TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

University of Zagreb
Faculty of Electrical Engineering and Computing

Master Thesis no. 776

Multi-label Document Classification using EuroVoc Thesaurus

Martin Tutek

Zagreb, June 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 10. ožujka 2014.

DIPLOMSKI ZADATAK br. 776

Pristupnik: **Martin Tutek**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Multi-label Document Classification using EuroVoc Thesaurus**

Opis zadatka:

Multi-label document classification, also known as document indexing, is the task of automatically assigning multiple labels to the same document. The multilingual and hierarchical EuroVoc thesaurus has been used to index legal documents to improve the access to legislation. However, multi-label classification using EuroVoc thesaurus has proven to be a particularly challenging task due to label sparsity, arising as a consequence of a large number of labels and unbalanced label distribution.
The topic of this thesis are the machine learning experiments on multi-label document classification using EuroVoc thesaurus. Provide a thorough overview of the existing approaches to EuroVoc document indexing, multi-label classification, hierarchical classification, and algorithms that combine the latter two approaches. Implement multi-label document classification algorithms based on, but not limited to, random forests, support vector machine (SVM), and the HOMER algorithm of Tsoumakas et al. (2008). Implementation may build on the existing open-source libraries such Mulan. Implement techniques for addressing the label sparsity problem, such as feature weighting and feature labeling. Evaluate and compare multi-labeling models with on the English and Croatian dataset, using the JRC AC corpus and NN13205, respectively. The evaluation should account for multi-label and hierarchical nature of the problem. Compare the models against multi-class SVM as the baseline and carry out a thorough error analysis on the test set. All algorithms, source code, and additional documentation must be provided with the thesis, and all references must be cited.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 30. lipnja 2014.

Mentor:

Doc. dr.sc. Jan Šnajder

Djelovođa:

Doc. dr.sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Prof. dr.sc. Siniša Srbljić

CONTENTS

1. Introduction

Multi-label classification, also known as document indexing, is the task of automatically assigning multiple labels to the same document. The multilingual and hierarchical EuroVoc thesaurus is a prime example of such a problem. Containing documents from various fields ranging from politics and law, over social and economical questions, geography, environment and science, it presents many obstacles such as label sparsity, a large number of labels and an unbalanced label distribution.

EuroVoc is a highly multilingual thesaurus containing over 6,800 hierarchically organised subject domains used by European Institutions and its Member States for classification and retrieval of official documents. Such a task is complex, slow and costly to be done manually, as human documentalists index around thirty documents per day. Thus, an automated approach proves to be the best option. Even with the disadvantages of having a significantly worse performance than its human counterparts, an automated approach is fast and consistent. The bad performance can be treated by using the automated approach along with human support, serving as a support tool of sorts.

There are two main methods of approaching the multi-label classification problem, namely problem transformation methods and algorithm adaptation methods. Problem transformation methods transform the multi-label problem into a set of binary classification problems, resulting in a one-vs-all approach *(or one-vs-rest, OvA or OvR)* where a single-class classifier is trained for each label. Such an approach is considered to be up to par with multi-class and multi-label algorithm adaptation approaches and has been explored considerably (Rifkin and Klautau, 2004, Read et al., 2011), some even on the EuroVoc dataset (Mencía and Fürnkranz, 2010). Algorithm adaptation methods modify algorithms to directly perform multi-label classification and treat the problem in its full form (Tsoumakas and Katakis, 2007, Li et al., 2006).

2. Related Work

2.1. Multi-label Classification

2.1.1. Mulan

An open source Java based library for learning from multi-label datasets, Mulan[1] is currently the best starting point for any multi-label researcher, as it provides a variety of algorithms easily applied to any multi-label problem. The only prerequisites are for the data to be formatted in the Attribute-Relation File Format *(further on: ARFF)*[2] format of Weka, however adapted to contain an EXtensible Markup Language *(further on: XML)* file describing the label hierarchy[3]. From the library, a number of algorithms will be considered for the classification task, namely Binary Relevance k Nearest Neighbor *(further on: BRkNN)*, Multi-label k Nearest Neighbor *(further on: MLkNN)*, Random k Label-sets *(further on: RAkEL)* and Hierarchy of Multi-label Classifiers *(HOMER)*. A short description of these algorithms follows.

BRkNN

BRkNN, from the family of lazy classifiers (Eleftherios Spyromitros, 2008) is a problem transformation approach, using the binary relevance *(further on: BR)* approach combined with the k Nearest Neighbor *(further on: kNN)* algorithm. However, instead of simply using the kNN algorithm $|L|$ times, the approach is optimized as to not calculate all distances every time for each label, but to calculate them once and store them in memory, speeding up the learning process by a factor of $|L|$. However, with such a high dimensionality of labels found in the EuroVoc thesaurus, even this approach proves to be extremely slow.

Another improvement is that BRkNN fixes a flaw found in many BR approaches

[1] http://mulan.sourceforge.net/

[2] http://www.cs.waikato.ac.nz/ml/weka/arff.html

[3] http://mulan.sourceforge.net/format.html

- an empty set as an output. In case that an empty set of labels is the output of the classifier due to none of the labels λ from $\mathbf{L}$ being included in at least half of the nearest k neighbors, the output label is then set to the label with the highest confidence.

MLkNN

MLkNN (Zhang and Zhou, 2007), also a member of the family of lazy classifiers is a multi-label algorithm adaptation of the kNN algorithm. MLkNN uses the maximum a posteriori principle to determine the label set of an unseen instance. The algorithm uses an Euclidean metric to measure distances between instances and then applies membership counting in order to determine the final label set. The paper, however, leaves a lot of open problems, such as using a different distance metric or a more complex method than membership counting to facilitate the usage of the a posteriori principle.

RAKEL

The multi-label learning method called Label Powerset *(further on: LP)* considers every distinct combination of labels existing in the training set as a different class value of a single-class classification task, therefore transforming the problem from a multi-label to a multi-class one. However, the LP algorithm has severe performance issues with larger numbers of labels and training examples, as is the case in the EuroVoc thesaurus. To reduce the dimensionality problem, the RAKEL (Tsoumakas and Vlahavas, 2007) algorithm proposes breaking the initial set of labels into a small number of random subsets, and then employing LP to train a corresponding classifier. The resulting algorithm proves to be competitive against other multi-label algorithms and reduces the training complexity greatly.

HOMER

The HOMER (Tsoumakas et al., 2008) algorithm is designed to combat domains with high label sets by constructing a hierarchy of multi-label classifiers, each one of them dealing with a smaller set of labels compared to the original problem. HOMER uses a balanced k means algorithm to cluster the labels while keeping a sorted list of labels based on importance to a specific cluster. However, the HOMER algorithm still proves to have some time problems with regard to an extremely large number of labels. Further on, a novel optimized approach based on the HOMER algorithm will be considered.

2.1.2. Probabilistic Clustering

To solve a hierarchical multi-label classification problem often appearing in protein function prediction, a novel approach named HMC-PC (Barros et al., 2013) based on probabilistic clustering is considered. In the protein function prediction the datasets can be structured both as trees and directed acyclic graphs, taking a form of a hierarchy where each step of the classification depends on the previous one. It is noteworthy to mention that a similar problem has been tackled by a family of Support Vector Machines *(further on: SVM)* trained seperately for each class, and then combined by using Bayesian Networks Friedman et al., 1997. Such an approach could be feasible for the EuroVoc classification problem, however this line of research will be left for future work.

Another clustering based algorithm was used for image annotation and retrieval (Nasierding et al., 2009). The amount of research in label clustering proves that high dimensionality of labels is indeed a very large issue in multi-label classification, and that an efficient way of clustering labels reduces training time, making various more complex approaches feasible on the smaller label sets.

2.1.3. Random Forests

Random Forests (Breiman, 2001) are a combination of tree predictors such that every tree in the forest depends on a random vector sampled from a same distribution, however independently for all trees in the forest. The random forest learner has proven to be good in application to multi-label classification problems, especially when combined with an error correcting code as seen in Kouzani and Nasireding, 2009. This learner was considered, however due to its weak performance when dealing with larger numbers of labels and no significant improvement in classification, it was discarded from further analysis.

2.1.4. Neural Networks

A notable mention is the Neural Network algorithm, which can also be used to solve the multi-label classification problem, as seen in Zhang and Zhou, 2006. The approach uses a backpropagation algorithm for ranking labels. However, as seen in the results section, the time performance for the algorithm is significantly (by a factor of over 10!) worse than other considered algorithms. Therefore, this approach was immediately discarded.

2.2. Support Vector Machine

A significant part of early approaches to the multi-label classification problem was to simply solve it by training one classifier for each label and treat the problem as a ranking task. A whole array of these approaches can be found in literature, most of them resorting to using SVM as the binary classifier, as seen in Boutell et al., 2004; Gonçalves and Quaresma, 2003; Lauser and Hotho, 2003, with the first usage originating from Elisseeff and Weston, 2001. Most of the approaches use either a ranking method or a cutoff based on confidence.

However, this method is approached sceptically for the problem of EuroVoc indexing. As will be shown in statistical data later on, the EuroVoc descriptors have a large amount of overlapping, some of them to the extent of being completely absorbed by other labels. Due to the nature of a SVM classifier which attempts a linear separation of classes based on features, and there being almost none decisive features (each EuroVoc label appears with at least one other label for all documents in the thesaurus), one can see that linear separation, even for the high feature dimensionality often found in text classification, will be very difficult here.

2.3. Feature Labeling

Feature labeling is an attempt to use the fact that some features can be almost absolutely certain indicators of a class. For example, the word *puck* in articles about sport is a very strong indicator that the article is talking about hockey. The approach found in Druck et al., 2008, 2009 takes advantage of this fact and attempts to label certain decisive features in order to give them stronger decision making power in document classification.

This approach has various obvious setbacks - one cannot simply classify a document based on one feature due to possible noise, problems with the learning set and also edge cases. However, some features can be stronger indicators of a certain class, and feature labeling presents a feasible supporting approach to classification algorithms which may contain such decisive features.

2.4. EuroVoc Thesaurus

Various research was carried out on the EuroVoc thesaurus, it in itself being a very interesting dataset with parallel translations on 22 different languages, possibilities for

document alignment through languages (Steinberger et al., 2002) and overall a very difficult multi-label classification problem. The freely available classification tool JEX Steinberger et al., 2013 has so far had the best performance by using a profile-based category-ranking approach.

Many others have tackled this problem, be it by using different classifiers (Mencía and Fürnkranz, 2010; Boella et al., 2012) by analysing various approaches such as lemmatisation Šarić et al., 2014 for slavic languages, using n-grams, part-of-speech *(further on: POS)* tags and synonyms (Mohamed et al., 2012; Moschitti and Basili, 2004).

Various approaches not related to EuroVoc, however related to specific languages show that different feature representation throughout languages can significantly improve classification performance, for instance, using character n-grams in a highly inflected language such as Croatian in Šilić et al., 2007, and completely ignoring linguistic preprocessing for languages such as Czech in Toman et al., 2006.

Research on a complex text classification problem on a large number of different languages has endless research options due to the finesses of those languages. In the scope of this research, we will focus on English and the corresponding dataset, JRC-Acquis (Steinberger et al., 2006).

3. Problem Analysis

3.1. Label Hierarchy

EuroVoc is, as mentioned before, a multilingual hierarchical thesaurus. Normally, a pre-defined hierarchy could help with classification as the labels in the same tree should be conceptually related. However, this is often not the case in EuroVoc. The labels are in general simply more specific descriptions of the matter being discussed in them, and more often than not a parent does not have an important semantic relation with the child. A sample of the hierarchy tree parsed during pre-processing can be found on figure 3.1.

```
2,breach of trust
    20,award of contract
        200,Yaoundé Convention
            2000,anti-racist movement
            2001,trends of opinion
            2002,capital movement
            2003,women's movement
            2004,youth movement
            2005,national liberation movement
            2006,complementarity agreement
            2007,ecology movement
            2008,European Movement
            2009,workers' movement
        201,UN convention
            2010,farmers' movement
            2011,social movement
            2012,means of communication
            2013,mass media
            2014,means of agricultural production
            2015,means of transport
            2016,medium-sized business
            2017,medium-sized holding
            2018,Mozambique
            2019,multilingualism
        202,economic convergence
```

Figure 3.1: A sample of the EuroVoc hierarchical tree

However, the label tree still serves a purpose, which will be used further on. If, for example, a child label is present in an document, that document will not contain any of the parent labels. This means, after the classification is done, a simple step can be used to eliminate possible mistakes, leaving only the furthermost children in the final determined label set.

3.2. Label Sparsity

An additional problem, which is found often in supervised classification is class imbalance. A specific example of class imbalance is class sparsity, or in the case of multi-label classification, label sparsity. Label sparsity indicates that some labels are underrepresented, and that is the case with the EuroVoc dataset as well.

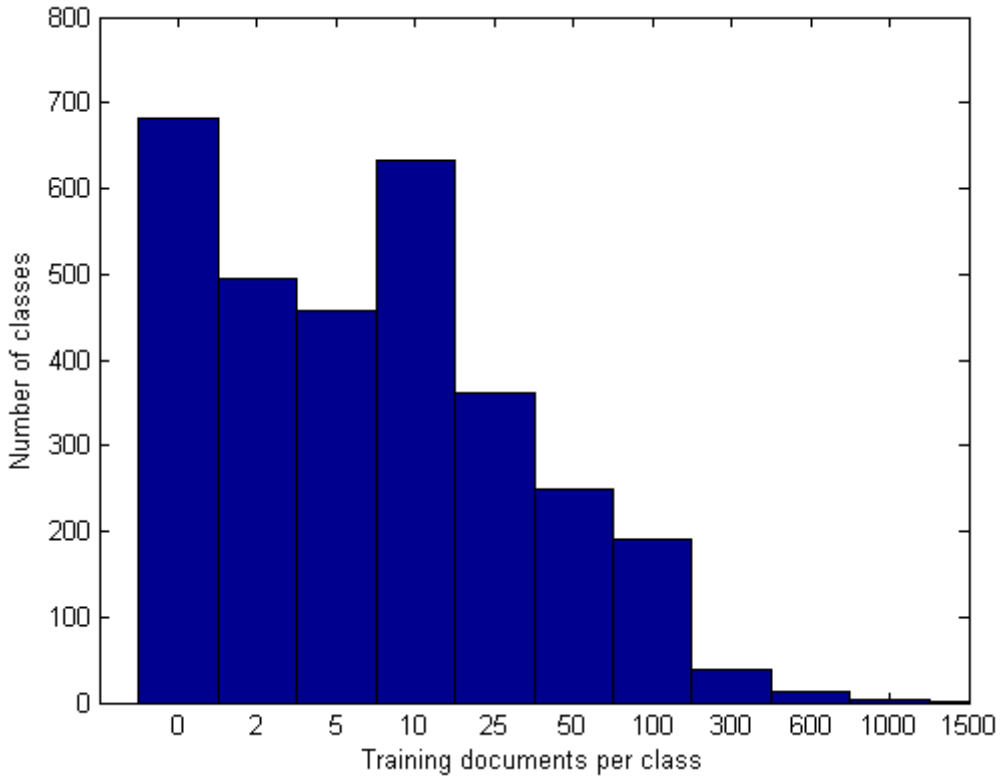

Figure 3.2: EuroVoc label imbalance

As can be seen in the figure, most labels have less than 15 training documents. Specifically, the number of labels that have only one or two training documents is 683, a very large number considering that only 3822 labels from the whole hierarchy can be found in the training set. Taking this into consideration, a minimum number

of documents necessary for a label to be considered in the classification process was set. As it will be shown during experiments, labels with less than 5 documents are extremely difficult to classify properly.

3.3. Descriptor distribution

Another issue to be considered is the inconsistent amount of descriptors for each document. Due to this, we are unable to set a constant parameter for the dimension of the output label set size, and have to resort to a ranking algorithm with a confidence factor cut-off in order to tackle this problem. Statistical analysis on the average number of labels in the training set might be an option to increase precision, as to restricting the number of labels if a document is found to be similar to a document family containing a lower number of labels on average, or reverse if that is the case. However, this approach will be left for further research.

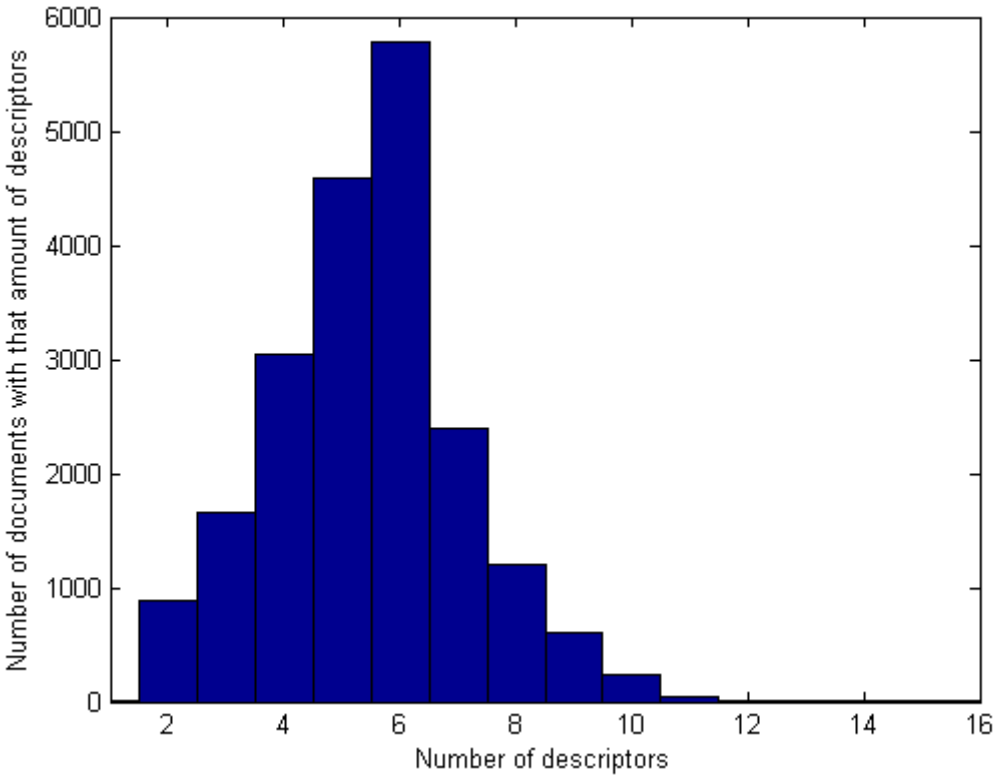

Figure 3.3: EuroVoc label distribution per document

The distribution of descriptors, as seen, ranges from a minimum of two (not a single document is assigned only one class label!) to a maximum of 10. On average, there

are 5.46 descriptors per document.

3.4. Absorbed Labels

A special case of label sparsity are absorbed labels. These labels are always found together with one (or more) other labels. This creates an issue, as the feature vectors for those documents will obviously be identical, and the non-zero feature sets for the absorbed labels will be a subset of non-zero feature sets for "parent" labels.

One would say that absorbed labels are a strong indicator of correlation between those labels, however after further analysis into the matter it has been found that out of 130 relations of absorbed labels, a large number of them do not have a semantic relationship. Therefore, it is safe to assume that this problem is in the dataset itself, and that it is simply by chance that currently the labels appear exclusively together. This is a special case of label sparsity and will be addressed later.

An example of absorbed labels follows. The absorbed label is on the left hand side of the arrow, while the "parent" label is found on the right hand side.

```
criminal_record -> information_transfer
Southern_Africa -> beef
Southern_Africa -> import_licence
Southern_Africa -> Kenya
South_America -> fresh_meat
South_America -> health_control
dismissal -> action_by_staff
comparative_analysis -> plant_propagation
foreign-exchange_reserves -> European_Central_Bank
biogas -> animal_product
cultural_pluralism -> cultural_policy
fodder_beet -> fodder_plant
labour_tribunal -> Court_of_Justice_of_the_European_Union
Euratom_Supply_Agency -> advisory_committee_(EU)
Euratom_Supply_Agency -> appointment_of_staff
denaturing -> alcohol
denaturing -> tax_exemption
denaturing -> excise_duty
Caucasus_countries -> the_EU's_international_role
Caucasus_countries -> diplomatic_representation
mass_media -> economic_concentration
mass media -> merger control
```

Figure 3.4: An example of absorbed labels

3.5. Label Relationships

Following absorbed labels and the HOMER algorithm, as well as various other multi-label classification algorithms that contained a clustering approach, a relation analysis between labels was made. In attempt to ease the heavy computational burden of having to calculate similarity measures over a large number of labels with an even larger number of features, an idea was created from seeing the large number of absorbed labels - the amount of times two labels appear together in the training set is a good indicator of their potential similarity, and can be used as a heuristic in order to speed up the learning process and only calculate similarity between labels when it is necessary.

Experimentally, it is shown that there is a significant correlation between the similarity of label profiles and the percentage they appear together in the training set. For the experiments, we used two fairly well known measures, cosine similarity and a slightly modified Jaccard index. The modified Jaccard index will be explained more in detail further on.

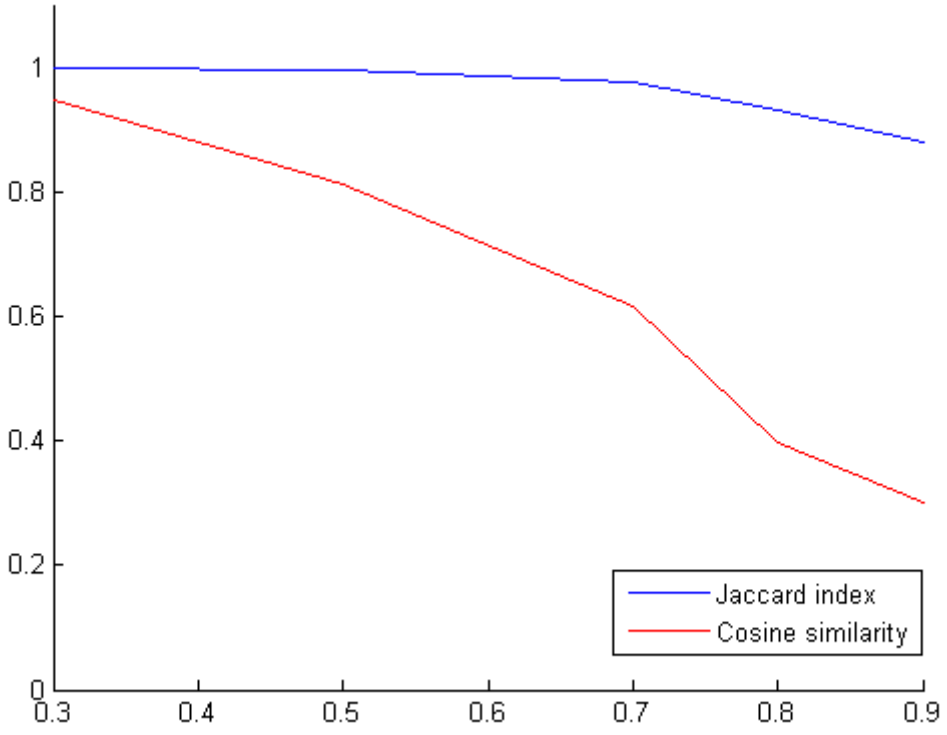

Figure 3.5: Correlation between label grouping and label profiles

The correlation was measured by calculating the amount of labels that have their similarity measure greater or equal than the ratio of times they appear together in the

dataset. Generally, the idea is to see whether it is safe to assume if two labels appear together often enough then they will have a large enough feature similarity. This is going to be useful as an optimization technique, as calculating the ratio of times two labels appear together is much faster than calculating any similarity measure between profiles of those labels.

The Pearson correlation coefficient indicates a weak, but existent positive correlation between those two variables, however a stronger connection is not necessary, as profile similarity will always be used as a final measure, while the ratio of times two labels appear together in the dataset serves merely as an indicator which labels could be grouped together, and not as a final measure.

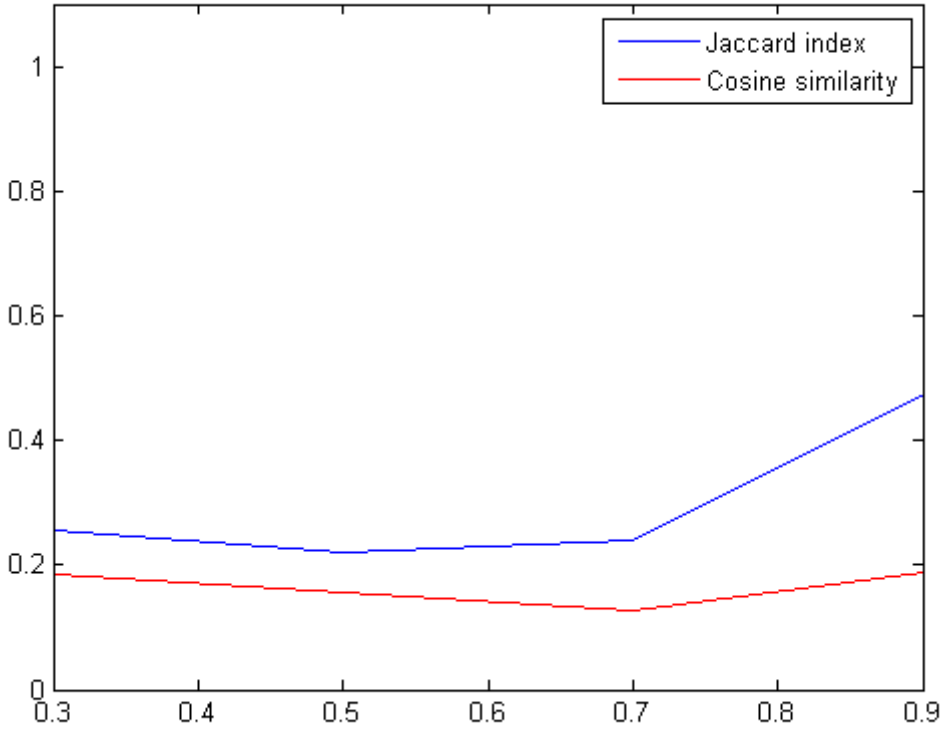

Figure 3.6: The Pearson correlation coefficient between label grouping and label profiles

3.6. Label Names

An interesting thing to note are the names of labels corresponding to the EuroVoc classification scheme. The full names often consist of multiple words (n-grams), however one could say that finding the same n-gram in a document would be a fairly good indi-

cator of the document being related to the corresponding class label. This leads us to a possibility of usage of weighted features extracted from label names, which is something we will analyse later on.

nut	olive oil	France
waste disposal	nuclear fusion	adoption of a child
wheat	food hygiene	wealth tax

Table 3.1: Examples of indicative label names

4. Preprocessing

4.1. JRC-Acquis

The Acquis Communautaire (AC) is the total body of European Union (EU) law applicable the the EU Member States. This is an everchanging collection of legislative text, dating from the 1950s until now. At the beginning of the year 2007, the EU has 27 Member States and 23 official languages. The parallelized Acquis Communautaire texts exist in these languages, and are recognizable between languages by their CELEX codes. The JRC-Acquis is a subset of the Acquis Communautaire, consisting of only those documents which have at least 10 translations in the languages of the Member States of the European Union.

In computational linguistics, parallel corpora is a very valuable resource, and usually it exists only for a small number of languages. The JRC-Acquis with its 22 languages and approximately 23,000 documents is the currently largest existing parralel corpus, taking into consideration the amount of languages and documents combined. The Language Technology team of the Joint Research Centre (JRC) has identified the documents which are part of the Acquis Communautaire, downloaded them and transformed them into the XML format. A sample of a document in the XML format is given below.

The XML format allows for easy extraction of various required fields. Since the point of research is the EuroVoc thesaurus, the fields extracted will be the CELEX code of the document, the EuroVoc classifications of the document the title and body of the text. The title is often a good indication of the topic matter and therefore is treated separately from the body of the text. The title, however, showed not to improve classification performance even when weighted, and was merged with the document body later on. All the extracted data was merged and saved as a single Comma Separated Value (CSV) file for easier parsing later on.

```
<TEI.2 id="jrc52006DC0014-en" n="52006DC0014" lang="en">
  <teiHeader lang="en" date.created="2007-04-24">
    <fileDesc>
      <titleStmt>
        <title>JRC-ACQUIS 52006DC0014 English</title>
        <title>Commission working document on a Community Action Plan on the
      </titleStmt>
      <extent>123 paragraph segments</extent>
      <publicationStmt>
        <distributor>
      </publicationStmt>
      <notesStmt>
        <note>Only European Community legislation printed in the paper
      </notesStmt>
      <sourceDesc>
    </fileDesc>
    <profileDesc>
      <textClass>
        <classCode scheme="eurovoc">11</classCode>
        <classCode scheme="eurovoc">3489</classCode>
        <classCode scheme="eurovoc">2300</classCode>
        <classCode scheme="eurovoc">5315</classCode>
        <classCode scheme="eurovoc">5462</classCode>
        <classCode scheme="eurovoc">2792</classCode>
        <classCode scheme="eurovoc">2830</classCode>
      </textClass>
    </profileDesc>
  </teiHeader>
  <text>
    <body>
      <head n="1">Commission working document on a Community Action Plan on
      <div type="body">
        <p n="2">[pic] | COMMISSION OF THE EUROPEAN COMMUNITIES |</p>
        <p n="3">Brussels, 23.01.2006</p>
        <p n="4">COM(2006) 14 final</p>
```

Figure 4.1: Sample of an XML document from the JRC-Acquis

4.2. Ignored Class Labels

Due to some labels being sparse, it is normal to expect those labels to underperform in classification. Statistical analysis over document count tresholds resulted in the following graph.

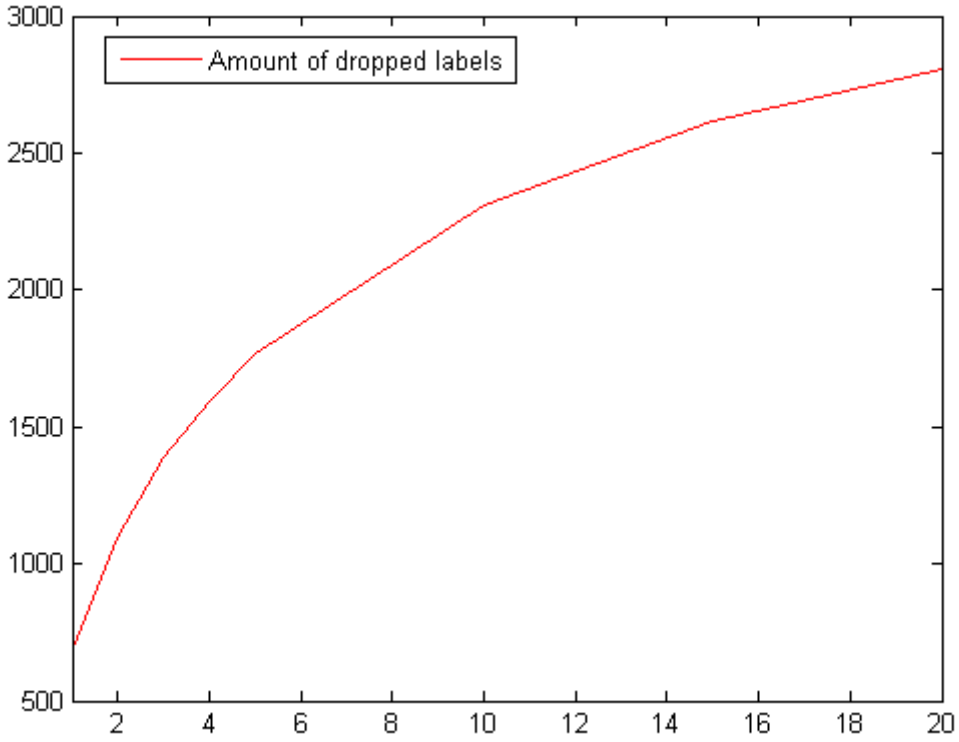

Figure 4.2: Amount of dropped labels depending on document count threshold

It is interesting to note the decline in the graph as the number increase, resulting from the fact that most sparse labels appear five or less times, with 686 labels having only one document indexed with that label.

4.3. Lexical Analysis

After extracting document bodies, the next necessary step is converting the text to a vector of words. This was done by using the Apache Lucene[1] package. By using the Apache Lucene TokenStream[2] , a sentence is automatically converted to a list of

[1] http://lucene.apache.org/core/

[2] http://lucene.apache.org/core/3_0_3/api/all/org/apache/lucene/analysis/TokenStream.html

tokens, in this case words to which a standard Porter Stem Filter (Porter, 1980) is applied, as well as the standard english stop word list. This results in a token vector for each document. In total, there are about 330 000 unique tokens, nearly 220 000 of them being words.

4.4. Feature Selection

Even though feature selection is often looked over in text classification, especially when using Support Vector Machines, as it is easier to create a definite border between class labels when there are more active features, in this case feature selection was a necessary step.

This is due to the fact that most multi-label classification algorithms implement some sort of a similarity analysis between labels, and calculating the similarity function for two sets P and Q is of $O(|S|)$, where S is the smaller of the two sets P and Q. Seeing as with 330 000 possible features and nearly 4000 possible labels, the overall complexity of finding similarities between every pair of labels is extremely large, some optimization methods have been used, one of them being feature selection through feature ranking by using various importance functions.

It is worth to note that even though feature selection was used a bit more severely in the case of multi-label algorithms, it was used lightly while combined with the one-vs-all SVM baseline algorithm, simply to fit memory constraints.

4.4.1. Feature selection Functions

Feature selection functions were used to assign a fitness value to each feature, in order to select only the features that will have the greatest impact in classification and clustering. The obvious candidate variables for the feature selection function are the number of times a feature appears in the dataset N_w, the maximum amount of times a feature has occured in any class label $\max_l(N_{w,l})$, the amount of labels the feature appears in $|L_w|$, the number of documents in which the feature appears in the dataset N_d and the minimum amount of documents the feature has occured in for all labels $\min_l(N_{d,l})$.

The reasoning for N_d and N_w are quite simple and obvious, the words that appear more often are going to be more valuable. However, the problem with those words is that they might appear in every label, making them eligible to be considered as stop

words. To skip the possible requirement of creating a manual stop word list, this was attempted to be done automatically, by creating a filter through which those words would not pass, and be considered as weak features. This was done by incorporating the amount of labels a feature has occured in, $|L_w|$.

However, even if a feature appears in many class labels, it is possible that it occured in many of them merely by accident. This is due to the fact that the same document always carries more than one class label, and therefore features are assigned membership to all of the class labels for the document. To evade that fact, we take into consideration the maximum amount of times a feature has occured in a label $\max_l(N_{w,l})$, and if the ratio of feature occurences for a specific label is large enough when compared to the total number of occurences, that means that the feature considered is still feasible.

Lastly, we come to the minimum number of documents the feature has occured in, $\min_l(N_{d,l})$. The reasoning behind this is not quite obvious, however it is added to combat label sparsity. In the cases of features which are significant for sparse labels, all of the aforementioned variables will be quite small in comparison to well represented labels, and the selection of a feature will be scaled by an amount inversely proportional to the minimum number of documents. Therefore, the features definin the sparse labels will almost always be on top of the feature selection ranking functions, since the scaling will be quite severe for well represented labels. However, this proves not to pose an issue, as this merely helps the sparse labels get attention, and in no way diminishes the performance for well represented labels, which will also have their more important features near the top.

Count Ranking

Count ranking is the simplest of ranking functions. The more times a feature appears, it is more valuable. This ranking function, of course, is not very efficient when combatting any of the aforementioned problems, however it was used in Mencía and Fürnkranz, 2010, and therefore is also included for comparison purpouses.

$$fitness_w = N_w \tag{4.1}$$

Improved Count Ranking

A possible improvenment to count ranking, and the second considered ranking function took into consideration the number of documents a feature appears in, the maximum number of times the feature appears for a label, and the square of the amount of labels

the word appears in. This was done to combat features dispersed over many labels, and it was quite efficient in doing so, however lacked performance for extracting features important for sparse labels.

$$fitness_w = \frac{N_w \times D_w \times \max_l(N_{w,l})}{|L_w|^2} \tag{4.2}$$

Diversity Threshold Ranking

The following ranking function incorporated a threshold for two of its variables, N_w and N_d. If the variables did not exceed the threshold, the fitness of the feature was set to zero, eliminating it from the feature set. This was done to combat features that either appeared a very low number of times, or in a few documents, but repeated often.

In combination with active thresholds on the number of documents required for a label to be considered in the classification process, this type of thresholding makes sense, and is related to the aforementioned document threshold for a label. Thresholding also relieves the ranking function of the necessity to take care of extremely sparse features, which would be quite difficult seeing as the point of the ranking function was to make features for sparse labels more visible. Creating a filter appeared to be the best solution.

If $N_w \leq W_{thresh}$ or $N_d \leq D_{thresh}$, $fitness_w = 0$, otherwise:

$$fitness_w = \frac{1}{\log\left(\max_l(N_{w,l})\right) \times \frac{\max_l(N_{w,l})}{N_w} \times \frac{1}{\min_l(N_{d,l})}} \tag{4.3}$$

Tf-idf

The standard term frequency - inverse document frequency function was also considered during feature selection, however due to many of its disadvantages compared to other methods it did not prove to be as efficient in extracting important features for sparse labels.

$$fitness_w = \frac{\log N_w + 1}{\log \frac{|D|}{N_d + 1}} \tag{4.4}$$

Delta Tf-idf

The delta tf-idf function was presented as to be used in sentiment analysis in Martineau and Finin, 2009.

Its advantage is differentiating between the amount of times a feature appears in a positive sentiment against the amount of times a word appears in a negative sentiment.

For multi-label classification, the definition of the function was extended to cover the amount of times a word appears in a label against the amount of times the word appears in all other labels.

$$fitness_w = \frac{\log N_w + 1}{\frac{N_{\bar{d}}}{N_d}} \tag{4.5}$$

Pearson's Chi-squared Test

The chi-squared test, also known as χ^2 is a statistical hypothesis test. The underlying idea of the chi-squared test in text analysis is to check whether the frequency distribution of certain events observed in a sample is consistent with some theoretical distribution.

Specifically, it is used to test if the occurence of a specific term and the occurence of a specific class label are independent. In cases of large χ^2 values, the hypothesis of independence is rejected, and therefore the term and the class label are dependent, in which case we select the feature for text classification.

We calculate the test value as follows for the general case:

$$\chi^2 = \frac{N(N_{11}N_{00} - N_{10}N_{01})}{(N_{11} + N_{01})(N_{11} + N_{10})(N_{10} + N_{00})(N_{01} + N_{00})} \tag{4.6}$$

Where

N_{11} = cases in which the term and the class label appear together
N_{00} = cases in which both the term and the class label don't appear
N_{10}, N_{01} = cases in which one of the two appears independent of the other
N = total number of cases

The χ^2 test was attempted to be used as a means of text classification, albeit unsuccessfully in Oakes et al., 2001, and is frequently used as a means of feature selection (ex. Meesad et al., 2011). The method, or the underlying idea of the method is used frequently throughout various functions in order to determine term - class label correlation.

4.5. Label Profiles

In order to determine the features associated with certain labels, and to be able to calculate various similarity functions, a profile is created for each label. The label

profile consists of the feature, the number of times the feature appears in all documents, and the number of documents for the label which contain the feature.

A sample of a text representation of the profile is as follows.

```
version 10 3
effect 3 3
until 3 3
forc 12 3
valid 9 3
from 4 3
have 7 3
trade 26 3
establish 3 3
good 6 3
code 7 3
come 6 3
numer 3 3
bind 3 3
document 3 3
base 3 3
```

Figure 4.3: A sample of a label profile

4.6. Post-processing

It is notable to mention the requirement for post-processing in the case of EuroVoc classification. In the EuroVoc hierarchy seen on Figure 3.1, in the documents where a child label appears, the parent label does not appear. It is assumed that when a child label is the document label, it implicates all of its hierarchical parents.

Therefore, an easy step to boost algorithm efficiency is to check after classification if a parent label as well as a child label are present in the resulting label set, and if they are, to remove the parent label.

However, due to the nature of the parent-child label relationships, those labels will never appear together, and in most cases, they will not be very similar. As seen on the aforementioned figure, even the label names of parents and children are not closely related. Therefore, even though this step is easily implemented, it does not affect the final classification efficiency noticeably.

5. Label Relationship Analysis

A problem which appeared often throughout EuroVoc classification, and is mentioned often earlier is the problem of label sparsity. To adress this problem, we will assume that the problem stems from the well-represented labels shadowing the sparse labels. Therefore, two approaches will be tested. The first approach is based on the idea of the HOMER algorithm as seen in Tsoumakas et al., 2008. The HOMER algorithm faces severe time complexity problems, as the balanced k-means approach does not converge very fast for a large number of labels and a large number of features. The idea taken from the algorithm was the hierarchical clustering approach, while instead of the cluster convergence method, a greedy approach was used.

The second method does not attempt to cluster labels, but instead build a full hierarchy from the more represented labels as the roots to the less represented ones as leaves. Thea idea of the method is that at various classification levels, the sparser labels will have better chances when competing only against other sparse labels, while the well represented labels will serve as guidelines which node should the classification tree visit next. Both of those algorithms will be analysed in detail further on.

Firstly, we will present the similarity functions used to analyse relations between label profiles. The relationships analysed will usually refer to parent-child label relationships, considering the well represented labels as parents, and the sparser ones as children in a hierarchy. This is done to balance the playing field by grouping labels of relatively same frequencies together for classification.

5.1. Similarity Functions

Similarity functions are used to determine similarity between profiles of labels. Profiles consist of feature sets, accompanied by normalized document and overall frequency of those features.

Based on those values, a similarity coefficient between two labels can be easily calculated. Various optimizations were used to attempt reduce overall complexity and

emphasise significant parts. Those optimizations will be adressed in detail further on.

5.1.1. Dot Product

The dot product is often defined in one of two ways: algebraically or geometrically. The geometric definition bases on the notion on the angle between vectors in space, and will be useful for the definition of the cosine similarity further on:

$$a \cdot b = ||a||\,||b|| \cos\phi \tag{5.1}$$

While the algebraic definition sums the multiplied values of vectors found on the same axis, as follows:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + ... + a_n b_n \tag{5.2}$$

5.1.2. Cosine Similarity

The cosine similarity function derives from the formula of the dot product, by dividing with the norms of a and b, we get the formula for cosine similarity:

$$\cos\phi = \frac{a \cdot b}{||a||\,||b||} \tag{5.3}$$

Or, in format of a sum:

$$\cos\phi = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.4}$$

A and B in the aforementioned formula represent label profiles. As we see, the cosine similarity function maps two sets to an interval $[0, 1]$, zero meaning no correlation, one meaning completely equal.

5.1.3. Penalized Cosine Similarity

Unfortunately, cosine similarity does not adress one important drawback of profile similarity. For instance, one would be correct to say that a feature that appears zero times for one set, but is significant for the other set, should be penalized more than a feature that appears a few times in one set, but is significant for the other set.

Simply, features which do not exist in one profile, but exist in the other could be treated as border features, ones that can easily differentiate between those two labels.

Therefore, the more border features exist, the easier will the two labels be separated. This was treated by adding a very simple addition to the cosine similarity, a weight factor called $penalization$. The idea stems from the penalty method approach for solving constrained optimization problems [1].

If A_i or B_i are equal to zero, $p_i = 5$. Otherwise, $p_i = 1$. The formula changes as follows:

$$\cos\phi = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} p_i \cdot A_i^2} \times \sqrt{\sum_{i=1}^{n} p_i \cdot B_i^2}} \tag{5.5}$$

5.1.4. Jaccard Index

Often discussed in literature are the pros and cons of using cosine similarity or the Jaccard Index (Hamers et al., 1989). The Jaccard index disregards numeric values of the set intersection, and only uses the relative sizes of the set intersection with regard to the set union. This formula is an interesting approach, and appears to be a route to an easier way of determining border features.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.6}$$

The drawback of the Jaccard index is that for the comparison of sparse labels to well-represented labels, the index will undoubtably be quite low due to the well-represented label having a larger amount of features, following the law of large numbers. Therefore, an optimization for time complexity and performance can be found via Mander's overlap coefficient.

5.1.5. Mander's overlap coefficient

The Mander's overlap coefficient (Manders et al., 1993) favors sparse label profiles by taking the smaller set size as the denominator instead of the union. This stems from the idea of absorption - we wonder to which measure is a profile of a sparser label absorbed by the profile of another label. From the idea, we notice a parent-child relationship forming through the idea of absorption.

The more well-represented label is often easier noticed, while the less represented one will get less attention due to the smaller amount of features. In this hierarhical route of classification we notice that as deeper we go through the relationship tree, we will have to be more careful about classifying. This notion will be adressed further on.

[1] http://en.wikipedia.org/wiki/Penalty_method

The formula for the Mander's overlap coefficient is as follows:

$$M(A,B) = \frac{|A \cap B|}{|\min_{A,B}|} \tag{5.7}$$

5.2. Importance Functions

Computing the similarity between two large vectors can be a time consuming task, and in case of label profiles, those vectors range up to 23 000 elements. Due to this, an attempt to use already existing information which is easier to calculate was made in the form of a measure called label connection. Label connection is the ratio of two labels appearing together in the training dataset against the total number of times either those labels appear. We will talk more about this measure in the next section.

5.2.1. Label Connection

Label connections are calculated by first creating a matrix of size $|L|^2$, where L is the set of labels of size n. For each label pair l_i, l_j, the amount of times those labels appear together in documents can be found on the index l_{ij}. It is easy to see that the diagonals will contain the amount of times a label appears in the training data. The matrix looks as follows:

$$\begin{pmatrix} l_{00} & \cdots & l_{0n} \\ \vdots & \ddots & \\ l_{n0} & & l_{nn} \end{pmatrix} \tag{5.8}$$

It is easy to see that the matrix is symmetrical, however in order to preserve an idea of a parent-child relationship, the label connection function between the label pair l_i, l_j will not be symmetric!

The label connection function between labels is defined as follows:

$$con(l_i, l_j) = \frac{l_{ij}}{|l_{ii}|} \tag{5.9}$$

Meaning, the label connection will be scaled by the norm of the first label of the pair. This allows for sparse labels to have very high label connection values to well-represented labels in cases of near-absorption and absorption, while this would not be the case had we scaled the fraction by a product of both norms, making the connection function symmetrical, however useless.

In cases of high values of label connection, it is safe to assume that the less represented label of the two can be put in a child relationship with the well represented label, skipping the part of calculating vector similarity. We will analyse two possible ways of using this technique, as a method of lowering the computational complexity and as a heuristic in combination with standard similarity functions.

5.2.2. Weighted Combination

One idea is to use the label connection function together with a similarity function, producing a weighted pair similarity measure as follows, for example with the cosine similarity:

$$similarity(l_i, l_j) = w_1 \cdot \frac{l_{ij}}{|l_{ii}|} + w_2 \frac{l_i \cdot l_j}{||l_i||\,||l_j||} \tag{5.10}$$

This approach does not reduce computational complexity, as both measures require calculation, however in edge cases the label connection function can be a good decisive measure. In tests, the arithmetic mean of those two measures was used, meaning $w_1 = w_2 = 0.5$, however the usage of both methods in combination resulted in severe time complexity and was discarded in favor of the thresholding approach.

5.2.3. Thresholding

Thresholding is an attempt to speed up clustering and hierarchy building algorithms. The general idea is to add a lower and upper threshold to the label connection function when determining similarity of labels. The assumption is that labels with values of the label connection function greater than the upper threshold will also have values of the similarity function greater than the upper threshold for the similarity function.

The lower threshold serves as a method of elimination - it is assumed that labels that do not appear often enough together have no strong relation, no matter how strong their similarity is. Values inbetween the thresholds will be further analysed by using a similarity function, as no definite decision was reached through the label connection function.

The general thresholding algorithm is as follows:

Algorithm 1 The thresholding method in combination with cosine similarity

```
1: function IsSimilar(l_i, l_j, w_l, w_u)
2:     s_0 ← con(l_i, l_j)
3:     if (s_0 < w_l) then
4:         return false
5:     else if (s_0 ≥ w_u) then
6:         return true
7:     else
8:         return IsSimilarCosine(l_i, l_j)
```

5.3. Clustering Algorithms

In the following section, a thorough analysis of existing clustering algorithms will be made, along with their weak and strong points, and a new greedy hierarchical clustering algorithm will be proposed. The main problem of label clustering is finding the right hierarchy and handling the outliers on the clusters. The idea for the greedy algorithm stems from the fact that instead of finding an optimal cluster distribution, in which the outliers will be difficult to reach from only one direction, it is possible to include the outliers into more than one cluster, the idea found in fuzzy clustering algorithms such as fuzzy c- and k- means.

However, another problem of those algorithms is their requirement of convergence. Calculating similarity between a lot of large vectors is a computationally heavy task, and therefore a greedy approach which does not reevaluate clusters might be more suitable for the task.

5.3.1. K-means Clustering

The k-means clustering algorithm, found often in literature since Hartigan and Wong, 1979. The general assumption of the algorithm is that n observed points can be partitioned into k clusters, in which each observation belongs to the cluster with the nearest mean. The problem is NP-hard, however with the use of various heristics, the convergence can be speeded up.

The algorithm starts by assigning k random cluster centers, finding points which belong to those centers, updating the cluster center as the arithmetic mean of all points

in the cluster, then repeating the process until the clusters converge. Since the distance is based on the cluster centre, the clusters are going to be circular in shape, as shown in the following example of clustering 500 points into 4 different clusters:

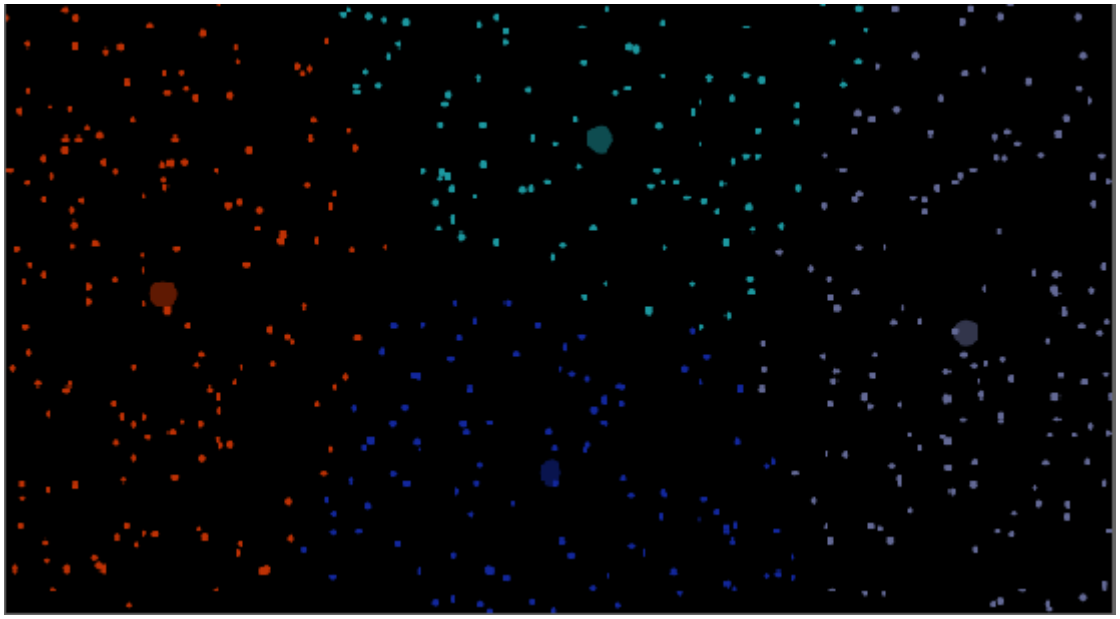

Figure 5.1: Points clustered using the k-means algorithm

As seen in the example, the main problem with the k-means clustering from a classification perspective are the outliers. The general idea behind clustering labels is to give sparse labels a better chance at being classified inside a cluster, however, since the outliers in the cluster are the points furthest away from the cluster center, it is hard to believe that they will get a good chance at being classified.

5.3.2. Balanced K-means Clustering

As stated before, the k-means algorithm has various problems, one of them being the speed of convergence. Therefore, various improvenments were considered to the algorithm (Kanungo et al., 2002), and some even in the field of label clustering, as found in Tsoumakas et al., 2008, containing the aforementioned HOMER algorithm.

The pseudocode of the balanced k-means algorithm for the HOMER algorithm is as follows

Input: number of clusters k, labels L_n, label data W_i, iterations it
Output: k balanced clusters of labels
for $i \leftarrow 1$ **to** k **do**
    // initialize clusters and cluster centers
    $C_i \leftarrow \emptyset$;
    $c_i \leftarrow$ random member of L_n ;
while $it > 0$ **do**
    foreach $\lambda \in L_n$ **do**
        for $i \leftarrow 1$ **to** k **do**
            $d_{\lambda i} \leftarrow \text{distance}(\lambda, c_i, W_i)$
        finished $\leftarrow$ false;
        $\nu \leftarrow \lambda$;
        while *not finished* **do**
            $j \leftarrow \arg\min_i d_{\nu i}$;
            Insert sort (ν, d_ν) to sorted list C_j;
            if $|C_j| > \lceil |L_n|/k \rceil$ **then**
                $\nu \leftarrow$ remove last element of C_j;
                $d_{\nu j} \leftarrow \infty$;
            else
                finished $\leftarrow$ true;
    recalculate centers;
    $it \leftarrow it - 1$
return $C_1, ..., C_k$;

Figure 5.2: Balanced k-means algorithm

The important points are the limit on the number of iterations (it), and the limit of labels in each cluster $\lceil |L_n| / k \rceil$, which assures that the outliers will be passed on to the next adequate cluster. These optimizations are good, however the speed of the algorithm is still an issue, and the algorithm has a fixed number of clusters - meaning that the outliers in one cluster are not going to create a new, separate cluster, but instead move to another one, possibly being outliers in that one aswell.

The problem is that since the algorithm in itself takes a significant amount of time, fine tuning the parameters of the algorithm takes an even more significant amount of time, and the algorithm is heavily not recommended for usage with higher numbers of labels such as is the case in the EuroVoc thesaurus. To combat this, the idea of a non-fixed number of clusters appeared, which will be addressed further on.

5.3.3. Fuzzy Clustering

Fuzzy clustering representes a class of clustering algorithms where the allocation of points to clasters is not hard in the sense that one point may belong only to one cluster. An open source implementation of the fuzzy k-means clustering algorithm exists in the Apache Mahout [2] package. Still, a downside of the implementation is the fixed number of clusters. An example of the result of the fuzzy clustering algorithm is as follows:

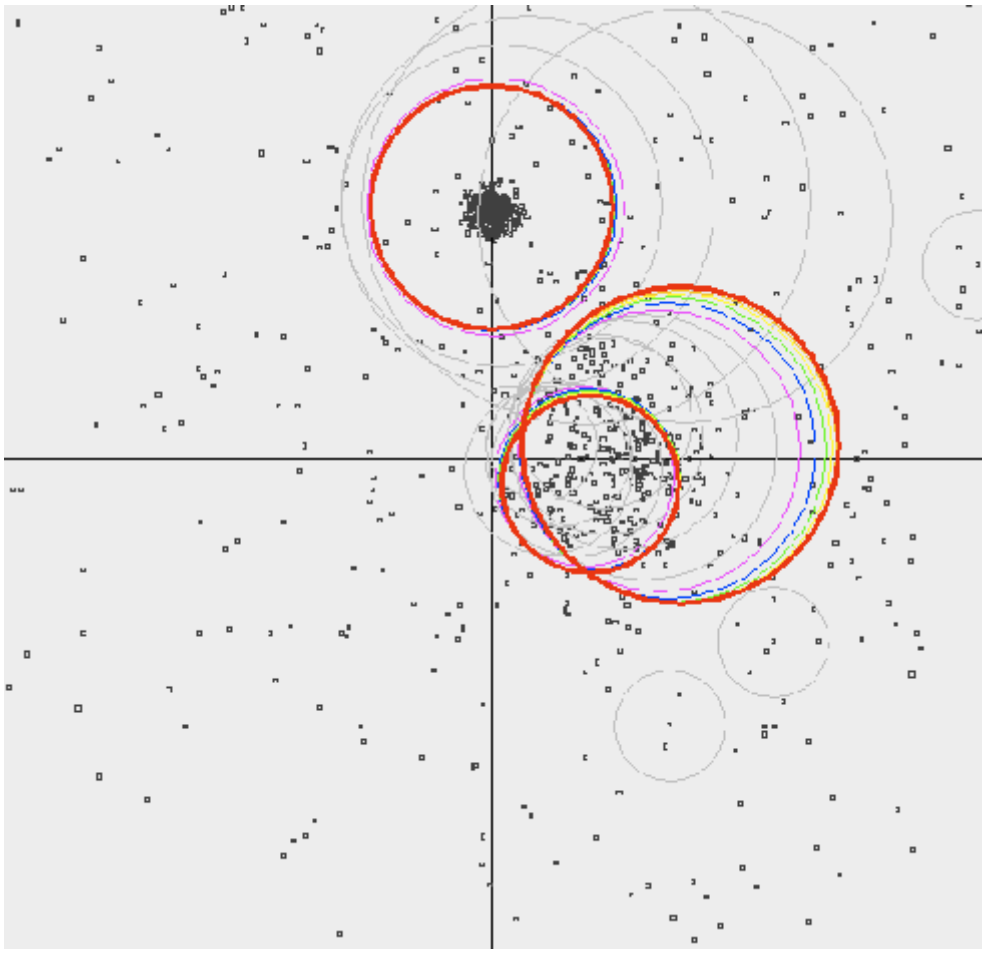

Figure 5.3: Fuzzy k-means algorithm example

The reason why fuzzy clustering is interesting is as follows. Imagine a class label l_i, existing as an outlier on a cluster C_0. The point is in itself close to outliers of clusters C_1, as well as C_2, meaning that it is borderline similar to those clusters as well.

A new document d_j will be classified as a member of a cluster if it is similar enough to the cluster centroid, however the outliers are not very similar to the cluster centroid. This sends an idea - what if the border for the outliers was extended to include other possible outliers from nearby clusters?

[2] https://mahout.apache.org

That way, the class label l_i can be reached not only through C_0, but also through C_1 and C_2, giving it adequate chances to be classified if d_j appears to carry significant similarity to l_i. This approach has no downside - as if d_j is not similar enough to l_i, it will not be classified with that class label no matter how many similarity checks are made, but if a similarity check is missed, it can easily be missclassified.

5.3.4. Greedy Hierarchical Clustering

The idea of greedy hierarhical clustering is to solve all of the aforementioned issues at the same time, while still retaining a good efficiency. It solves the time problems by having a linear complexity with regard to the number of labels. The algorithm only loops once through the labelset, constructs the cluster hierarchy and not once reconsiders it.

Possible errors are mitigated by adding the option of a label appearing in more than one cluster. As explained before, multiple occurences of labels in clusters do not hurt the classification, except by slowing the process down a bit, however it eliminates possible errors. The algorithm also does not have a fixed number of clusters, but rather constructs a new cluster when it deems necessary - when a label does not match the criteria to join any of the clusters, it becomes the center of a new cluster.

The mentioned looping through the labelset implicates that labels are added one by one to the hyperplane, and inserted into existing clusters, instead of adding them all initially, and then finding optimal clusters. The algorithm is as follows:

Algorithm 2 Greedy clustering method

```
1: function CLUSTER(labelSet, criteria)
2:     l_0 ← labelSet(0)
3:     c_0 = createCluster(l_0)
4:     clusterSet(0) ← c_0
5:     for all l_i ∈ labelSet do
6:         boolean success ← false
7:         for all c_j ∈ clusterSet do
8:             if similarity(l_i, clusterSet_j) > criteria then
9:                 success ← true
10:                insert(l_i, c_k)
11:        if ¬success then
12:            createCluster(l_i)
13:
14: function INSERT(l_i, clusterSet_j)
15:    boolean clustered ← false
16:    boolean success ← false
17:    for all c_k ∈ clusterSet_j do
18:        if similarity(l_i, c_k) > criteria_j then
19:            success ← true
20:            insert(l_i, c_k)
21:    if ¬success then
22:        clustered ← attemptCluster(labelSet_j, l_i)
23:    if ¬clustered then
24:        addToLabels(l_i)
25:
26: function ATTEMPTCLUSTER(labelSet_j, l_i)
27:    clusterSet ← l_i
28:    boolean added ← false
29:    for all l_k ∈ labelSet_j do
30:        if similarity(l_k, l_i) > criteria_j then
31:            clusterSet ← l_k
32:            added ← true
33:    return added
```

To keep the description of the algorithm as short as possible, extensive check methods have been omitted from the pseudocode, such as checking if a child cluster is the same size as the parent cluster, as creating another cluster identical to the parent cluster is useless, and the critera function for inserting into clusters.

The insertion into clusters is done by finding all similarities greater than the initial criteria, sorting them descending and gradually incrementing the criteria for each insertion. This is done to prevent a label from being inserted in too many clusters and therefore having a seriously negative effect on computational complexity (as each cluster will have a risk of becoming seriously overpopulated).

An example of a cluster with named labels is given in the following image:

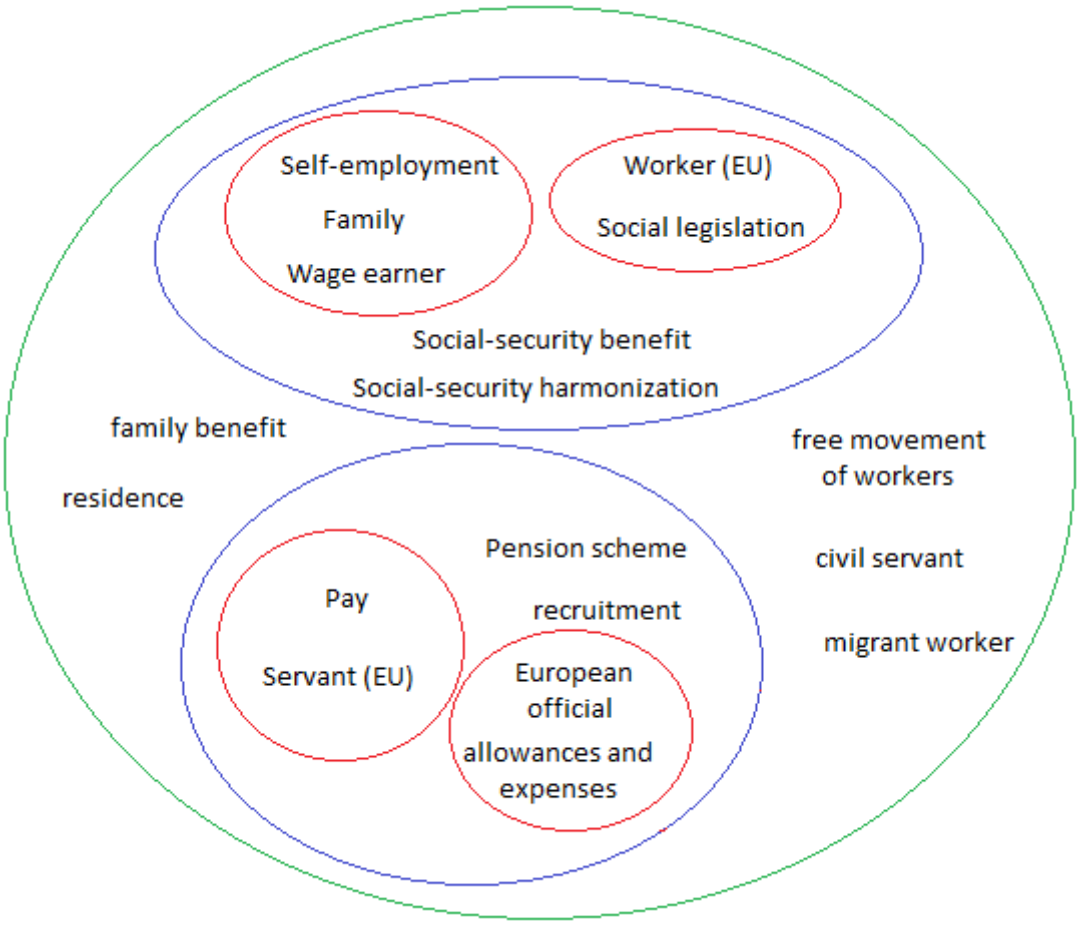

Figure 5.4: Resulting cluster example

The labels on the image represent the second (green), third (blue) and fourth(red) levels of hierarchy inside one cluster. The complete cluster at level one consists of 28 labels, all related to worker regulation and personnel administration. The full list of labels can be found in the addendum. In the end, there were 408 clusters at level 1,

271 clusters at level 2, 150 at level 3 and 70 at level 4.

Unfortunately, even with the optimization, the clustering process took 2 hours, 51 minute and 42 seconds, still a very large amount of time. With a stricter criteria, allowing only one level of cluster depth, the process took 1 hour and 4 minutes, however it was significantly worse.

For comparative purpouses, the BRkNN algorithm finishes a 5-fold evaluation in 48 minutes, the MLkNN algorithm takes 3 hours and 10 minutes for the same task, while the HOMER algorithm does not complete clustering with 50 centers in over two days. Various time problems will be analysed further later on.

6. Evaluation Measures

To determine the efficiency of classification algorithms it is required to calculate the ratio of actual class labels as opposed to ones determined by the classification algorithm. This is done by calculating the confusion matrix, which is also known as the contingency table or the error matrix.

Columns in the matrix represent predicted instances, while the rows represent actual data. It is easy to see which class labels the system is confusing, labeling them wrongly, and from this notion the name of the confusion matrix was derived.

	Condition Positive	Condition Negative
Outcome Positive	True Positive	False Positive
Outcome Negative	False Negative	True Negative

Table 6.1: Confusion matrix

An outcome is true positive in the case that the predicted class label is in fact present in the document, true negative in case the class label is not present either in the document or in the prediction, false positive if a class label is predicted, however not present and false negative if a class label is not predicted, however is present in the document.

6.1. Precision and Recall

The confusion matrix by itself is not a very often used measure of algorithm efficiency, but is however used as a means to calculate more complex measures. Two of those measures are precision and recall.

Precision, also called the *positive predictive value*, is the probability that a selected retrieved document is relevant. In other words, it is the probability of a classified document actually having the class label it was classified into. We define precision by using true positive and false positive amounts from the confusion matrix as follows:

$$Precision = \frac{tp}{tp + fp} \tag{6.1}$$

Recall, also known as *sensitivity*, is the probability that a selected document is retrieved in search. In other words, it is the probability of a document classification to be retrieved as such. Recall is defined by using the true positive and false negative amounts from the confusion matrix as follows:

$$Recall = \frac{tp}{tp + fn} \tag{6.2}$$

6.2. F-measure

F-measure, also known as the F_1-score or F-measure is a measure of a tests accuracy. It considers both the precision and the recall of the test in order to compute the score. The traditional F-measure is the harmonic mean of precision and recall and is calculated as follows:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{6.3}$$

In a general case, for a F_β measure, the formula changes to a general form as follows:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{6.4}$$

Converted to the confusion matrix value format, the equation takes the following form:

$$F_\beta = \frac{(1 + \beta^2) \cdot tp}{(1 + \beta^2) \cdot tp + \beta^2 \cdot fn + fp} \tag{6.5}$$

Also worth noting is the G-measure, the geometric mean between precision and recall, defined as follows:

$$G = \sqrt{precision \cdot recall} \tag{6.6}$$

6.3. Micro and Macro measures

The problem with applying the aforementioned measures is that they are defined for rather simple sets of data, single-label single-class or single-label multi-class classi-

fication problems (and various other similar problems). A problem appears when attempting to apply those measures to multi-label classification. There are two methods used to fix that problem, and they are micro- and macro- averaging.

The difference between those two is that macro-averaging gives equal weight to each class, while micro-averaging gives equal weight to each per-document classification decision. Due to the fact that F1 measure ignores true negatives and the fact that its magnitude comes from true positives, one can easily see that the micro-averaging method is dominated by well-represented class labels, as they will provide more true positives.

Macro-averaging, on the other hand, first calculates the measures, and then averages over them, equalizing the power of small and large classes in the final measure.

Imagine a measure M which is a function of values found in the confusion matrix, $M = f(tp, fp, tn, fn)$. Any one of the tree aforementioned measures (precision, recall, F-measure) falls under this generalisation. Then we would define the micro-measure of M as follows:

$$M_{micro} = M\left(\sum_{i=1}^{L} tp_i, \sum_{i=1}^{L} fp_i, \sum_{i=1}^{L} tn_i, \sum_{i=1}^{L} fn_i\right) \tag{6.7}$$

Where L would be, for instance, the number of class labels in a multi-label classification problem. On the other side, the macro-measure of M will be defined with the following equation:

$$M_{macro} = \frac{1}{L}\sum_{i=1}^{L} M(tp_i, fp_i, tn_i, fn_i) \tag{6.8}$$

7. Implementation

The implementation was divided in three steps. The first step was data extraction and formatting, the second a implementation of a indexing algorithm, and the third step, unrelated to the second step, was data conversion in order for the data to be used by open-source classification software.

7.1. Data Extraction and Formatting

Data extraction and formatting was required due to the fact that every document in the EuroVoc thesaurus was a single XML file, and the time cost of opening that many files is huge. Also, the XML files contained a lot of unnecessary trailing data which was simply discarded in the process. The extraction of data from XML files was done by using the SAXParser[1], and during the extraction some pre-processing steps were already taken.

The stop word list was applied on the extracted texts, as well as a stemming algorithm. The whole text was saved in one line of a file in the format of a word vector separated by spaces. At the end of the word vector, there was a single comma after which the relevant class labels for the document were listed, separated by spaces. A sample of the file is given in the following image.

```
council decis 23 januari 1995 concern common posit defin basi art
commiss decis 2 februari 1995 amend decis 93 231 eec author respe
commiss decis 10 februari 1995 lai down special condit govern imp
council decis 19 decemb 1994 conclus agreement form exchang lette
commiss decis 20 februari 1995 list establish former yugoslav rep
council decis 6 march 1995 conclus agreement form exchang letter
commiss decis 10 march 1995 amend decis 93 231 eec author respect
commiss decis 20 march 1995 concern implement annex council regul
commiss decis 17 march 1995 amend decis 94 621 ec protect measur
```

Figure 7.1: A sample of the ssv (space separated values) file

[1] http://docs.oracle.com/javase/7/docs/api/javax/xml/parsers/SAXParser.html

The whole process can be simply described by using a flowchart as follows:

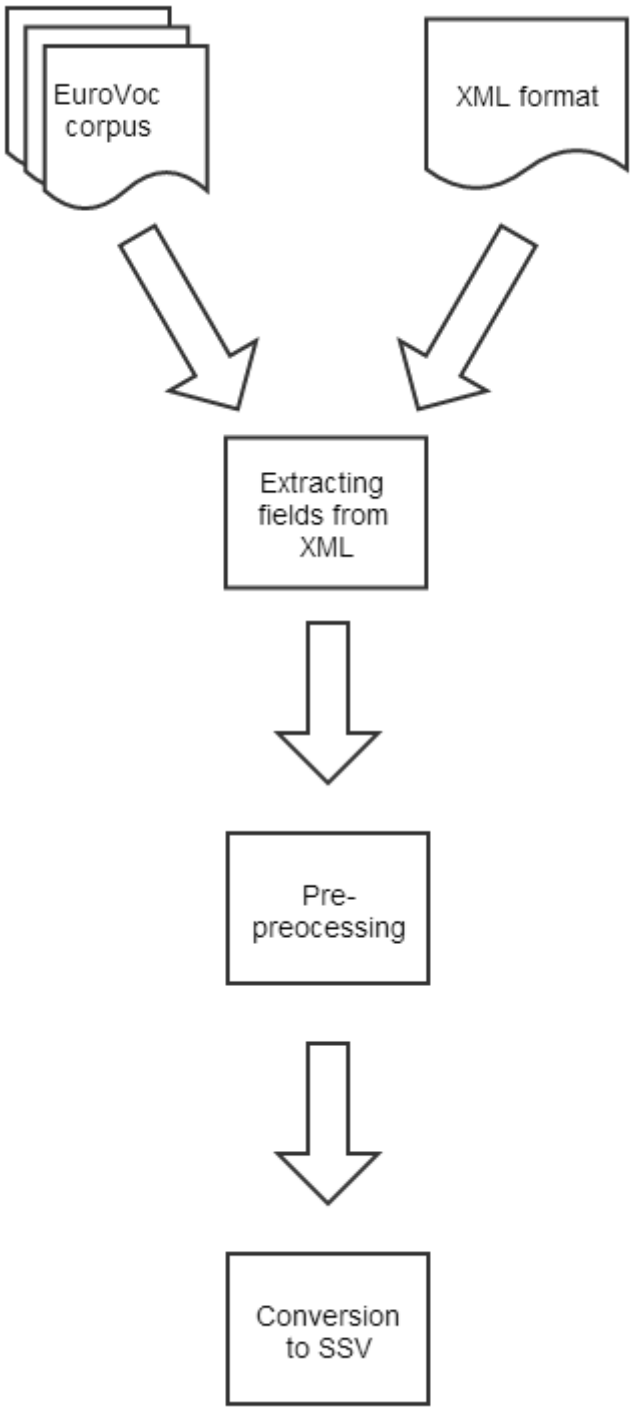

Figure 7.2: Data extraction flowchart

7.2. Algorithm implementation

The second step was the implementation of the document indexing system. This was done by splitting the dataset into a training set and a testing set with the ratio 9:1 in favor of the training set. The implementation can be shown by using a simple flowchart as follows:

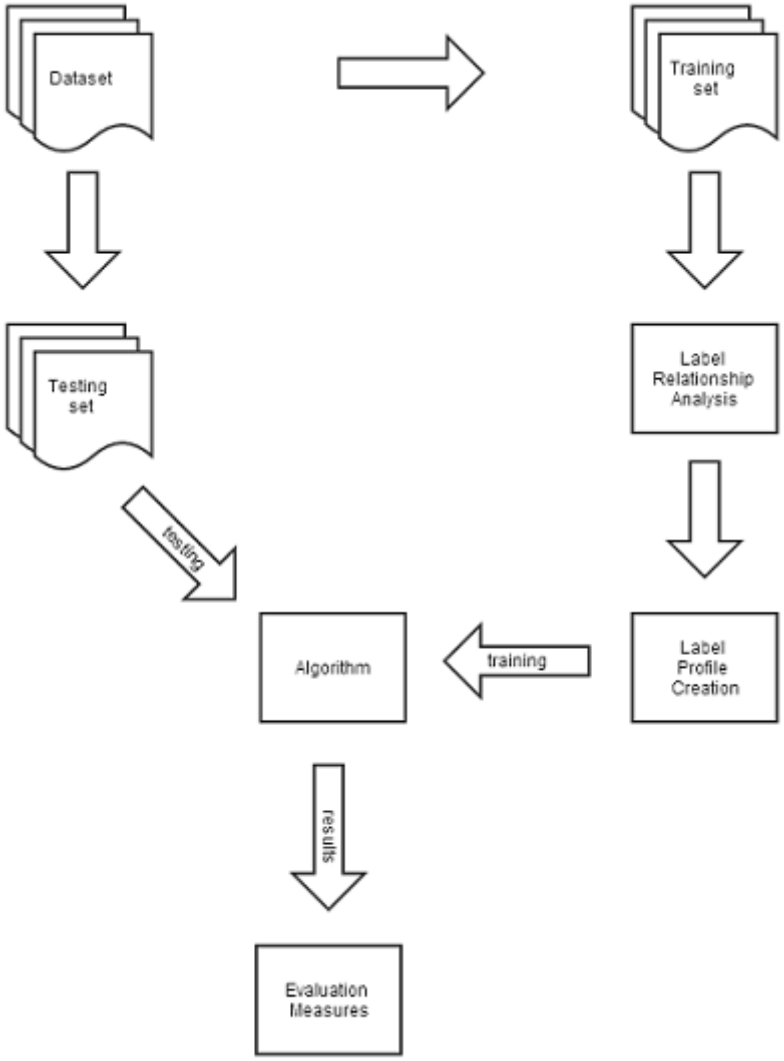

Figure 7.3: System implementation flowchart

7.2.1. Label Profile Similarity Ranking

Label Profile Similarity Ranking is an algorithm based on the work found in Cvitaš, 2005, where a combination of label similarity methods is used in order to index documents. Due to label clustering and hierarchical label organization, each of the clusters

and nodes in the hierarchy already has a profile calculated. This fact can easily be used in classification by using a combination of various similarity methods in order to complete the ranking.

The function is a combination of cosine similarity $cos(a, b)$, Mander's overlap coefficient $M(a, b)$ and the dot product of the vectors $a \cdot b$. The resulting linear function was tested with weights ranging from 0.1 to 0.7, by a step of 0.1, for each factor in the linear combination.

The function is as follows:

$$R(d_j, l_i) = 0.5 \cdot \cos(d_j, l_i) + 0.2 \cdot M(d_j, l_i) + 0.3 \cdot d_j \cdot l_i \tag{7.1}$$

The method was at first tested alone, without a backing clustering algorithm, to fine tune the parameters faster (as clustering takes a larger amount of time), and then the best parameters were tested in combination with the clustering algorithm.

However, the results of the combination were worse in comparison to the algorithm alone. The problem might lie in tuning the required confidence for a document to be classified with a label depending on the level of the hierarchy, however one run of the algorithm takes 5 hours, and there wasn't enough time during the research to test all possible options.

7.3. Data Conversion

Most open-source classification algorithms rely on a simple universal format for data input, the document-term matrix. Due to the high dimensionality of the problem, for every classifier, the sparse format of this matrix was used, in which the terms with value zero were omitted from the matrix, and only the non-zero terms remained. A brief description of the document-term matrix can be found in the following section.

7.3.1. Document-term Matrix

The document-term matrix is a matrix which describes the frequency of documents that occur in a collection of documents. In a document-term matrix, rows correspond to documents, while the columns correspond to terms. In the formats analyzed, we will add the class labels in the columns after the terms. The cells in the matrix are not necessarily just term frequencies of the terms in the document, as they are usually scaled or normalized by some method. For all algorithms, we used only the tf-idf method and the delta tf-idf method.

The general layout of the matrix is as follows:

$$\mathbf{D} = \overbrace{\begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n_d} \\ f_{21} & f_{22} & \cdots & f_{2n_d} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n_w 1} & f_{n_w 2} & \cdots & f_{n_w n_d} \end{bmatrix}}^{\text{documents}} \Bigg\} \text{words}$$

Figure 7.4: Document-term matrix

7.3.2. ARFF Format

The Atrribute-relation file format, or the ARFF format is a format used by Weka, an open-source machine learning software library written in Java. The difference between the ARFF format and the document-term matrix is that the ARFF format requires for each feature (or, term), as well as each class, to be named beforehand, and for their data type to be included. Data types can be numeric, enumerations, strings or dates with a corresponding date format. The class value attribute has to be named class, as the library was made for single-label learning.

The multi-label extension of the Weka library is the Mulan library, which in addition to the ARFF file, requires an additional XML file containing the label hierarchy, as well as the names of labels in the dataset. This is due to the fact that weka does not accept multiple class labels, and a workaraound is made by passing them as attributes, and considering them for classification later.

The ARFF file is split into two parts, the header which contains the information about classes and features, and the data part, which contains the document-term matrix. An example of those two parts is given as follows:

```
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

Figure 7.5: Arff header sample

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Figure 7.6: Arff document-term matrix sample

8. Results

During analysis, various algorithms have proven to have far too large time complexity for the fine tuning of their parameters.

The RAkEL (Tsoumakas and Vlahavas, 2007) algorithm successfully finished a 2-fold evaluation after 10 hours with a macro-averaged precision and recall of 0,32 and 0,35 respectively for the base parameters. However, tuning the parameter which controls the amount of random label-sets results in a greater time complexity, and the algorithm did not finish even the first fold in the same time, after which it was terminated and discareded.

The HOMER algorithm from Tsoumakas et al., 2008 did not finish the clustering process for three different cluster sizes for the balanced k-means algorithm, namely 3, 50 and 200. In each case the algorithm was terminated after 24+ hours without finishing the clustering process.

8.1. BRkNN

The Binary Relevance k-Nearest Neighbour algorithm proved to be the fastest performing algorithm. Proposed in Eleftherios Spyromitros, 2008, along with two modifications to the base algorithm, BRkNN-A and BRkNN-B.

The BRkNN-A algorithm uses a simple method, which however yields a significant improvement with regard to the standard algorithm, the addition is that in case of an empty prediction set due to none labels exceeding the threshold, the top ranked label is predicted as the result.

The BRkNN-B algorithm is more suitable for problems with less label variance, as the algorithm predicts the top n ranked labels based on the size of the neighbour's labelsets.

The time complexity of the algorithm is shown in the following table with respect to the size of the feature set.

Both algorithms both performed best on the feature set of size 100 000, and on that

Num. of features	5000	37000	100 000	200 000
Time required	22 $\pm$ 1 min	33 $\pm$ 1min	44 $\pm$ 1min	48 $\pm$ 2min

Table 8.1: Time Performance of the BRkNN algorithm

set an analysis on the parameter k (the number of neighbours) was carried out. The results can be seen in the following graphs, first for BRkNN-A, and then for BRkNN-B, both for 5-fold validation:

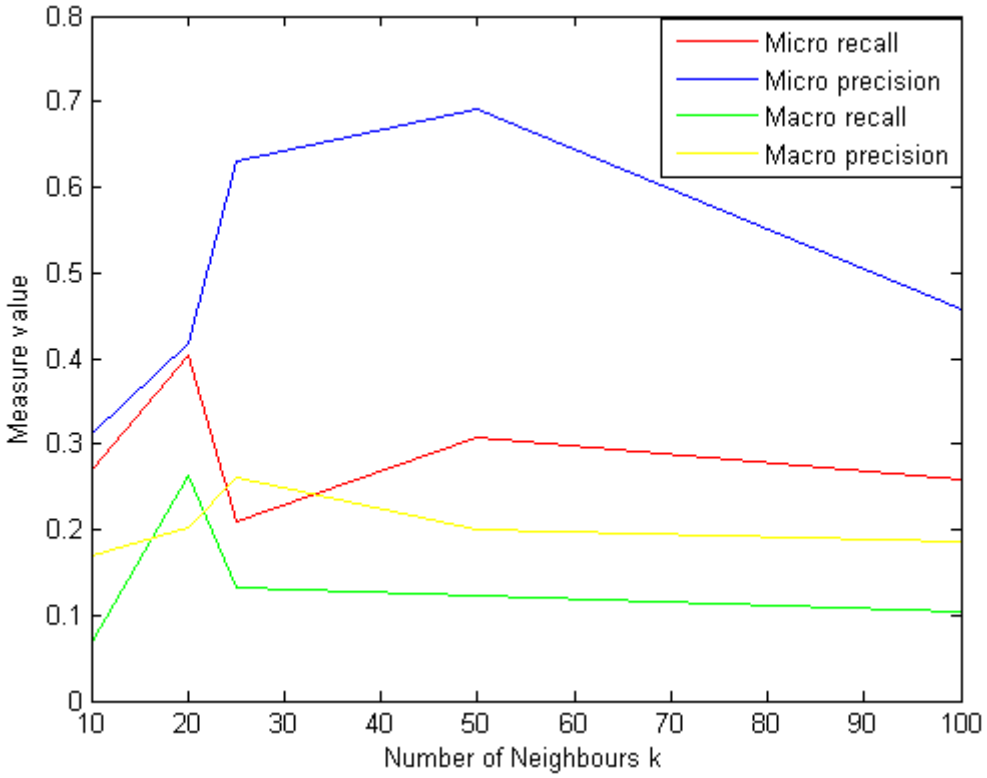

Figure 8.1: Performance of the BRkNN-A algorithm

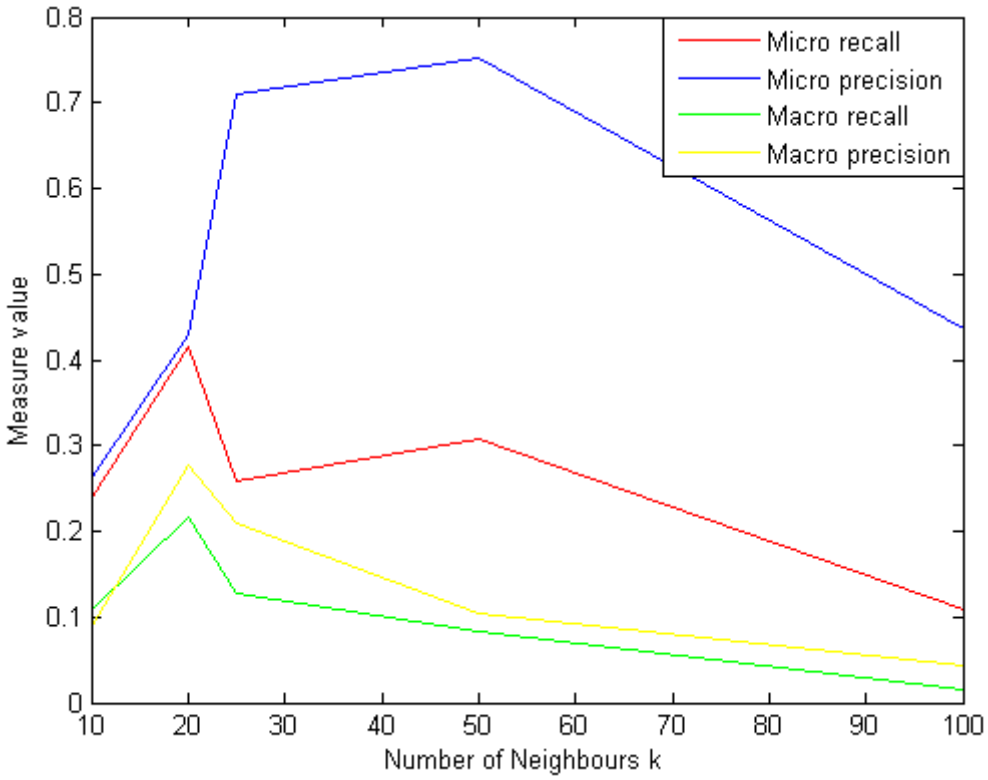

Figure 8.2: Performance of the BRkNN-B algorithm

8.2. MLkNN

Due to the underperformance in various tests and larger time complexity, the MLkNN algorithm was discarded after three different classification attempts. The average time to do a 2-fold validation of the MLkNN algorithm was 2 hours and 10 minutes for 200 000 features, and 1 hour and 30 minutes for 100 000 features.

The agorithm performed as follows on those two sets with $k = 25$ neighbours:

Num. of features	100 000	200 000
Micro F1	0.3218	0.3025
Macro F1	0.2827	0.2655

Table 8.2: Performance of the MLkNN algorithm

8.3. Support Vector Machine

A one-vs-all implementation of the Support Vector Machine algorithm was also used for a baseline approach, F1 measure of the algorithm are in the following table:

Micro	0.5491
Macro	0.4444

Table 8.3: F1 measures for the one-vs-all SVM algorithm

8.4. Label Profile Similarity Ranking

Label profile similarity ranking results will be shown as a standalone algorithm and in combination with the clustering algorithm for the best tuned hyperparameters of the algorithms. For feature selection, diversity threshold ranking was selected as it outperformed the count ranking feature selection method, both methods were used in combination with the top 100 000 features.

A feature labeling method was used in combination with the algorithm in order to scale the vastly dominant features for a certain label profile up, and to downscale them for other label profiles. Unfortunately, this method proved to increase performance in some cases, while decrease it in many other. The results were often of either close to 100% correct labels, or, more often, close to 0%. The feature labeling method was discarded pending further analysis.

Feature selection	Diversity treshold	Count ranking
Micro F1	0.3918	0.3725
Macro F1	0.3427	0.3355

Table 8.4: Performance of the standalone algorithm

The results were analysed over 5-fold evaluation, where firstly an instance of every label was taken into the learning set, in order to prevent labels which do not exist in the training set to appear in the testing set.

An interesting notion is that the algorithm is trained in less than 4 minutes, while one fold of evaluation takes 10 minutes. This brings the algorithm close to the speed of the BRkNN algorithm, and even improving its performance.

The results of the algorithm used in combination with the label clustering algorithm are as follows:

Fold	Fold 1	Fold 2
Micro F1	0.2918	0.3025
Macro F1	0.2627	0.2455

Table 8.5: Performance of the hybrid algorithm

9. Conclusion

Unfortunately, most of the algorithms could not compete with the Support Vector Machine. The Binary Relevance k-Nearest Neighbour algorithm has a good micro performance measure, however it lacks the macro performance in order to catch up completely. The problem of sparse labels, unfortunately, still persists.

The clustering algorithm did not yield satisfactory results as the performance is not as good compared to its counterparts, even compared to the underlying algorithm without the clustering method. However, the resulting clusters make sense label-wise, and the problem might be just to find the right way to exploit the label relations.

One of the ideas of this is a pure hierarchical composition of labels. A problem that the clustering algorithm faced was that it still needed to calculate similarities between labels and cluster centers occasionaly, something that would rather be avoided in order to keep a reasonably low computational complexity, and performance time. A purely hierarchical tree-wise composition would be able to use the label connection derived from the training set to a greater extent, and would possibly avoid some of the time trouble. Using the tree hierarchy in combination with a classifier, and having separate thresholds for classification and going deeper in a tree node is an idea for further research.

Otherwise, one of the bigger problems is the label cardinality per document when combined with sparse labels. The fact that each document has at least two labels attached to it, in combination with a large number of analysed sparse labels means that discriminatory features are going to be rare to find. This is why the feature labeling method is not working as efficiently as it was supposed to on the global level. Working on a more local level would require separate training for each tree node, which might be a problem with the large number of class labels.

A problem that was started to be tackled in Tsoumakas et al., 2008 with the use of the HOMER algorithm still exists and is not adequetely solved. The algorithms which have the means to fight sparse labels are too slow and therefore unusable for larger tasks such as this. A faster way of organizing a label hierarchy is the first step towards

solving the problem, and greedy methods and heuristics might just be the next step.

BIBLIOGRAPHY

Rodrigo C Barros, Ricardo Cerri, Alex A Freitas, and André CPLF de Carvalho. Probabilistic clustering for hierarchical multi-label classification of protein functions. In *Machine Learning and Knowledge Discovery in Databases*, pages 385–400. Springer, 2013.

Guido Boella, Luigi Di Caro, Leonardo Lesmo, and Daniele Rispoli. Multi-label classification of legislative text into eurovoc. In *Legal Knowledge and Information Systems: JURIX 2012, the Twenty-fifth Annual Conference*, volume 250, page 21. IOS Press, 2012.

Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Ana Cvitaš. Automatsko indeksiranje dokumenata u modelu vektorskog prostora. *diplomski rad, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, srpanj*, 2005.

Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602. ACM, 2008.

Gregory Druck, Burr Settles, and Andrew McCallum. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 81–90. Association for Computational Linguistics, 2009.

Ioannis Vlahavas Eleftherios Spyromitros, Grigorios Tsoumakas. An empirical study of lazy multilabel classification algorithms. In *Proc. 5th Hellenic Conference on Artificial Intelligence (SETN 2008)*, 2008.

André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687, 2001.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

Teresa Gonçalves and Paulo Quaresma. A preliminary approach to the multilabel classification problem of portuguese juridical documents. In *Progress in Artificial Intelligence*, pages 435–444. Springer, 2003.

Lieve Hamers, Yves Hemeryck, Guido Herweyers, Marc Janssen, Hans Keters, Ronald Rousseau, and André Vanhoutte. Similarity measures in scientometric research: the jaccard index versus salton's cosine formula. *Information Processing & Management*, 25(3):315–318, 1989.

John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.

Abbas Z Kouzani and Gulisong Nasireding. Multilabel classification by bch code and random forests. *International journal of recent trends in engineering*, 2(1):113–116, 2009.

Boris Lauser and Andreas Hotho. Automatic multi-label subject indexing in a multilingual environment. In *Research and Advanced Technology for Digital Libraries*, pages 140–151. Springer, 2003.

Tao Li, Chengliang Zhang, and Shenghuo Zhu. Empirical studies on multi-label classification. In *IcTAI*, volume 6, pages 86–92, 2006.

EMM Manders, FJ Verbeek, and JA Aten. Measurement of co-localization of objects in dual-colour confocal images. *Journal of microscopy*, 169(3):375–382, 1993.

Justin Martineau and Tim Finin. Delta tfidf: An improved feature space for sentiment analysis. In *ICWSM*, 2009.

Phayung Meesad, V NuiPian, and P Boonrawd. A chi-square-test for word importance differentiation in text classification. *Proceedings of Computer Science and Information Technology*, 6:110–114, 2011.

Eneldo Loza Mencía and Johannes Fürnkranz. *Efficient multilabel classification algorithms for large-scale problems in the legal domain*. Springer, 2010.

Ebrahim Mohamed, Maud Ehrmann, Marco Turchi, and Ralf Steinberger. Multi-label eurovoc classification for eastern and southern eu languages. *Multilingual Processing in Eastern and Southern EU Languages–Low-resourced Technologies and Translation. Cambridge Scholars Publishing, Cambridge, UK*, 2012.

Alessandro Moschitti and Roberto Basili. Complex linguistic features for text classification: A comprehensive study. In *Advances in Information Retrieval*, pages 181–196. Springer, 2004.

Gulisong Nasierding, Grigorios Tsoumakas, and Abbas Z Kouzani. Clustering based multi-label classification for image annotation and retrieval. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 4514–4519. IEEE, 2009.

Michael Oakes, Robert Gaaizauskas, Helene Fowkes, Anna Jonsson, Vincent Wan, and Micheline Beaulieu. A method based on the chi-square test for document classification. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 440–441. ACM, 2001.

Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

Frane Šarić, Bojana Dalbelo Bašić, Marie-Francine Moens, and Jan Šnajder. Multi-label classification of croatian legal documents using eurovoc thesaurus. In *Proceedings of SPLeT-Semantic processing of legal texts: Legal resources and access to law workshop*, 2014.

Artur Šilić, Jean-Hugues Chauchat, Bojana Dalbelo Bašić, and Annie Morin. N-grams and morphological normalization in text classification: A comparison on a croatian-english parallel corpus. In *Progress in Artificial Intelligence*, pages 671–682. Springer, 2007.

Ralf Steinberger, Bruno Pouliquen, and Johan Hagman. Cross-lingual document similarity calculation using the multilingual thesaurus eurovoc. In *Computational Linguistics and Intelligent Text Processing*, pages 415–424. Springer, 2002.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. *arXiv preprint cs/0609058*, 2006.

Ralf Steinberger, Mohamed Ebrahim, and Marco Turchi. Jrc eurovoc indexer jex-a freely available multi-label categorisation tool. *arXiv preprint arXiv:1309.5223*, 2013.

Michal Toman, Roman Tesar, and Karel Jezek. Influence of word normalization on text classification. *Proceedings of InSciT*, pages 354–358, 2006.

Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Machine Learning: ECML 2007*, pages 406–417. Springer, 2007.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, pages 30–44, 2008.

Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 18(10):1338–1351, 2006.

Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

Multi-label Document Classification using EuroVoc Thesaurus

Sažetak

EuroVoc je višejezični konceptualni pojmovnik pravnih dokumenata zemalja članica Europske Unije. Sastoji se od preko 6000 različitih oznaka klasa koje se dodjeljuju dokumentima, zadatak koji spada u područje strojnog učenja. Problem pojmovnika EuroVoc je visoko rasipanje oznaka te njihova neravnomjerna raspodjeljenost. Slabije reprezentirane oznake su teže za prepoznati i klasificirati, te najviše škode procesu klasifikacije. Razni načini poboljšanja postupka su analizirani s naglaskom na povezivanje sličnih oznaka u grupe te potom strožu diskriminaciju među članovima iste grupe. Rezultati su uspoređeni s osnovnim algoritmom stroja s potpornim vektorima, te su izneseni zaključci i analizirani mogući novi smjerovi istraživanja.

Ključne riječi: EuroVoc, Klasifikacija Teksta, Neravnomjerna Distribucija Klasa, Strojno Učenje

Multi-label Document Classification using EuroVoc Thesaurus

Abstract

EuroVoc is a multilingual conceptual thesaurus covering the activities of the European Union. It consists of over 6000 different class labels which are assigned to documents, a problem which is tackled by machine learning algorithms for multi-label classification. Problems with the thesaurus include label sparsity, inconsistent descriptor distribution and various other. The main line of research was an attempt to group similar labels in order to split the problem into differentiating between groups of labels and differentiating between similar labels. The model was compared to the SVM baseline and various other multi-label classification algorithms, the results and possible new lines of research were analysed.

Keywords: EuroVoc, Text Classification, Multi-label Classification, Label Sparsity, Machine Learning