k-nearest neighbours technique (k-nn) to solve the multiaspect text categorization problem. From the point of view of the statistical pattern classification theory, this method belongs to the group of the nonparametric techniques. This type of approaches seems to be most promising for the task at hand due to a limited set of assumptions which have to be adopted to apply them. One of the characteristic MTC features is the sparse training data present and thus, e.g., assuming a specific family of (conditional) probability distributions of data and estimation of its parameters may be difficult, if possible at all. The k-nn technique proved to be effective for many different classification tasks, including the topic tracking and detection problem [20] which is closely related to our MTC problem as discussed in section 2.2.

Basically, the k-nn technique may be described in the context considered here as follows. A set of categories C and a set of training documents D (cf. section 2.1), for which the category assignment is known, are assumed. For a new document d^* to be classified the k most similar to it documents in D are found. The similarity measure is usually defined as the inverse of some distance measure; usually the Euclidean distance is adopted. The category to which the majority of the k closest documents belong to is assigned to d^*. Formally, using the notation introduced in (1)-(4), the category c^* assigned to the document d^* is defined as follows:

$$c^* = \arg\max_{c_i} |\{d \in D : (\text{Category}(d) = c_i) \wedge (d \in \mathbf{NN}_k(d^*))\}| \quad (5)$$

where Category(d) denotes the category $c \in C$ assigned to a training document d and $\mathbf{NN}_k(d^*)$ denotes the set of k documents $d \in D$ which are the closest to d^*, i.e.,

$$\mathbf{NN}_k(d^*) = \{d_{\delta(1)}, d_{\delta(2)}, \ldots, d_{\delta(k)}\}$$

where δ is such a permutation of the set $\{1, \ldots, n\}$ that $d_{\delta(j)}$ is j-th most similar to d^* document in the set D.

The k-nn is a very popular classifier, often used also in the context of the text categorization, cf., e.g., [10, 19]. An inspiration for our work is in particular the paper by Yang et al. [20] on the application of the k-nn technique for the purposes of the topic tracking and detection. The authors adopt standard document representation of the stories within the framework of the vector space model, using a variant a variant of the tf x IDF keyword weighting schema.

Yang et al. study in particular the use of the k-nn for the solution of the topic tracking problem of the TDT (cf. section 2.2). They proposed some modifications to the basic algorithm which proved to yield better results on some benchmark datasets. Namely, in [20] the following improvements to the basic k-nn algorithm have been proposed. First of all, instead of the multiclass problem they consider t binary classification problems, one for each category $c \in C$ (cf. (2)). Moreover, instead of simply counting the number of documents belonging to that category among k most similar documents $\mathbf{NN}_k(d^*)$ they compute the following index, called kNN.sum (we slightly modify the original notation used in [20] to adjust it to the context of our MTC problem):

$$r(d^*, c, k, D) = \sum_{d \in P_k^c} sim(d^*, d) - \sum_{d \in Q_k^c} sim(d^*, d) \quad (6)$$

where $P_k^c = \{d \in \mathbf{NN}_k(d^*) : Category(d) = c\}$, i.e., it is a subset of the set of documents most similar to d^* which belong to the category c (are positive), $Q_k^c = \mathbf{NN}_k(d^*) \setminus P_k^c$, i.e., it is a subset of documents being negative examples with respect to the category c, and $sim(d^*, d)$ denotes the similarity measure between the documents which is assumed to be the cosinus of the angle between the vectors representing documents in question. Then, the document is assigned to a category for which this index is the highest, provided it exceeds some threshold value (otherwise document is treated as starting a new topic). From this point of view this approach is a kind of the weighted k-nn technique [6].

Yang et al. notice the difficulty in setting an appropriate value of the parameter k. In case of the TC problem, i.e., when there is usually a large enough number of positive examples for each category, an experimentally verified recommended value for k is rather large, higher than 30 and less than 200. When the number of positive examples for a given category is small, as it is the case for topic tracking in TDT or assigning a document to a case in our MTC, this recommendation is not valid. If k is high the set $\mathbf{NN}_k(d^*)$ will be dominated by negative examples as their number in D is much greater than the number of positive examples. However, also choosing k low may lead to the set $\mathbf{NN}_k(d^*)$ comprising only negative examples unless the document to be classified d^* is very similar to some positive examples. Yang et al. [20] proposed to overcome this difficulty introducing modified versions of the index (6).

The first version is called kNN.avg1 and is defined as follows:

$$r'(d^*, c, k, D) = \frac{1}{|P_k^c|} \sum_{d \in P_k^c} sim(d^*, d) - \frac{1}{|Q_k^c|} \sum_{d \in Q_k^c} sim(d^*, d) \quad (7)$$

where $|\cdot|$ denotes the cardinality of a set. In this case the similarity to the positive and to the negative examples is averaged and thus even for a large k the dominance of the negative examples in the neighbourhood of the classified document d^* does not pose a problem.

The second modified version of the kNN.sum technique (6) is called kNN.avg2 and is defined as follows:

$$r''(d^*, c, k, D) = \frac{1}{|U_{k_p}^c|} \sum_{d \in U_{k_p}^c} sim(d^*, d) - \frac{1}{|V_{k_n}^c|} \sum_{d \in V_{k_n}^c} sim(d^*, d) \quad (8)$$

In this case k_p positive examples, i.e., belonging to the category c, most similar to d^*, and k_n negative examples most similar to d^* are considered and they form the sets $U_{k_p}^c$ and $V_{k_n}^c$, respectively. Similarity between d^* and the documents from these two sets is averaged as in case of kNN.avg1. Thanks to that a small number of the nearest training examples may be taken into account and there is no risk that the negative examples will dominate. The kNN.avg2 techniques gives a higher flexibility making it possible to independently choose the parameters k_p and k_n but these two parameters have to be tuned instead of just one k as in kNN.avg1. When the kNN.avg2 is going to be applied to the MTC problem there is a risk that there are not enough positive examples, i.e., their number is lower than the value of k_p. In particular, if c corresponds to a very short case and is considered as a candidate for the assignment of d^* then for the reasonable value of k_p it may easily happen. In such a situation, our implementation of the kNN.avg2 reduces the value of k_p to the number of existing positive examples.

Yang et al. [20] offer some recommendations as to the tuning of the parameters k or k_p and k_n. Their main concern is a limited number of training documents making challenging the usual splitting of the training data set into a genuine training set and a validation set. Thus, they devise the tuning in the framework of an ensemble of tested classifiers comprising kNN.avg1 and kNN.avg2 as well as a Rocchio type classifier belonging to the class of the profile based classifiers [16]. The details can be found in [20]. In the current paper we test a more standard way of tuning the parameters and we check how effective it is; cf. the next section 3.2.

3.2. Our Approach to Improving the k-nn Classifier for the Purposes of the MTC Problem Solution

In our previous work [25] on the solution of the MTC problem we already proposed to use the k-nn technique for the first stage of the solution, i.e., deciding on the category to which the newly classified document d^* belongs as well as to the second stage, i.e., assigning the document d^* to a case. Here we study some extensions to the basic k-nn procedure and compare them experimentally with the approaches proposed by Yang et al. [20] which are presented in section 3.1.

In the TDT problem, more specifically the topic tracking problem, considered by Yang et al. [20] the documents related to a topic are assumed to arrive over some time but the order in which they come is not essential. Namely, some documents may describe exactly the same aspect of a topic/event but come from different sources and thus it should not be expected that the order in which they appear on the input carries out some information useful for their classification. In the MTC problem the situation is different and the order of the documents within a case may be assumed to convey some extra information which may be exploited for their proper classification. In [25] we have shown that the documents can be quite successfully classified to the cases by their comparison just to the last document of candidate cases. Thus, it seems to confirm that the similarity to the most recent documents in a case should influence most the classification decision. This observation reminds what have been confirmed in some computational experiments on the comparison of the path in the tree of an XML document [17]: that the similarity of the last segments of these paths is decisive for establishing the coreference of respective XML elements.

Here, we develop this idea and propose to take into account during the comparison all documents belonging to a candidate case but with different weights: the closer to the end of the case given document is located the higher its weight is. Formally, we propose to use the following index to evaluate the matching of the document d^* against a candidate case $\sigma \in \Sigma$. Let us first cast it in a strict but linguistically expressed form as the truth value of the following *linguistically quantified proposition* [12, 21]:

The document d^* is similar to most of the *important* documents of the case σ (9)

According to the Zadeh's calculus of linguistically quantified propositions the truth value of the proposition (9) for a document to be classified d^* and a candidate case σ is computed as follows:

$$m(d^*, \sigma) = \mu_Q \left(\frac{\sum_{d \in \sigma} \min(sim(d^*, d), imp(d))}{\sum_{d \in \sigma} imp(d)} \right) \quad (10)$$

where $sim(\cdot, \cdot)$ denotes a similarity measure between documents and $imp(\cdot)$ denotes the importance of the document d belonging to the case σ. The linguistic quantifier Q in (10) represents the concept of linguistically expressed majority, exemplified in (9) with the word "most". Such a quantifier may be formally represented in many different ways (cf., e.g., [5]) and in what follows we adopt the original Zadeh's approach [21]. Thus, a linguistically quantified proposition fits one of the generic templates:

$$QX's \text{ are } A, \text{ or} \tag{11}$$
$$QBX's \text{ are } A \tag{12}$$

and expresses that, e.g., for $Q = most$, "most of the elements of a universe X possess a property A", in case of (11), or that "most of the elements of a universe X possessing a property B possess also a property A", in case of (12). Properties A and B are in general fuzzy and are represented by their membership functions defined on the universe of discourse X. A linguistic quantifier Q is formally represented as a fuzzy set in the interval [0,1]. For example, the membership function of $Q = most$ may be expressed as follows:

$$\mu_Q(x) = \begin{cases} 1 & \text{for } x \geq 0.8 \\ 2x - 0.6 & \text{for } 0.3 < x < 0.8 \\ 0 & \text{for } x \leq 0.3 \end{cases} \tag{13}$$

The value of the membership function $\mu_Q(x) = y$ is interpreted as meaning that if $x * 100\%$ of elements of X possess the property A then the truth value of (11) is equal y, or that if $x * 100\%$ of elements of X possessing the property B possess also the property A then the truth value of (12) is equal y.

General formulae for the truth value of (11) and (12) are thus the following, respectively:

$$\text{truth}(QX's \text{ are } A) = \mu_Q\left(\frac{\sum_{x\in X}\mu_A(x)}{n}\right) \tag{14}$$

$$\text{truth}(QBX's \text{ are } A) = \mu_Q\left(\frac{\sum_{x\in X}\min(\mu_A(x),\mu_B(x))}{\sum_{x\in X}\mu_B(x)}\right) \tag{15}$$

Notice that the definition of our indicator of matching between the document d^* and a case σ, as expressed with (9), may be rephrased as:

Most of the *important* documents of σ are *similar* to d^* (16)

and thus fits the general template (*a protoform*) (12) [12]. It may be also easily seen that the formula (10) is an instantiation of the formula (15) where X is a set of documents belonging to the case σ (treated in the following formulae as a *set* of documents), A is a fuzzy property of a document $d \in \sigma$ with the membership function:

$$\mu_A : \sigma \rightarrow [0,1]$$
$$\mu_A(d) = sim(d, d^*)$$

and the fuzzy property B corresponds to the importance of document d with respect to the case σ, i.e.:

$$\mu_B : \sigma \rightarrow [0,1]$$
$$\mu_B(d) = imp(d)$$

The index (10) introduced here is used to assign new document D^* to a case in a straightforward way (we assume that before that d^* is assigned to a category c using, e.g., the basic k-nn algorithm, as in our previous paper [25]):

1) the matching index m defined by (10) is computed for all candidate (on-going) cases belonging to category c; the set of such cases is denoted as Σ_c,
2) the document d^* is assigned to the case σ^* such that:
$$\sigma^* = \arg\max_{\sigma_i \in \Sigma_c} m(d^*, \sigma_i)$$

Thus, our approach may be treated as another way of using the k-nn technique for classification of documents, although to some extent it may be also interpreted as a kind of a profile-based classification [16]. We employ the weighted similarity of the document d^* with respect to the documents of a case – similarly as the kNN.avg1 and kNN.avg2 are doing. However, we compute the *weighted average* and with respect to *all* the training documents comprising a particular candidate case. Moreover, in our approach two different types of weights are involved: one related to the similarity $sim(d, d^*)$ and another one related to the importance $imp(d)$ of a document within a case.

In order to use effectively the introduced index (10) we need to devise the way to set its parameters, i.e.:
1) the form of the quantifier Q,
2) the form of the similarity measure sim used therein,
3) the importance weights assigned to particular documents of a case.

Concerning the linguistic quantifier employed, the very nature of the proposed index m (10) suggests the use of the quantifier expressing the concept of the (fuzzy) majority, such as "most". More generally a so-called *Regular Increasing Monotonic (RIM)* quantifier should be used [18], i.e., one with the monotone increasing membership function μ_Q, such as, e.g., (13), i.e.:

$$\forall x, y \quad x < y \Rightarrow Q(x) \leq Q(y) \tag{17}$$

Thus, the choice of a specific quantifier seems to be of a limited importance. The replacement of a linguistic quantifier Q_1 with Q_2 in (10) may change the assignment of a document to the case only due to the assumed weak monotonicity of the membership function μ_Q (cf. (17)), i.e., if $\mu_{Q_i}(x) = \mu_{Q_i}(y)$ and $\mu_{Q_j}(x) < \mu_{Q_j}(y)$, where $i, j \in \{1,2\}$ and $x < y$. Thus, we assume the *unitary linguistic quantifier* in (10), i.e., defined by the membership function:

$$\mu_Q(x) = 1, \forall x \in X$$

It has to be noted that the choice of a linguistic quantifier may play a more important role if a threshold value of the index (10) is set and meant to decide if document d^* may be assigned to a given case or should start a new case, i.e., when the first story detection problem is considered. However, this goes beyond the scope of this paper.

The similarity measure sim in (10) may be defined in many ways. In our previous work we are most often using the Euclidean distance between the vectors representing documents under comparison. Here we adopt it also and take its complement as the measure of the similarity. We assume the vectors representing documents to be normalized in such a way that their Euclidean norms are equal 1. Thus, the highest possible Euclidean distance between two vectors representing documents equals $\sqrt{2}$ and the similarity between two documents $d = [d_1, \ldots, d_l]$ and $d^* = [d_1^*, \ldots, d_l^*]$, assuming the number of keywords used to represent the documents to be equal l, may be expressed as follows:

$$sim(d, d^*) = \frac{\sqrt{2} - \sqrt{\sum_{i=1}^{l}(d_i - d_i^*)^2}}{\sqrt{2}} \tag{18}$$

$sim(d, d^*) \in [0, 1]$.

Finally, the importance of each document in a case is assumed to be an increasing function of the position of the document in the case. The extreme cases are the following:

- important is only the last document in the case; cf. our paper [25] studying this case;
- all documents are equally important, i.e., effectively the importance is not taken into account

In the current paper we consider and test experimentally the following options (x denotes the position of a document in the case, len denotes the length of this case, and a and b are the parameters):

1) linear importance:

$$imp(x) = \frac{x}{len} \tag{19}$$

2) quadratic importance:

$$imp(x) = a * (\frac{x}{len})^2 + 1 - a \tag{20}$$

3) radical importance:

$$imp(x) = a * \sqrt{\frac{x}{len}} + 1 - a \tag{21}$$

4) piecewise linear importance:

$$imp(x) = \begin{cases} 0 & \text{if } x \leq a \\ (x-a)/(b-a) & \text{if } a < x \leq b \\ 1 & \text{if } x > b \end{cases} \tag{22}$$

All options assume that the importance degree of the last document of the case, i.e., the one most recently assigned to this case, is equal 1, i.e., is the highest. Also all are monotone: documents located later in a case get not lower importance than those located earlier.

The first option is the simplest one making the last document most important and gradually reducing the importance of earlier documents with the constant rate. No parameters have to be set. The second option, the quadratic importance, for high values of the parameter $a \in [0, 1]$ relatively, with respect to the linear importance, reduces the importance of the documents behind the last document of the case. This reduction is highest for the documents located in the middle of the case. For small values of a the importance of documents in the case is increased relative to the linear importance. This increase is highest for the documents at the beginning of the case. The radical importance, for any value of the parameter $a \in [0, 1]$ increases importance of all documents in the case. The smaller the value of a the higher is this increase. The increase is relatively highest for the documents located at the beginning of the case. Finally, the piecewise linear importance makes it possible to set the importance of the documents located at the beginning of a case to 0 what effects in ignoring them during the computation of the index (10). At the same time, the documents located closer to the end of the case can get importance degree equal 1 - the highest importance degree is thus not anymore reserved for the last document of the case. This option requires setting of two parameters a and b which decide what proportion of the documents will get the importance degree equal 0 and 1, respectively.

An option for the importance degree may be chosen based on the experience of the user or may be tuned using the training dataset. In our experiments reported in section 4 we follow the latter way.

3.3. Oversampling of Short Cases to Circumvent the Imbalance of Classes

In the previous section we have introduced a variant of the k-nn method which is meant to better account for the relation of the subsequent documents in a case. This is a rather far going modification of the original algorithm which, in a sense, replaces the comparison of the document to be classified d^* against training documents with the comparison of d^* against the whole cases with a proper account for the sequential character of documents forming a case. In this section we propose another, modest, modification of the original k-nn algorithm which is expected to improve its working for the MTC problem.

Namely, usually when a document d^* is going to be assigned to a case, particular candidate (on-going) cases are of different length. Some have just started and comprise a small number of documents while other are already well developed and may comprise tens of documents. Thus, when each case is treated as a separate class then we have usually to deal with the classes of imbalanced sizes in the training dataset. We propose to duplicate documents of short candidate cases thus increasing their visibility during the execution of the regular k-nn method and its variants described earlier, including our approach presented in section 3.2.

Formally, a threshold $caselen$ is set and all cases in a given category whose length is shorter that this threshold are replicated. Several strategies may be applied: all documents of such a short case may be replicated the same number of times or the number of replicated copies may depend on the location of the document within the case. Following the similar reasoning as in section 3.2 we may put more emphasis on the most recent documents in the case and replicate them more times. In our experiments in section 4 we try a few variants.

This technique is basically meant for the original k-nn algorithm. Its variants proposed by Yang et al. [20] and described in section 3.1 are resistant somehow to this problem thanks to averaging of the similarity (and dissimilarity) over the documents neighbouring the document d^*. It is also easy to notice that oversampling corresponds to considering the importance in our approach presented in section 3.2 and eventually boils down to the weighted averaging of the similarities of documents neighbouring d^*.

In the experiments discussed in the next section we employ the oversampling in case of the regular k-nn technique to compare its effectiveness with the effectiveness of the apparently more sophisticated methods discussed in sections 3.1 and 3.2.

4. Computational Experiments

4.1. Data and Software Used

The are no benchmark datasets yet for the multiaspect text categorization problem dealt with here. In our work we are using a collection of papers about computational linguistics which have been structured using XML and made available on the Internet as the ACL Anthology Reference Corpus (ACL ARC) [4]. In our experiments we use 113 papers forming a subset of the ACL ARC. We group papers to obtain categories (cf. section 2). After some trials we decided to look for 7 categories (clusters) using the standard k-means clustering algorithm. The clustering is applied to the papers represented according to the vector space model (cf., e.g., [2]) ignoring the XML markup. In particular, the following operations are executed to obtain the final representation (cf. also our earlier papers, e.g., [25]). The text of the papers is normalized, i.e., the punctuation, numbers and multiple white spaces are removed, stemming is applied, the case is changed to the lower case, stopwords and words shorter than 3 characters are dropped. The document-term matrix is created for the whole set of the papers using *tf* $\times$ *IDF* terms weighting scheme. Next, the keywords present in less than 10% of the papers are removed from the document-term matrix. The vectors representing particular papers are normalized by dividing each coordinate by the Euclidean norm of the whole vector and thus the Euclidean norm of each vector equals 1.

Next, we produce a set of cases based on the papers, in the following way. The papers are originally partitioned into sections (segments) and each section forms the content of the XML element `Section`. We treat each paper as a case while its sections are considered to be documents of this case, preserving their original order within the document. This way we obtain a collection of 113 cases comprising 1453 documents, cf. also [22–25]. The documents, i.e., the sections of the original papers, are represented using the vector space model. Thus, again the operations such as the punctuation, numbers and multiple white spaces removal, stemming, changing all characters to the lower case, stopwords and words shorter than 3 characters elimination are applied to the documents. A document-term matrix is constructed for the above set of documents using *tf* $\times$ *IDF* terms weighting scheme. Again, sparse keywords appearing in less than 10% of documents are removed from this matrix. and as a results 125 keywords are used to represent the documents. The vectors representing documents are normalized in the same way as in case of the papers, i.e., their Euclidean norm equals 1.

The dataset obtained this way is then split into the training and testing datasets. To this aim, a number of cases are randomly chosen as the on-going cases which are thus the candidate cases for the document d^* to be assigned to. In each on-going case a cut point is selected randomly: the document located at the cut point and all subsequent documents are removed from the case and serve as the testing dataset. All remaining documents from the collection serve as the training dataset.

All computations are carried out using the R platform [15] with the help of several packages. In particular, the text processing operations are are implemented using the `tm` package [7]. The `FNN` package [3] is employed to classify documents to cases with the use of the original k-nn algorithm. The algorithms mentioned in sections 3.1 and 3.2 are implemented ourselves in the form of an R script.

4.2. The Goals of the Experiments and How the Parameters Are Chosen

Our goal is to compare the effectiveness of the basic k-nn algorithm and its variants discussed in sections 3.1 and 3.2 for the solution of the MTC problem presented in section 2, and in particular, of its second stage consisting in assigning the document d^* to the proper case. Of a special interest is, of course, our approach presented in section 3.2. Thus, we assume here the representation of the collection of documents described in section 4.1 and we assume the two-stage approach with two stages consisting in assigning the document d^* to a category and a case within this category, respectively.

We run a number of experiments and based on the results of each run we evaluate the effectiveness of the assignment of documents to cases. As the evaluation of the effectiveness of the assignment we use the microaveraged recall, i.e.,

$$accuracy = \frac{\text{number of documents properly assigned to their cases}}{\text{number of all documents being classified}} \tag{23}$$

The cardinalities of the sets of test documents belonging to particular cases do not differ extremely and thus microaveraging seems to be the measure best illustrating the quality of particular classification algorithms under consideration.

Each experiment consists in choosing on-going cases and classifying all documents located behind the cut points in these cases.

An important aspect of the successful application of the classifier is the question of tuning of its parameters. Thus in the first series of our experiments we tune the parameters of the five classifiers under comparison (cf. section 3):

1) the regular k-nn algorithm with the parameter tuned to be k;
2) the kNN.avg1 technique proposed by Yang et al. [20] with the parameter tuned to be k;
3) we consider also a simplified variant of kNN.avg1, cf. r' given by (7), which may be expressed as follows:

$$r'_0(d^*, c, k, D) = \frac{1}{|P^c_k|} \sum_{d \in P^c_k} sim(d^*, d) \qquad (24)$$

   i.e., instead of taking account the similarity of the document d^* to the closest, both, positive and negative documents of the training data set, as r' does, the index r'_0 takes into account only the closest positive neighbours. The simplified variant is more in the spirit of the basic k-nn technique and we would like to check if taking into account also the similarity of d^* with respect to the negative examples really increases the effectiveness of the classification; here again tuning concerns choosing the value of k,
4) the kNN.avg2 technique proposed by Yang et al. [20] with the parameters tuned to be k_p and k_n; for this technique the variant simplified along the lines proposed for kNN.avg1 may be also of interest but for large k's it becomes identical with r'_0 introduced above, and in our preliminary tests it turned out to be inferior to the original kNN.avg2 technique.
5) our approach given by (10) with the parameter tuned to be the importance function imp.

Tuning the Parameters k One may choose and fix the parameters values based on his or her experience, some extra knowledge concerning the characteristic of the problem at hand, or on historical data. It is also possible to dynamically and automatically choose the parameters values each time a classification decision is to be made, again based on the available data. In the first series of experiments we check if such a dynamic tuning really helps in comparison to using fixed value for the parameter k. We compare the results obtained for all five algorithms under consideration: basic k-nn, kNN.avg1 and its modified variant defined by (24), and kNN.avg2, with the tuning of the parameter k and without tuning it, using fixed values $k = 1, 5,$ and 10 for k-nn, and $k = 5,$ and 10 for the kNN.avg1 and its modified variant (for $k = 1$ these two latter methods coincide with the 1-nn). For kNN.avg2 all 9 combinations for $k_p, k_n \in \{1, 3, 5\}$ are investigated.

An option to consider for the dynamic tuning is which part of the data set to use. Basically, it should be a separate validation data set. However, it is difficult to obtain due to the limited size of classes (cases) and the requirement that the training data set have to be formed of the prefixes of the cases (i.e., original cases up to a cut off point). Thus, for the purposes of the tuning we have employed training and testing datasets formed as for the original dataset but assuming that all the cut off points in the on-going cases are one position earlier than in the original dataset (if such a new cut off point happens to be the first position of the case then such a case is not used during the tuning). Then, we compared two tuning procedures, both based on adopting subsequent values of k from the interval [1,10] and checking if the testing documents are assigned to proper cases but differing in the number of testing documents taken into account. In a simpler procedure the testing set comprises only documents located at the new cut-off points while in the second procedure also the preceding documents are used - down to the document located at the second position in a given case. The former procedure, to be called *simple* in what follows, may benefit from employing for tuning a dataset most similar – in terms of the length of cases and their content – to the actual test dataset. This procedure is also cheaper computationally as less documents are classified. The latter procedure, to be called *complex* in what follows, may better reflect the properties of the cases belonging to a given category but is more expensive.

In Tables 1-4 we show the results of the tuning of the parameter k for all the methods under comparison (or k_p and k_n in case of the kNN.avg2 technique). Table 1 shows that for the basic k-nn technique the best results are obtained for fixed value of k equal 1 and for the simple tuning procedure. The former is however much cheaper computationally and thus we will use k fixed to 1 for the basic k-nn in our further comparisons with other techniques discussed in this paper. In case of the kNN.avg1 technique the results obtained for various values of k are more uniform, as shows Table 2 (k=1 is omitted as the method then coincides with 1-nn). This seems to be the effect of the averaging employed by the kNN.avg1 technique. For further comparisons we choose the complex tuning procedure. The same happens for our simplified version of the kNN.avg1 technique and we again choose the complex tuning. In case of the kNN.avg2 technique the best results are obtained for largest tested number of k_p, i.e., $k_p = 10$ (in some extra tests for even higher values of k_p we have not obtained better results). The value of k_n does not make much difference so we choose the following setting $(k_p, k_n) = (10, 1)$ for further comparisons.

Tab. 1. The averaged results of 100 runs of the basic k-nn algorithm for the following fixed values of k: 1, 5, 10, and for the values tuned using the simple and complex procedures. First row shows the mean value of the accuracy over all the runs while the second row shows the standard deviation

k				
1	5	10	simple	complex
0.6338	0.5186	0.4566	0.6077	0.5741
0.0656	0.0607	0.0524	0.0641	0.0656

Tab. 2. The averaged results of 100 runs of the kNN.avg1 algorithm for the following fixed values of k: 5, 10, and for the values tuned using the simple and complex procedures. First row shows the mean value of the accuracy over all the runs while the second row shows the standard deviation

k			
5	10	simple	complex
0.6079	0.5961	0.6164	**0.6196**
0.0564	0.0582	0.0599	0.0572

Tab. 3. The averaged results of 100 runs of the modified (simplified) kNN.avg1 algorithm for the following fixed values of k: 5, 10, and for the values tuned using the simple and complex procedures. First row shows the mean value of the accuracy over all the runs while the second row shows the standard deviation

k			
5	10	simple	complex
0.6150	0.6020	0.6307	**0.6329**
0.0596	0.0634	0.0615	0.0553

Choosing the Importance Function This parameter applies only to our technique proposed in section 3.2. We have to choose one of the importance functions (19)-(22). Besides choosing the very function, with exception of the linear importance, we have also the freedom to choose its parameters. In case of the quadratic and radical functions (20)-(21) there is only one parameter $a \in [0,1]$ which we sample every 0.1. In case of the piecewise importance two parameters $a, b \in [0,1]$ are to be selected.

In our experiments we are tuning this parameter comparing some fixed settings and the dynamic tuning procedure (using the simple procedure as described earlier in case of tuning the parameter k for the other techniques under comparison). The fixed settings are the following:
1) the linear importance (19),
2) the quadratic (20) and radical (21) importances with the parameters (a, b) set to $(0.1, 0.9)$, $(0.5, 0.5)$, $(0.9, 0.1)$ each
3) the piecewise linear importance with the parameters (a, b) set to $(0.0, 0.5)$, $(0.3, 0.8)$, $(0.5, 1.0)$

The dynamic tuning checks the whole space of the possible settings for the importance function. In particular, all four importance functions are taken into account and tested on the test dataset formed following the *simple* procedure, i.e., making the cut-off points one position earlier than in the original test data set and using only the documents located at these new cut off points for tests. During this testing the linear importance does not need any parameters, the quadratic and radical importance functions with all the pairs from $U = \{(a, b) : a = 0.1, 0.2, \ldots, 1.0 \text{ and } b = 1 - a\}$ are tested while in case of the piecewise importance all the pairs from $U = \{(a, b) : a = b - 1.0, b - 0.9, \ldots, b - 0.1 \text{ and } b = 0.1, 0.2, \ldots, 1.0\}$ are tested. The combination of the parameters, i.e., the importance function together with the parameters a and b setting, where applicable, is chosen for the actual classification o a newly arrive document.

Table 5 shows the results of the tuning of the importance function parameter. Several combinations give equally good results. Also using our approach with no importance, what is equivalent to assigning highest importance of 1.0 to all documents of a case in question, yields good results. In the latter case, the index (10) underlying our approach boils down to averaging the similarity of a document to classify over all documents of a candidate case, what makes it close to the kNN.avg1 and kNN.avg2 techniques of Yang et al. [20]. For the further comparisons of our approach with respect to the other techniques we choose the radical importance function with the parameters $(a, b0$=(0.5, 0.5) what corresponds to the importance function $imp(x) = 0.5\sqrt{\frac{x}{len}} + 0.5$.

All five techniques under consideration employ a similarity measure. In our experiments we adopted the Euclidean distance, also for kNN.avg1 and kNN.avg2 which are originally defined in [20] with the use of the cosine measure. We leave the experiments with other similarity measures for the future research.

Oversampling Variants In section 3.3 we propose to to use the oversampling of documents from short cases in the training data set to remedy the imbalance of cases sizes. In our experiments we have applied the oversampling to the cases of length lower than 3, i.e., effectively the only the cases in the training data set comprising one or two documents are affected. We have tested the following three variants:

over1 in which the oldest documents in the case is oversampled more, i.e., effectively the first document in a short case is tripled while the second (if exists) is doubled,

over2 in which the newest documents in the case is oversampled more, i.e., effectively the first document in a short case is doubled while the second (if exists) is tripled; this is a strategy in-line with our general assumption that the newest documents, located closer to the end of the case, matter the most for the successful classification,

over3 in which all documents in short cases are equally oversampled; effectively the first and the second document in a short case are doubled.

4.3. The Results

After choosing the fixed parameters or their dynamic tuning, as described earlier, we finally compare the effectiveness of the techniques discussed in sections 3.1 and 3.2. In the Table 6 we show the accuracy of the 20 following variants of the earlier discussed algorithms, averaged over 200 runs:
1) basic 1-nn technique,
2) basic 5-nn technique

***Tab. 4.** The averaged results of 100 runs of the kNN.avg2 algorithm for the following fixed values of (k_p, k_n): (1,1), (1,5), (1,10), (5,1), (5,5), (5,10), (10,1), (10,5), (10,10), and for the values tuned using the simple and complex procedures. First rows show the mean value of the accuracy over all the runs while the second rows show the standard deviation*

(k_p, k_n)					
(1,1)	(1,5)	(1,10)	(5,1)	(5,5)	(5,10)
0.6264	0.6339	0.6307	0.6552	0.6534	0.6495
0.0562	0.0565	0.0570	0.0564	0.0569	0.0564
(k_p, k_n)					
(10,1)	(10,5)	(10,10)	simple	complex	
0.6682	0.6639	0.6614	0.6429	0.6411	
0.0546	0.0533	0.0551	0.0548	0.0571	

***Tab. 5.** The averaged results of 100 runs of our algorithm for the following fixed choices of the importance function: linear, quadratic with (a, b) = (0.1, 0.9), (0.5, 0.5), (0.9,0.1), radical with a = (a, b) = (0.1, 0.9), (0.5, 0.5), (0.9,0.1), piecewise with (a, b) = (0.0, 0.5), (0.3, 0.8), (0.5, 1.0), dynamically tuned, and with importance identically equal 1.0 (no importance). First rows show the mean value of the accuracy over all the runs while the second rows show the standard deviation*

Importance functions					
linear	quadratic			radical	
	(0.1,0.9)	(0.5,0.5)	(0.9,0.1)	(0.1,0.9)	(0.5,0.5)
0.6332	0.6545	0.6577	0.6164	0.6532	**0.6595**
0.0599	0.0651	0.0593	0.0640	0.0642	0.0623
Importance functions					
radical	piecewise			tuned	no importance
(0.9,0.1)	(0.0,0.5)	(0.3,0.8)	(0.5,1.0)		
0.6557	0.6525	0.5888	0.5368	0.6279	0.6518
0.0609	0.0610	0.0624	0.0663	0.0653	0.0636

3) kNN.avg1 with dynamically tuned value of k using complex tuning
4) kNN.avg1 with k fixed and equal 5
5) the simplified version of the kNN.avg1 technique with dynamically tuned value of k using complex tuning
6) the simplified version of the kNN.avg1 technique with k fixed and equal 5
7) kNN.avg2 with (k_p, k_n) fixed and set to (10,1)
8) our algorithm presented in section 3.2
9) 5-nn with oversampling in variant 1 (cf. section 3.3)
10) kNN.avg1 with k fixed and equal 5 and with oversampling in variant 1
11) the simplified version of the kNN.avg1 technique with k fixed and equal 5 and with oversampling in variant 1
12) kNN.avg2 with (k_p, k_n) fixed and set to (10,1) and with oversampling in variant 1
13) 5-nn with oversampling in variant 2
14) kNN.avg1 with k fixed and equal 5 and with oversampling in variant 2
15) the simplified version of the kNN.avg1 technique with k fixed and equal 5 and with oversampling in variant 2
16) kNN.avg2 with (k_p, k_n) fixed and set to (10,1) and with oversampling in variant 2
17) 5-nn with oversampling in variant 3
18) kNN.avg1 with k fixed and equal 5 and with oversampling in variant 3
19) the simplified version of the kNN.avg1 technique with k fixed and equal 5 and with oversampling in variant 3
20) kNN.avg2 with (k_p, k_n) fixed and set to (10,1) and with oversampling in variant 3

Our goal was to compare our method (10) with other techniques, check how the simplified version of the kNN.avg1 compares with its original form, check if oversampling discussed in section 3.3 increases the effectiveness of the techniques to which it is applicable and if there is a difference between its variants. Of course, as the test has been executed on one dataset any far going conclusions are not fully justified.

The best results are obtained for the kNN.avg2 (for all parameters tested). Our approach produces slightly, but statistically significant, worse results according to the paired Wilcoxon signed-rank test at 0.05 significance level (we use this statistical test in what follows, too). In particular, in 108 runs out of 200 reported in Table 6 the kNN.avg2 without sampling (algorithm no. 7 in Tab. 6) was better than ours while our was better in 63 runs. The third is the simple k-nn algorithm with $k = 1$, i.e., 1-nn, which is however

Tab. 6. The averaged results of 200 runs of the compared algorithms. First rows show the mean value of the accuracy over all the runs while the second rows show the standard deviation

The algorithm					
1	2	3	4	5	6
1-nn	5-nn	kNN.avg1 tuned	kNN.avg1 k=5	simp kNN.avg1 tuned	simp kNN.avg1 k=5
0.6309	0.5230	0.6264	0.6100	0.6253	0.6053
0.0566	0.0611	0.0567	0.0582	0.0576	0.0592
The algorithm					
7	8	9	10	11	12
kNN.avg2 k_p=10,k_n=1	our approach	5-nn over1	kNN.avg1 k=5 over1	simp kNN.avg1 k=5 over1	kNN.avg2 k_p=10,k_n=1 over1
0.6667	0.6569	0.6084	0.6125	0.6081	**0.6671**
0.0558	0.0585	0.0620	0.0576	0.0579	0.0560
The algorithm					
13	14	15	16	17	18
5-nn over2	kNN.avg1 k=5 over2	simp kNN.avg1 k=5 over2	kNN.avg2 k_p=10,k_n=1 over2	5-nn over3	kNN.avg1 k=5 over3
0.6084	0.6124	0.6079	**0.6665**	0.6084	0.6130
0.0620	0.0578	0.0580	0.0558	0.0620	0.0580
The algorithm					
19	20				
simp kNN.avg1 k=5 over3	kNN.avg2 k_p=10,k_n=1 over3				
0.6085	**0.6667**				
0.0583	0.0558				

significantly worse than two previously mentioned algorithms while is better than the kNN.avg1 algorithm and its simplified version.

Concerning the simplification of the kNN.avg1 algorithm which we have considered, our experiments seem to show statistically significant reduction of the quality of the classification due to its use for most of the parameters settings, i.e., when the pairs of algorithms (4,6), (10,11), (14,15) and (18,19) in Table 6 are compared. However, for the setting where both techniques perform the best, i.e., when the parameter k is dynamically tuned (pair (3,5)), there is no statistically significant difference between the original kNN.avg1 technique and its simplified version.

Concerning the oversampling, the most striking effect is visible in case of the basic 5-nn algorithm. Its performance without oversampling is poor while if coupled with oversampling, in any of the variants over1, over2 or over 3, it produces results not significantly worse than e.g., the kNN.avg1 technique. For kNN.avg1 itself and its simplified version adding oversampling also produces the significantly better results, again in case of any variant. For kNN.avg2 no significant impact of oversampling is visible.

5. Conclusions

We have studied the application of the k-nn technique and its variants to the problem of the multi-aspect text categorization (MTC), in particular with respect to the classification of a document to a case. One of the variants known from the literature [20] proved to be the best when applied to a data set we prepared for our experiments with the solutions to MTC. We proposed also our technique which makes it possible to take into account the importance of the documents within a case, in an intuitively appealing way. This approach also yields good results in our experiments.

We have also studied various ways of tuning of the parameters of the classifiers employed, as well as we have checked if the oversampling of data may help to increase the accuracy of the classification. The results are mixed in this respect: for some classifiers the dynamic tuning of the parameters works while for other there is no improvement. The oversampling supports better classification but the results are convincing mainly for the basic 5-nn classifier.

Further research is surely needed concerning the tuning of the considered techniques. Our experiments with an ACL ARC dataset have confirmed some limited usefulness of the parameters tuning. However, in another setting adjusting parameters to a given collection may turn worth consideration. Thus, it may be important to devise the tuning algorithms in the computationally optimal way. In our experiments the dynamic tuning has been performed by a direct repetition of the functions implementing particular techniques. This can be surely improved. The ways to more efficiently sample the parameters space should be looked for as well as combining the sampling with the implementation of a given technique may be advantageous. We have also discussed the question of the form of the test/validation dataset and here there is also some room for further investigations.

AUTHORS

Sławomir Zadrożny* – Systems Research Institute, Polish Academy of Sciences, 01-447 Warszawa, ul. Newelska 6, Poland, e-mail: Slawomir.Zadrozny@ibspan.waw.pl.
Janusz Kacprzyk – Systems Research Institute, Polish Academy of Sciences, 01-447 Warszawa, ul. Newelska 6, Poland, e-mail: Janusz.Kacprzyk@ibspan.waw.pl.
Marek Gajewski – Systems Research Institute, Polish Academy of Sciences, 01-447 Warszawa, ul. Newelska 6, Poland, e-mail: gajewskm@ibspan.waw.pl.

*Corresponding author

ACKNOWLEDGEMENTS

This work is supported by the National Science Centre (contract no. UMO-2011/01/B/ST6/06908).

REFERENCES

[1] J. Allan, ed., *Topic Detection and Tracking: Event-based Information*, Kluwer Academic Publishers, 2002.

[2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval*, ACM Press and Addison Wesley, 1999.

[3] A. Beygelzimer, S. Kakadet, J. Langford, S. Arya, D. Mount, and S. Li. *FNN: Fast Nearest Neighbor Search Algorithms and Applications*, 2013. R package version 1.1.

[4] S. Bird, R. Dale, B. Dorr, B. Gibson, M. Joseph, M.-Y. Kan, D. Lee, B. Powley, D. Radev, and Y. Tan, "The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics". In: *Proc. of Language Resources and Evaluation Conference (LREC 08)*, Marrakesh, Morocco, 1755–1759.

[5] M. Delgado, M. D. Ruiz, D. Sánchez, and M. A. Vila, "Fuzzy quantification: a state of the art", *Fuzzy Sets and Systems*, vol. 242, 2014, 1–30, http://dx.doi.org/10.1016/j.fss.2013.10.012.

[6] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, no. 4, 1976, 325–327, http://dx.doi.org/10.1109/TSMC.1976.5408784.

[7] I. Feinerer, K. Hornik, and D. Meyer, "Text mining infrastructure in R", *Journal of Statistical Software*, vol. 25, no. 5, 2008, 1–54, http://dx.doi.org/10.18637/jss.v025.i05.

[8] A. Feng and J. Allan, "Hierarchical topic detection in tdt-2004".

[9] M. Gajewski, J. Kacprzyk, and S. Zadrożny, "Topic detection and tracking: a focused survey and a new variant", *Informatyka Stosowana*, to appear.

[10] E. Han, G. Karypis, and V. Kumar, "Text categorization using weight adjusted k-nearest neighbor classification". In: D. W. Cheung, G. J. Williams, and Q. Li, eds., *Knowledge Discovery and Data Mining - PAKDD 2001, 5th Pacific-Asia Conference, Hong Kong, China, April 16-18, 2001, Proceedings*, vol. 2035, 2001, 53–65.

[11] J. Kacprzyk, J. W. Owsiński, and D. A. Viattchenin, "A new heuristic possibilistic clustering algorithm for feature selection", *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 8, no. 2, 2014, http://dx.doi.org/10.14313/JAMRIS_2-2014/18.

[12] J. Kacprzyk and S. Zadrożny. "Power of linguistic data summaries and their protoforms". In: C. Kahraman, ed., *Computational Intelligence Systems in Industrial Engineering*, volume 6 of *Atlantis Computational Intelligence Systems*, 71–90. Atlantis Press, 2012. http://dx.doi.org/10.2991/978-94-91216-77-0_4.

[13] D. Olszewski, J. Kacprzyk, and S. Zadrożny. "Time series visualization using asymmetric self-organizing map". In: M. Tomassini, A. Antonioni, F. Daolio, and P. Buesser, eds., *Adaptive and Natural Computing Algorithms*, volume 7824 of *Lecture Notes in Computer Science*, 40–49. Springer Berlin Heidelberg, 2013. http://dx.doi.org/10.1007/978-3-642-37213-1_5.

[14] D. Olszewski, J. Kacprzyk, and S. Zadrożny. "Asymmetric k-means clustering of the asymmetric self-organizing map". In: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, eds., *Artificial Intelligence and Soft Computing*, volume 8468 of *Lecture Notes in Computer Science*, 772–783. Springer International Publishing, 2014. http://dx.doi.org/10.1007/978-3-319-07176-3_67.

[15] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[16] F. Sebastiani, "Machine learning in automated text categorization", *ACM Computing Survys*, vol. 34, no. 1, 2002, 1–47, http://dx.doi.org/10.1145/505282.505283.

[17] M. Szymczak, S. Zadrożny, A. Bronselaer, and G. D. Tré, "Coreference detection in an XML schema", *Information Sciences*, vol. 296, 2015, 237 – 262, http://dx.doi.org/10.1016/j.ins.2014.11.002.

[18] R. Yager, "Quantifier guided aggregation using OWA operators", *International Journal of Intelligent Systems*, vol. 11, 1996, 49–73, http://dx.doi.org/10.1002/(SICI)1098-111X(199601)11:1%3C49::AID-INT3%3E3.0.CO;2-Z.

[19] Y. Yang, "An evaluation of statistical approaches to text categorization", *Information Retrieval*, vol. 1, no. 1-2, 1999, 69–90, http://dx.doi.org/10.1023/A:1009982220290.

[20] Y. Yang, T. Ault, T. Pierce, and C. W. Lattimer, "Improving text categorization methods for event tracking". In: *SIGIR*, 2000, 65–72, http://dx.doi.org/10.1145/345508.345550.

[21] L. Zadeh, "A computational approach to fuzzy quantifiers in natural languages", *Computers and Mathematics with Applications*, vol. 9, 1983, 149–184, http://dx.doi.org/10.1016/0898-1221(83)90013-5.

[22] S. Zadrożny, J. Kacprzyk, M. Gajewski, and M. Wysocki, "A novel text classification problem and its solution", *Technical Transaction. Automatic Control*, vol. 4-AC, 2013, 7–16.

[23] S. Zadrożny, J. Kacprzyk, and M. Gajewski, "A novel approach to sequence-of-documents focused text categorization using the concept of a degree of fuzzy set subsethood". In: *Proceedings of the Annual Conference of the North American Fuzzy Information processing Society NAFIPS'2015 and 5th World Conference on Soft Computing 2015, Redmond, WA, USA, August 17-19, 2015*, 2015.

[24] S. Zadrożny, J. Kacprzyk, and M. Gajewski. "A new approach to the multiaspect text categorization by using the support vector machines". In: G. De Tré, P. Grzegorzewski, J. Kacprzyk, J. W. Owsiński, W. Penczek, and S. Zadrożny, eds., *Challenging problems and solutions in intelligent systems*, to appear. Springer, Heidelberg New York, 2016.

[25] S. Zadrożny, J. Kacprzyk, and M. Gajewski, "A new two-stage approach to the multiaspect text categorization". In: *2015 IEEE Symposium on Computational Intelligence for Human-like Intelligence, CIHLI 2015, Cape Town, South Africa, December 8-10, 2015*, to appear, 2015.