Hierarchical, multi-label classification of scholarly publications: modifications of ML-KNN algorithm

Michał Łukasik, Tomasz Kuśmierczyk, Łukasz Bolikowski, and Hung Son Nguyen

Interdisciplinary Centre for Mathematical and Computational Modelling,
University of Warsaw
{m.lukasik, t.kusmierczyk, l.bolikowski}@icm.edu.pl

Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw
son@mimuw.edu.pl

Abstract. One of the common problems when dealing with digital libraries is lack of classification codes in some of the documents. In the following publication we deal with this problem in a multi-label, hierarchical case of Mathematics Subject Classification System. We develop modifications of ML-KNN algorithm and show how they improve results given by the algorithm on example of Springer textual data.

Keywords: document classification, multilabel classification, hierarchical classification, ML-KNN, YADDA2 software platform

1 Introduction

Document classification is an old problem and does not require a computer to be solved. A good example is a library, in which categories are assigned to books. A problem that occurs with manual approach is scalability. Automatic text classification is considered since 1960s [1].

Nowadays, document classification is a common problem. Sebastiani [1] brings up such examples as: document indexing, document filtering, meta-data extraction, word sense disambiguation, creating hierarchical catalogue of Internet websites. This list can be extended with analysis of emotions expressed by a text's author [2]. An important problem which appears when dealing with text corpora is assigning classification codes to documents, based on previously classified documents.

In this paper we inspect an established multi-label classification algorithm: ML-KNN. We show a problem that might occur when dealing with noisy data and develop a new KNN-based algorithm that is more resistant to noise. We also use an established method of dealing with hierarchical classification problem and join the method with ML-KNN modifications. We show on real data how the new algorithms perform better than ML-KNN.

The rest of this work is organised as follows: in section 2 we formally define what a classification problem is. We show different classification measures for multi-label classification and specify a measure for hierarchical classification. In section 3 we review literature on multi-label and hierarchical classification. Section 4 contains a detailed description of ML-KNN algorithm. We show a problem, which might occur when working with noisy data using this algorithm. We propose and describe novel modifications of ML-KNN which are not prone to this specific problem. Section 5 describes the data, on which we tested our algorithms. Section 6 shows what experimental settings have been taken, whereas section 7 contains the results for the data. We finish our work with summary and propositions of future work which might improve the algorithms.

2 Problem statement

In this section we formalize a classification problem of documents. We consider an example of such problem in section 5.

2.1 Classification problem

In the classification problem of scientific documents a set of documents D is considered. Furthermore, k attribute functions are defined, each mapping a document into a value from a domain specific for an attribute:

$$\forall_{i \in 1, \cdots, k} : a_i : D \to D_{a_i} \tag{1}$$

Let $Q = \{q_1, \cdots, q_n\}$ be the set of n labels describing documents from set D. We can then specify a function $K : D \times Q \to \{0, 1\}$, such that $K(d, q) = 1 \Leftrightarrow$ q describes document d. Let $K(d)$ be the set of labels describing d.

The solution to the classification problem is, based on some finite set of training documents $D_{train} \subset D$, creating a function $K' : D \times Q \to \{0, 1\}$, which is as similar to K as possible.

2.2 Evaluation

Testing is evaluating, how similar function K' is to K. It is based on comparing values $K'(d)$ and $K(d)$ returned for documents d from some set of documents $D_{test} \subset D$.

Different approaches to evaluation exist in the literature. It is worth noting, that many labels may be assigned to a single document, which brings even more complexity to the problem. We have shown some of the existing evaluation methods below [3].

$$Accuracy(K, K', D_{test}) = \frac{1}{|D_{test}|} \sum_{d_{test} \in D_{test}} \frac{|K(d_{test}) \cap K'(d_{test})|}{|K(d_{test}) \cup K'(d_{test})|} \tag{2}$$

Accuracy measures classification quality, not distinguishing the errors resulting from choosing too many labels from errors resulting from not choosing the label that should be chosen.

Let P and R (defined by equations: (3) and (4)) be: Precision and Recall evaluated for a document. Precision measures the percentage of correct decisions made by a classifier. Recall measures what percentage of documents that describe the document has been chosen.

$$P(K, K', d_{test}) = \frac{|K(d_{test}) \cap K'(d_{test})|}{|K'(d_{test})|} \tag{3}$$

$$R(K, K', d_{test}) = \frac{|K(d_{test}) \cap K'(d_{test})|}{|K(d_{test})|} \tag{4}$$

Based on P and R, analogous variables can be specified for the whole data set: these are the arithmetic means of measures calculated for single documents (equations: (5) and (6)).

$$\mathit{Precision}(K, K', D_{test}) = \frac{1}{|D_{test}|} \sum_{d_{test} \in D_{test}} P(K, K', d_{test}) \tag{5}$$

$$\mathit{Recall}(K, K', D_{test}) = \frac{1}{|D_{test}|} \sum_{d_{test} \in D_{test}} R(K, K', d_{test}) \tag{6}$$

F-measure is a popular classification measure, which deals with a problem of imbalanced label representation. F-measure for a single document is defined as a harmonic mean of Precision and Recall. In equation (7) F-measure is defined as an arithmetic mean of such variables calculated for each of the documents.

$$\mathit{F\text{-}measure}(K, K', D_{test}) = \frac{1}{|D_{test}|} \sum_{d_{test} \in D_{test}} 2\frac{P(K, K', d_{test})R(K, K', d_{test})}{P(K, K', d_{test}) + R(K, K', d_{test})} \tag{7}$$

The last 2 measures we list show how much error is in the classification. Hamming Loss (8) returns number of labels, for which wrong answer has been returned as for it describing a document, averaged over all documents.

$$\mathit{Hamming\text{-}loss}(K, K', D_{test}) = \frac{1}{|D_{test}|} \sum_{d_{test} \in D_{test}} \frac{|(K(d_{test}) - K'(d_{test})) \cup (K'(d_{test}) - K(d_{test}))|}{|Q|} \tag{8}$$

Subset Zero-One loss (9) returns amount of documents for which at least one error has been made in the classification process.

$$\mathit{Zero\text{-}One\text{-}loss}(K, K', D_{test}) = \frac{1}{|D_{test}|} \sum_{d_{test} \in D_{test}} K(d_{test}) \neq K(d_{test}) \tag{9}$$

2.3 Label dependencies

In the classification problem dependencies amongst the labels may be defined. We can specify them by the relation: $R \subset Q \times Q$. An example of such a relation is as follows: $q_1 R q_2 \Leftrightarrow q_1$ is a sub-category of q_2. In such a case, labels with their dependencies form a graph. In general, each label may have a few parental labels. When adding a constraint that each label can have only one parental node, categories form a tree. Only the case of a balanced tree will be considered in this paper.

In a problem stated in such a way one of 2 possibilities can occur. The first possibility is that each of the labels assigned has to be a leaf node in a category tree. The second possibility is the opposite. In this paper we will consider the first option.

In the literature, many approaches to evaluating hierarchical classifiers exist. The common idea behind most of the evaluation methods is that the closer the labels in the category graph are, the less punishment in the evaluation process they should bring when the mistake is made for the other label. There is no established method for hierarchical evaluation yet [4].

In this paper, we evaluate the classifier by comparing the functions K and K' in the following way: for each height h of nodes in the tree of labels (where maximum height is taken by the leaves, and minimum value of 0 by the root) project all the leaf nodes to their ancestor nodes of height h. Such projected labels can be compared in the traditional way, for example using F-measure. We will receive as many results as there are levels in the tree. Because we consider only the case of balanced tree, where only leaf nodes are assigned, we will not encounter a problem that an assigned label does not have an ancestor node of given height.

3 Previous work

In this section we review methods for solving multi-label and hierarchical classification problem.

3.1 Multi-label classification

Multi-label classification methods can be divided into 2 groups: problem transformation methods and algorithm adaptation methods. Problem transformation methods are algorithms that decompose a multi-label problem into one or more single-label problems. Algorithm adaptation methods are about extending an established single-label algorithm to deal with multiple labels.

Problem Transformation Methods One of the simple approaches to the multi-label classification is creating a single-label classifier for each label, which discriminates between a label and all the other labels. There are $|Q|$ classifiers, each of which is trained on the whole data set. At the classification time each classifier answers whether a given label should be assigned to an object.

Another approach is training $|Q|^2 - |Q|$ classifiers, which distinguish between all pair of labels q_1 and q_2. Set of positive samples consists of those objects to which label q_1 has been assigned whereas set of negative samples contains only objects with label q_2 assigned. There is therefore much more classifiers to be trained and less training examples for each of them, comparing to previous approach.

It is worth noting that each of the methods listed above assumes independence between labels. What seems an appropriate approach to multi-label classification, which does not make such an assumption is creating a new set of labels Q', which contains the power set of set Q. The problems with this approach are: small number of training samples for each of the new categories and exponential number of new labels.

Read et al describe a method called classifier chains in [5], which assigns labels one after another, at each step using information about labels assigned this far. A problem how to determine the order of labels is solved by randomly choosing several options.

Zhang proposes in [6] an algorithm based on creating a Bayesian network, which models dependencies between the labels. Information about the dependencies is fetched from correlations between errors given by single-label classifiers.

Algorithm Adaptation Methods There are various algorithm adaptation methods for multi-label classification. Clare and King described in [7] a modification of a C4.5 algorithm, with appropriately modified formula for entropy calculation.

There exist modifications of Ada-Boost approach that allow multi-label classification, that are not transformation based [8]. The idea is very similar to the idea behind basic Ada-Boost approach.

A popular approach in Multi-label classification based on algorithm adaptation is Multi-label KNN, introduced in [9]. It is a Bayesian classifier based on distance features, calculated on neighbours from the training set. The classifier is described in more details in next section.

3.2 Hierarchical classification

In case, when labels form a tree, use of the information about the dependencies might increase the efficiency of a classifier. There exist various approaches to how to use such information [10] [4].

First approach is the flat method, which is about ignoring the hierarchical dependencies and just using some standard classification algorithm.

Another approach is dividing nodes by their distance from the root. Each set of nodes is then treated as a separate flat classification problem.

The most popular approach is creating one classifier per node of a label tree [10]. Each of the classifiers is trained on appropriately narrowed data set.

There are different subclasses of this approach. There might be a single-label classifier in each node, returning a truth value whenever a class describes a given object. There might also be multi-label classifiers in each parental node. In such approach a classifier can choose multiple labels from node's children.

The last category of hierarchical classification listed in [10] is the Big-Bang approach. It is about training one single classifier for the whole hierarchy, which is somehow built in the algorithm. The arguments for using such an approach are savings in time and space complexity. There has not been made much research about such kind of classifiers [10]. An example of the Big-Bang approach is to cast the hierarchical classification into a multi-label problem, saving the information about the hierarchy by adding the labels that are parents of those describing objects [11]. The post-processing step enforces consistency with the hierarchy.

4 ML-KNN

ML-KNN (Multi-Label KNN)[9] is a popular multi-label classification algorithm [3]. It uses 2 popular approaches to classification [12]: Naive Bayes and KNN.

Naive Bayes is an algorithm, which is popular because of its efficiency: in case of simple features it uses only linear time for training.

KNN is an algorithm, which achieves efficiency close to the best classifiers. In case of problems, for which Bayes Error Rate equals 0, 1NN algorithm converges to the optimal classifier as the training data becomes larger [12].

It is worth noting, that already in 1998 Joachims noticed some serious arguments for using SVM for text classification [13]. However, SVM depends heavily on solving a quadratic programming problem, which makes it a computationally demanding task. At the same time, KNN-based algorithms can be efficiently implemented, for example using k-d trees. When working with big text corpora it is therefore worth considering more efficient methods than SVM, such as ML-KNN.

In this section we describe in detail ML-KNN algorithm and inspect its nature. We point at a possible problem when working with ML-KNN on real data. We deal with this problem, developing novel modifications of ML-KNN that do not increase asymptotic time complexity.

4.1 Basic algorithm

Let us use the notation defined in section 2 and moreover let us define:

- S_x - neighbourhood of object x, e.g. its k nearest neighbours (where k is earlier defined)
- $S_x(q)$ - number of occurrences of label $q \in Q$ amongst the objects from S_x

Let H_q be an event, that a given object belongs to class q, and let $\neg H_q$ be the opposite event. Let $E_{S_x(q)}$ be an event, that an object has $S_x(q)$ neighbours belonging to class q.

Category q is being assigned to a given object, if $P(H_q|E_{S_x(q)}) > P(\neg H_q|E_{S_x(q)})$. Bayes theorem states, that this inequality is equivalent to the following:

$$P(E_{S_x(q)}|H_q)P(H_q) > P(E_{S_x(q)}|\neg H_q)P(\neg H_q) \tag{10}$$

It is possible to estimate variables from the inequality (10) using the training set.

The training algorithm for ML-KNN is shown in listing: Algorithm 1 (D is the training set, m is its size, K is the known classification of the training objects, k is the neighbourhood size, s is the smoothing parameter). The algorithm works as follows. First, the a-priori probabilities for each label occurrence are calculated. This is performed by calculating occurrences of categories in the training set.

Next, in the double-nested loop, values $c[q][i]$ are calculated. They denote, how many times the following situation occurs: object belonging to class q has exactly i neighbours, which belong to class q. Similarly, $c'[q][i]$ can be evaluated. They correspond to situations, when object not belonging to class q has exactly i neighbours belonging to class q.

In the end, the posterior probabilities are calculated using values from arrays c and c'.

Algorithm 1 ML-KNN(D, m, K, k, s)

```
Initialize 2-dimensional arrays $c$ and $c'$, both of size $|Q| \times k$
for $q \in Q$ do
    $P(H_q) = \frac{s+\sum_{x \in D} K(x,q)}{2s+m}$
    $P(\neg H_q) = 1 - P(H_q)$
end for
for $x \in D$ do
    $S_x$ = find k nearest objects to x in the training set
    for $q \in Q$ do
        $i$ = how many times class q occurs among objects in $S_x$
        if $K(x,q)$ then
            increment $c[q][i]$
        else
            increment $c'[q][i]$
        end if
    end for
end for
for $q \in Q$ do
    for $i \in 0..k$ do
        $P(E_i|H_q) = \frac{s+c[q][i]}{s(k+1)+\sum_{p \in \{0..K\}} c[q][p]}$
        $P(E_i|\neg H_q) = \frac{s+c'[q][i]}{s(k+1)+\sum_{p \in \{0..K\}} c'[q][p]}$
    end for
end for
return estimated probabilities
```

Classification of an object is implemented as the inequality (10) defines.

Each category is considered separately. Therefore, label independence is assumed.

It can be noticed, that when smoothing parameter s equals 0, the algorithm is equivalent to comparing counts $c[q][i]$ and $c'[q][i]$ for given i and choosing class q iff $c[q][i] > c'[q][i]$. This can be shown by the following series of equivalent inequalities:

$$P(E_{S_x(q)}|H_q)P(H_q) > P(E_{S_x(q)}|\neg H_q)P(\neg H_q) \tag{11}$$

$$\frac{c[q][S_x(q)]}{\sum_{p\in\{0..K\}} c[q][p]} \frac{\sum_{p\in\{0..K\}} c[q][p]}{m} > \frac{c'[q][S_x(q)]}{\sum_{p\in\{0..K\}} c'[q][p]} \frac{\sum_{p\in\{0..K\}} c'[q][p]}{m} \tag{12}$$

$$c[q][S_x(q)] > c'[q][S_x(q)] \tag{13}$$

4.2 Threshold ML-KNN

Let us consider the following situation: 2 objects x_1 and x_2 are given for classification and the following inequality holds for them: $S_{x_1}(q) < S_{x_2}(q)$ for some label q. In such case, ML-KNN algorithm allows the following to happen: $P(H_q|E_{S_{x_1}(q)}) > P(\neg H_q|E_{S_{x_1}(q)})$ and at the same time $P(H_q|E_{S_{x_2}(q)}) < P(\neg H_q|E_{S_{x_2}(q)})$. Example of data, where such a situation should be allowable is shown in figure 1. Nevertheless, intuitively such a situation corresponds to noise. Therefore, it seems reasonable to consider a modification of ML-KNN, which does not allow such a situation to happen.

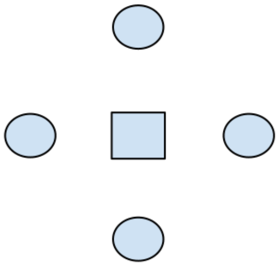

Fig. 1. An example, where situation described should be allowable. When considering neighbourhood of size 4, each of the circles has only 3 neighbouring circles, whereas the square has 4 neighbouring circles.

In order to achieve this, we propose the following modification. Instead of estimating the probabilities shown in inequality (10), threshold number of neighbours p can be chosen for each category, such that $K(x,q) = 1 \Leftrightarrow S_x(q) > p$. In the training data it is possible that such a value does not exist. Common situation is such as shown in table 1. It shows, that in almost all cells non-zero value exists. Nevertheless, intuitively the threshold in the case of data shown in table 1 should be chosen for p=1, because $\forall_{t>1} : c[l] > c'[l]$ and $\forall_{t\leq1} : c[l] \leq c'[l]$. It is less obvious, what the threshold should be like in case of data shown in table 2, because: $c[1] < c'[1]$, $c[2] > c'[2]$, $c[3] < c'[3]$ and $c[4] > c'[4]$.

Table 1. Example table with counts for category q.

neighbours count	c	c’
0	0	100
1	20	40
2	30	24
3	10	8
4	8	1

Table 2. Example table with counts for category q: complicated situation.

neighbours count	c	c’
0	0	100
1	20	40
2	30	21
3	10	14
4	8	1

We propose to choose threshold by maximizing the F-measure. Let us use the following notation:

- FN (false negatives), number of objects incorrectly classified as not belonging to class q. These are the samples incrementing the count c[q][i] for $i \leq p$.
- TP - (true positives), number of objects correctly classified as belonging to class q. These are the samples incrementing the count c[q][i] for $i > p$.
- TN - (true negatives), number of objects correctly classified as not belonging to class q. These are the samples incrementing the count c’[q][i] for $i \leq p$.
- FP - (false positives), number of objects incorrectly classified as belonging to class q. These are the samples incrementing the count c’[q][i] for $i > p$.

Now, threshold p can be chosen in such a way, that F-measure is maximized. F-measure is calculated using the following formula:

$$F1 = \frac{2PR}{P + R} \tag{14}$$

In the formula (14), P means precision and R means recall. They are calculated in the following way: $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$.

Complete training algorithm for Threshold ML-KNN is shown in listing: Algorithm 2. The agorithm works as follows. In lines 3 - 13 counts c and c' are calculated in a similar way as in case of ML-KNN. Next, for each category q a value for a threshold maximizing F-measure value is being chosen. An array of such values is returned.

Algorithm 2 THRESHOLD-ML-KNN(D, K, k, s)

```
 1: Initialize 2-dimensional arrays $c$ and $c'$, both of size $|Q| \times k$
 2: Initialize 1-dimensional array $p$ of size $|Q|$ and floating point variable $best_{f1}$
 3: for $x \in D$ do
 4:     $S_x$ = find k nearest objects to x in the training set
 5:     for $q \in Q$ do
 6:         $i$ = how many times class q occurs among objects in $S_x$
 7:         if $K(x, q)$ then
 8:             increment $c[q][i]$
 9:         else
10:             increment $c'[q][i]$
11:         end if
12:     end for
13: end for
14: for $q \in Q$ do
15:     $p[q] = -1$
16:     $best_{f1} = -1$
17:     for $i \in 0..k$ do
18:         $FN = \sum_{l \in \{0,\cdots,i-1\}} c[q][l]$
19:         $TP = \sum_{l \in \{i,\cdots k\}} c[q][l]$
20:         $TN = \sum_{l \in \{0,\cdots,i-1\}} c'[q][l]$
21:         $FP = \sum_{l \in \{i,\cdots k\}} c'[q][l]$
22:         $F1$ = Calculate F-measure based on FN, TP, TN and FP
23:         if $F1 > best_{f1}$ then
24:             $p[q] = i$
25:             $best_{f1} = F1$
26:         end if
27:     end for
28: end for
29: return p
```

4.3 Ensemble Threshold ML-KNN

Choice of value for parameter k is a problem that appears each time a KNN based algorithm is used. The way how Threshold ML-KNN has been defined allows to cope with the problem, by using different values at the same time. A few Threshold ML-KNN classifiers may be constructed simultaneously, not making the asymptotic time complexity larger.

Algorithm 3 ENSEMBLE-OF-THRESHOLD-ML-KNN$(D, K, k\text{-}list, s)$

```
Initialize 3-dimensional arrays $c$ and $c'$, both of size $k \times |Q| \times k$
Initialize 2-dimensional arrays $p$ and $best_{f1}$, both of size $k \times |Q|$
for $x \in D$ do
  $S_x$ = find $max(k\text{-}list)$ nearest objects to x in the training set
  for $q \in Q$ do
    for $j \in k\text{-}list$ do
      $i$ = how many times class q occurs among first j objects from $S_x$
      if $K(x, q)$ then
        increment $c[j][q][i]$
      else
        increment $c'[j][q][i]$
      end if
    end for
  end for
end for
for $j \in k\text{-}list$ do
  for $q \in Q$ do
    $p[j][q] = -1$
    $best_{f1}[j][q] = -1$
    for $i \in 0..j$ do
      $FN = \sum_{l \in \{0,\cdots,i-1\}} c[j][q][l]$
      $TP = \sum_{l \in \{i,\cdots j\}} c[j][q][l]$
      $TN = \sum_{l \in \{0,\cdots,i-1\}} c'[j][q][l]$
      $FP = \sum_{l \in \{i,\cdots j\}} c'[j][q][l]$
      $F1$ = Calculate F-measure based on FN, TP, TN and FP
      if $F1 > best_{f1}[j][q]$ then
        $p[j][q] = i$
        $best_{f1}[j][q] = F1$
      end if
    end for
  end for
end for
return $p$, $best_{f1}$
```

Let us assume we have a list of natural numbers $k\text{-}list = [k_1, k_2, \cdots, k_n]$ and that we can find k nearest neighbours in sorted order, which allows to quickly take j nearest neighbours out of them, for $k > j$. In such case, it is enough to find a number of $max(k\text{-}list)$ nearest neighbours once in order to calculate the parameters needed to train an ensemble of Threshold ML-KNN classifiers.

After the training process, it is possible to use different Threshold ML-KNN's in the voting process. We propose to use the estimated F-measure values to choose the best Threshold ML-KNN for each class.

It is important, that there exists danger of over-fitting - such a classifier has more dimensions in the training process, which makes it more vulnerable to changes in the form of training data.

In the listing: Algorithm 3 the training algorithm for Ensemble Threshold ML-KNN is shown. First, counts c and c' are calculated. After calculations, $c[j][q][i]$ shows how many times occurs the following situation: object belonging to class q has exactly i neighbours out of the closest j neighbours, which belong to class q. On the other hand, $c'[j][q][i]$ shows how many times object not belonging to class q has exactly i neighbours out of the closest j neighbours, which belong to class q.

Next, arrays p and $best_{f1}$ are calculated. After calculations, $p[j][q]$ shows, what minimum number of neighbours out of the closest j neighbours should be in class q in order to assign class q to an object. $best_{f1}[j][q]$ shows what F-measure value has been achieved for such threshold. Based on these arrays the best value $k_q \in$ *k-list* in terms of achieved F-measure can be chosen for each class q.

5 Data description

In this section we describe data on which we evaluated algorithms described in previous sections. The classification task has been evaluated on Springer data[1], which has been made available to the ICM[2].

5.1 General description

The data made available consisted of 1342882 records, describing consecutive scientific papers. Each of the records has been described by meta-data such as: list of authors, title, abstract, keywords. The full list of fields has been shown in table 3.

Each record is described by a list of labels. There are different categorization systems in the data, such as: MSC[3], PACS[4] (Physics and Astronomy Classification Scheme) etc. In table 4 we show basic information about how each categorization system exists in the data.

The categories are very rare in the corpus. Therefore, in the evaluation process we consider MSC only.

5.2 MSC codes

MSC (Mathematics Subject Classification) is the classification system for documents on mathematics. It consists of more than 5000 categories, each represented by a string of 2, 3 or 5 characters. Each document can be described using more than one category (in such case, the first category is considered as most important). Categories form a hierarchy, in which each category represented by a

[1] http://www.springer.com/

[2] http://www.icm.edu.pl/

[3] http://www.ams.org/mathscinet/msc/msc2010.html

[4] http://publish.aps.org/PACS

Table 3. Fields describing the records.

Field	Description
an	Unique identifier.
py	Publication year.
ti	Title.
ut	Keywords.
ab	Abstract. Consisting of 1-4 short sentences describing the paper.
au	List of authors.
jy	Year of journal publication.
mc	MSC classification tags.
jp	Journal publisher.
ps	Page numbers in the journal.
jt	Journal title.
uv	Affiliations of authors.
vl	Volume of a journal.
jc	ISSN number of a journal.

Table 4. Statistics about various categorization systems in the corpus.

Categorization	No. of documents	No. of occurrences	No. of distinct codes
ZDM	297	783	125
PACS	13715	38639	3970
CLC	8536	8536	2495
QICS	37	66	40
MSC	20275	54410	5130
JEL	7927	22349	860

string of 2 or 3 characters is divided into subcategories. In picture 2 we show part of the MSC tree. Each node corresponds to some field of mathematics. In the example shown, 60 corresponds to probability theory and stochastic processes, 60E to distribution theory, and 60E15 to inequalities and stochastic orderings.

5.3 Analysis of record content

In the meta-data there are many fields containing various information concerning the publications. It seems that some introduce relevant information (such as keywords and title) and some don't (such as page numbers).

It can be stated, that the most important fields are textual ones: title, abstract and keywords. List of authors can also contain relevant information, since if a person published in some field, then it is very possible that he or she is still going to work on it.

In table 5 we showed how many records contain all fields from given sets. We excluded fields, which do not seem to introduce important information for choosing the topic of a paper.

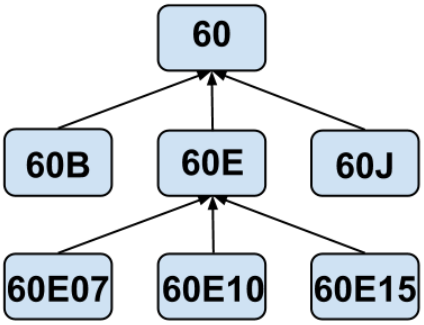

Fig. 2. Part of an MSC category tree.

After counting the fields it can be noticed, that number of records containing all the textual information (title, abstract and keywords) is not much smaller then number of records containing MSC codes.

Table 5. Number of records containing subsets of fields.

ab	1105609
au	1289023
jt	14959
mc	20275
py	1342065
ti	1281409
ut	851240
ab au mc py ti	20155
ab au mc py ut	17977
ab au mc ti ut	17959
au mc py ti ut	18017
au jt mc py ti ut	470
ab au jt mc py ti	502
ab au jt mc py ut	468
ab au jt mc ti ut	468
ab au mc py ti ut	17959
ab jt mc py ti ut	468
ab au jt mc py ti ut	468
ab au jt mc py ti ut	468

5.4 Data filtering

In picture 3 we show a histogram showing number of codes with consecutive occurrence numbers. It can be noticed, that there is a big number of codes with very few occurrences. Training a classifier with such small number of training samples is a very hard task.

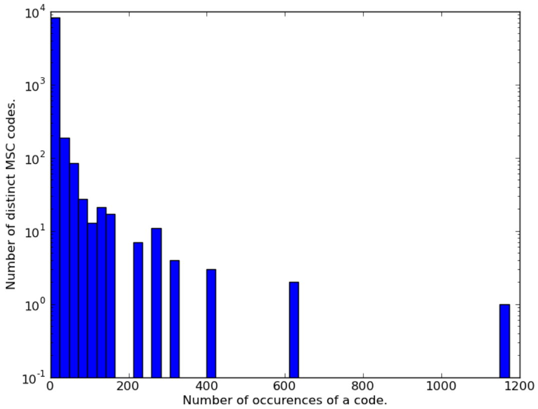

Fig. 3. Number of MSC codes of a given number of occurrences in the corpus before filtering.

Because of low number of occurrences of some codes, we performed data filtering. We left only these codes, which appeared at least 30 times and at the same time have not been the only subcategory of their parental category. The criterion has been applied recursively to codes of 2nd and 3d level. As a result, some codes have been left without any labels, therefore they have been removed from the corpus. Finally, 240 different codes for the bottom level remained. We got 9180 records.

5.5 How multi-label and how hierarchical data is

Tsoumakas in [3] describes measures which describe, how multi-label data is: *label cardinality* and *label density*.

Label cardinality shows, how many labels on average are being assigned to a record. In our case the value is 1.56.

Label density describes, what part of all the labels is on average assigned to a document. The value for the data is 0.65%.

The corpus can also be described by measures describing, how hierarchical it is. In table 6 we showed in how many documents there are at least 2 similar codes (similar in the hierarchical sense) and at least 2 different codes.

Table 6. The percentage amount of records with at least 2 similar (different) codes. The criterion for similarity is being subcategory of the same category.

Property	Percentage
Contains at least 2 sub-codes of the same highest level category	35.39%
Contains at least 2 sub-codes of different highest level categories	13.66%
Contains at least 2 sub-codes of the same 2nd level category	29.59%
Contains at least 2 sub-codes of different 2nd level categories	24.78%

6 Experimental settings

We decided to use only textual fields in the experiments. We joined keywords, title and abstract of each record to form single text describing each document. We removed stop words and projected words to their base forms using Porter stemmer. We then performed a popular text analysis technique called TF-IDF [12]. This way, we got vectors of numbers describing each document.

In each algorithm, we used cosine distance, which is a good measure to distinguish between high dimensional objects, such as texts described by TF-IDF vectors [12].

As for ML-KNN algorithms, we used the following settings for the parameters:

- smoothing parameter (s): 1
- k parameter: 5
- k list parameter: [3, 5, 8]

As was the case in [9], we also noticed, that value for parameter k does not influence the relative performance of the algorithms.

7 Experimental results

In this section we show results for classifiers described earlier.

We have evaluated the following classifiers: ML-KNN, Threshold ML-KNN and Ensemble Threshold ML-KNN, where the 2 latter algorithms are our modifications of ML-KNN algorithm (they have been described in section 4).

Furthermore, we have evaluated modifications, where we put each of the 3 listed classifiers into parental nodes of the hierarchical structure of labels. This is one of the most popular approaches to hierarchical classification, as stated in section 3. We list tested algorithms in table 7.

The evaluation has been performed on data described in section 5, prepared as stated in section 6. 5-fold cross validation has been used to estimate efficiency of classifiers.

Table 7. Evaluated classifiers. We separately test flat approaches and hierarchical approaches. Novel methods that have been proposed in this paper have been highlighted.

Approach	Base Algorithm	Modification	Ensemble Modification
Flat	ML-KNN	Threshold ML-KNN	Ensemble Threshold ML-KNN
Hierarchical	Hierarchical ML-KNN	Hierarchical Threshold ML-KNN	Hierarchical Ensemble Threshold ML-KNN

Table 8. Evaluation measures for results projected to the 1st level of labels.

Classifier	Accuracy	Precision	Recall	F-measure	Hamming Loss	Subset 0/1 Loss
ML-KNN	21.30%	23.08%	21.35%	21.90%	0.38%	80.49%
Threshold ML-KNN	44.36%	47.48%	45.23%	45.68%	0.30%	59.55%
Ensemble Threshold ML-KNN	63.33%	66.59%	72.38%	67.43%	0.29%	48.59%
Hierarchical ML-KNN	44.62%	48.09%	45.91%	46.20%	0.32%	60.03%
Hierarchical Thresh. ML-KNN	50.95%	54.73%	51.75%	52.47%	0.28%	53.55%
Hierarchical Ens. Thresh. ML-KNN	66.59%	70.66%	71.33%	69.54%	0.25%	42.06%

Table 9. Evaluation measures for results projected to the 2nd level of labels.

Classifier	Accuracy	Precision	Recall	F-measure	Hamming Loss	Subset 0/1 Loss
ML-KNN	18.40%	21.52%	18.51%	19.44%	0.45%	84.59%
Threshold ML-KNN	36.49%	41.76%	37.85%	38.67%	0.40%	69.72%
Ensemble Threshold ML-KNN	49.42%	54.19%	61.72%	55.14%	0.49%	66.44%
Hierarchical ML-KNN	35.27%	40.69%	37.38%	37.76%	0.44%	71.78%
Hierarchical Thresh. ML-KNN	40.91%	46.95%	42.78%	43.53%	0.41%	66.59%
Hierarchical Ens. Thresh. ML-KNN	51.83%	57.57%	61.75%	57.00%	0.46%	62.41%

Table 10. Evaluation measures for results directly on leaf nodes.

Classifier	Accuracy	Precision	Recall	F-measure	Hamming Loss	Subset 0/1 Loss
ML-KNN	13.83%	18.87%	14.03%	15.45%	0.59%	90.61%
Threshold ML-KNN	25.12%	32.60%	27.71%	28.43%	0.60%	83.92%
Ensemble Threshold ML-KNN	31.74%	37.14%	46.55%	38.56%	0.84%	85.82%
Hierarchical ML-KNN	24.78%	32.63%	27.11%	28.04%	0.63%	84.03%
Hierarchical Thresh. ML-KNN	26.60%	34.32%	30.39%	30.45%	0.66%	83.82%
Hierarchical Ens. Thresh. ML-KNN	31.94%	37.37%	48.62%	38.92%	0.96%	85.34%

We used measures described in section 2.2. Furthermore, we used method for evaluating hierarchical classification problems, described in section 2.3.

Results for each consecutive level of the label tree are shown in tables: 8, 9, 10. By separately comparing the flat approaches and the hierarchical approaches we can notice, that proposed modifications give better results than basic ML-KNN algorithm.

8 Future work

There are many directions, in which our work can be improved. We listed some of the possibilities below.

- In our work, we assumed a bag of words model, which in turn assumes independence between the words. What can be done to improve this is to try extracting semantic information from the text.
- We excluded non-textual attributes from data. It seems that information about authors can be very useful to categorize documents.
- We used TF-IDF algorithm to generate weights. There exist other algorithms (such as LSA or LDA), which may improve the results.
- The most popular approach to deal with the hierarchical structure between the labels has been chosen. One can try other solutions.
- We narrowed our work to a single algorithm (ML-KNN). We could try use other approaches and join them with our algorithms. For example we could form a hierarchy of different classifiers, where our algorithm deals with the first classification step, and more difficult situations to decide are delegated to more effective classifiers (such as SVM),
- Evaluating hierarchical classification is not a trivial problem and it can be further explored.

9 Summary

In this paper we dealt with multi-label hierarchical classification problem of documents. We chose to work on ML-KNN and supported our decision with the efficiency of this algorithm. Analysis of the classification has been performed. It turns out, that when there is a specific type of noise in the data, the algorithm fits to it too much. We proposed modifications of ML-KNN, which help to deal with such noise.

The problem has been noticed when working on the data, on which the algorithms have been evaluated. It turns out, that a situation, where document's neighbourhood is not very stable is not rare. The algorithm's ability to learn about such instabilities causes degradation in classification.

What is worth mentioning is that our modifications, apart from giving better results, sustain low computational cost of the algorithm. This is important, because as we pointed out earlier, this is one of the biggest strengths of this approach.

10 Implementation in the YADDA2 architecture

Results of research presented in this paper are currently implemented as a module in the SYNAT system. SYNAT is a strategic project commissioned by the Polish National Centre for Research and Development, with the goal of building "Interdisciplinary System for Interactive Scientific and Scientific Technical Information." YADDA2 framework, developed at ICM UW, is a core part of that system.

YADDA2 [14] has a two-tier architecture, with base services tier providing generic fuctionalities independent of the type of content being processed, and application tier where business logic and user interfaces are located. YADDA2 facilitates creation of several types of products:

- stand-alone repositories with a web front-end and a publication application in the back-end;
- repository federations containing multiple autonomous collections, accessed through a central front-end;
- publication data warehouses aggregating content from multiple repositories in order to provide long-term preservation of data and access for researchers and analysts.

Several configurable components are already implemented and are ready to be used, for example: meta-data and content storage, full-text indexing, batch processing engine, relational index, user annotation service. Results of this research are being implemented as yet another reusable module, providing hierarchical, multi-label classification tailored for scholarly publications.

The classification module is intended to be part of back-end work-flows for improving meta-data quality and enriching it with inferred information. In a typical setting, one back-end process fetches from a storage all the documents that are already classified using codes from a given classification scheme, passes them to the module in question in order to train it and places results of the training in a storage. Another back-end process fetches all documents from a given domain lacking codes from a given classification scheme, pipes them to the module for classification (configured to use the results of an earlier training) and updates document meta-data using output from the classifier.

11 Acknowledgements

This work is supported by the National Centre for Research and Development (NCBiR) under Grant No. SP/I/1/77065/10 by the Strategic scientific research and experimental development program: "Interdisciplinary System for Interactive Scientific and Scientic-Technical Information."

References

1. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys **34**(1) (March 2002) 1–47
2. Melville, P., Gryc, W., Lawrence, R.D.: Sentiment analysis of blogs by combining lexical knowledge with text classification. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '09, New York, NY, USA, ACM (2009) 1275–1284
3. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. IJDWM **3**(3) (2007) 1–13
4. Costa, E., Lorena, A., Carvalho, A., Freitas, A.: A review of performance evaluation measures for hierarchical classifiers. In Drummond, C., Elazmeh, W., Japkowicz, N., Macskassy, S., eds.: Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop, AAAI Technical Report WS-07-05, AAAI Press (July 2007) 1–6
5. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II. ECML PKDD '09, Berlin, Heidelberg, Springer-Verlag (2009) 254–269
6. Zhang, M.L., Zhang, K.: Multi-label learning by exploiting label dependency. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '10, New York, NY, USA, ACM (2010) 999–1008
7. Clare, A., Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: In: Lecture Notes in Computer Science, Springer (2001) 42–53
8. Zhu, J., Rosset, S., Zou, H., Hastie, T.: Multi-class adaboost. Technical report (2005)
9. Zhang, M.L., Zhou, Z.H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition **40**(7) (2007) 2038–2048
10. Silla, C., Freitas, A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery **22** (2011) 31–72 10.1007/s10618-010-0175-9.
11. Kiritchenko, S., Matwin, S., Nock, R., Famili, A.F.: Learning and evaluation in the presence of class hierarchies: application to text categorization. In: Proceedings

of the 19th international conference on Advances in Artificial Intelligence: Canadian Society for Computational Studies of Intelligence. AI'06, Berlin, Heidelberg, Springer-Verlag (2006) 395–406

12. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA (2009)
13. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In Nédellec, C., Rouveirol, C., eds.: Proceedings of ECML-98, 10th European Conference on Machine Learning, Heidelberg et al., Springer (1998) 137–142
14. Sylwestrzak, W., Rosiek, T., Bolikowski, L.: YADDA2 – Assemble Your Own Digital Library Application from Lego Bricks. In: Proceedings of the 2012 ACM/IEEE Joint Conference on Digital Libraries. (2012)