

Contents

1	Introduction	8
1.1	Problem statement	9
1.2	Publications	9
1.3	Related work	10
1.4	Authorship	10
1.4.1	On the language of this thesis	11
2	Theoretical foundations of SVM classifiers	12
2.1	Statistical learning	12
2.2	Classification as supervised learning method	13
2.3	Support Vector Machines	13
2.3.1	Linearly separable Support Vector Machines	14
2.3.2	Non-linearly separable Support Vector Machines	17
3	System workflow	20
3.1	The input	22
3.2	Structure extraction	22
3.2.1	Character extraction	22
3.2.2	Page segmentation	22
3.2.3	Reading order resolving	22
3.3	Initial zone classification	25
3.4	Metadata classification	26
3.5	References extraction	27
3.6	Content extraction	28
4	Classification workflow	29
4.0.1	Generating samples	29
4.0.2	Data scaling	29
4.0.3	Data sampling	30

4.0.4	Transformation to SVM ^{LIGHT} format	30
4.1	Parameter optimization	31
4.1.1	Training of the final classifier	32
4.2	Feature selection	32
5	Dataset	44
5.1	Motivation	44
5.2	Related work	45
5.3	Pubmed database	45
5.4	Dataset creation	45
5.5	GROTOAP2 properties	46
5.6	The methodology of creating GROTOAP2	46
5.7	Filtering	50
5.8	TrueViz file format	50
5.9	Dataset evaluation	57
6	Evaluation	59
6.1	Initial classification evaluation	59
6.2	Metadata classification evaluation	61
6.3	Future work	63
6.3.1	GROTOAP2	63
6.3.2	CERMINE	63
 Appendices		
A	Description of the training workflow	66
B	Description of classification features	69
C	Reading order resolving	74
D	Project resources	84
D.1	CERMINE repository	84
D.2	CERMINE interface	84
D.3	GROTOAP2	86
D.4	This thesis	86
E	GROTOAP2 classes distributions	87

Chapter 1

Introduction

Digital libraries gained a lot of momentum in the recent years. The amount of publications held there is enormous and constantly growing. Collecting them in digital repositories raised many problems, whereas many of them go beyond acquisition issues - they have to be organized and classified in order to improve various retrieval procedures. Moreover, with time the purpose of the digital libraries has been evolving from a place where publications were stored, over search engines used to quickly filter out publications relevant for certain topic, to frameworks used for analysis of collaboration patterns, researchers' professional network analysis, calculation of productivity and efficiency factors such as H-index ([J. 05]) or G-index or looking for emerging trends and topics in science. All this research depends on a reliable source of full article texts and corresponding metadata, being both an input for analysis algorithms.

Even though there was a huge progress in science as such, scholarly publishing relies on methods invented more than two decades ago, which is an immense time in digital world. The most common format in digital publishing is PDF¹, which focuses on how the document is rendered in reader's browser making it highly portable across hardware and software platforms, but without putting too much focus on metadata, which are understood here literally as *data about data*. Articles are very often published and distributed in this format resulting in losing metadata. Their recovery can be performed either manually by a human worker or by a computer program.

Thus, a modern, fully functional framework for digital libraries, in order to be useful and to provide high quality services, has to have access not only to the full texts of the articles and books that it stores, but also to their metadata. The metadata, data describing data, include such information as: document authors,

¹Portable File Format

document title, document abstract, keywords, document references and others. The purpose of obtaining the metadata is twofold. Firstly, they are indispensable when implementing a search engine, being an invaluable tool for all researchers looking for articles needed in their research field. Secondly, a subset of metadata, i.e. article authors and references are needed for various ranking algorithms based on citation count, such H-index [J. 05], Impact Factor and other bibliometric indicators.

1.1 Problem statement

Regrettably, usually a digital library has to deal with the resources without any metadata provided by the publisher. Even when we consider specialized libraries, whose collections are usually modest, the number of articles makes it barely possible to process them manually. What is more, it might also happen that the metadata available are not trustworthy, are faulty or partially missing. In these cases a digital library needs a way to obtain them automatically or semi-automatically.

This thesis presents CERMINE - a system for automatic extraction of metadata from scholarly publications, as well as GROTOAP2 - Ground-truth publication dataset. At its core CERMINE uses supervised machine learning algorithms, where labeled training samples from GROTOAP2 are used to train the system to recognize and assign individual elements of an article into classes depending on their function in an article. In a multi-stage process CERMINE extracts metadata, full text and bibliographic references from an input article. The system was developed jointly by the author and his team mates as a part of work duties in Interdisciplinary Center for Mathematical and Computational Modeling at the University of Warsaw. However, their work will not be described extensively in this document and the authorship of each piece of the system will be marked explicitly.

1.2 Publications

The system has been recently presented at the Force2015 meeting ([For15]). Furthermore, an article on CERMINE was presented in the 11th IAPR International Workshop on Document Analysis Systems ([Dom14]). GROTOAP2 article was published in D-Lib magazine, volume 20. ([DB14]). Information on the access to the source code and other resources is included in the appendix D.

1.3 Related work

Extracting metadata from scholarly publications is a well-studied problem. In the past, the algorithms expected scanned documents as an input and were generally formulated as image recognition problem. Those were built in the period when scholarly communication was present mainly or entirely in a printed form and each article had to be scanned before becoming available in a digital way. Such approaches were presented in [Tho01] and [FZM⁺07].

Nowadays, a digital library has to cope with born-digital documents, where the letter recognition stage is omitted and processing starts with building up words and lines of text based on single characters. In the past several approaches using machine learning methods were presented.

In [EFB08] there is presented a method for processing PostScript and PDF files using various machine learning methods in order to extract metadata.

[Mar09] extracts basic metadata from scientific papers by, first of all, using various rules to perform segmentation of characters extracted from text and then employing a neural network to assign classes to blocks of text. Results are corrected by comparing them with an external bibliographic database.

In the approach proposed by [CX10] PDF files are first transformed into HTML format, then metadata is extracted using Hidden Markov Model. The authors test their approach using a set of 458 articles for VLDB conferences.

Both [HGM⁺03] and [HMZ⁺05] present an approach of employing mixed Hidden Markov Model/Support Vector Machines algorithm to perform metadata and references extraction. Unlike in our approach, they do not take sequence-oriented information into consideration.

Finally, the TeamBeam algorithm, portrayed in [KJHG12], uses a Maximum Entropy classifier to include sequence-related information in the classification. This algorithm is applied only to the first page of an article, therefore focusing exclusively on metadata.

1.4 Authorship

The work enclosed in this thesis has been implemented as a part of author's work duties in the Interdisciplinary Center for Mathematical and Computation Modeling at the University of Warsaw, in a team managed by Łukasz Bolikowski and technically lead by Dominika Tkaczyk. Although the system has been created by a team of people, the attention was always paid to keeping distinctive portions of work sharply separated in terms of the authorship. Therefore it is possible to indicate which parts of the system were authored by whom. As a general rule, components of the system that were not created by the author of this thesis are not enclosed in

the thesis or are described very briefly, with the authorship named explicitly. Since CERMINE is a successor of a legacy system based on Hidden Markov Models, some parts of it were inherited and possibly remodeled. Below we follow components that were not created by the author of the thesis:

- Interface with PDFBox for character extraction,
- Text segmentation and structure extraction,
- Web interface,
- Reference extraction.

On the contrary, the following components were entirely designed and implemented by the thesis' author:

- Research on the related work (section 1.3),
- Design and implementation of the reading order resolution (section 3.2.3, appendix C),
- Implementation of the classification features (section 4.2, appendix B),
- Design and implementation of the classification workflow using Support Vector Machines (chapter 4),
- Design and implementation of the classifiers' training workflow (appendix A)
- Preparation and characterization of the training data set, i.e. GROTOAP2 (chapter 5, appendix E),
- Optimization of the classifiers (section 4.1),
- Evaluation of the initial and metadata classifiers (chapter 6).

1.4.1 On the language of this thesis

In this thesis the first-person pronoun in plural form (“we”) has been used on several occasions. This form was adopted to follow the guidelines of scientific writing in english language. This thesis was written entirely by its author. Similarly, the depicted components of CERMINE were designed, developed and tested entirely by the author of this thesis.

Chapter 2

Theoretical foundations of SVM classifiers

CERMINE at its core uses Support Vector Machines - a machine learning approach to classify data from the input articles, optimized and trained on an immense set of scholarly publications. Below we describe its theoretical foundations to encourage the reader to get acquainted with the algorithm in details.

2.1 Statistical learning

The main objective of statistical learning is to find a non-trivial description of dependencies between data gathered by measuring objects and the objects themselves. The measurement, known also as input data, is known for all of the objects under research. The property that is being looked for, on the contrary, is known uniquely for a subset of objects. The main goal of the machine learning is to figure out, in an automated way, an algorithm allowing to reason the values of the properties of interest for all available input objects.

As a classical example of machine learning we might take an application in the medical diagnosis. In this case, the input data are the data obtained by performing medical analysis and medical interview. The outcome of the algorithm is a probability of patient suffering from certain disease.

2.2 Classification as supervised learning method

The goal of all classification methods is to determine to which category a new observation belongs, being given a training set consisting of samples belonging to some predefined categories. Training samples have form of n -dimensional vectors, where n is the number of explanatory variables, known as well as *features*. Features are values that express some traits of the classified elements. They may be real-numbers (e.g. 1.23, 3.14), integer-numbers (e.g. 1,2,3) or categorical values (e.g. Circle, Square, Trapezoid, Large, Medium, Small, 1,0 etc.).

In the theory of statistical learning, classification is considered as an example of supervised learning tasks, i.e. learning on the basis of training set for which the categories of elements are available. An example of unsupervised learning task is clustering, which consists in joining input elements into cluster, without any prior knowledge nor training data, using only objects' features and similarities between them. One need to note here that clustering and classification can be interchanged with each other.

2.3 Support Vector Machines

One of the most successful and broadly known techniques in machine learning are Support Vector Machines, whose theory was developed by Cortez and Vapnik in [C.C95]. The basic idea is as follows: being given a set of two classes of N -dimension points find a hyperplane which separates an N -dimensional space into two subspaces, so that each of them contains points belonging to only one class (also called category). Since the two classes might not always be linearly separable, SVM introduces an idea of kernel-induced space which casts the points into a higher dimensional feature space where the new points are linearly-separable. In the section 2.3.1 there is a detailed description of the theoretical foundations. Originally the SVM classifier is a binary classifier, meaning that each point in the training set, as well as all the unknown sample, belong to one and only one of two object classes. However, many implementations bypass this constraint by using a series of SVM classifiers, for instance $n - 1$ classifiers to classify points into n classes. Their approach is as follows: first classifier classifies points as class 1 or non-class-1. If a point is classified as non-class-1, it is classified by the second classifier as class 2 or non-class-2 etc. The classifier $n - 1$ classifies points as $n - 1$ or n .

2.3.1 Linearly separable Support Vector Machines

In the very basic example of SVM, namely a linear SVM, the task is to assign each new sample to one of the two classes, being given a set of learning samples. All the points, both learning samples and the unknown samples, are usually tuples of values of certain features.

Let us assume that we have a set of learning samples L , where each sample is labeled l_i , whereas i is the index of the learning sample and I is the set of these indices. Each training sample has d values, meaning that it is a d -dimensional tuple ($x_i \in \mathbb{R}^d$) and a class label y_i being one of two values $-1, 1$. We can sum up these definitions with the following equation:

$$\forall i \in I \quad (x_i, y_i) : x_i \in \mathbb{R}^d, y_i \in \{-1, 1\} \quad (2.1)$$

For the sake of simplicity let us assume that the set of learning samples is linearly separable i.e. for two groups of points X_{-1} and X_1 , belonging to decision classes -1 and 1 respectively, there exist $d + 1$ real numbers, such that every point $x \in X_0$ satisfies $\sum_{i=1}^n w_i x_i > k$ and every point $x \in X_1$ satisfies $\sum_{i=1}^n w_i x_i < k$, where x_i is the i -th component of x . The property of linear separability can be easily visualized in a 2-dimensional space - the points are linearly separable if there exists a line that divides a plane so that one group of points is on one side and the other group is on the other side.

We will label the separating hyperplane as H . Clearly, in a general case there is an infinite number of such hyper-planes what is shown in the figures 2.1 (in this case $I = 5$ and $d = 2$) and 2.2. The two dimensions are marked as $x[0]$ and $x[1]$.

We can define the hyperplane H with a general formula:

$$w \cdot x + b = 0 \quad (2.2)$$

Noticeably, the number of such hyperplanes is infinite. In addition, let us consider two hyperplanes parallel to each other that separate the space. They will be labeled as H_{-1} and H_{+1} , whereas the former is stuck to a point from class -1 and the latter to a point belonging to class $+1$ and both are parallel to the hyperplane H . These hyperplanes are shown in the figure 2.3.

They can be characterized with the following equations:

$$w^T x_i + b = 1 \quad (2.3)$$

$$w^T x_i + b = -1 \quad (2.4)$$

Because H_{-1} and H_{+1} are decision boundaries, there are no points in between them. This can be summarized in the following equations:

$$w^T x_i + b \geq 1 \quad \forall (x_i, y_i) : y_i = 1 \quad (2.5)$$

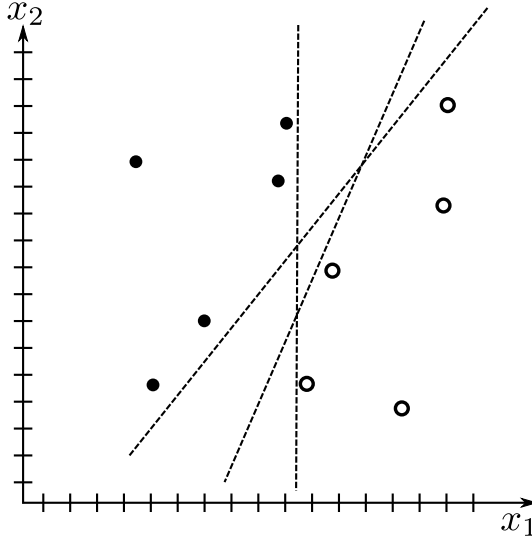


Figure 2.1: There might exist an infinite number of hyperplanes separating two groups of points. SVM's task is to find a hyperplane that maximizes distance to all data points.

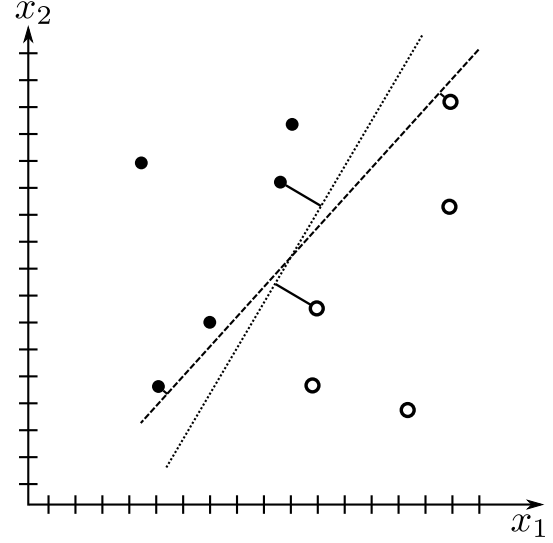


Figure 2.2: There were chosen two alternative lines separating the training points. The dotted line maximizes the distance to support points, whereas the dashed one is suboptimal.

$$w^T x_i + b \leq -1 \quad \forall (x_i, y_i) : y_i = -1 \quad (2.6)$$

We can unify equations 2.5 and 2.6 by one generalized equation:

$$y_i (w^T x_i + b) - 1 \geq 0 \quad (2.7)$$

For each data point we might calculate the distance to any of the hyperplanes:

$$d((w, b), x_i) = \frac{y_i (x_i \cdot w + b)}{\|w\|} \geq \frac{1}{\|w\|} \quad (2.8)$$

To make the decision boundary most accurate, we are intuitively looking for a hyperplane that maximizes the distance to all the data points. This in turn minimizes the probability of an erroneous classification of unknown data and avoids introducing a bias. According to the equation 2.8 this can be achieved by maximizing $\frac{1}{\|w\|}$ or alternatively by minimizing the $\|w\|$ vector. We can leverage the fact that $\|w\|$ is a non negative value and, for sake of simplicity of calculations, minimize $\frac{1}{2}\|w\|^2$ subject to $y_i (w^T x_i + b) \geq 1 \forall i$ (note that $\|w\|^2 = w^T w$).

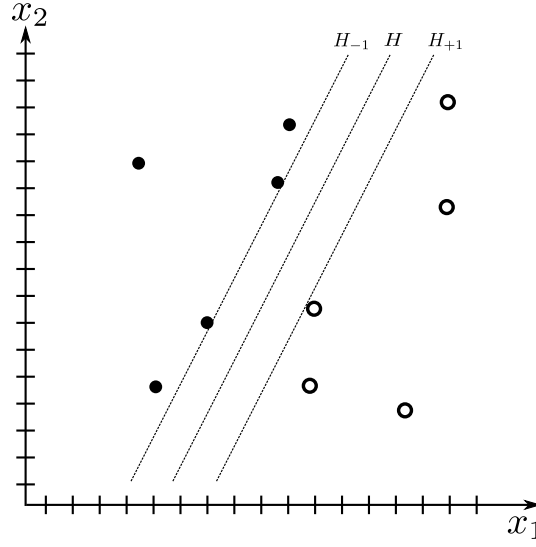


Figure 2.3: Hyperplane H separates perfectly points belonging to two classes. It is equally distanced to hyperplanes H_{-1} and H_{+1} which are stuck to objects of classes -1 and $+1$ respectively.

We will solve this problem with the method of Lagrange multipliers. The Lagrangian is calculated as follows:

$$\mathcal{L} = \frac{1}{2}w^T w + \sum_{i=1}^n \alpha_i \left(1 - y_i (w^T x_i + b)\right) \quad (2.9)$$

In this equation α is a vector of Lagrange multipliers. The problem is reduced to finding a saddle point where the function has its maximum value. Thus, by setting the Lagrangian to 0 with respect to w and b we get:

$$w = \sum_{i=1}^n \alpha_i (-y_i) x_i = 0 \Rightarrow w = \sum_{i=1}^n \alpha_i y_i x_i \sum_{i=1}^n \alpha_i y_i = 0 \quad (2.10)$$

If we substitute $w = \sum_{i=1}^n \alpha_i y_i x_i$ in the above formula with \mathcal{L} we have:

$$\begin{aligned} \mathcal{L} &= \frac{1}{2} \sum_{i=1}^n \alpha_i y_i x_i^T \cdot \sum_{j=1}^n \alpha_j y_j x_j + \sum_{i=1}^n \alpha_i \left(1 - y_i \left(\sum_{j=1}^n \alpha_j y_j x_j^T x_i + b\right)\right) \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \alpha_i y_i \sum_{j=1}^n \alpha_j y_j x_j^T x_i - b \sum_{i=1}^n \alpha_i y_i \\ &= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^n \alpha_i \end{aligned}$$

This equation can be used to define a dual problem:

$$\begin{aligned} \max \quad & -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^n \alpha_i \\ \text{subject to } & \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

The problem above is a quadratic programming problem that always has a solution. Points x_i with non-zero α_i are called support vectors. These are the points that determine the decision boundary.

2.3.2 Non-linearly separable Support Vector Machines

The assumption regarding linear separability of training points defined in 2.3.1 cannot be always met. In the section below we will present an approach that does not require points to be linearly separable. The basic idea is to allow misclassification, i.e. wrong classification, but also to minimize the classification error on the same time. This method is known as Cost-SVM or soft margin method. In general, it allows the points to be on the *wrong* side of the hyperplane by introducing a penalty function. Let us label with ξ_i the error of classification of the i -th point, i.e. its distance to the classification plane in case when it is misclassified. This variable is called *error variable* or *slack variable*. In addition, let us label with C a trade-off parameter between the error and the margin. Now, the minimized function is expressed as follows:

$$\frac{1}{2} ||w||^2 + C \cdot \sum_{i=1}^n \xi_i \quad (2.11)$$

with the following constraints:

$$y_i (w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i > 0 \quad (2.12)$$

The dual problem becomes the following:

$$\begin{aligned} \max. \quad & W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{subject to } & C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i x_i = 0 \end{aligned}$$

Non-linear Support Vector Machines

So far we considered only linear SVM, i.e. those with linear decision boundaries. While this solution is very simple from the computational and conceptual point

of view, the results yielded by the method might be suboptimal. In this section we will present a generalized version of the algorithm, in which training points x_i will be transformed to a higher dimensional space. This will allow performing a linear division of the transformed feature space, which will be equivalent to a non-linear operation in the original space. In other words, we will use a hyper-surface instead of a hyper-plane, e.g a curve in a 2-dimensional space. An example of such transformation is shown in the figure 2.4.

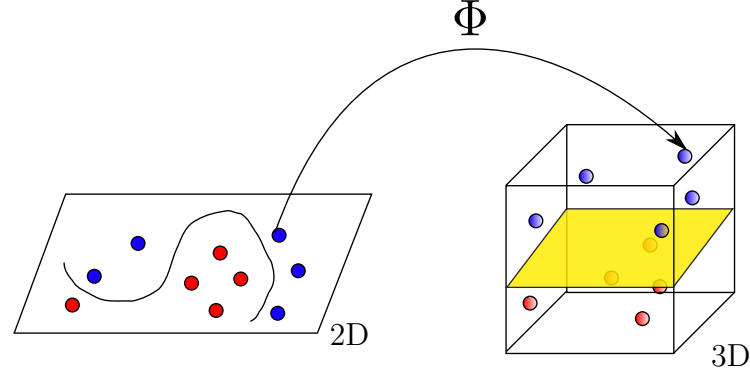


Figure 2.4: An example of a transformation from a two-dimensional to a three-dimensional space. In the original space the points are clearly not linearly separable. After applying a transformation kernel, they can be separated with a plane marked with yellow color. With SVM several common kinds of kernels are applied to find the best way of transforming the input data into a problem with distinct borders.

If we come back to the equation 2.11 we will see that the data points appear only as inner products $(x_i^T x_j)$. This means that there is need to know the mapping to the higher dimensional feature space explicitly.

Let us define the kernel function \mathcal{K} as follows:

$$\mathcal{K}(x_i, x_j) = \phi(x_i)^T \cdot \phi(x_j) \quad (2.13)$$

Under certain circumstances it is feasible to obtain ϕ analytically being given \mathcal{K} . As an example let's take $\mathcal{K}(u, v) = (1 + u^T v)^2$. Then when $n = 2$, i.e. $u = (u_1, u_2)$ and $v = (v_1, v_2)$ we get:

$$\begin{aligned} \mathcal{K}(u, v) &= (1 + u^T v)^2 \\ &= 1 + u_1^2 v_1^2 + 2u_1 v_1 u_2 v_2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 \\ &= (1, u_1^2, \sqrt{2}u_1 u_2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2)^T (1, v_1^2, \sqrt{2}v_1 v_2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2) \\ &= \phi(u)^T \phi(v) \end{aligned}$$

A kernel function \mathcal{K} must be symmetric, continuous and have a positive definite Gramian Matrix. If the kernel function does not satisfies these conditions, then the dual problem might have no solutions. We can distinguish three most commonly used kernel function families: polynomial, sigmoid and radial (called also RBF). A polynomial kernel is expressed as follows:

$$\mathcal{K}(x_i, x_j) = (\gamma x_i^T x_j + c)^d \quad (2.14)$$

where the parameters are:

- c is a zero-degree term,
- d is the degree of the polynomial,
- γ is a generic coefficient used to parametrize the kernel.

When $d = 1$ we say that a kernel is linear. This case has been referred to in the section 2.3.1.

The most common variant of radial kernels is a gaussian distribution:

$$\mathcal{K}(x_i, x_j) = e^{-\gamma |x_i - x_j|^2} \quad (2.15)$$

where γ is often replaced with $\frac{1}{2\sigma^2}$, so it can be interpreted as a variation around distribution's deviation.

Chapter 3

System workflow

CERMINE implements a workflow which is composed of a multi-part pipeline, where every stage is treated as a separate module and can be maintained and developed independently. For implementation of the majority of non-trivial problems we used supervised and unsupervised machine learning algorithms, which gave us a lot of elasticity and allowed adapting to new document layouts.

CERMINE workflow consists of the following distinguishable stages:

1. Character extraction - takes place at the very beginning of metadata extraction process. CERMINE uses `iText` - an external library for manipulating PDF content - to extract a stream of characters together with their geometrical properties (size and position) from the input files.
2. Text segmentation - builds up blocks of text from a set of characters using their position and size. This stage uses Docstrum algorithm, coded by Krzysztof Rusek and described in details in [O’G93]. It is therefore beyond the scope of this work and will not be mentioned hereafter.
3. Resolution of the reading order - figures out the order in which blocks of text are meant to be read. While this task is trivial to a human reader, it is tricky to implement an elastic algorithm suitable for different article layouts.
4. Basic structure extraction - takes a PDF file on the input and produces a geometric hierarchical structure representing the document. The structure is composed of pages, zones, lines, words and characters. The reading order of all elements is determined. Every zone is labeled with one of four general categories: `METADATA`, `REFERENCES`, `BODY` and `OTHER`.
5. Metadata extraction - part analyses parts of the geometric hierarchical structure labeled as `METADATA` and extracts a rich set of document’s metadata

from it.

6. References extraction - part analyses parts of the geometric hierarchical structure labeled as **REFERENCES** and the result is a list of document's parsed bibliographic references.
7. Content extraction - part analyses parts of the geometric hierarchical structure labeled as **BODY** and extracts document's body structure composed of sections, subsections and paragraphs.

A diagram of the above-mentioned workflow can be found in the figure 3.1.

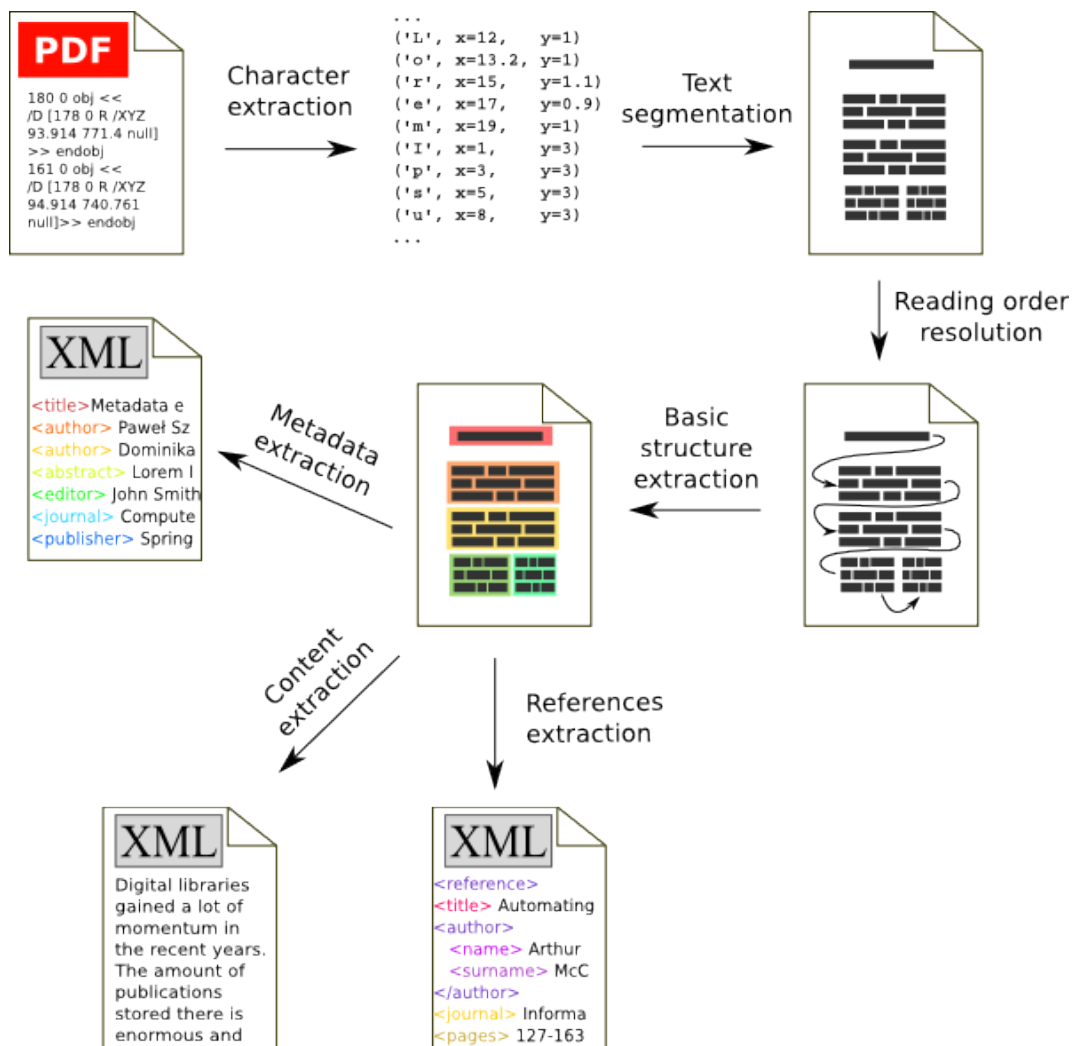


Figure 3.1: A schematic diagram of CERMINE's workflow.

3.1 The input

CERMINE uses TrueViz documents as an input for the classification training workflow. We motivate choice of this format in 5.8. As depicted in 5.2 TrueViz files consist of document's text together with its geometrical properties and classification label. The text is divided hierarchically into: pages, zones, lines, words and chunks of text.

3.2 Structure extraction

Below we characterize superficially the consequent stages of the CERMINE workflow, from PDF files as input to ready-to-use metadata as output.

3.2.1 Character extraction

PDF format as such consists of a stream of characters and their geometrical properties. Order of the character in the stream does not have to be related, and usually is not, to its position in the text. Extraction of the characters is implemented using `iText` library and consists in obtaining a set of characters together with their geometrical properties from an input document. This part of the system was entirely implemented by Dominika Tkaczyk and will not be characterized in this work.

3.2.2 Page segmentation

Page segmentation is a task of clustering a set of characters into blocks of text. CERMINE uses internally the Docstrum algorithm, described in details in [O'G93]. It is a bottom-up approach based on nearest-neighborhood clustering of connected components extracted from the document. The text lines are found based on a threshold of the angle between component centroids. Then, histograms of components spacing are used to detect inter-character distance in a word and between words. The algorithm has a set of thresholds that have to be estimated based on experiments with various documents. The algorithm was implemented by Krzysztof Rusek and tuned by Dominika Tkaczyk and therefore will not be described in this document in details.

3.2.3 Reading order resolving

Resolution of Reading Order is a process aiming at transforming text zones from a two-dimensional space, i.e. as they are laid out on the paper, to a single-dimensional

space, i.e. in an order meant by its author and as they are read by a human. Intuitively, this is done by going from left to right and from top to bottom. However, there are a lot of corner cases that make this naïve approach inaccurate, e.g. multi-column layouts, mixed single-multi column layouts, page numbers, textual elements of figures, tables, figures' and equations' labels.

As we already described in [Dom14], an input PDF file contains a stream of characters that undergo processes of extraction and segmentation. This results in a list of pages consisting of zones, lines, words and chunks of text. These elements are meant to be put together in the same order as that would be done by a human reader.

Algorithm for resolution of the reading order

Below a heuristic responsible for setting order of text zones is characterized. The full implementation of the algorithm is included in the appendix C.

1. For each pair of zones inside a single page boundaries the amount of free space between them is calculated. On the figure 3.2 this space is illustrated as the black area between two zones.
 - (a) Formally the free space S can be expressed as $S = A_d - (A_1 + A_2)$, where A_1 and A_2 are areas of the two zones and A_d is area of the smallest rectangle in which two objects can fit. This rectangle has its sides parallel to the X and Y axes.
 - (b) To include preference for vertical clustering (and to avoid joining blocks horizontally while working with multi-column layouts) cosine of the angle α between the vector \vec{v} connecting two objects' geometrical centers and vector \vec{x} being a projection of the vector \vec{v} on the X axis is calculated. This was illustrated in the figure 3.3. The formula $\cos \alpha = \frac{\vec{v} \cdot \vec{x}}{|\vec{v}| |\vec{x}|}$ is employed.
 - (c) Free space S is multiplied by sum of coefficient M and $\cos \alpha$. Coefficient M is introduced in order to prevent situations where more than two zones have the same width and lay in-line with respect to the Y axes. Then, for each of them the calculated distance would be equal to zero. In such cases we prefer to join these two groups between which the Euclidean distance is minimized.
2. A list L containing triples (O_1, O_2, D) is created (where O_1 and O_2 are zones on the page and D is the distance between zones). Initially there are $\binom{N}{2}$ elements in the list, where N is the total number of zones on a page. graphics

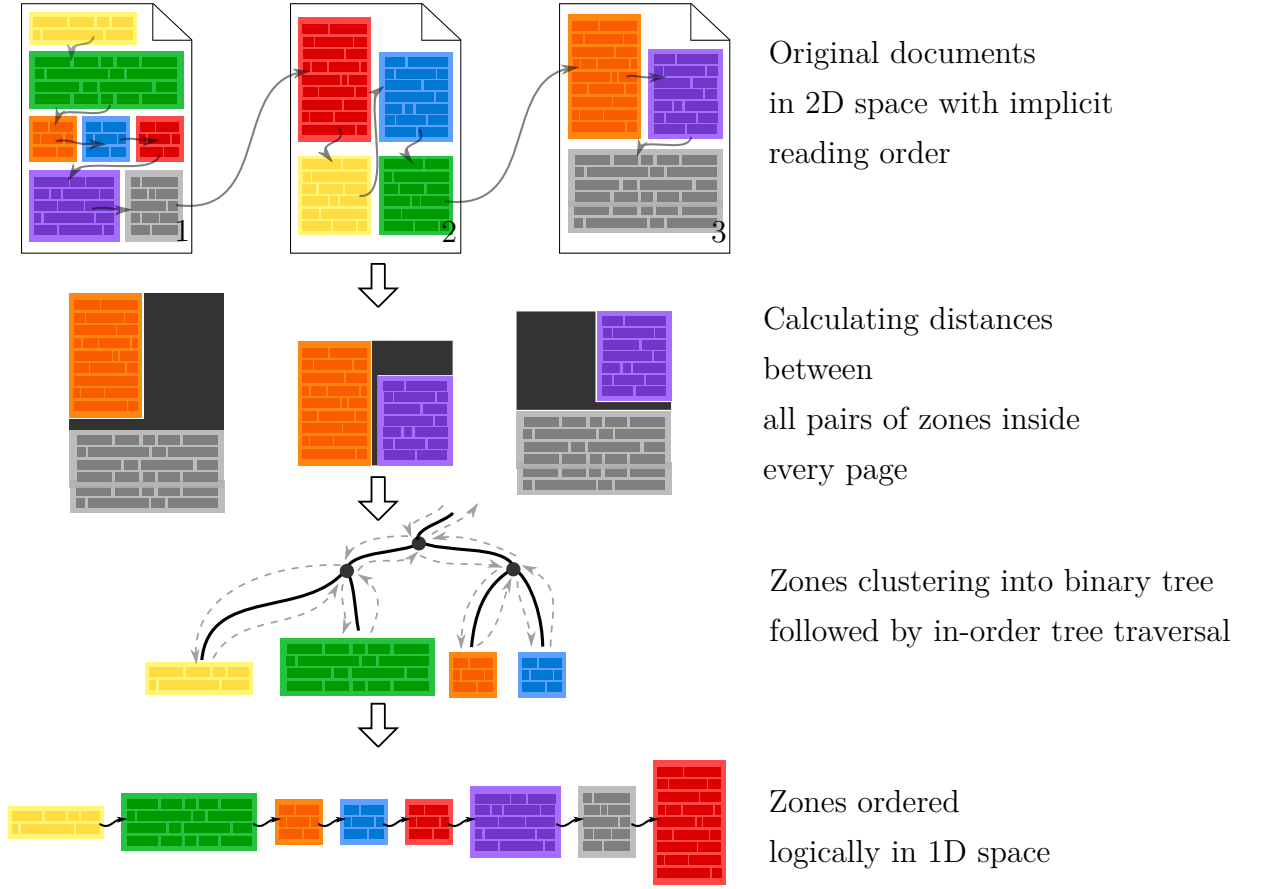
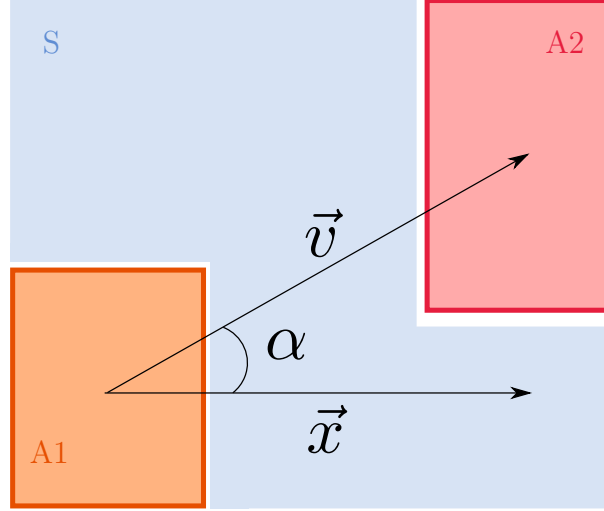


Figure 3.2: A schematic diagram of reading order resolving. Firstly, distances between text zones are calculated. Then, zones are hierarchically grouped according to distances between them. Finally, the tree is in-order traversed and a list of zones is created.

3. List L is sorted in ascending order with respect to the distance D .
4. Until this list is empty, the first triple is picked (i.e. with the smallest distance D) and two associated objects, O_1 and O_2 , are merged into a new object O_{1-2} .
5. For each element E in the list L recalculate the distance D to the new group iff E was based on O_1 or O_2 (i.e. is in form of $(O_*, *, D)$ or $(*, O_*, D)$). Then, a new triple (O_1O_2, X, D) is inserted in the list L , where O_{1-2} is the new group.

6. The list L is sorted in ascending order.



$$D = S \cdot \left(M + \frac{\vec{v} \cdot \vec{x}}{|\vec{v}| |\vec{x}|} \right)$$

Figure 3.3: An example of calculating distance between two zones. To include preference for vertical clustering a cosine of the angle α between the vector \vec{v} and vector \vec{x} (horizontal axis) is calculated. Parameter M is used to prevent zeroing of the distance for zones laying in the same vertical axis.

3.3 Initial zone classification

Once the workflow stages described in sections 3.2.1, 3.2.2 and 3.2.3 are done, the document can undergo two phases of classification. For the sake of clarity, in the remaining part of this document all labels from the first classification phase will be typed in upper case, while those from the second phase will be typed in lower case. Both will be typed using `verbatim`.

In the first phase, all the zones are classified into one of four general classes: `METADATA`, `REFERENCES`, `BODY` and `OTHER`. After this stage, more specific classifiers and algorithms can perform fine-grained analysis on the zones falling into each class. Semantics of these classes is as following:

- **METADATA** contains all the data which do not constitute the proper content. This includes publication elements such as: title, author name, affiliation, bibliographic information, abstract etc.
- **BODY** contains text of the publication. This includes the chapters, figures, tables, equations, plots, acknowledgments, headings, contribution statements, conflict statements and attachments.
- **REFERENCES** contains article's references,
- **OTHER** contains misclassified zones and page numbers. This class is a result of a non-ideal dataset, which tagged only to some extent, while some text zones are not classified to any class. When training the classifier, those unclassified zones are labeled as **OTHER**. This in turn introduces some kind of additional noise. We decided that it is better to exclude a handful of zones from the further processing and classify them as **OTHER**, rather than to classify them incorrectly to one of the remaining classes. Employed dataset contains 3.6% of unknown labels, therefore we might expect a similar number of zones classified to **OTHER** in the classifier's output.

Initial classification uses Support Vector Machines as classification algorithm, with the parameters given in the table 4.1. A method for obtaining these parameters is reported in 4.1.

3.4 Metadata classification

Once the initial coarse-grained classification is done, we employ another classifier, which is specialized in classifying metadata. Its input is made of those zones which were classified in the first stage as **METADATA**. This class consists of 15 labels:

- **abstract** - document's abstract, usually one or more zones placed in the first page of the document;
- **acknowledgments** - sections containing information about document's acknowledgments or funding;
- **affiliation** - authors' affiliations sections, which sometimes contain contact information (emails, addresses) as well;
- **author** - a list of document's authors,

- **bib_info** - zones containing all kinds of bibliographic information, such as journal/publisher name, volume, issue, DOI, etc.; often includes pages' headers or footers; headers and footers containing document's title or author names are also labeled as **bib_info**;
- **conflict_statement** - conflict statement declarations;
- **copyright** - copyright or license-related sections;
- **correspondence** - the authors' contact information, such as emails or addresses;
- **dates** -important dates related to the document, such as received, revised, accepted or published dates;
- **editor** - the names of the document's editors;
- **equation** - equations placed in the document;
- **keywords** - keywords listed in the document;
- **title** - the document's title;
- **title_author** - zones containing both document's title and the list of authors;
- **type** -the type of the document, usually mentioned on the first page near the title, such as *research paper*, *case study* or *editorial*;

3.5 References extraction

Reference extraction path is responsible for processing zones that were labeled as **REFERENCE** during the initial classification; its purpose is to extract bibliographic references together with their metadata.

To this end, the blocks of text are divided into strings containing one bibliographic reference each. This phase employs unsupervised learning whose task is to cluster all lines of references into two group - one group containing only first lines of references and the second group containing other lines. The clustering algorithm uses 5 features based on the text layout.

Afterwards, individual references are parsed and corresponding bibliographical data are yield back. This algorithm is based on Conditional Random Fields, thus a supervised learning algorithm, and is implemented on the top of GRMM and Mallet packages.

Reference extraction was implemented by Mateusz Fedoryszak and reported in [Dom14].

3.6 Content extraction

CERMINE is capable of extracting article's raw text. To this end, it processes all the zones labeled by the initial classifier as **BODY** and outputs the text that they contain according to the reading order.

Chapter 4

Classification workflow

Obtaining a fully operational and robust classifier is a time-consuming and resource-intensive process. Below we describe the full workflow from raw training files as input to optimal classification parameters and ready-to-use classification models as output.

4.0.1 Generating samples

Under the hood CERMINE does not operate on raw text - instead, for each input text zone CERMINE calculates values of various features (58 and 79 for the initial and metadata classifications respectively). Samples, i.e. vectors of feature values, are used as an input to for the classifiers. In the training process, samples are additionally associated with a decision class. As already said in the theoretical introduction (see chapter 2), the role of the classifier is to figure out non-trivial dependencies between training input samples, i.e. these providing a decision class, and to extrapolate them to yet unseen cases, i.e. those with no decision class assigned.

4.0.2 Data scaling

A very important step towards learning a classifier is scaling of the input data. The main motivation behind that, as mentioned in [CL10], is not to allow variables in greater numerical ranges to dominate those in smaller numerical ranges. What is more, very large values might cause numerical problems. One has to take into account that the closer to zero floating point variables are, the more precisely are they represented. Scaling recommended by [CL10] is in range $[-1, +1]$ or $[0, 1]$. In the CERMINE we implemented the latter approach. Clearly, both learning and

testing data have to be scaled using the same transformation. Thus, it is necessary to retain extreme values encountered in learning process.

4.0.3 Data sampling

In machine learning it happens very often that the decision classes are unequally represented in the training data. The approaches used to tackle this problem can be divided into three groups ([Cho10]):

1. Changing class distribution by modifying the data itself in order to rebalance it. It can be reached by either getting rid of samples from over-represented classes or by cloning samples from the under-represented classes. The former can greatly reduce the time spent on the optimization phase. The latter prevents not common samples from falling out of the training set, making the resulting classifier more versatile.
2. Adjusting the classifier itself by applying different misclassification costs for different classes. These costs might be for instance inversely linear to number of samples in each class.
3. Ensemble learning methods, i.e. using multiple classifiers trained with multiple training data and then combining output of the classifier, for instance by a simple voting.

In CERMINE two first mentioned approaches were tested and the second one was included in the final code.

4.0.4 Transformation to SVM^{LIGHT} format

Various SVM packages, including libsvm, require that the data are represented as vectors of real numbers. In consequence, if the data contains categorical attributes (such as **red**, **green**, **blue** or **big**, **small**) they have to be converted to numerical values. There are two common approaches to this problem:

- Using a single value to encode the category. This can be very easily implemented by using the integer value behind *enum* types in C or Java implementation;
- Using N different boolean variables to represent a category with N possible values. According to [CL10], this approach yields more stable results.

SVM^{LIGHT} format is a text file format that requires every training sample to be in a separate line (so, they must be split with new line characters). Each line consists of a certain number of fields separated with white signs, e.g. spaces. Each line has to begin with a number representing the decision class followed by feature values. There are no constraints regarding encoding of the decision classes, as long as this encoding is coherent. Feature values have a form of $f_n : v_n$, where f_n is the feature number and v_n is its value. Not every feature has to be listed, but only these whose values are non-zero. This format is beneficial for the scenarios where features are sparse.

4.1 Parameter optimization

It is impossible to tell *a priori* which kernel function is the most appropriate for the given training data. There are kernels that behave usually well (and RBF is one of them) and these should be a choice when the learning process is constrained by time.

SVM classifiers have a handful of parameters, including kernel type. In general, when optimizing a classifier one has to find a triple (K, C, γ) that maximizes classifier's accuracy for the given training data. In this triple K is the kernel type, C is a generic cost of misclassifying data and γ is a kernel parameter. One of the techniques of obtaining them is an extensive parameter space search by checking classifier's accuracy in every point on a parameter grid. Commonly this kind of search is called *grid search* and does not allow to omit any of the points in the parameter space without a risk of missing a possibly optimal point.

In the grid search for every point in the space a k -fold cross-validation is applied. It means that for a set of parameters the training set is divided into k subsets, each of equal or nearly-equal size. At every step $k-1$ of these subsets constitute the training set and the remaining k -th set is treated as the unknown data employed to verify correctness of the trained classifier. The trained classifier is applied to classification of the pseudo-unknown data and finally their *de facto* class is compared to the output of the classification. After k iterations the mean efficiency is calculated and it serves as a criterion for evaluating the current (K, C, γ) triple.

In the case of CERMINE grid search was performed on 16'384 samples from GROTOAP2. The training set was limited to this size to possibly keep classifier's training within reasonable time constraints. For both classifiers four kernels were tested with $\log_2 C$ and $\log_w \gamma$ varying with a unit step in range $[-5, 15]$ and $[-15, 3]$ respectively, resulting in 399 points tested for every classifier. In total there were 3192 ($= 399 \cdot 4 \cdot 2$) points tested in the first pass, which yielded candidates for accuracy maxima. Later, a fine-grained search was performed with the best kernels

so far (both RBF) around the candidates for maximum in distance of 10dB with a step equal to 2.5dB resulting in additional 81 points being checked for each of the classifier. Results of this procedure are presented in the figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10. Optimal parameters for the classifiers are presented in the table 4.1.

	Initial classifier	Metadata classifier
SVM type	C_SVC	C_SVC
kernel type	RBF	RBF
$\log_2\gamma$	-4.0	-2.5
\log_2C	2.75	4.25
$coef_0$	0.5	0.5
mean accuracy	93.66	86.98

Table 4.1: Results of the grid search in parameter space for the initial and metadata classifiers. Values above were used to train the classifiers and to perform final evaluation of the systems.

4.1.1 Training of the final classifier

Once the optimal parameters had been found using a grid search, the classifiers were trained to perform the final evaluation. This process is described command by command in the appendix A. For this task we used classes `SVMInitialZoneClassificationEvaluator` and `SVMMetadataClassificationEvaluator` that build the classifier and perform cross-validation at a time. Its results are presented in the chapter 6.

4.2 Feature selection

In both stages of classification we employed a set of features including almost 80 values. Appendix B contains a detailed description of how their values are calculated. Briefly, they can be divided into following groups:

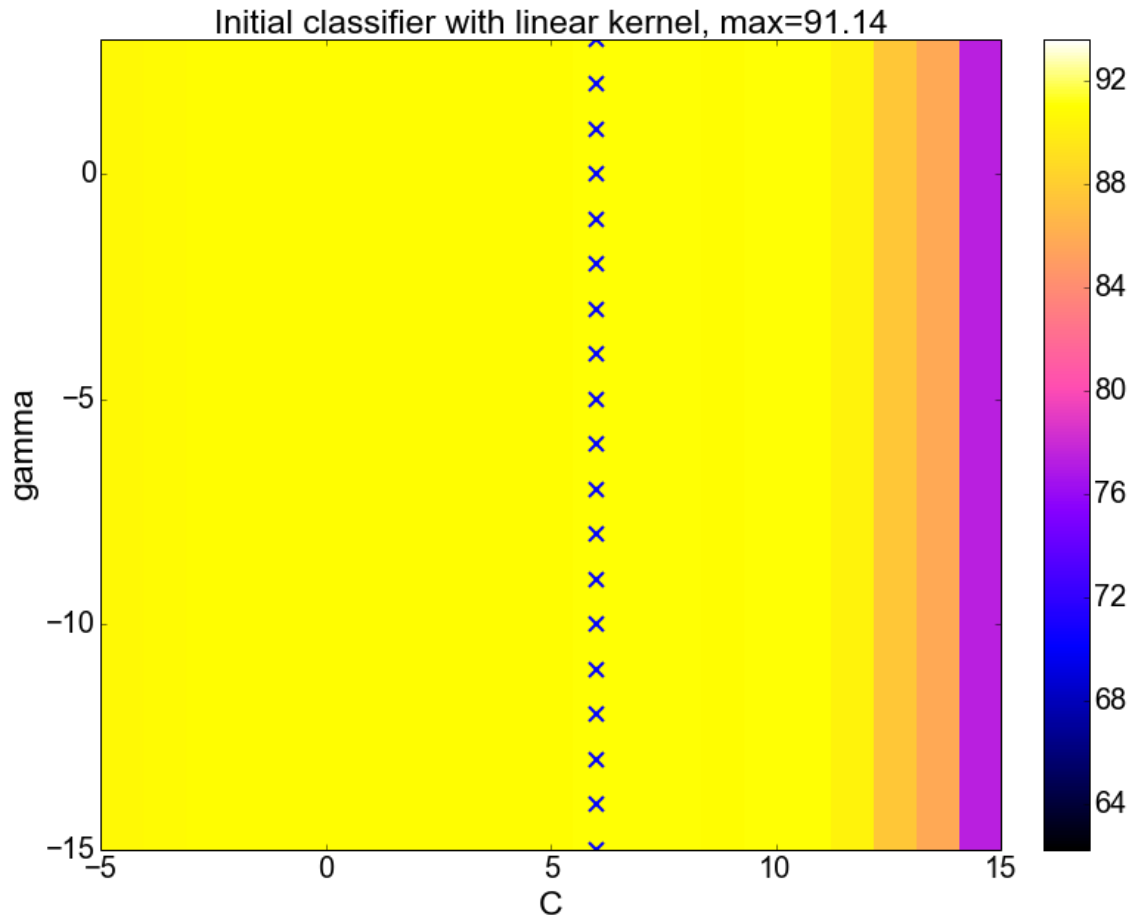


Figure 4.1: Grid search over two-dimensional parameter space (C, γ) with linear kernel for the initial classifier. The maximal accuracy in 5-fold cross-validation is equal to 91.14 and is found for $\log_2 C = 6$

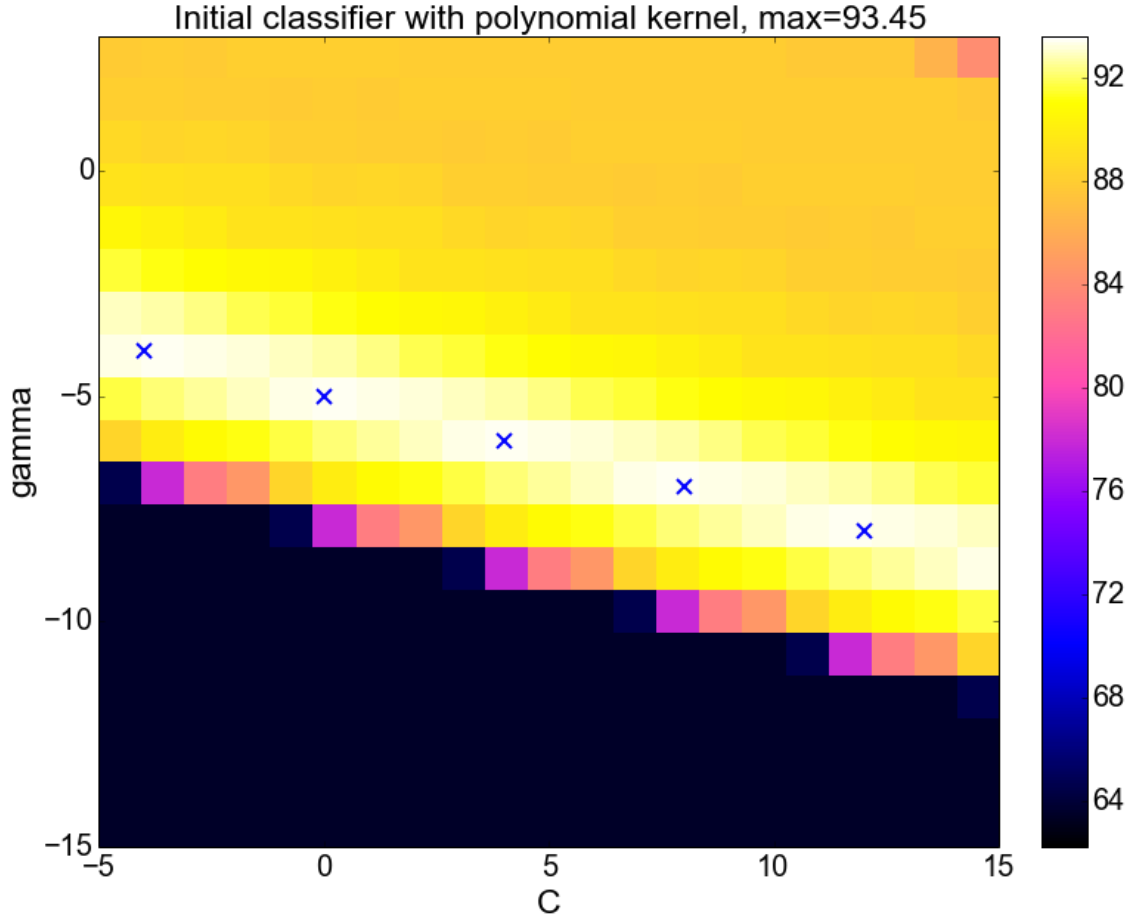


Figure 4.2: Grid search over two-dimensional parameter space (C, γ) with polynomial 4^{th} -degree kernel for the initial classifier. In the investigated interval there are 5 maxima. Classifier accuracy over 5-fold validation is equal to 93.45 and is found for instance for $\log_2 C = 3$ and $\log_2 \gamma = -7$.

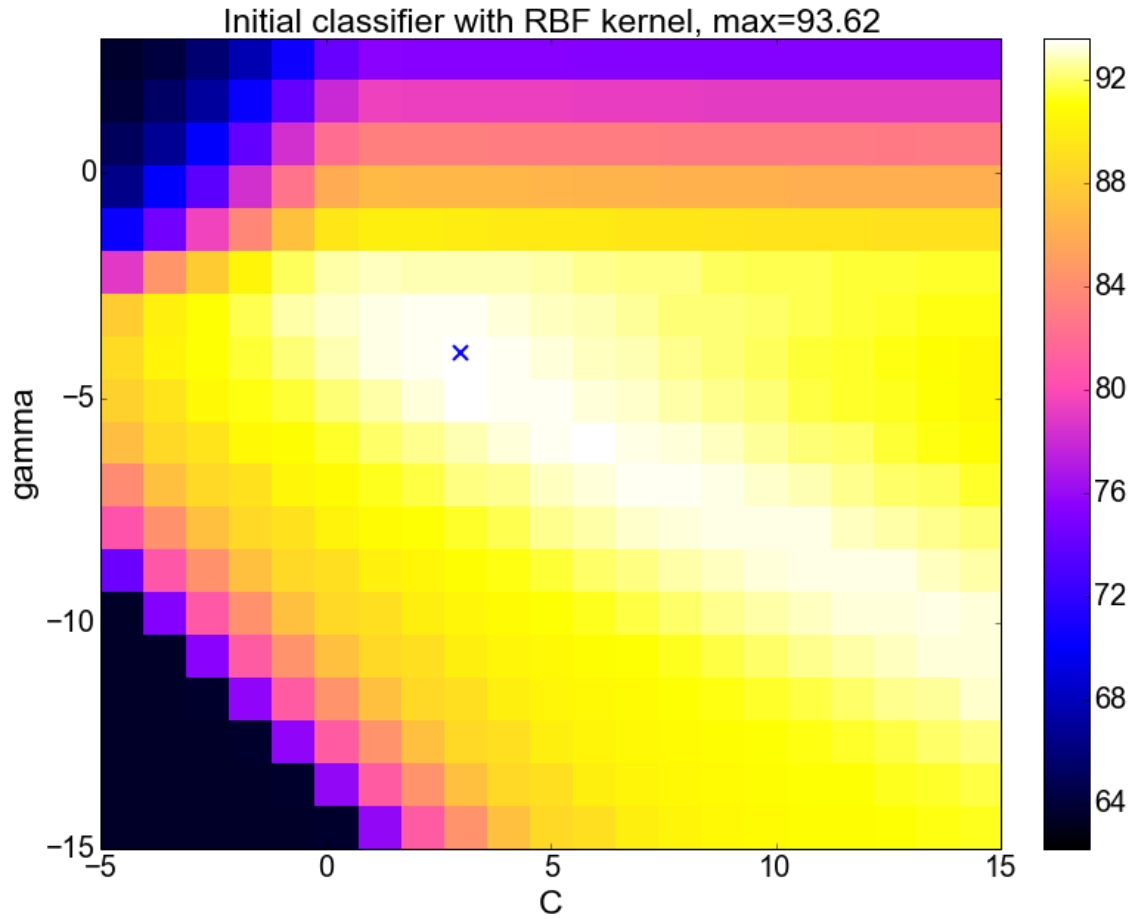


Figure 4.3: Grid search over two-dimensional parameter space (C, γ) with a radial-basis function kernel for the initial classifier. The maximal accuracy in 5-fold cross-validation is equal to 93.62 and is found for $\log_2 C = 3$ and $\log_2 \gamma = -4$. This was the highest accuracy found in the first pass. The second pass refined the maximum around the initial result.

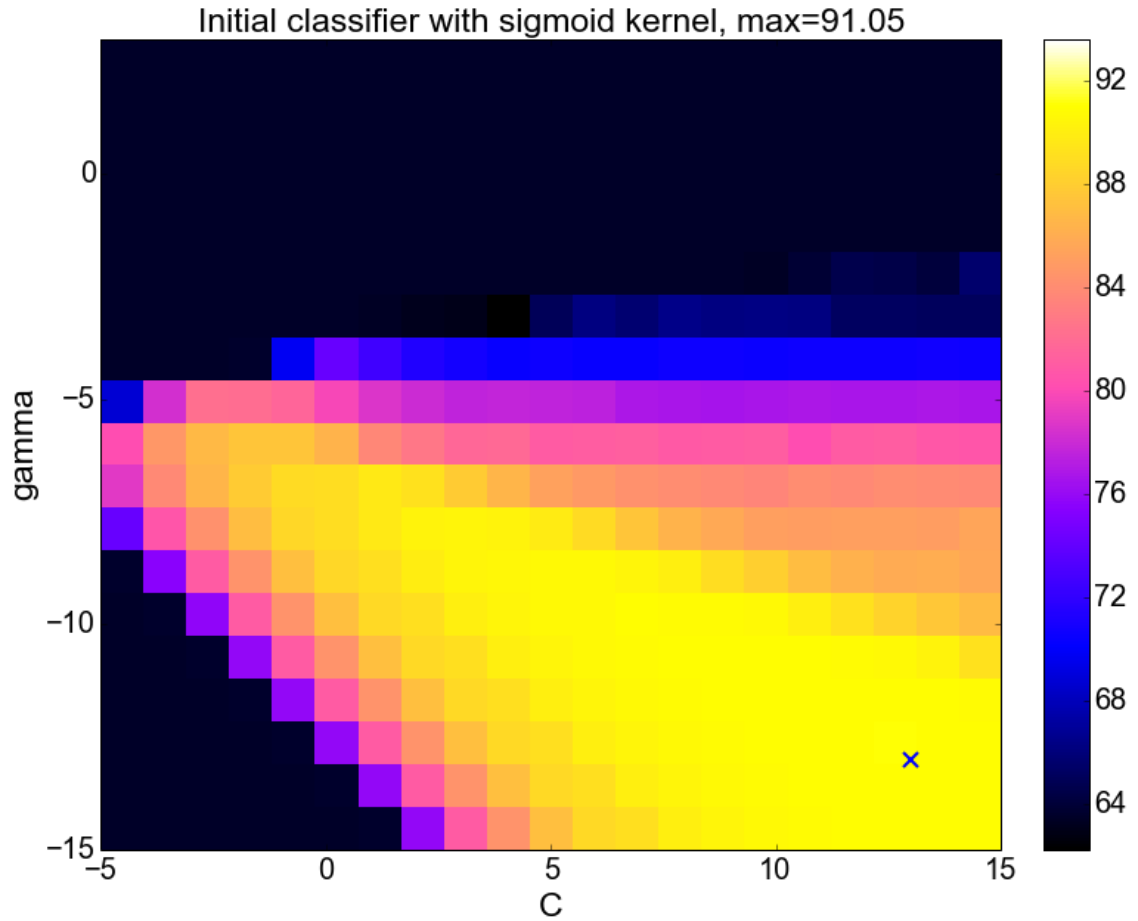


Figure 4.4: Grid search over two-dimensional parameter space (C, γ) with a sigmoid kernel for the initial classifier. The maximal accuracy in 5-fold cross-validation is equal to 91.05 and is found for $\log_2 C = 13$ and $\log_2 \gamma = -13$

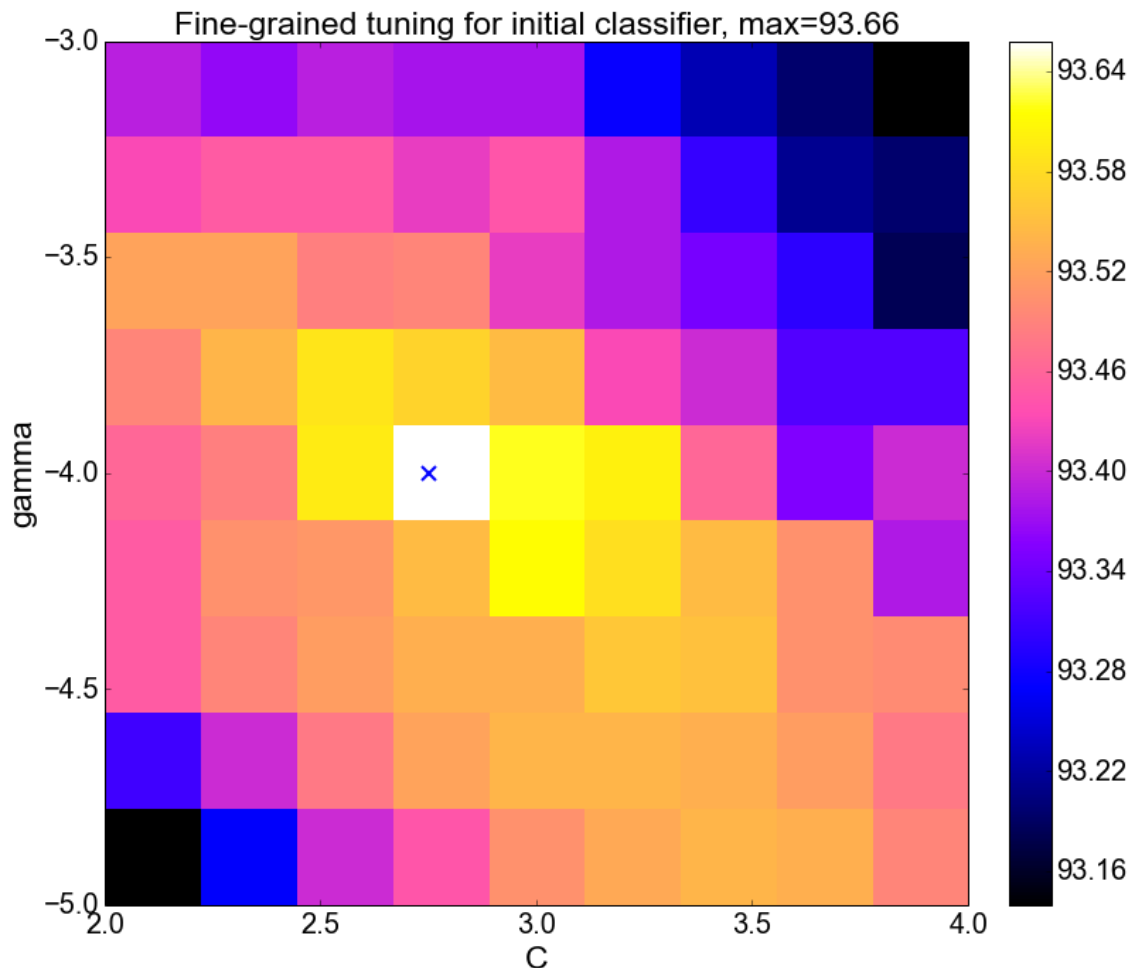


Figure 4.5: Fine-grain search around initial classifier with RBF kernel for the initial classifier (the color-mapping scheme is not coherent with the previous plots). Initial coarse-grain search yielded the maximum in $(\log_2 C = 3, \log_2 \gamma = -4)$ equal to 93.62. This search refines the search with $\log_2 \text{step} = 0.25$ yielding maximum in $(\log_2 C = 2.75, \log_2 \gamma = -4)$ equal to 93.66. This parameters are used for training of the final classifier.

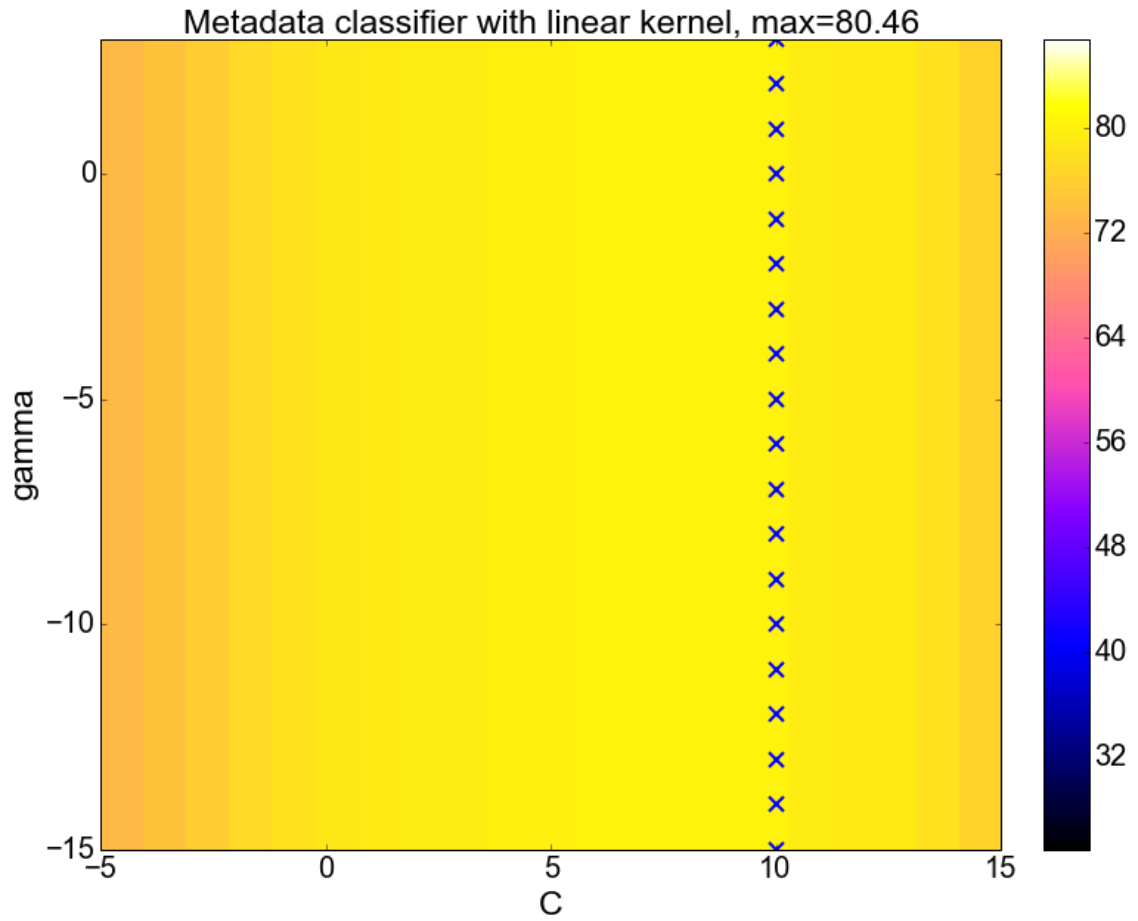


Figure 4.6: Grid search over two-dimensional parameter space (C, γ) with linear kernel for the metadata classifier. The maximal accuracy in 5-fold cross-validation is equal to 80.46 and is found for $\log_2 C = 19$

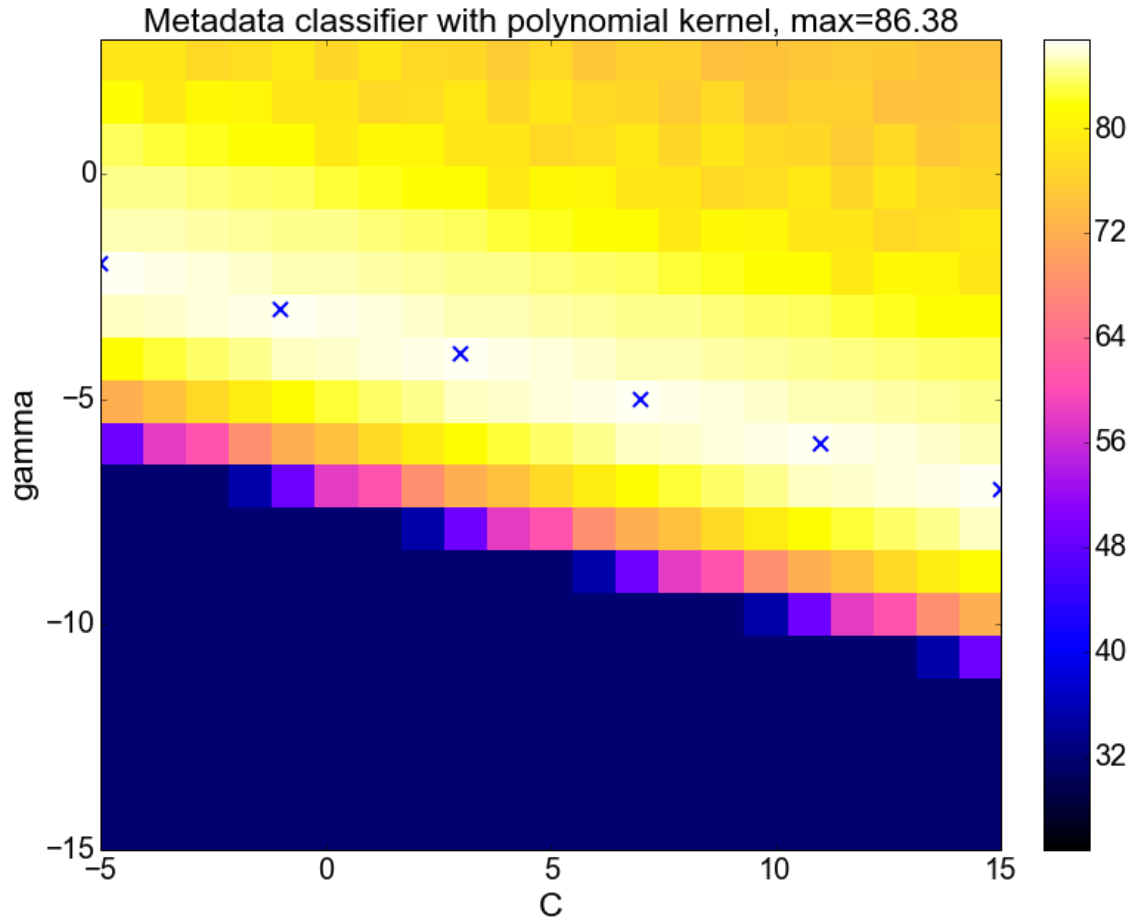


Figure 4.7: Grid search over two-dimensional parameter space (C, γ) with polynomial 4^{th} -degree kernel for the metadata classifier. In the investigated interval there are 5 maxima. Classifier accuracy over 5-fold validation is equal to 86.38 and is found for instance for $\log_2 C = 7$ and $\log_2 \gamma = -5$.

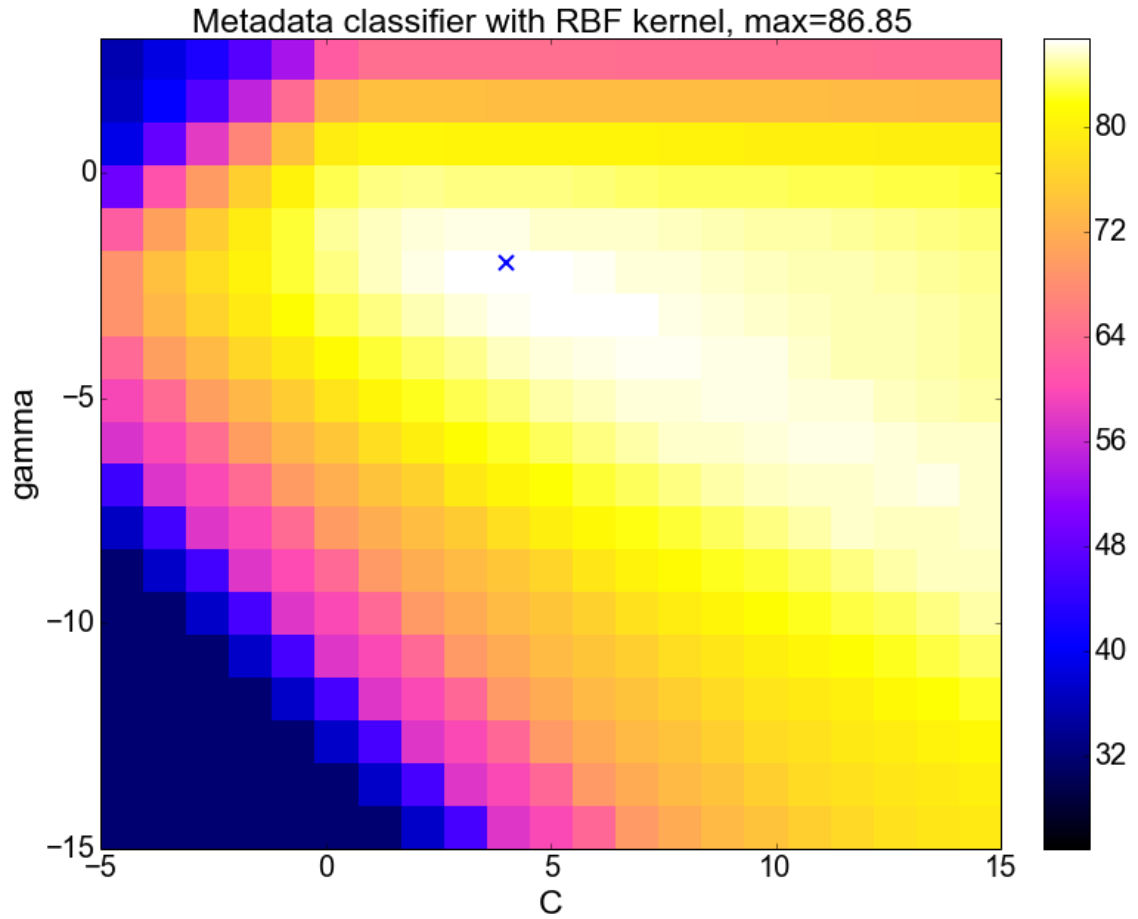


Figure 4.8: Grid search over two-dimensional parameter space (C, γ) with a radial-basis function kernel for the metadata classifier. The maximal accuracy in 5-fold cross-validation is equal to 86.85 and is found for $\log_2 C = 4$ and $\log_2 \gamma = -2$. This was the highest accuracy found in the first pass. The second pass refined the maximum around the initial result.

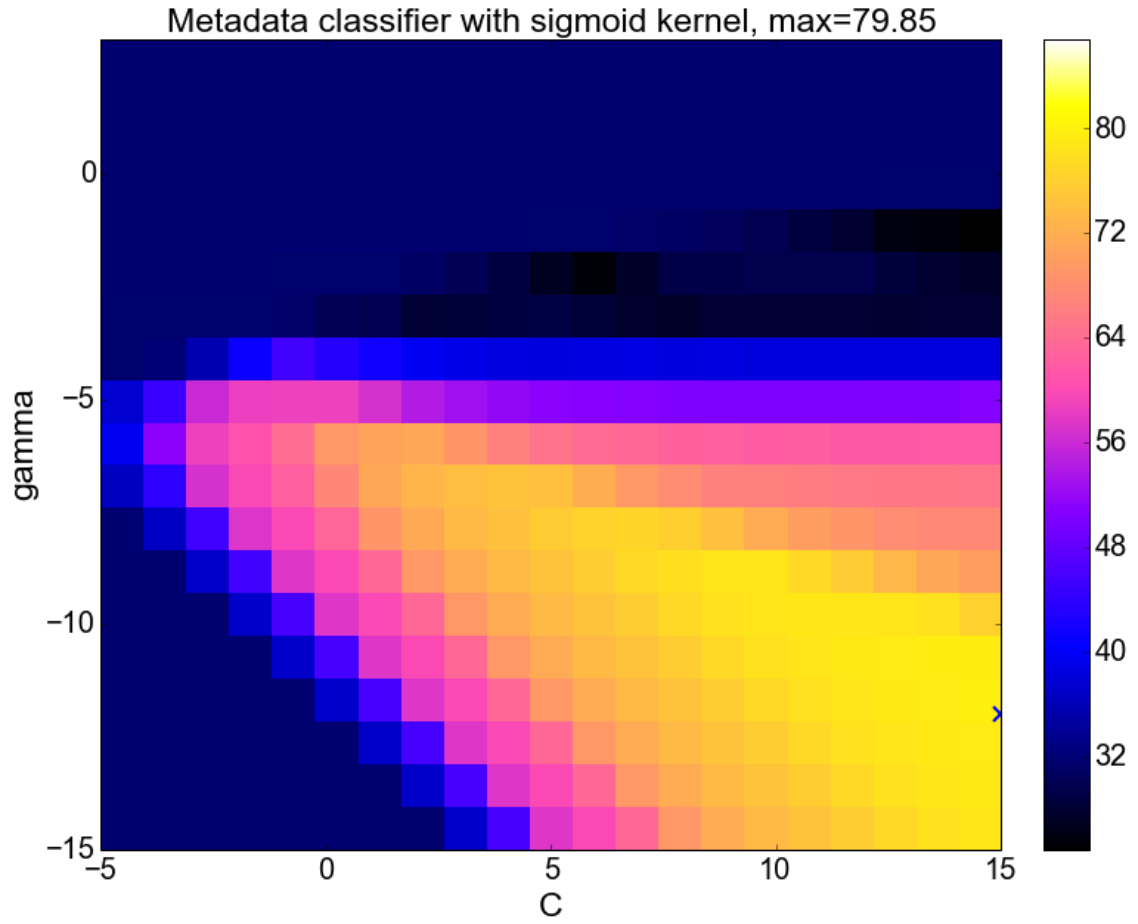


Figure 4.9: Grid search over two-dimensional parameter space (C, γ) with a sigmoid kernel for the initial classifier. The maximal accuracy in 5-fold cross-validation is equal to 79.85 and is found for $\log_2 C = 15$ and $\log_2 \gamma = -12$

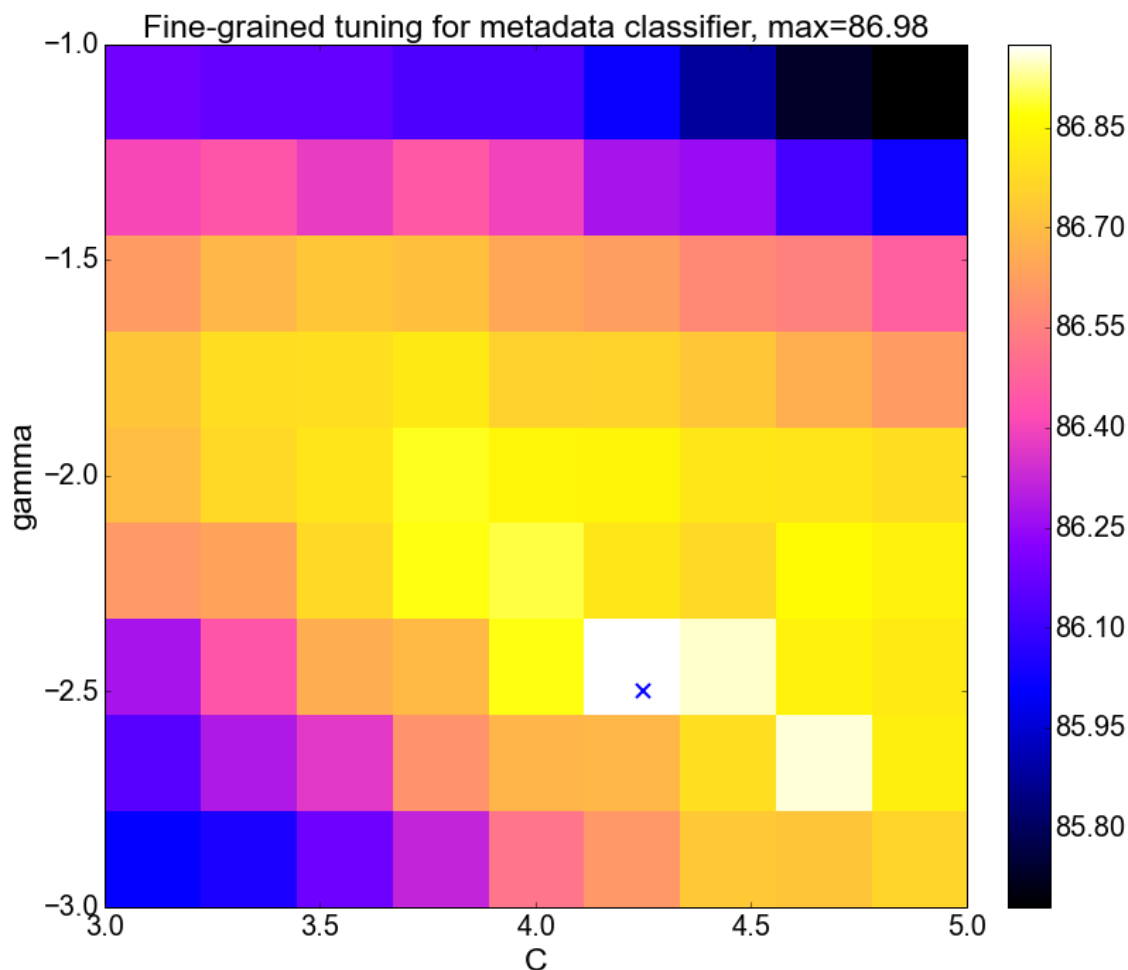


Figure 4.10: Fine-grain search around initial classifier with RBF kernel for the metadata classifier (the color-mapping scheme is not coherent with the previous plots). Initial coarse-grain search yielded the maximum in $(\log_2 C = 4, \log_2 \gamma = -2)$ equal to 86.85. This search refines the search with $\log_2 \text{step} = 0.25$ yielding maximum in $(\log_2 C = 4.25, \log_2 \gamma = -2.5)$ equal to 86.98. This parameters are used for training of the final classifier.

Sequence features

This group contains only two features: *PreviousZoneFeature* and *LastButOneZoneFeature*. It leverages the fact that the zones are classified after the process of reading order resolution is done. The features contain numerical value of two previous labels. In turn, this allows to incorporate sequence information into classification, which is not done in SVM by design (as opposed to for instance Hidden Markov Model).

Formatting features

This group of features contains information about graphical properties of text. It can be very helpful since certain elements in scholarly publications are usually typed with bigger font size.

Layout features

These features encode information about graphical layout on the page. This includes properties of the zone itself (e.g. *WidthFeature*, *HeightFeature*, *LineMean-WidthFeature*) as well as features of the area between text zones (e.g. *HorizontalRelativeProminenceFeature*, *VerticalRelativeProminenceFeature*, *IsHighestOnThePage*, *IsGreatestOnThePage*, *DistanceFromNearestNeighbourFeature*).

Semantic features

This group of features encodes appearance of certain key-words that very often characterize certain parts of articles written with scientific English, e.g. *keywords*, *terms*, *distributed*, *reproduction*, *open*, *commons*, *license*, *creative*, *copyright*, *cited*, *distribution*, *access*, *references*, *author*, *bibliography*, *figure*, *table*, *editor*, *email*, *correspondence*, *address*, *abstract*, *author details*, *university*, *department*, *school*, *institute*, *affiliation*, *research article*, *review article*, *editorial*, *review*, *debate*, *case report*, *research*, *original research*, *methodology*, *clinical study*, *commentary*, *article*, *hypothesis*.

Special features

This group contains features that do not fit other groups mentioned above. This includes e.g. *IsAnywhereElseFeature*, *BracketCountFeature*, *CharCountFeature*, *CommaCountFeature*.

Chapter 5

Dataset

5.1 Motivation

Despite a strong need of performing automated analyses of scholarly publications, there are no big and reliable open access datasets of annotated publications publicly available. Such a dataset would greatly facilitate the process of building automated systems for metadata extraction, which, even when boiled down to scientific publishing, is error prone and not trivial.

By far this is caused by two factors:

- scientific publishing is inherently characterized by richness of layouts, formats and designs. No one is to be blamed for this variety which is to the most extreme degree natural. Any tool that aims at being reliable needs to take this diversity of styles into account;
- PDF, being a *de facto* standard in the scientific publishing, is a format designed to hold its content together with geometrical data. However, it is not suitable for carrying content's metadata. Its main design goal was to hold geometrical metadata only allowing a faithful reproduction in various environments (tools, platforms etc.) of how the source document was displayed. It holds no relevant metadata and the structure of the input document is broken down to the level of single characters.

Each publisher and even each journal follows its own layout. For instance, in the Open Access subset of PubMed Central there are more than 800 publishers, each of them very likely following a different layout. One has to keep in mind that the articles included in PMC are biology-related which is only a small fraction of scientific publishing. It is safe to assume that there are thousands of publishers

worldwide. We decided to put together a diverse set of articles that could serve as a basis for automatic approach for content analysis.

5.2 Related work

The vast majority of the data sets publicly available are based on images, i.e. there are no born-digital documents available. As an example here we might give MARG, which contains scanned first pages of biomedical articles. This makes its usability in our case very limited. PRIMa is a data set containing images of various documents, whose category goes beyond scientific publishing. Again, these are images which are not helpful in case of chosen architecture. Conversely, The UW-III data set contains both articles' images and born-digital data. It is not available freely, though. What is more, we preferred to focus in our work on pure scientific articles as those are the target input of the designed system.

5.3 Pubmed database

Pubmed [Puba] is a collection of more than 24 millions articles from MEDLINE - life science journals and online books maintained by the US National Library of Medicine. This collection is digitized which means that the articles are stored in form of PDF files and can serve as a source of full texts without a need for performing Optical Character Recognition. The PMC Open Access Subset (abbreviated to PMC-OAS) is a part of Pubmed Central available publicly without any fees. These articles are kept protected by copyrights, but can be used under the Creative Commons license that allows more liberal usage of the resources. For the time of creating this report (June 2014) there were more than 400,000 articles available.

5.4 Dataset creation

With very limited budget it is clearly impossible to create a rich set of annotated articles by human means. A predecessor of the thereafter described data set was created at ICM in [DCR⁺12]. GROTOAP (*GROund Truth for Open Access Publications*) created there is a collection of 113 articles in digital form with corresponding ground-truth files. This data set was created semi-automatically. First, a handful of articles was tagged manually and supplied to a predecessor of CERMINE as a training input. Then, the remaining part of articles was tagged by the trained classifier and corrected by experts later on, guaranteeing high precision and accuracy.

Its downside was that the included articles cover not more than 15 different layouts, which is certainly not enough when aiming to build a tool applicable to the whole spectrum of publishers and layouts. Initial tests have proven that this data set did not provide sufficient diversity for the classifier to perform well enough on a broader documents collection. This is why we decided to create a new dataset, called GROTOAP2, that could serve as a base for classifiers.

5.5 GROTOAP2 properties

GROTOAP2 is made of ground-truth files in TrueViz format, which extends XML, containing publications' content structured hierarchically and tagged according to its role. This is accompanied by geometrical data on 4 levels of granularity: text zone, text line, word and character. The role of the geometrical data is to reflect how the content should be displayed according to the PDF files. Since function of a piece of text can be distinguished not only by the text itself, but also by its appearance, i.e. the geometrical features, a dataset applicable for training a classifier must preserve the information related to dimensions and positions of the objects.

In GROTOAP2 there are in total 13210 articles made of 119334 pages and 1640973 zones coming from 208 different publishers and 1170 different journals. The data set is distributed under the CC-BY license and can be downloaded from the server of Center for Open Science [CeO]. It is available together with a list of URLs for the corresponding PDF files (which are not included in the data set itself) and scripts downloading both input XMLs and the PDFs.

5.6 The methodology of creating GROTOAP2

As stated before, it was assumed that GROTOAP2 has to be based on open access articles. Pubmed was chosen as source of input data, as a great majority of contained articles is associated with extracted metadata. Sometimes their quality might be put in doubt, but the process was designed to automatically filter them out. The method presented here is scalable to immense sets as the human factor was ruled out.

Each article in the Pubmed dataset comes as a tarball containing the article itself (in PDF file format), a metadata file (in XML file format) and possibly some resource files, mainly including pictures extracted from the PDF. A sample content of a metadata XML is showed in the listing 5.1. A complete description of possible elements can be found in [Pubb].

```

<?xml version="1.0" encoding="UTF-8"?>
<front>
  <journal-meta>
    <journal-id journal-id-type="nlm-ta">Genome Biol</journal-id>
    <journal-title>Genome Biology</journal-title>
    <issn pub-type="ppub">1465-6906</issn>
    <issn pub-type="epub">1465-6914</issn>
    <publisher>
      <publisher-name>BioMed Central</publisher-name>
    </publisher>
  </journal-meta>
  <article-meta>
    <article-id pub-id-type="pmid">18341703</article-id>
    <article-id pub-id-type="pmc">2397496</article-id>
    <article-id pub-id-type="publisher-id">gb-2008-9-3-213</article-id>
    <article-id pub-id-type="doi">10.1186/gb-2008-9-3-213</article-id>
    <article-categories>
      <subj-group subj-group-type="heading">
        <subject>Minireview</subject>
      </subj-group>
    </article-categories>
    <title-group>
      <article-title>On the nature of thumbs</article-title>
    </title-group>
    <contrib-group>
      <contrib id="A1" corresp="yes" contrib-type="author">
        <name>
          <surname>Wagner</surname>
          <given-names>Gunter P</given-names>
        </name>
        <xref ref-type="aff" rid="I1">1</xref>
        <email>gunter.wagner@yale.edu</email>
      </contrib>
      <contrib id="A2" contrib-type="author">
        <name>
          <surname>Vargas</surname>
          <given-names>Alexander O</given-names>
        </name>
        <xref ref-type="aff" rid="I1">1</xref>
      </contrib>
    </contrib-group>
    <aff id="I1">
      <label>1</label>
      Department of Ecology and Evolutionary Biology, Yale University, Prospect Street, New Haven, CT
      06520-8106, USA.
    </aff>
    <pub-date pub-type="ppub">

```

```

    <year>2008</year>
  </pub-date>
  <pub-date pub-type="epub">
    <day>4</day>
    <month>3</month>
    <year>2008</year>
  </pub-date>
  <volume>9</volume>
  <issue>3</issue>
  <fpage>213</fpage>
  <lpage>213</lpage>
  <permissions>
    <copyright-statement>Copyright 2008 BioMed Central Ltd</copyright-statement>
    <copyright-year>2008</copyright-year>
    <copyright-holder>BioMed Central Ltd</copyright-holder>
  </permissions>
  <abstract abstract-type="short">
    <p>Differential Hox gene expression make the thumb special</p>
  </abstract>
  <abstract>
    <p>Asymmetric regulation of Hox gene expression pre-dates the appearance of tetrapod digits, and was co-opted in the development of 'thumbness'. This asymmetric expression correlates with independent morphological evolutionary variation of digit 1.</p>
  </abstract>
</article-meta>
</front>

```

Listing 5.1: Sample content of a PUBMED XML file

The general strategy for creating ground-truth files out of the PMC-OAS was to match content of the metadata files to blocks of text extracted and segmented from the provided PDF files resulting in a tagged TrueViz file. This allowed us to mix together both data, metadata and geometric properties of the articles.

PMC-AOS metadata format is not fully clear. Full understanding of its structure was achieved by thorough and laborious manual analysis of many randomly picked sample files. There are many situations that one has to take into account:

- structure of the documents varies between documents,
- random fields might be missing or incomplete,
- same content (e.g. article's title) might be located under different paths, e.g. contributor's e-mail address might be found under `/article/front/article-meta/contrib` or `/article/front/article-meta/contrib-group/contrib/address/email`.
- same content might have different level of granularity, e.g. editor's name might be given as a flat string, whereas author's name might be divided into surname

and given names,

- elements of the same kind be found under different parents, e.g. figures might be located in `/article/floats-wrap//fig`, `/article/floats-group//fig`, `/article/back//fig`, `/article/body//fig`, `/article/back/app-group//fig`

As being said, we created the data set in an automatic way ([DB14]):

1. First, a large set of files was downloaded from Open Access Subset of PubMed Central. We obtained both source articles in PDF and corresponding NLM files containing metadata, full text and references.
2. PDF files were automatically processed by tools provided by CERMINE and their hierarchical geometric structure along with the natural reading order was constructed.
3. The text content of every extracted zone was compared to labeled data from corresponding NLM files, which resulted in attaching the most probable labels to zones.
4. Files containing a lot of zones with unknown labels, that is zones for which the automatic labeling process was unable to determine the label, were filtered out.
5. A small random sample of the remaining documents was inspected by a human expert. We were able to identify a number of repeated problems and errors and develop heuristic-based rules to correct them.
6. From the corrected files the final dataset was chosen randomly.

We use `xpath` to extract text entries from predefined paths. These paths are common for the documents in the Pubmed dataset. Each path is mapped to a label in our labeling system. The task consists in finding a mapping between NLM entries and PDF zones, so that each zone could obtain a label dependent on its path in the NLM file.

For each pair (`pdf_zone`, `nlm_entry`) Smith-Waterman distance is calculated. Afterwards, for each zone a set of different approaches are followed in order to assign a correct label:

1. Take NLM entry with the highest ratio of SW distance to the number of tokens. If both PDF zone and NLM entry are not empty and their size expressed in number of tokens is comparable (ratio not smaller than 0.7) and length of the common substring constitutes more than 70% of the NLM entry, then assign corresponding label,

2. If the previous approach failed, take the entry with the highest SW distance. If the length of the common substring is bigger than 50% of the number of tokens in the PDF zone, then assign corresponding label,
3. If the previous approaches failed, a *cumulative* distance is calculated. This makes it possible to assign a label to these zones that form together one NLM entry, but were segmented into several parts. This applies mostly to **BODY** zones. For each zone these values are aggregated by summing them up. From all the cumulative distance the biggest one is taken. If it is greater than 0.5, the corresponding label is assigned.

5.7 Filtering

After the matching stage we obtained initial set of TrueViz files whose quality was very diverse, i.e. spanning from fully labeled to almost unlabeled. There are two reasons for that:

1. NLM files in PMC vary in quality. Those with less detailed metadata resulted in barely labeled TrueViz files,
2. With certain layouts our segmentation algorithm was failing, producing very finely-grained text zones with just a handful of letters each. In these cases it was barely possible to match NLM metadata with the actual text, what again lead to sparsely labeled TrueViz files.

In order to assure good quality of the final set we needed to filter out documents with TrueViz files of bad quality. In the figure 5.1 one can see a distribution of documents with a given fraction of labeled zones. Based on this distribution we decided to keep the documents having at least 90% of labeled zones.

5.8 TrueViz file format

We decided to choose TrueViz file format as the data set's main data format. This was to maintain the choice taken while working on GROTOAP data set. The motivation behind this choice is discussed in details in [DCR⁺12]. TrueViz is a plain text file based on the XML format that holds in an extremely inefficient way the content of a document together with its geometrical properties and classification metadata. An example of a dumb file containing a single character is shown on the listing 5.2.

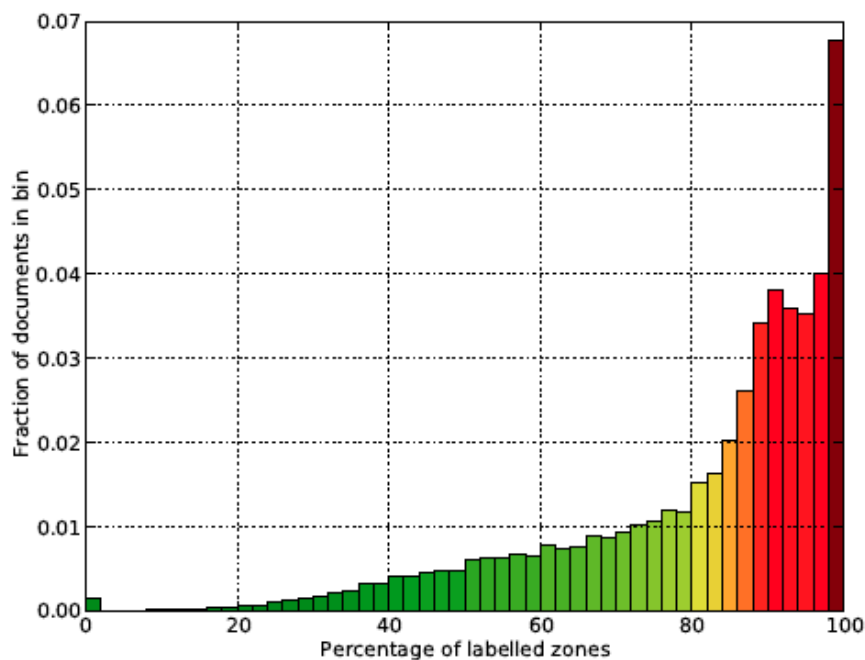


Figure 5.1: The histogram presents distribution of successful zone coverage after mapping NLM metadata to PDF files in the Open Access Subset of PubMed. Coverage different than 100% can be caused by imperfect metadata, wrong PDF segmentation or inefficient mapping algorithm. Image source: [DB14].

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Document SYSTEM "Trueviz.dtd">
<Document>
  <Page>
    <Zone>
      <ZoneID Value="52"/>
      <ZoneCorners>
        <Vertex x="45.4" y="743.8"/>
        <Vertex x="197.4" y="751.3"/>
      </ZoneCorners>
      <ZoneNext Value="53"/>
      <ZoneInsets Bottom="" Left="" Right="" Top=""/>
      <ZoneLines Value=""/>
      <Classification>
        <Category Value="BIB_INFO"/>
        <Type Value=""/>
      </Classification>
    </Zone>
  </Page>
</Document>
```

```

<Line>
  <LineID Value="506"/>
  <LineCorners>
    <Vertex x="45.4" y="743.8"/>
    <Vertex x="197.4" y="751.3"/>
  </LineCorners>
  <LineNext Value="507"/>
  <LineNumChars Value=""/><name />
  <Word>
    <WordID Value="4796"/>
    <WordCorners>
      <Vertex x="45.4" y="743.9"/>
      <Vertex x="77.0" y="751.3"/>
    </WordCorners>
    <WordNext Value="4797"/>
    <WordNumChars Value=""/>
    <Character>
      <CharacterID Value="26121"/>
      <CharacterCorners>
        <Vertex x="45.4" y="743.9"/>
        <Vertex x="49.5" y="751.3"/>
      </CharacterCorners>
      <CharacterNext Value="26122"/>
      <GT_Text Value="F"/>
    </Character>
  </Word>
</Line>
</Zone>
</Page>
</Document>

```

Listing 5.2: Sample content of a TrueViz file with one letter of content - *F*

The resulting TrueViz files are obviously much bigger than the input PDF file. This is the price that has to be paid for the readability that it offers.

Each of the text zones in a document is labeled with one out of 22 categories:

- **abstract** - document's abstract,
- **acknowledgments** - acknowledgment for article's funding
- **affiliation** - authors' affiliations,
- **author** - document authors,
- **author_title** - zones containing both document's title and the list of authors;
This zone has been introduced, as very often our segmentation algorithm

- **bib_info** - bibliographic information that was not assigned into any other category, usually concerning the journal issue e.g. journal name, publisher name, volume, DOI,
- **body_content** - the text of the document that did not match any other category,
- **conflict_statement** - conflict statement declaration,
- **copyright** - copyright section,
- **correspondence** - contact information to the article's authors, usually includes e-mail addresses, but physical address is also possible,
- **dates** - dates related to the document life-cycle - reception date, review date, publishing date etc.,
- **editor** - name of document's editor,
- **equation** - mathematical equation,
- **figure** - figure captions and figures elements which are embedded in a PDF as text, e.g. axes labels,
- **glossary** - article's glossary,
- **keywords** - article's keywords,
- **page_number** - page number of the volume;
- **references** - article's references with all the data (titles, names, years) put into one bag,
- **table** - table's content and caption;
- **title** - article's title;
- **type** - the type of the document, usually mentioned on the first page near the title, such as *research paper*, *case study* or *editorial*;
- **unknown** - used for the text zones that couldn't be assigned to any other category. This might be because they are very rare or couldn't be matched to any metadata.

This list is richer than the set of labels used by CERMINE, as we were not targeting such a fine-grained granularity in the system output. On the other hand, Pubmed XMLs are very rich and contain a huge number of metadata elements. It would be a pity to create a data set which would lose silently these data. This is why we decided to keep extended set of categories in GROTOAP2. In the appendix E one can see distributions of these categories for the chosen documents.

The **acknowledgment** section has a very similar distribution, **affiliation** is mostly focused in two or three zones with a pretty short tail, **author** section has a very narrow distribution with most of the articles having only one such zone. On the contrary, **author_title** is in most cases represented by 4 or 5 zones. **editor** is, as expected, in the majority of cases present in only one zone per document. It is interesting to see what is the distribution of **unknown** zones, i.e. those whose content couldn't be matched with the data from Pubmed XML. The most numerous bins are 1 and 2, containing in total more than 12000 documents. This can be seen as an indicator of good quality of the data set.

Figure 5.2 shows the fraction of documents in the data set that contain given label. One can see that all the documents contain labels **bib_info**, **body_content**, **references** and **affiliation**.

Figure 5.3 shows the total count of labels across all documents (logarithmic scale). One can see that **table** is the most numerous label, followed by **body_content**. This is caused by inability to cluster very small and distant text zones together, as they usually appear so in the tables embedded in the PDF documents.

Figures 5.5 and 5.4 show the contribution in the data set of the top 15 publishers and top 25 journals respectively. In total there are 208 publishers included and 1170 journals. As expected, the biggest publisher is a medical one (BioMed Central), but the remaining ones might be associated with a general scientific publishing. As different journals are stuck to their layouts, we might be sure that the variety of the layouts in the data set is sufficient to allow good mean performance on unknown articles.

Figures 5.6 and 5.7 show the distribution of number of pages and publication years respectively. The first distribution reaches its maximum at 8 pages, which is a standard in the scientific publishing. The second distribution is focused around the year 2010, but has a tail reaching year 1995. Therefore, we might expect that the classifier trained using this set will show better performance for more recent publications.

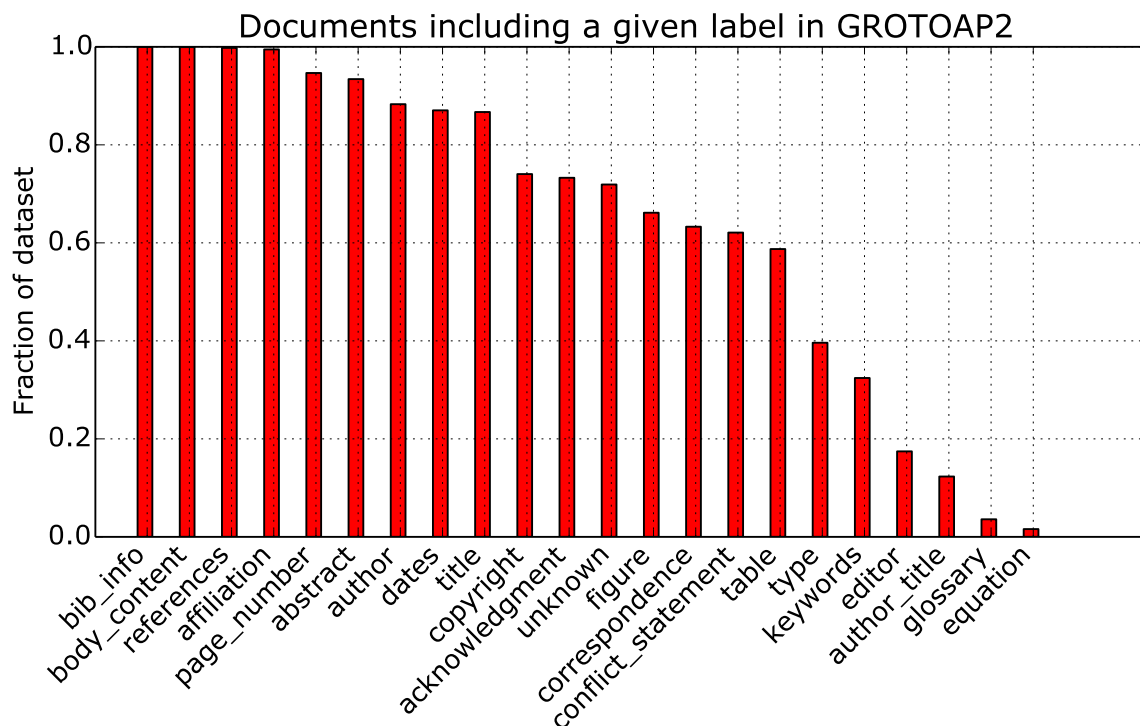


Figure 5.2: Fraction of documents containing given label

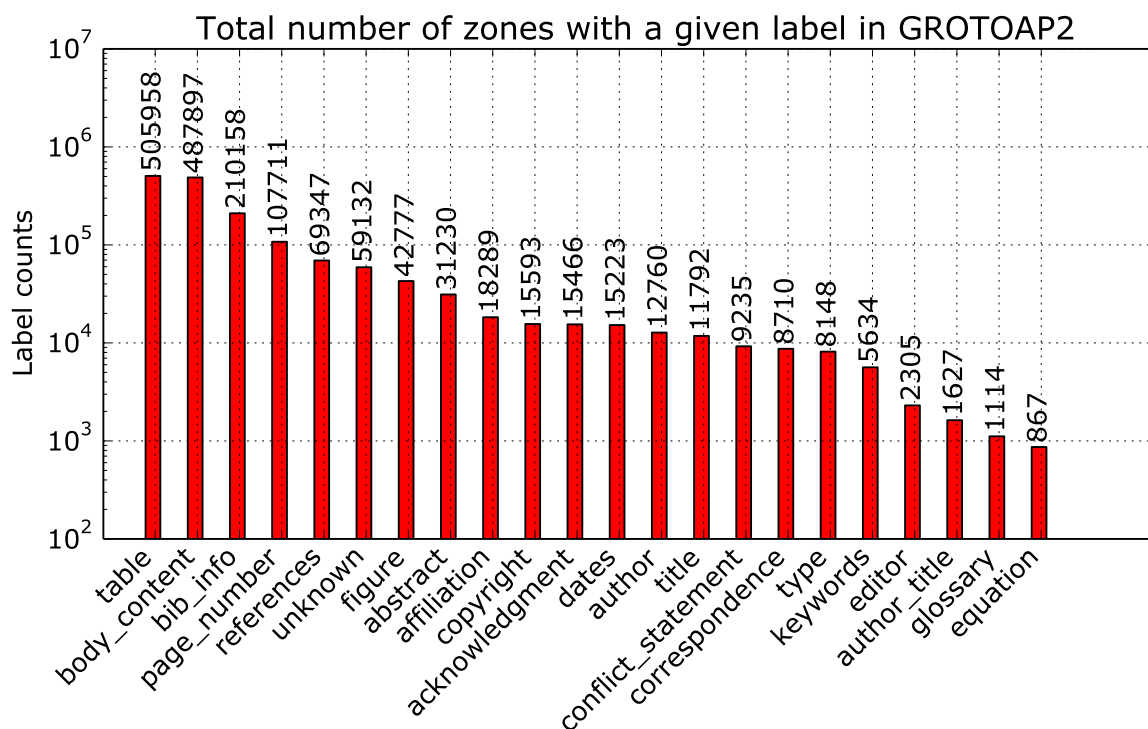


Figure 5.3: Total count of zones labeled with given label in GROTOAP2

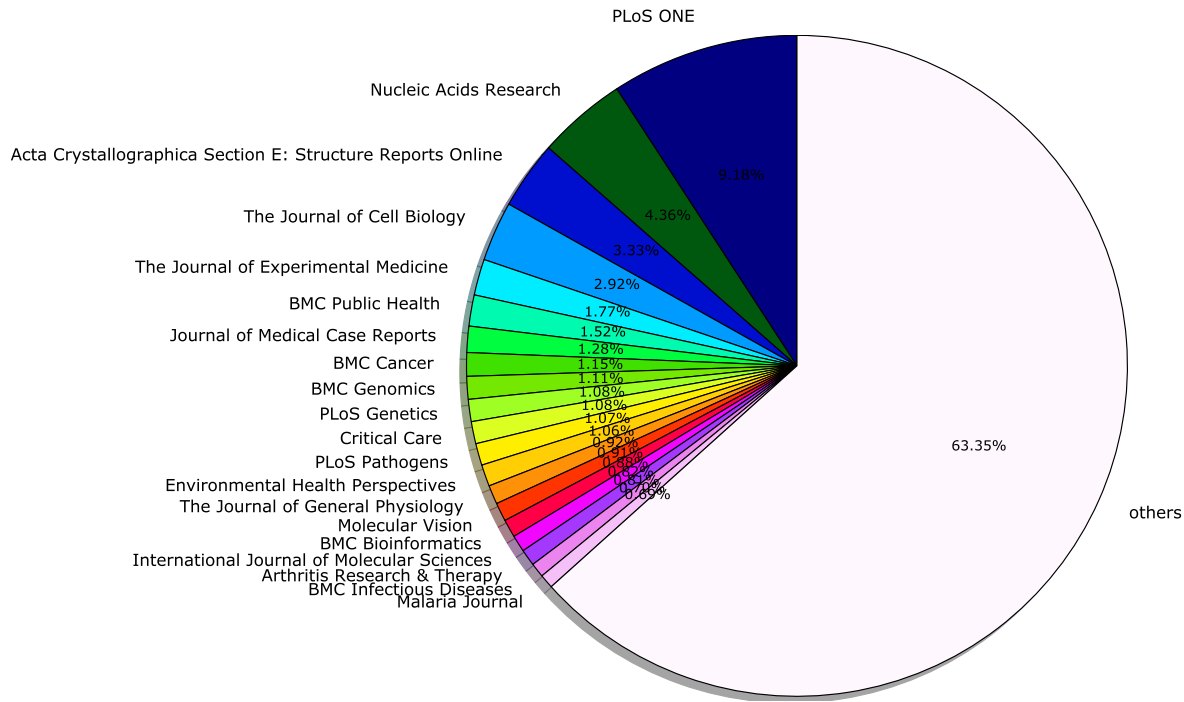


Figure 5.4: Top 25 journals in the GROTOAP2 dataset

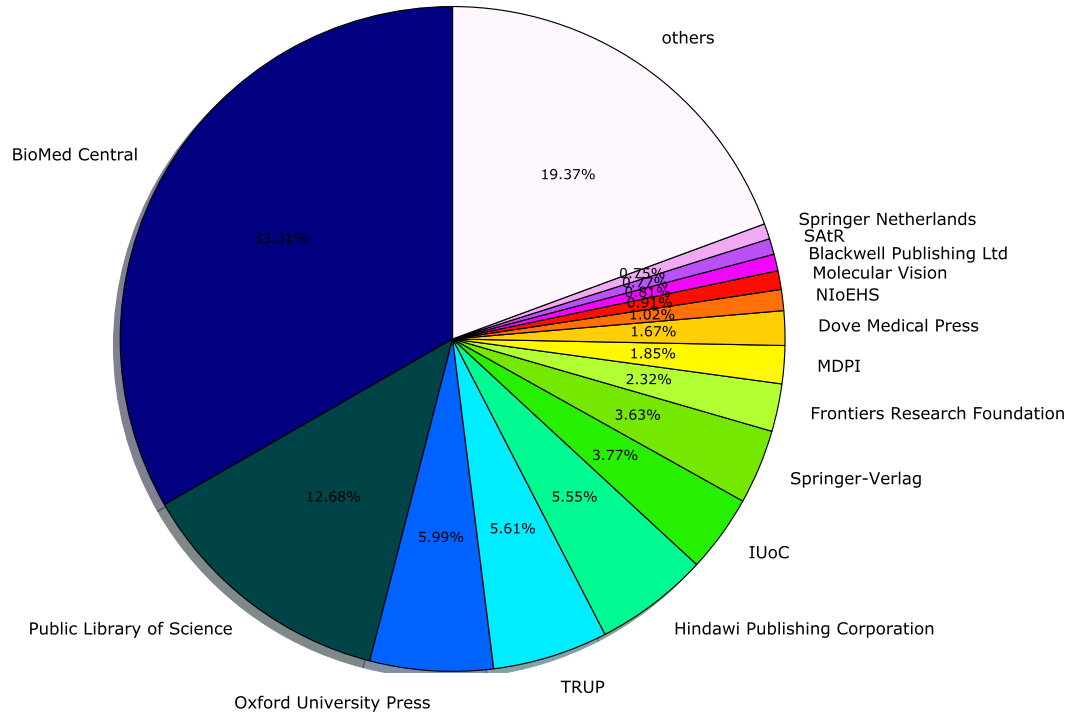


Figure 5.5: Top 15 publishers in the GROTOAP2 dataset

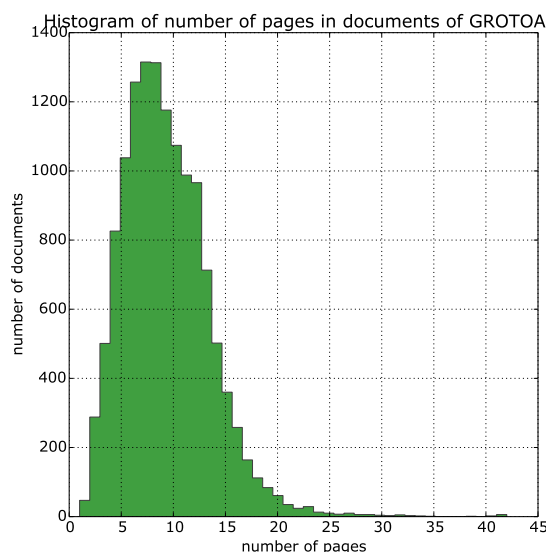


Figure 5.6: Histogram of number of pages of the documents in the GROTOA2 dataset

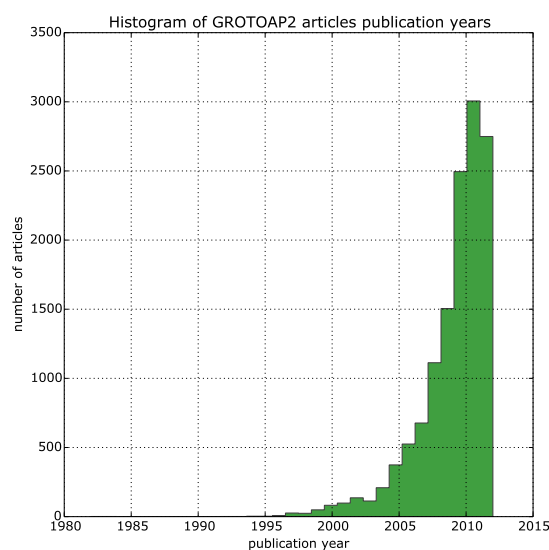


Figure 5.7: Histogram of publication year of the documents in the GROTOA2 dataset

5.9 Dataset evaluation

Evaluation of the created dataset is precisely described in [DB14]. Evaluation was not done by the author of this thesis. It was entirely performed by author's team mates from the Interdisciplinary Center for Mathematical and Computation Modeling. Nevertheless, its results and methodology will be quoted here in order to make the description more complete.

For the purpose of evaluation there was a handful of 50 documents picked. Those documents were automatically labeled and then errors were rectified. Subsequently, the corrected documents and the original ones were compared and the factors of precision and recall were calculated. Table 5.1 shows the values obtained for each label.

Another evaluation of the dataset was done by comparing CERMINE performance in two scenarios: trained with GROTOA2 and a random handful of 1000 documents from GROTOA2. Test documents were randomly selected from the Pubmed Central collection. Table 5.2 shows results of the evaluation. One can notice that the F-score of classifier based on GROTOA2 has grown by 16.93%.

label	precision	recall	label	precision	recall
abstract	0.98	0.98	equation	-	-
acknowledgment	1.00	0.90	figure	0.99	0.46
affiliation	0.95	0.95	glossary	1.00	1.00
author	1.00	0.98	keywords	1.00	0.94
author_title	1.00	1.00	page_number	0.98	0.97
bib_info	0.96	0.94	references	0.99	0.95
body	0.88	0.99	table	0.98	0.96
conflict_statement	0.82	0.89	title	1.00	1.00
copyright	0.93	0.78	type	0.89	0.47
correspondence	1.00	0.97	unknown	0.62	0.94
dates	0.94	1.00	*	0.95	0.91
editor	1.00	1.00			

Table 5.1: Evaluation metrics for the GROTOAP2 zone labeling process. The automatically generated metrics were manually compared with the actual labels on a subset of GROTOAP2. For each class precision and recall were calculated.

Source: [DB14]

	GROTOAP	GROTOAP2
precision	77.13%	82.22%
recall	55.99%	76.96%
F-score	62.41%	79.34%

Table 5.2: Evaluation of GROTOAP2 using CERMINE trained with GROTOAP and GROTOAP2.

Chapter 6

Evaluation

Evaluation of the final system was performed in an automatic way using GROTOAP2 documents as input. We used a 5-fold cross-validation to better simulate behavior of the classifiers on unknown data and to rule out “lucky” division into subsets. In this way we employed all the samples for both training and validation. In an n -fold cross-validation the data set is randomly partitioned into n subsets. In each of n stages, a single subset is kept as the validation data and the remaining $n - 1$ subsets are used as training data. This process is repeated n times and the results are summed up together in a form of a confusion matrix. This matrix summarizes classification by saying what were the recognized and actual classes for the classified objects. To quantify goodness of the classification process for every class we use precision and recall, which are two metrics complementing each other.

Precision of classification is the fraction of correctly classified elements from a class to all samples/zones classified as this class. In other words, precision says what is the chance of a sample belonging to some class if it was classified as this class. In the context of confusion matrices 6.1 and 6.2, precision is the fraction of the number typed in bold to the sum of the whole column.

Recall of classification is the fraction of correctly classified elements from a class to all samples from the class. It intuitively indicates what fraction of a class was classified properly. In the context of confusion matrices 6.1 and 6.2, recall is the fraction of the number typed in bold to the sum of the whole row.

6.1 Initial classification evaluation

For the evaluation of the initial classification we used 262'144 samples. This is much less than available in GROTOAP2, which includes 1'640'258 samples, since we

	BODY	METADATA	REFERENCES	OTHER	precision	recall
BODY	167467	1394	350	769	96.95	98.52
METADATA	2094	51707	52	392	95.99	95.32
REFERENCES	1457	45	9641	97	95.67	87.77
OTHER	1724	719	34	24202	95.06	90.72

Table 6.1: Confusion matrix for the initial zone classification in CERMINE obtained in a 5-fold cross-validation. Rows and columns contain desired and obtained labels respectively.

needed to limit number of samples to keep the evaluation in a reasonable timescale. With the original number of samples the evaluation would not be possible, because:

- The training process involves matrix inversion, which has $\mathcal{O}(n^3)$ complexity with respect to the training set. This means that for the full documents set from GROTOAP2 it would take around 250 times more. With the limited number of samples the evaluation was done within 15 hours on a decent server machine.
- `libsvm` has a single-thread implementation only. It is therefore impossible to fully use a multi-core machine with this tool. In our research we tried to use `jlibsvm`, which is multi-threaded implementation of `libsvm` in Java, but it was not stable enough.

Table 6.1 presents results of a 5-fold cross-validation, including a confusion matrix, as well as precision and recall values. We can state, that the first stage of classification yields good results. All the classes have precision over 95%, which is an extraordinary number. Conversely, recall of the **REFERENCE** class is equal to 87%, since a lot of references were classified as body elements. This issue needs to be addressed in next releases of CERMINE.

6.2 Metadata classification evaluation

For the evaluation of the metadata classification we used 341'368 samples, this is all the samples available in GROTOAP2. Evaluation of the metadata classification was done independently from the first stage, i.e. from initial classification. We therefore assumed that the first stage is done without errors (precision = recall = 100%) and all the metadata are passed to this stage. Similarly to what was done in 6.1, we performed a 5-fold cross-validation whose results are presented in the table 6.2. All the classes were classified with precision greater than 90%. The one with the lowest precision is **correspondence**, since often **bib_info**, **author** and **affiliation** were classified as this class.

Recall is in most cases held at a high level (over 92%), with the exception of **type** and **keywords**. The former is often misclassified as **bib_info**, the latter is misclassified as **bib_info** and **abstract**. These errors are very likely caused by imprecise segmentation algorithm, as well as errors in GROTOAP2 matching algorithm, what should be addressed in the next release in order to achieve better performance.

In order to evaluate complete CERMINE performance, we need to take into account the fact that not all the zones in the initial classification are classified correctly. Let us consider **title** as an example in this analysis. Only 95% of all metadata zones are correctly classified and only 95% of all zones classified as metadata are real metadata. **title**'s recall and precision are equal to 98%. We might estimate that 93% of all zones classified as **title** are real titles, the remaining 7% are random. Good classification results have to be considered from

two perspectives:

- the results are comparable to the state-of-the-art system presented in [KJHG12]

	abstract	affiliation	author	bib_info	copyright	correspondence	dates	editor	keywords	title	title_author	type	precision[%]	recall[%]
abstract	30583	27	8	403	33	42	5	2	85	7	0	25	98.18	97.96
affiliation	40	17483	39	257	19	365	26	19	22	3	1	11	95.55	97.96
author	12	59	12116	270	5	264	2	3	3	6	5	14	97.71	94.46
bib_info	278	117	98	208099	117	496	317	2	133	66	10	359	97.62	99.05
copyright	40	13	13	956	14419	58	40	0	5	1	14	25	95.72	92.58
correspondence	48	535	62	239	12	7783	15	1	12	1	0	0	90.04	89.38
dates	13	5	2	1115	35	6	14026	1	11	4	1	0	97.15	92.16
editor	1	34	4	1	0	0	1	2264	0	0	0	0	98.69	98.22
keywords	104	23	21	880	1	9	3	2	4546	21	0	23	93.67	80.70
title	17	1	11	66	4	0	1	0	3	11642	25	21	98.61	98.74
title_author	3	0	1	22	6	0	0	0	0	33	1565	0	96.53	96.00
type	12	0	25	872	33	0	2	0	33	22	0	7148	93.73	87.74

Table 6.2: Confusion matrix obtained for classification of the metadata in CERMINE in a 5-fold cross-validation using documents from GROTOAP2. Rows and columns contain desired and obtained labels respectively.

6.3 Future work

6.3.1 GROTOAP2

GROTOAP2 is a rich data set created in a semi-automatized way from Open Access publications. It contains 13210 articles representing a big variety of layouts and issued by more than 200 publishers. We find this set very useful for every data mining algorithm willing to extract metadata, data and other various features from scholarly articles in an automated way. GROTOAP2 can be easily applied to testing or learning every algorithm designed for processing born-digital documents. We believe that GROTOAP2 can be made more versatile by:

- Growing its size by an order or two orders of magnitude.
- Improving the algorithm used to assign NML metadata to text zones by tuning the magic constants used there.
- Tuning the segmentation algorithm and improving segmentation quality. This would result in removing the `author_title` tag.
- Dividing the `body_content` into subcategories based on the level of their headers.
- Including fine-grained author metadata, e.g. first name, middle name, surname.
- Introducing association between authors and theirs affiliations, i.e. pointing out which author is associated with which affiliation.
- Including fine-grained bibliographic reference data, making it applicable for reference-extraction algorithms.

Most of the above-mentioned cases are related to losing as little information as possible from the original Pubmed XML files.

6.3.2 CERMINE

CERMINE is a robust system for metadata extraction. It can be used to accurately extract fine-grained metadata from scientific articles en masse. CERMINE can be very useful in every digital library application, as it is flexible and does not depend on any particular layout. What is more, it can be easily adapted in a semi-automatic way to some particular layout, if such need arises.

We consider that there are many ways to improve the system, e.g.:

-
- increase classification accuracy and recall. This can be done by:
 - choosing a different classification algorithm,
 - using an ensemble of algorithms,
 - performing lexical analysis on the zones classified as unknown,
 - increasing quality of the training set.
 - extract detailed bibliographic references,
 - extract figures and tables,
 - perform sentiment analysis with respect to the included references,
 - improve segmentation algorithm's on unusual layouts.

This page is intentionally printed only with this statement.

Appendix A

Description of the training workflow

Below one can find description of the complete procedure for training the system classifiers.

1. Get CERMINE source code. It is available publicly on the Web.

```
git clone https://github.com/pszostek/CERMINE.git
cd CERMINE
```

2. Build CERMINE using maven by running the following Bash script from the CERMINE main directory. It assumes that the PATH variable contains a path to mvn.

```
#!/usr/bin/env bash
curdir=$PWD
cd cermine-impl && mvn package -DskipTests && mvn assembly:single && cd
    $curdir
cd cermine-tools && mvn package -DskipTests && mvn assembly:single && cd
    $curdir
```

3. Create files with samples for the classifiers using GROTOAP2. To this end, one need to run the following script. Note that the GROTOAP2_PATH variable has to point to a directory containing cxml files from the dataset.

```
#!/usr/bin/env bash
BASE=$(dirname $(pwd))
```

```
java -Djava.util.Arrays.useLegacyMergeSort=true -cp \
    $BASE/cermine-tools/target/cermine-tools-1.2-SNAPSHOT-jar-with-dependencies.jar:\
    $BASE/cermine-impl/target/cermine-impl-1.2-SNAPSHOT-jar-with-dependencies.jar:\
    $BASE/cermine-impl/target/cermine-impl-1.2-SNAPSHOT.jar
    pl.edu.icm.cermine.libsvm.LibSVMExporter $GROTOAP2_PATH
```

4. Scale down the samples' feature values so that those with relatively huge absolute values do not dominate those with smaller values. This is a step recommended in [CL10]. It can be done by using the svm-scale tool from libsvm package.

```
svm-scale -l -1 -u 1 -s range1 meta_GROTOAP2 > meta_GROTOAP2.scale
svm-scale -l -1 -u 1 -s range1 initial_GROTOAP2 > initial_GROTOAP2.scale
```

5. Find the best parameters for the classifiers using grid.py from the libsvm package. The below script can facilitate iterating over different kernel types. One need to pick the classifier with the lowest error rate.

```
#!/usr/bin/env bash
for classifier in meta initial; do
    for kernel in $(seq 0 3); do
        printf "kernel=$kernel: "
        if [ $kernel -eq 1 ]; then
            degree=" -d 4 "
        else
            degree=""
        fi
        /root/grid.py -t $kernel -gnuplot /usr/local/bin/gnuplot
        -svmtrain /root/libsvm-3.18/svm-train $degree
        ${classifier}_GROTOAP.dat.scale
    done
done
```

6. Using the optimal parameters, train the classifiers and save them to files. The below command has to be run from the main CERMINE directory. The variables KERNEL_I, G_I, C_I and D_I indicate respectively the kernel type, the γ parameter, the C parameter and possibly polynomial degree of the initial zone classification kernel. Variables KERNEL_M, G_M, C_M and D_M indicate respectively the kernel type, the γ parameter, the C parameter and possibly polynomial degree of the metadata classification kernel.

```
#!/usr/bin/env bash

java -Djava.util.Arrays.useLegacyMergeSort=true -cp \
./cermine-tools/target/cermine-tools-1.2-SNAPSHOT-jar-with-dependencies.jar\
./cermine-impl/target/cermine-impl-1.2-SNAPSHOT-jar-with-dependencies.jar\
./cermine-impl/target/cermine-impl-1.2-SNAPSHOT.jar\
pl.edu.icm.cermine.tools.classification.svm.SVMInitialBuilder -input\
initial_GROTOAP2.dat.scale -output CERMINE_initial_classifier\
-kernel $KERNEL_I -g $G_I -C $C_I [-degree $D_I]

java -Djava.util.Arrays.useLegacyMergeSort=true -cp \
./cermine-tools/target/cermine-tools-1.2-SNAPSHOT-jar-with-dependencies.jar\
./cermine-impl/target/cermine-impl-1.2-SNAPSHOT-jar-with-dependencies.jar\
./cermine-impl/target/cermine-impl-1.2-SNAPSHOT.jar\
pl.edu.icm.cermine.tools.classification.svm.SVMMetadataBuilder -input\
meta_GROTOAP2.dat.scale -output CERMINE_meta_classifier\
-kernel $KERNEL_M -g $G_M -C $C_M [-degree $D_M]
```

7. Parameters from the previous step can be used to evaluate the classifier. CERMINE contains full classes that perform n-fold cross-validation. To obtain full validation reports, one need to run the commands that follow.

```
#!/usr/bin/env bash

java -Djava.util.Arrays.useLegacyMergeSort=true -cp \
./cermine-tools/target/cermine-tools-1.2-SNAPSHOT-jar-with-dependencies.jar\
./cermine-impl/target/cermine-impl-1.2-SNAPSHOT-jar-with-dependencies.jar\
./cermine-impl/target/cermine-impl-1.2-SNAPSHOT.jar\
pl.edu.icm.cermine.evaluation.SVMInitialClassificationEvaluator -full\
-input initial_GROTOAP2.dat \
-kernel $KERNEL_I -g $G_I -C $C_I [-degree $D_I]

java -Djava.util.Arrays.useLegacyMergeSort=true -cp \
./cermine-tools/target/cermine-tools-1.2-SNAPSHOT-jar-with-dependencies.jar\
./cermine-impl/target/cermine-impl-1.2-SNAPSHOT-jar-with-dependencies.jar\
./cermine-impl/target/cermine-impl-1.2-SNAPSHOT.jar\
pl.edu.icm.cermine.evaluation.SVMMetadataClassificationEvaluator -full\
-input metadata_GROTOAP2.dat \
-kernel $KERNEL_I -g $G_I -C $C_I [-degree $D_I]
```

Appendix B

Description of classification features

Below the reader can find description of all the features used in the zone classification process.

Feature Name	Type	Description
Affiliation	bool	Tells if the zone contains one of the <i>{author details, university, department, school, institute, affiliation}</i>
Author	bool	Tells if a text zone contains a word starting with <i>author</i>
Bibinfo	int	Number of occurrences of <i>{cite, pages, article, volume, publishing, journal, doi, cite this article, citation, issue, issn}</i>
BracketRelativeCount	float	Ratio of the number of brackets to the number of all characters
BracketedLineRelativeCount	float	Ratio of number of lines starting with a bracket to the number of all lines
CharCount	int	Counts number of characters in a zone
CharCountRelative	float	Ratio of number of characters in a zone to the number of character on its page
CommaCount	int	Counts number of commas in a zone

CommaRelativeCount	int	Ratio of commas to all characters in a zone
ContainsPageNumber	bool	Does the zone consists of a number or is like <i>Page %d</i> or <i>page %d</i> , where %d is a number
CuePhrasesRelativeCount	float	Ratio of cue phrases from { <i>although, therefore, therein, hereby, nevertheless, to this end, however, moreover, nonetheless</i> } to all words
Date	bool	Tells if the zone contains a string being a date with months expressed as text or as a number, with either European or American order
DigitCount	int	Number of digits is in the zone
DigitRelativeCount	float	Ratio of digits to all characters in the zone
DistanceFromNearestNeighbour	float	Distance to the nearest neighbour zone or returns Double.MAX_VALUE if there is no other zone on the page
DotCount	int	Counts number of dots (.) in the zone
DotRelativeCount	float	Ratio of number of dots to all characters in the zone
Email	bool	Tells if the zone contains a valid e-mail address
EmptySpaceRelative	float	Calculates a ratio of the surface taken by the characters' bounding boxes to the area of the zone
FontHeightMean	float	Calculates average font size in the zone
FreeSpaceWithinZone	float	Difference between zone's area and total area taken by the characters
Height	int	Height of a zone
HeightRelative	float	Ratio of zone's height to the page's height
HorizontalRelativeProminence	float	Area taken by the zone and free area to the West and East of the page. This features tries to fit into the page a bounding box of the same height as the zone and sticking either to neighbour zones or to the page boundaries in E and W

IsAnywhereElse	bool	Tells if this zone is duplicated anywhere in the investigated document
IsFirstPage	bool	Tells if the zone is located on the first page of the document
IsFontBiggerThanNeighbours	bool	Tells whether the font of the zone is bigger than in neighbour zones
IsGreatestFontOnPage	bool	Tells if the zone has the greatest font on the page
IsHighestOnThePage	bool	Tells if the zone is the highest on the page
IsItemize	bool	Tells if the zone contains any kind of indexing characteristic for section or enumeration numbering
IsWidestOnThePage	bool	Tells if the zone is the widest on the page
IsLastButOnePage	bool	Tells if the zone is located on the last but one page
IsLastPage	bool	Tells if the zone is located on the last page
IsLeft	bool	Tells if the zone is in the left part of the page
IsLongestOnThePage	bool	Tells if the zone contains maximal number of characters among all the zones in the page
IsLowestOnThePage	bool	Tells if the zone is most southern on the page (southern bounding box border is taken into account)
IsOnSurroundingPages	bool	Tells if the same zone is on the previous or on the next page
IsPageNumber	bool	Tells if a zone is entirely a number
IsRight	bool	Tells if a zone is on the east side of the page
IsSingleWord	bool	Tells if a zone is one word
LastButOneZone	bool	Tells if the zone is last but one with respect to the reading order
LineCount	int	Number of lines in the zone
LineRelativeCount	float	Ratio of lines in the to all lines on the page
LineHeightMean	float	Mean height of the zone
LineWidthMean	float	Mean width of the zone's lines

LineXPositionMean	float	Mean distance of zone's lines to the zone's bounding box
LineXPositionDiff	float	The feature looks at the left border of all lines within a zone and finds the most-eastern and most-western among them. A positive difference between these two values is returned
LineXWidthPositionDiff	float	Difference between mean value of lines' left border X coordinate and zone's
LetterCount	int	Number of letters in the zone
LetterRelativeCount	float	Ratio of letters in the zone to all letters on the page
LowercaseCount	int	Number of letters in lower case in the zone
LowercaseRelativeCount	float	Number of letters in lower case in the zone
PageNumber	int	Page number wrt. the reading order
PreviousZone	int	Value of label assigned to the previous zone wrt. the reading order
Proportions	float	Width to height ratio
PunctuationRelativeCount	float	Ratio of punctuation marks to all characters in the zone
References	int	Number of characteristic enumeration indices in the zone, e.g. 1..
StartsWithDigit	bool	Tells if the zone's first character is a digit
UppercaseCount	int	Number of upper case characters in the zone
UppercaseRelativeCount	float	Ratio of upper case letters to all characters
UppercaseWordCount	int	Number of words starting with upper case
UppercaseWordRelativeCount	float	Ratio of upper case words to all words
VerticalProminence	float	Space taken by the zone and free area to the North and South of the page. This features tries to fit into the page a bounding box of the same height as the zone and sticking either to neighbor zones or to the page boundaries in N and S

Width	int	Width of the bounding box in pixels
WordCount	int	Number of words in the zone
WordCountRelative	float	Ratio of number of words in the zone to all words in the page
WordWidthMean	float	Mean width of words in the zone expressed as pixels
WordLengthMean	float	Mean width of words in the zone expressed as number of characters
WordLengthMedian	int	Median of number of characters in all words
WhitespaceCount	int	Number of white characters in the text
WhitespaceRelativeCountLog	float	Feature calculated as $-\log \frac{S}{L}$ where S is zone's empty area and L is zone's text length
WidthRelative	float	Ratio of zone's width to page's width
XPosition	int	X coordinate
XPositionRelative	float	Ratio of zone's X coordinate to page's width
YPosition	int	Y coordinate
YPositionRelativeFeature	float	Ratio of zone's Y coordinate to page's height

Appendix C

Reading order resolving

Below we include the code for resolution of the reading order. This codes serves as a reference for a potential thesis' reviewer willing to assess author's coding skills.

```
public class HierarchicalReadingOrderResolver implements ReadingOrderResolver {

    static final int GRIDSIZE = 50;
    static final double BOXES_FLOW = 0.5;
    static final double EPS = 0.0001;
    static final int MAX_ZONES = 1000;
    static final Comparator<BxObject> Y_ASCENDING_ORDER = new
        Comparator<BxObject>() {

        @Override
        public int compare(BxObject o1, BxObject o2) {
            if((o1.getY() - o2.getY()) > EPS ) {
                return 1;
            } else if(Math.abs(o1.getY() - o2.getY()) < EPS) {
                return 0;
            } else {
                return -1;
            }
        }
    };
};
```

```

static final Comparator<BxObject> X_ASCENDING_ORDER = new
    Comparator<BxObject>() {

    @Override
    public int compare(BxObject o1, BxObject o2) {
        if(o1.getX()-o2.getX() > EPS) {
            return 1;
        } else if(Math.abs(o1.getX() - o2.getX()) < EPS) {
            return 0;
        } else {
            return -1;
        }
    }
};

static final Comparator<BxObject> YX_ASCENDING_ORDER = new
    Comparator<BxObject>() {

    @Override
    public int compare(BxObject o1, BxObject o2) {
        int yCompare = Y_ASCENDING_ORDER.compare(o1, o2);
        return yCompare == 0 ? X_ASCENDING_ORDER.compare(o1, o2) : yCompare;
    }
};

@Override
public BxDocument resolve(BxDocument messyDoc) {
    BxDocument orderedDoc = new BxDocument();
    List<BxPage> pages = messyDoc.getPages();
    for (BxPage page : pages) {
        List<BxZone> zones = page.getZones();
        for (BxZone zone : zones) {
            List<BxLine> lines = zone.getLines();
            for (BxLine line : lines) {
                List<BxWord> words = line.getWords();
                for (BxWord word : words) {
                    List<BxChunk> chunks = word.getChunks();

```

```

        Collections.sort(chunks, X_ASCENDING_ORDER);
    }
    Collections.sort(words, X_ASCENDING_ORDER);
}
Collections.sort(lines, Y_ASCENDING_ORDER);
}
List<BxZone> orderedZones;
if (zones.size() > MAX_ZONES) {
    orderedZones = new ArrayList<BxZone>(zones);
    Collections.sort(orderedZones, YX_ASCENDING_ORDER);
} else {
    orderedZones = reorderZones(zones);
}
page.setZones(orderedZones);
orderedDoc.addPage(page);
}
setIdsAndLinkTogether(orderedDoc);
return orderedDoc;
}

/**
 * Builds a binary tree from list of text zones by doing a hierarchical
 * clustering and converting the result tree to
 * an ordered list.
 *
 * @param zones is a list of unordered zones
 * @return a list of ordered zones
 */
private List<BxZone> reorderZones(List<BxZone> unorderedZones) {
    if (unorderedZones.isEmpty()) {
        return new ArrayList<BxZone>();
    } else if (unorderedZones.size() == 1) {
        List<BxZone> ret = new ArrayList<BxZone>(1);
        ret.add(unorderedZones.get(0));
        return ret;
    } else {
        BxZoneGroup bxZonesTree = groupZonesHierarchically(unorderedZones);
        sortGroupedZones(bxZonesTree);
    }
}

```

```

        TreeToListConverter treeConverter = new TreeToListConverter();
        List<BxZone> orderedZones = treeConverter.convertToList(bxZonesTree);
        assert unorderedZones.size() == orderedZones.size();
        return orderedZones;
    }
}

/**
 * Generic function for setting IDs and creating a linked list by filling
 * references. Used solely by
 * setIdsAndLinkTogether(). Can Handle all classes implementing Indexable
 * interface.
 *
 * @param list is a list of Indexable objects
 */
private <A extends Indexable<A>> void setIdsGenericImpl(List<A> list) {
    if (list.isEmpty()) {
        return;
    }
    if (list.size() == 1) {
        A elem = list.get(0);
        elem.setNext(null);
        elem.setPrev(null);
        elem.setId("0");
        elem.setNextId("-1");
        return;
    }

    //unroll the loop for the first and last element
    A firstElem = list.get(0);
    firstElem.setId("0");
    firstElem.setNextId("1");
    firstElem.setNext(list.get(1));
    firstElem.setPrev(null);
    for (int idx = 1; idx < list.size() - 1; ++idx) {
        A elem = list.get(idx);
        elem.setId(Integer.toString(idx));
        elem.setNextId(Integer.toString(idx + 1));
    }
}

```

```

        elem.setNext(list.get(idx + 1));
        elem.setPrev(list.get(idx - 1));
    }
    A lastElem = list.get(list.size() - 1);
    lastElem.setId(Integer.toString(list.size() - 1));
    lastElem.setNextId("-1");
    lastElem.setNext(null);
    lastElem.setPrev(list.get(list.size() - 2));
}

/**
 * Function for setting up indices and reference for the linked list. Causes
 *   objects of BxPage, BxZone, BxLine,
 * BxWord and BxChunk to be included in the document's list of elements and
 *   sets indices according to the
 * corresponding list order.
 *
 * @param doc is a reference to a document with properly set reading order
 */
private void setIdsAndLinkTogether(BxDocument doc) {
    setIdsGenericImpl(doc.asPages());
    setIdsGenericImpl(doc.asZones());
    setIdsGenericImpl(doc.asLines());
    setIdsGenericImpl(doc.asWords());
    setIdsGenericImpl(doc.asChunks());
}

/**
 * Builds a binary tree of zones and groups of zones from a list of
 *   unordered zones. This is done in hierarchical
 * clustering by joining two least distant nodes. Distance is calculated in
 *   the distance() method.
 *
 * @param zones is a list of unordered zones
 * @return root of the zones clustered in a tree
 */
private BxZoneGroup groupZonesHierarchically(List<BxZone> zones) {
    /*

```

```

    * Distance tuples are stored sorted by ascending distance value
    */
List<DistElem<BxObject>> dists = new
    ArrayList<DistElem<BxObject>>(zones.size()*zones.size()/2);
for (int idx1 = 0; idx1 < zones.size(); ++idx1) {
    for (int idx2 = idx1 + 1; idx2 < zones.size(); ++idx2) {
        BxZone zone1 = zones.get(idx1);
        BxZone zone2 = zones.get(idx2);
        dists.add(new DistElem<BxObject>(false, distance(zone1, zone2),
            zone1, zone2));
    }
}
Collections.sort(dists);
DocumentPlane plane = new DocumentPlane(zones, GRIDSIZE);
while (!dists.isEmpty()) {
    DistElem<BxObject> distElem = dists.get(0);
    dists.remove(0);
    if (!distElem.isC() && plane.anyObjectsBetween(distElem.getObj1(),
        distElem.getObj2())) {
        dists.add(new DistElem<BxObject>(true, distElem.getDist(),
            distElem.getObj1(), distElem.getObj2()));
        continue;
    }

    BxZoneGroup newGroup = new BxZoneGroup(distElem.getObj1(),
        distElem.getObj2());
    plane.remove(distElem.getObj1()).remove(distElem.getObj2());
    dists = removeDistElementsContainingObject(dists, distElem.getObj1());
    dists = removeDistElementsContainingObject(dists, distElem.getObj2());
    for (BxObject other : plane.getObjects()) {
        dists.add(new DistElem<BxObject>(false, distance(other,
            newGroup), newGroup, other));
    }
    Collections.sort(dists);
    plane.add(newGroup);
}

```

```

        assert plane.getObjects().size() == 1 : "There should be one object left
            at the plane after grouping";
        return (BxZoneGroup) plane.getObjects().get(0);
    }

    /**
     * Removes all distance tuples containing obj
     */
    private List<DistElem<BxObject>>
        removeDistElementsContainingObject(Collection<DistElem<BxObject>> list,
            BxObject obj) {
        List<DistElem<BxObject>> ret = new ArrayList<DistElem<BxObject>>();
        for (DistElem<BxObject> distElem : list) {
            if (distElem.getObj1() != obj && distElem.getObj2() != obj) {
                ret.add(distElem);
            }
        }
        return ret;
    }

    /**
     * Swaps children of BxZoneGroup if necessary. A group with smaller sort
     * factor is placed to the left (leftChild).
     * An object with greater sort factor is placed on the right (rightChild).
     * This plays an important role when
     * traversing the tree in conversion to a one dimensional list.
     *
     * @param group
     */
    private void sortGroupedZones(BxZoneGroup group) {
        BxObject leftChild = group.getLeftChild();
        BxObject rightChild = group.getRightChild();
        if (shouldBeSwapped(leftChild, rightChild)) {
            // swap
            group.setLeftChild(rightChild);
            group.setRightChild(leftChild);
        }
    }

```

```

    if (leftChild instanceof BxZoneGroup) // if the child is a tree node,
        then recurse
    {
        sortGroupedZones((BxZoneGroup) leftChild);
    }
    if (rightChild instanceof BxZoneGroup) // as above - recurse
    {
        sortGroupedZones((BxZoneGroup) rightChild);
    }
}

private boolean shouldBeSwapped(BxObject first, BxObject second) {
    double cx, cy, cw, ch, ox, oy, ow, oh;
    cx = first.getBounds().getX();
    cy = first.getBounds().getY();
    cw = first.getBounds().getWidth();
    ch = first.getBounds().getHeight();

    ox = second.getBounds().getX();
    oy = second.getBounds().getY();
    ow = second.getBounds().getWidth();
    oh = second.getBounds().getHeight();

    // Determine Octant
    //
    // 0 | 1 | 2
    // __|___|__
    // 7 | 9 | 3 First is placed in 9th square
    // __|___|__
    // 6 | 5 | 4

    if (cx + cw <= ox) { //2,3,4
        return false;
    } else if (ox + ow <= cx) { //0,6,7
        return true; //6
    } else if (cy + ch <= oy) {
        return false; //5
    } else if (oy + oh <= cy) {

```

```

        return true; //1
    } else { //two zones
        double xdiff = ox+ow/2 - cx-cw/2;
        double ydiff = oy+oh/2 - cy-ch/2;
        if (xdiff + ydiff < 0) {
            return true;
        }
        return false;
    }
}

/**
 * A distance function between two TextBoxes.
 *
 * Consider the bounding rectangle for obj1 and obj2. Return its area minus
 * the areas of obj1 and obj2, shown as
 * 'www' below. This value may be negative.
 * (x0,y0) +-----+www
 * | obj1 |www
 * +-----+www+-----+
 * www| obj2 |
 * www+-----+ (x1,y1)
 *
 * @return distance value based on objects' coordinates and physical size on
 * a plane
 */
private double distance(BxObject obj1, BxObject obj2) {

    double x0 = Math.min(obj1.getX(), obj2.getX());
    double y0 = Math.min(obj1.getY(), obj2.getY());
    double x1 = Math.max(obj1.getX() + obj1.getWidth(),
        obj2.getX() + obj2.getWidth());
    double y1 = Math.max(obj1.getY() + obj1.getHeight(),
        obj2.getY() + obj2.getHeight());
    double dist = ((x1 - x0) * (y1 - y0) - obj1.getArea() - obj2.getArea());

    double obj1CenterX = obj1.getX();

```

```
double obj1CenterY = obj1.getY() + obj1.getHeight() / 2;
double obj2CenterX = obj2.getX();
double obj2CenterY = obj2.getY() + obj2.getHeight() / 2;

double obj1obj2VectorCosineAbs = Math.abs((obj2CenterX - obj1CenterX) /
    Math.sqrt((obj2CenterX - obj1CenterX) * (obj2CenterX - obj1CenterX) +
        (obj2CenterY - obj1CenterY) * (obj2CenterY - obj1CenterY)));
final double MAGIC_COEFF = 0.5;
return dist * (MAGIC_COEFF + obj1obj2VectorCosineAbs);
}
}
```

Appendix D

Project resources

D.1 CERMINE repository

CERMINE was a collaborative project implemented as a part of author's work in ICM UW. Repository for the project can be found under <https://github.com/CeON/CERMINE/>. In total there were 217 commits added between 25.06.2012 and 10.01.2015 by the author of this work under usernames "Pawel Szostek" and "pszostek". They can be easily listed with the procedure below:

```
#!/usr/bin/env bash
git clone https://github.com/CeON/CERMINE.git
cd CERMINE
git log --author="Pawel Szostek"
git log --author="pszostek"
```

D.2 CERMINE interface

CERMINE interface was created as a side project to make the engine available for public use. It can be found under <http://cermine.ceon.pl/>. Figures D.1 and D.2 show the main interface and results from a run on a sample file.



Figure D.1: Main web interface for CERMINE as seen at <http://cermine.ceon.pl>

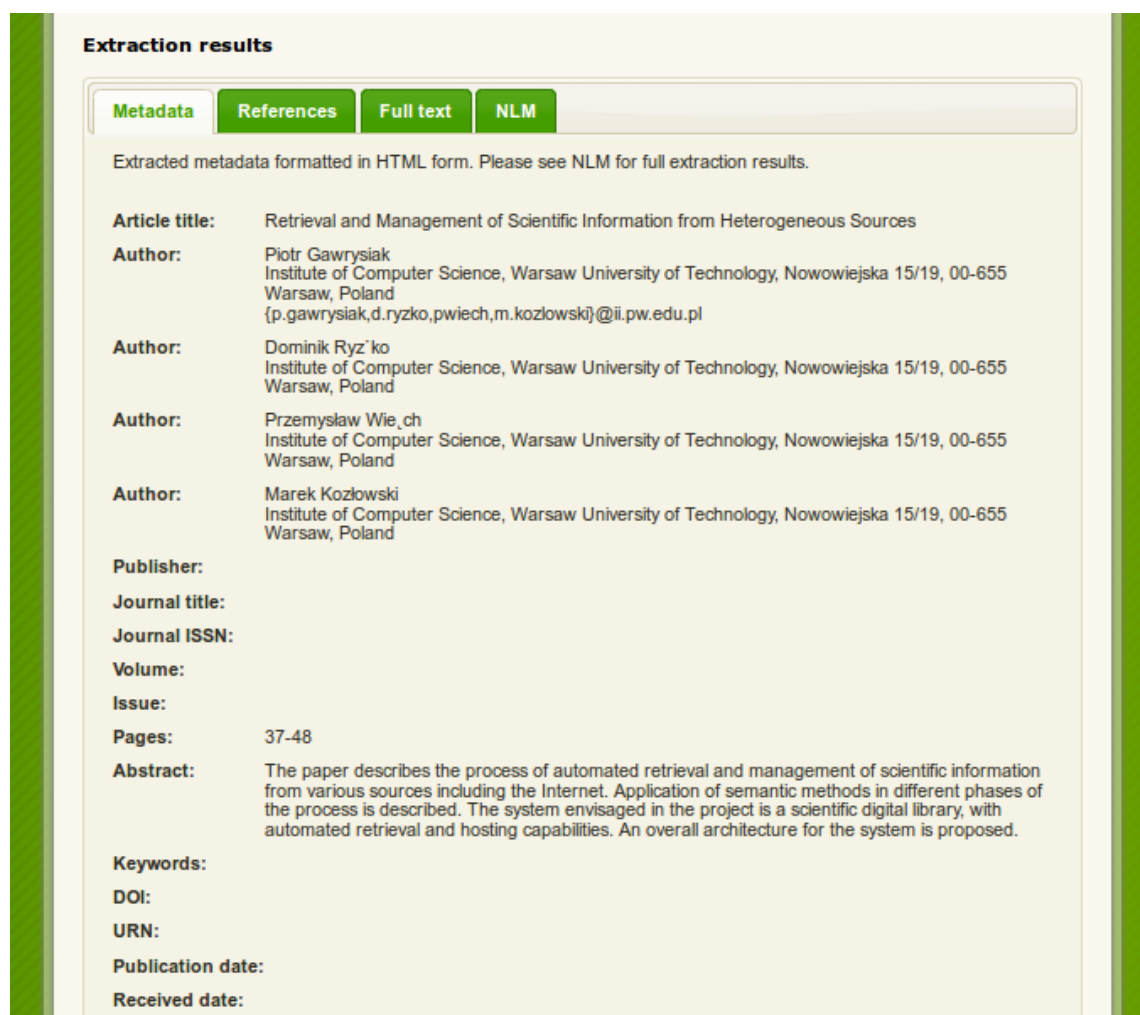


Figure D.2: Example of an output generated by CERMINE via the web interface

D.3 GROTOAP2

GROTOAP2 can be downloaded from the servers of the Center for Open Science (CeON) under <http://cermine.ceon.pl/grotoap2/>

D.4 This thesis

Sources of the thesis together with plots, graphics, data and processing scripts can be found on author's Github: <https://github.com/pszostek/msc>.

Appendix E

GROTOAP2 classes distributions

Figures E.1, E.2, E.3, E.4, E.5, E.6, E.7, E.8, E.9, E.10, E.11, E.12, E.13, E.14, E.15, E.16, E.17, E.18, E.19, E.20, E.21, E.22 show the density of different labels across the data set. One can see that in most of the documents the abstract is enclosed in only one zone. Nevertheless, there is a tail of the distribution going up to 40 zones. This can mean that there are articles where abstract is excessively long or that the segmentation algorithm failed.

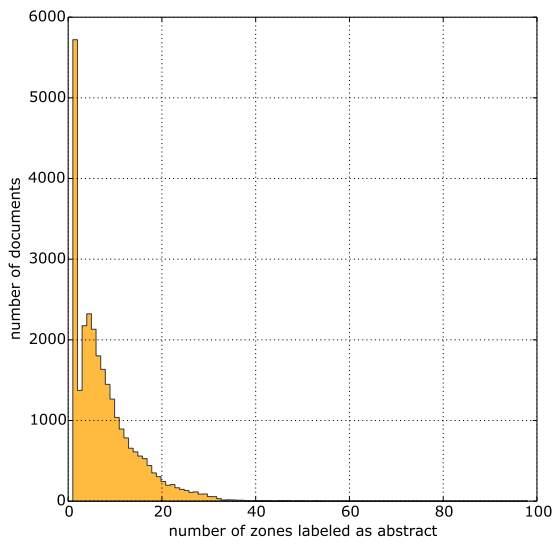


Figure E.1: Histogram of number of zones labeled as **abstract**

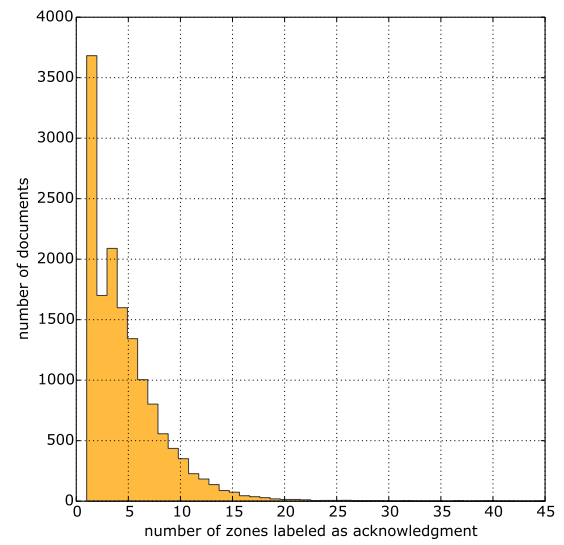


Figure E.2: Histogram of number of zones labeled as **acknowledgment**

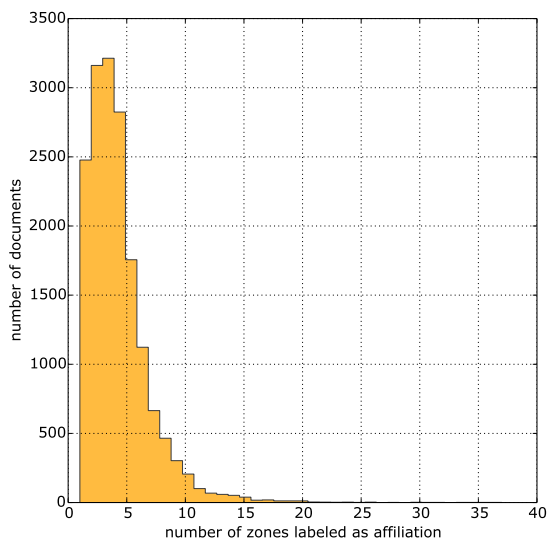


Figure E.3: Histogram of number of zones labeled as **affiliation**

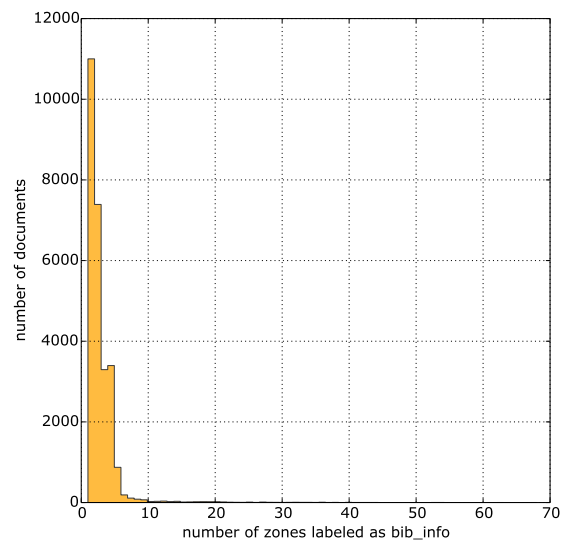


Figure E.4: Histogram of number of zones labeled as **bib_info**

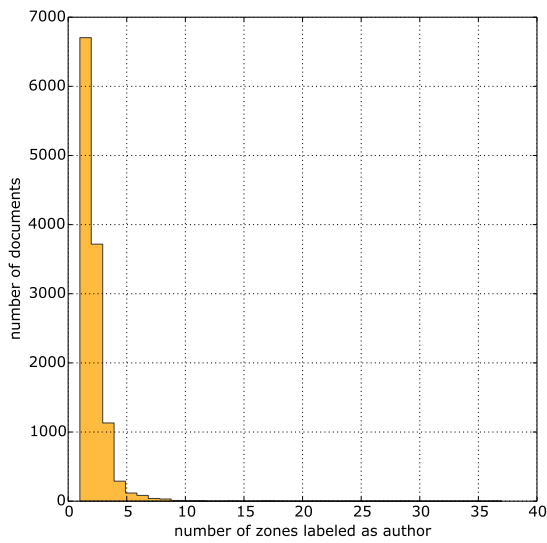


Figure E.5: Histogram of number of zones labeled as `author`

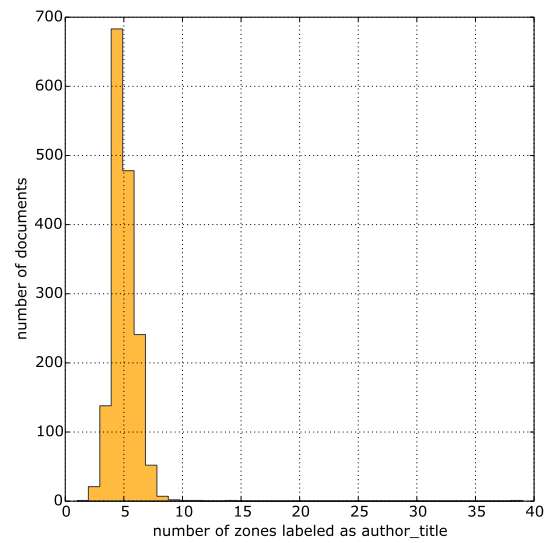


Figure E.6: Histogram of number of zones labeled as `author_title`

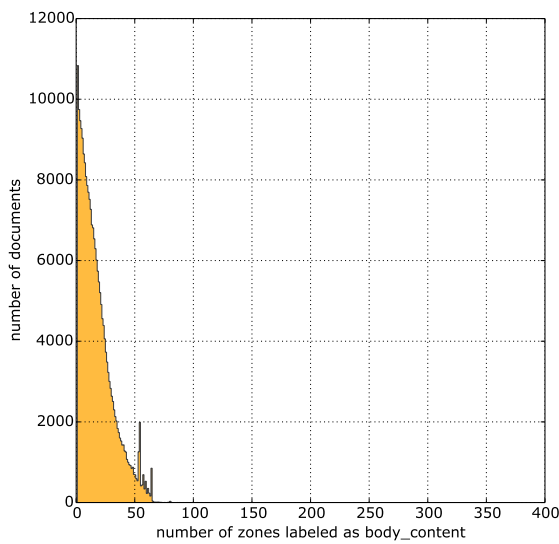


Figure E.7: Histogram of number of zones labeled as `body_content`

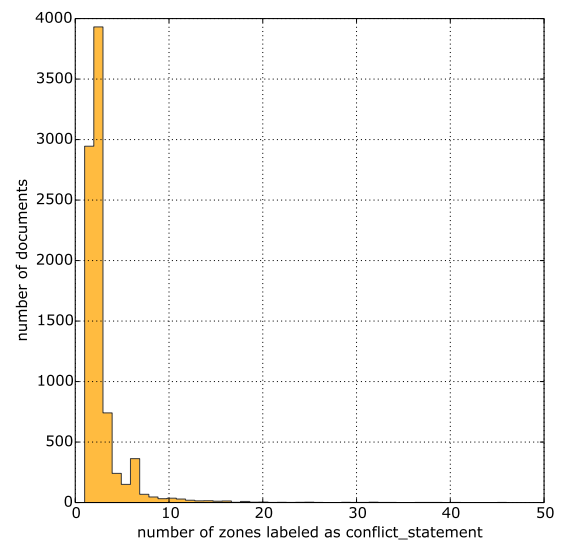


Figure E.8: Histogram of number of zones labeled as `conflict_statement`

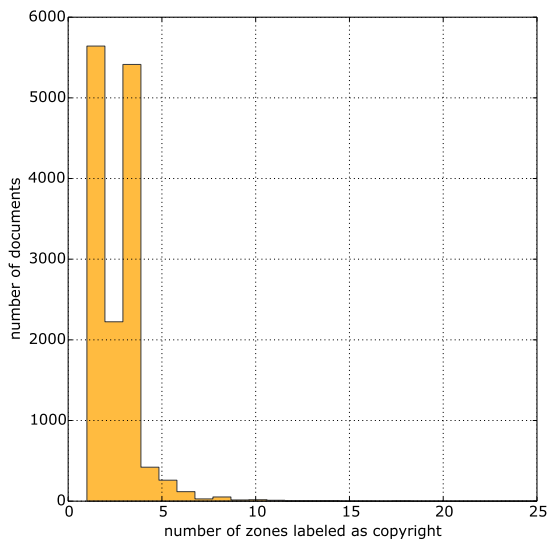


Figure E.9: Histogram of number of zones labeled as **copyright**

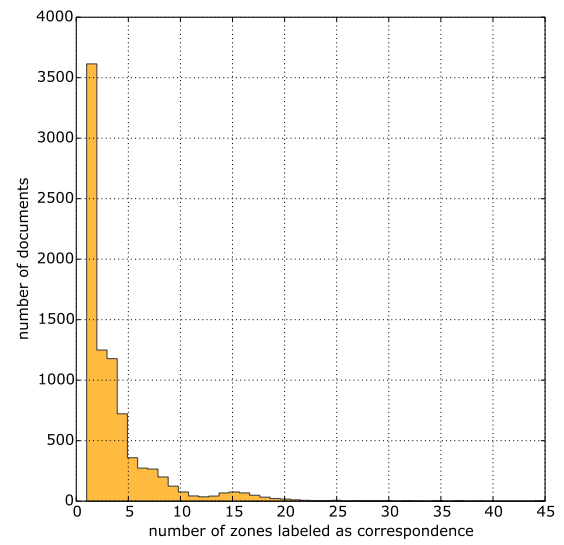


Figure E.10: Histogram of number of zones labeled as **correspondence**

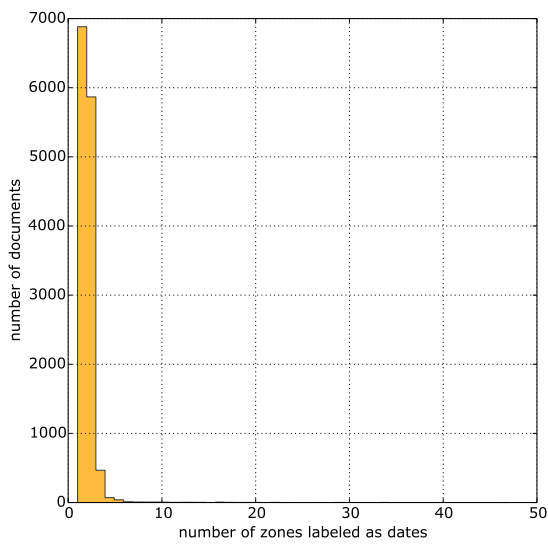


Figure E.11: Histogram of number of zones labeled as **dates**

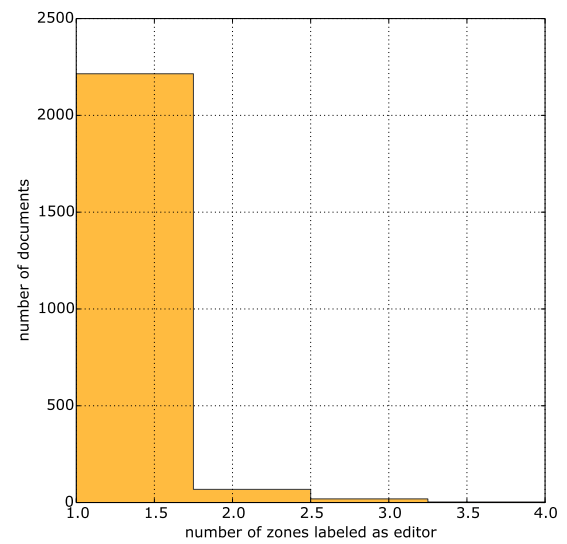


Figure E.12: Histogram of number of zones labeled as **editor**

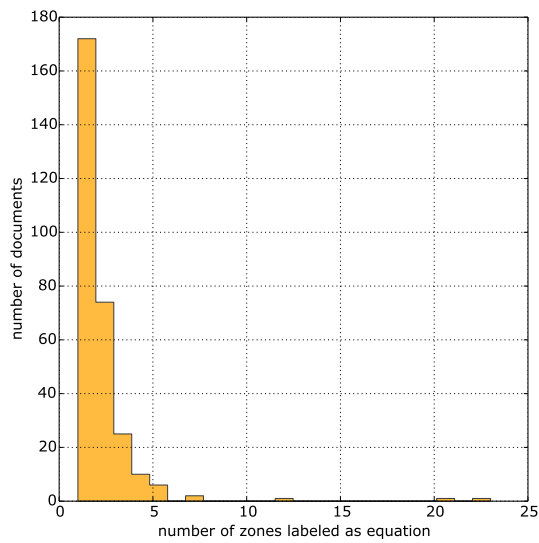


Figure E.13: Histogram of number of zones labeled as `equation`

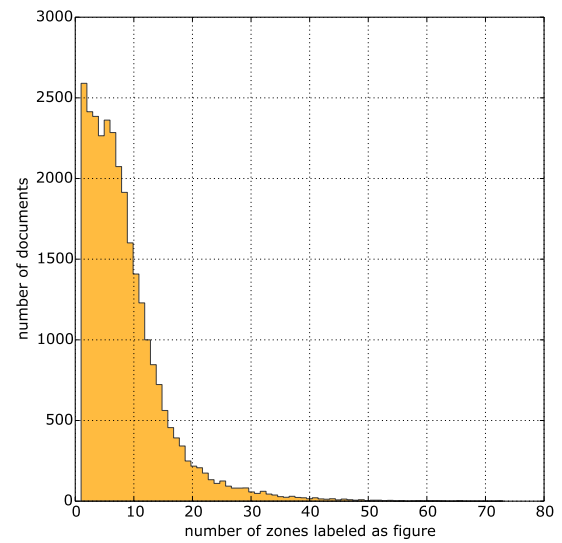


Figure E.14: Histogram of number of zones labeled as `figure`

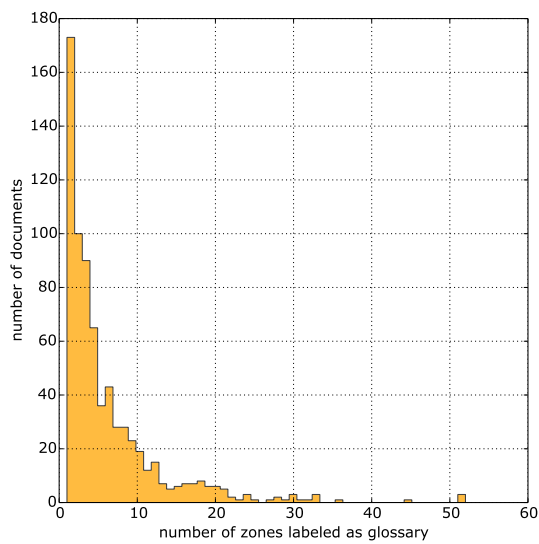


Figure E.15: Histogram of number of zones labeled as `glossary`

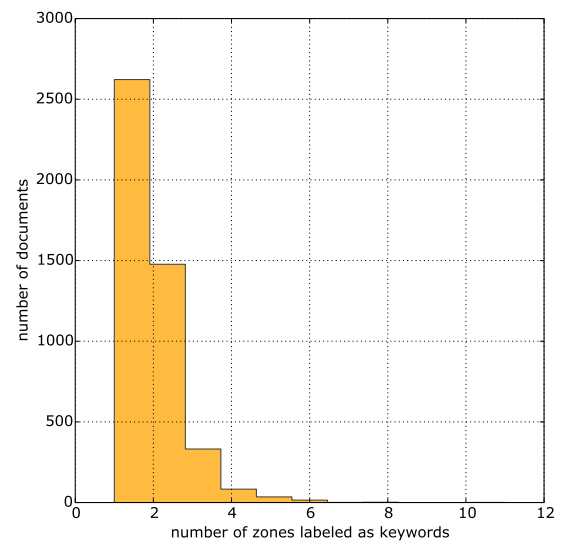


Figure E.16: Histogram of number of zones labeled as `keywords`

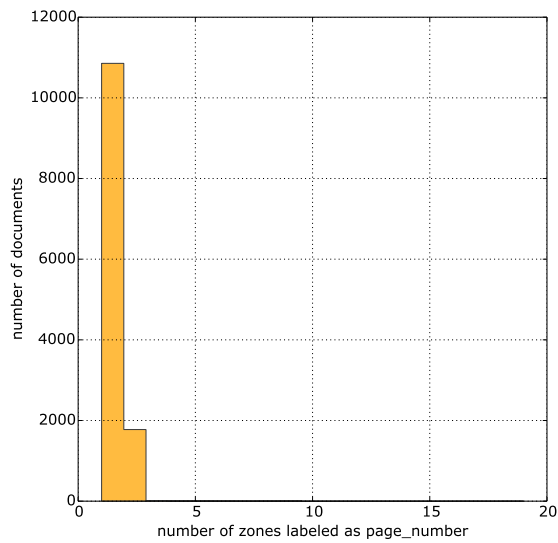


Figure E.17: Histogram of number of zones labeled as `page_number`

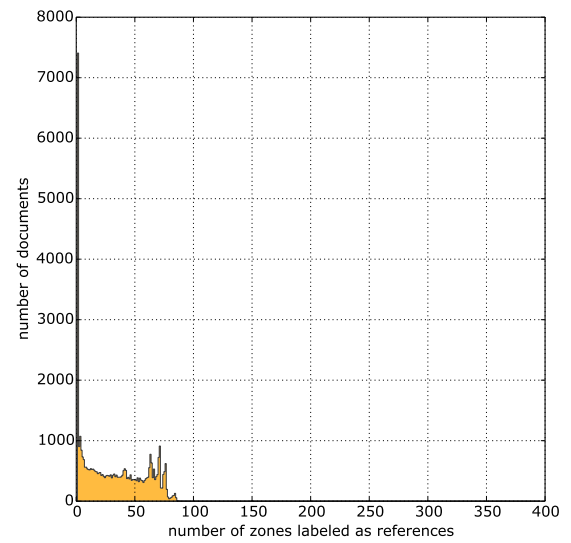


Figure E.18: Histogram of number of zones labeled as `reference`

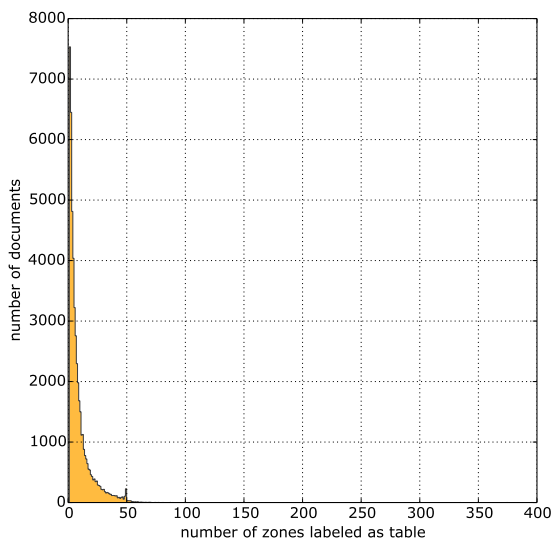


Figure E.19: Histogram of number of zones labeled as `table`

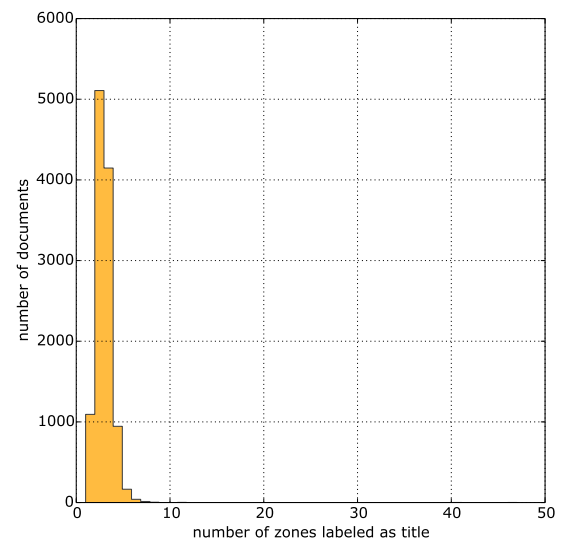


Figure E.20: Histogram of number of zones labeled as `title`

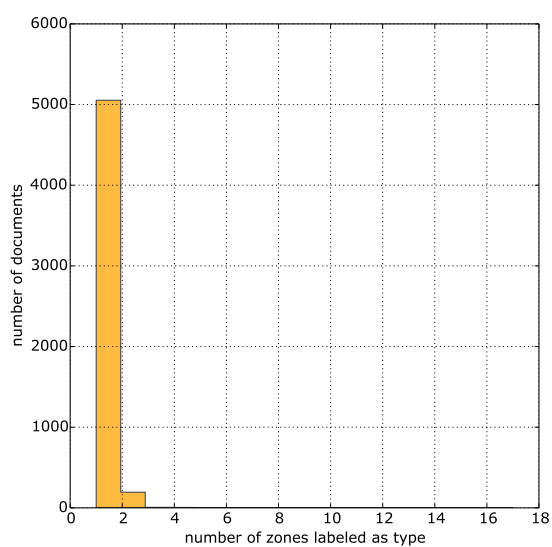


Figure E.21: Histogram of number of zones labeled as `type`

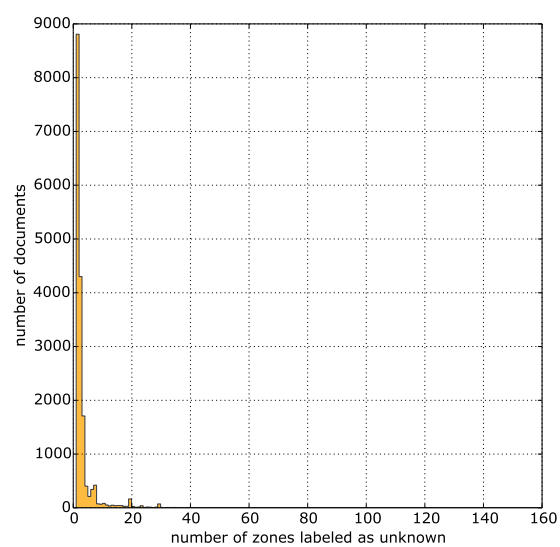


Figure E.22: Histogram of number of zones labeled as `unknown`

Bibliography

- [ABPP09] A. Antonacopoulos, D. Bridson, C. Papadopoulos, and S. Pletschacher. A Realistic Dataset for Performance Evaluation of Document Layout Analysis. *2009 10th International Conference on Document Analysis and Recognition*, 2009.
- [AD09] M. Agrawal and D. Doermann. Voronoi++: A dynamic page segmentation approach based on voronoi and docstrum features. pages 1011–1015, July 2009.
- [AR00] K. Nigam A. McCallum and J. Rennie. Automating the construction of internet portals with machine learning. *Information Retrieval*, pages 127–163, 2000.
- [ASC09] I. Guyon A. Statnikov, D. Hardin and C. Aliferis. A Gentle Introduction to Support Vector Machines in Biomedicine. In *AMIA Annual Symposium Proceedings*, 2009.
- [BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.
- [C.C95] V. Vapnik C.Cortes. Support Vector Networks. *Machine Learning*, 20:273–297, 1995.
- [CeO] Grotoap2 at ceon. <http://cermine.ceon.pl/>.
- [Cho10] Jong Myong Choi. *A Selective Sampling Method for Imbalanced Data Learning on Support Vector Machines*. PhD thesis, 2010.
- [CL10] Chih-Chung Chang Chih-Wei Hsu and Chih-Jen Lin. A Practical Guide to Support Vector Classification. Technical report, 2010.

- [CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 2011.
- [CX10] B. G. Cui Chen. and X. An Improved Hidden Markov Model for Literature Metadata Extraction. *Advanced Intelligent Computing Theories and Applications*, 6215:205–212, 2010.
- [DB14] Pawel Szostek Dominika Tkaczyk and Lukasz Bolikowski. GROTOAP2 - The methodology for creating a large ground truth dataset of scientific articles. 2014.
- [DCR⁺12] Dominika Tkaczyk, Artur Czekczo, Krzysztof Rusek, Lukasz Bolikowski, and Roman Bogacewicz. GROTOAP: ground truth for open access publications. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 381–382, 2012.
- [Dom14] Lukasz Bolikowski Dominika Tkaczyk, Paweł Szostek, Mateusz Fedoryszak, Piotr Jan Dendek. CERMINE - automatic extraction of metadata and references from scientific literature. In *11th IAPR International Workshop on Document Analysis Systems*, pages 11–16, 2014.
- [EFB08] Floriana Esposito, Stefano Ferilli, and Teresa M A Basile. Machine Learning for Digital Document Processing : From Layout Analysis To Metadata Extraction. *World Wide Web Internet And Web Information Systems*, 2008.
- [For15] Force2015: The future of Research Communication and e-Scholarship. <https://www.force11.org/meetings/force2015/demos-and-posters>, 2015.
- [FZM⁺07] P. Flynn, L. Zhou, K. Maly, S. Zeil, and M. Zubair. Automated template-based metadata extraction architecture. 2007.
- [GBL98] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. CiteSeer: An Automatic Citation Indexing System. In *ACM Conference on Digital Libraries*, pages 89–98, 1998.
- [GSY00] Giovanni Giuffrida, Eddie C. Shek, and Jihoon Yang. Knowledge-based metadata extraction from PostScript files. *International Conference on Digital Libraries*, page 77, 2000.
- [HBAT07] P. Heroux, E. Barbu, S. Adam, and E. Trupin. Automatic Ground-truth Generation for Document Image Analysis and Understanding.

- Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 1, 2007.
- [HGM⁺03] Hui Han, C.L. Giles, E. Manavoglu, Hongyuan Zha, Zhenyue Zhang, and E.A. Fox. Automatic document metadata extraction using support vector machines. pages 37–48, May 2003.
- [HK03] Chang Ha Lee and Tapas Kanungo. The architecture of TrueViz: A groundTRUth/metadata editing and VISualizing ToolKit. *Pattern Recognition*, 36:811–825, 2003.
- [HMZ⁺05] H. Han, E. Manavoglu, H. Zha, K. Tsioutsouliklis, C. L. Giles, and X. Zhang. Rule-based word clustering for document metadata extraction. 2005.
- [Ite] iText. <http://itextpdf.com/>.
- [J. 05] J. E. Hirsch. An index to quantify an individual’s scientific research output. *Proceedings of National Academy of Science of USA*, 2005.
- [KJHG12] Roman Kern, Kris Jack, Maya Hristakeva, and Michael Granitzer. Teambeam - meta-data extraction from scientific literature. 2012.
- [Mar] MARG. <http://marg.nlm.nih.gov>.
- [Mar09] S. Marinai. Metadata Extraction from PDF Papers for Digital Library Ingest. *2009 10th International Conference on Document Analysis and Recognition*, 2009.
- [O’G93] L. O’Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993.
- [Puba] Pubmed. <ftp://ftp.ncbi.nlm.nih.gov/pub/pmc>.
- [Pubb] Pubmed XML description. http://www.nlm.nih.gov/bsd/licensee/elements_descriptions.html.
- [PZ07] Steven Zeil Paul Flynn, Li Zhou, Kurt Maly and Mohammad Zubair. Automated Template-Based Metadata Extraction Architecture. *ICADL*, 4822:327–336, 2007.
- [RBH⁺05] M. Rigamonti, J.L. Bloechle, K. Hadjar, D. Lalanne, and R. Ingold. Towards a canonical and structured representation of PDF documents through reverse engineering. *Eighth International Conference on Document Analysis and Recognition (ICDAR’05)*, 2005.

- [SHK⁺97] J. Sauvola, S. Haapakoski, H. Kauniskangas, T. Seppanen, M. Pietikainen, and D. Doermann. A distributed management system for testing document image analysis algorithms. *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, 2, 1997.
- [SMR99] Kristie Seymore, Andrew McCallum, and Ronald Rosenfeld. Learning hidden markov model structure for information extraction. In *In AAAI 99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.
- [Tho01] G. Thoma. Automating the production of bibliographic records for medline. 2001.
- [Uva] UvA. <http://www.science.uva.nl/UvA-CDD/>.
- [Uw-] UW-III. <http://www.science.uva.nl/research/dlia/datasets/uwash3.html>.
- [VV99] Olivier Chapelle Vapnik and Vladimir. Model selection for support vector machines. In *NIPS*, pages 230–236, 1999.
- [Yad] The virtual library of sicence. <http://http://yadda.icm.edu.pl/>.