Information-theoretic anomaly detection and authorship attribution in literature

Master Thesis

By

José Vargas Quiros

Department of Information and Computing Sciences
Utrecht University

Supervisors:
prof. dr. A.P.J.M. Siebes
dr. A.J. Feelders

December 2017

Abstract

KRIMP is an algorithm based on information theory capable of capturing arbitrary length co-occurrence relations between itemsets in a database. KRIMP code tables have shown to be useful models of databases for multiple Machine Learning tasks including classification and clustering. Cross-compression sizes obtained from KRIMP code tables are a generalization of cross-entropy capable of taking into account such co-occurrence relations. This work focuses on the application of KRIMP cross-compression in two different but overlapping areas. First, an unsupervised anomalous database detection algorithm is presented, capable of taking into account database structure. The algorithm is tested on itemset databases with a significant amount of structure to characterize its behavior and an experiment is performed on text data. Second, the application of KRIMP to natural language is investigated further, in the context of authorship attribution. The KRIMP classifier is a generalization of the Naive Bayes classifier capable of combining frequently co-occurring words or items into itemsets, delivering an appealing comparison between the two algorithms. Different ways of transforming text into itemset form were explored, as well as two different ways of applying Laplace smoothing. Experiments indicated that co-occurrence relations are important for attribution, with KRIMP being more robust and in most cases more accurate on prose text when the complete alphabet is considered and itemsets are created per sentence. The behavior suggests that there is relevant syntactic structure at the sentence level involving punctuation and function words that is captured in KRIMP code tables, and is characteristic of individual authors. The decrease in performance when splitting sentences at punctuation characters and considering only function words is an indication that language grammar is too restrictive at that level, causing stylistic choices to be obscured rather than exposed.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

An important problem in machine learning is how to capture, in a model, the most important dependency relations between variables or their values. It is an important problem because most real databases exhibit such relations, and in many cases accounting for them leads to increased performance of Machine Learning algorithms, with respect to simplifying assumptions. It is an interesting problem, as solutions are not straightforward: attempting to take into account all possible partitionings of a set results in intractable algorithms. In fact, as this problem appears whenever modeling multivariate data, an immense number of solutions have been presented in data mining and machine learning literature. In some cases, it is acceptable to completely ignore dependency relations of order greater than one, or up to a specific order. More challenging is the problem of capturing the most important relations regardless of the number of variables involved.

A similar but not equivalent problem appears in the itemset database domain, where it is desired to capture a model of a database that incorporates the most important relations between itemsets, or values of variables. KRIMP is an algorithm with grounds on information theory that addresses this problem by capturing a reduced set of important co-ocurrence relations between items. A KRIMP model is interpretable, and naturally defines quantities that share analogies with entropy, cross-entropy and the Kullback-Leibler divergence, very well-known quantities applicable specially to single-variate distributions.

One kind of readily available data with a degree of structure is natural language. The main focus of this work is the application of these KRIMP-induced quantities to written text, first with an anomaly detection algorithm capable of differentiating databases based on their structure, and second, in the authorship attribution of literature.

The rest of this document is structured as follows:

- Chapter 2 is a brief explanation of the KRIMP algorithm, a compression algorithm for itemset databases with a theoretical basis on information theory, specifically on the Minimum Description Length principle. The induced dissimilarity measure for databases analyzed and used in chapters 3 and 4 is presented, as well as some applications of the KRIMP algorithm, including classification. References to the relevant papers are given for each of these applications.
- Chapter 3 presents a KRIMP-based algorithm to perform detection of anomalous databases. The algorithm performs unsupervised detection of anomalies on a set of databases, based on several assumptions, an important one being that most of the databases have a structure that can be considered normal. The rationale for the use of KRIMP cross-compression sizes is presented, as well as a characterization of the algorithm's behavior on synthetic and real data.
- Chapter 4 studies the use of KRIMP to capture structure present in natural language, in particular for authorship attribution: finding the most likely author of a disputed text. It presents the results of the first application of KRIMP to literature, in attribution of both Victorian novels and Dutch poems. The algorithm is compared with Naive Bayes classification due to its similar theoretical grounds, which facilitate interpretation. Conclusions are drawn regarding the applicability of KRIMP on text data.

Chapter 2

Preliminaries

This chapter introduces the KRIMP algorithm and the different dissimilarity measures for itemset databases that can be defined based on it. These constitute the basis of the algorithms proposed, and the experiments performed in subsequent chapters.

2.1 Definitions

Some definitions used in itemset databases and pattern set mining will now be introduced. The task of finding frequent itemsets (a special case of theory mining), of a database $\mathcal{D}$ has been defined in [41] as: given an alphabet $\mathcal{I}$ and a database $\mathcal{D}$ over $\mathcal{I}$ (that is, the transactions in $\mathcal{D}$ are subsets of $\mathcal{I}$), and a pattern language $\mathcal{L}$ which consists of all subsets of $\mathcal{I}$, find all the elements of $\mathcal{L}$ that meet a minimum support:

$$\{X \in \mathcal{L} | supp_{\mathcal{D}}(X) \geq minsup\}$$

where the support of an itemset X is the number of transactions of $\mathcal{D}$ containing X:

$$|\{X \in \mathcal{D} | X \subseteq t\}|$$

2.2 Kolmogorov Complexity

The Kolmogorov complexity of a string x is defined as the length of the shortest computer program that can produce x as output, and is denoted as $K(x)$. Program lengths are of course dependent on the programming language that is used to code the program. However, it can be proven that differences are up to a fixed additive constant [39]. Hence $K(x)$ is the minimum amount of information necessary to generate x.

Conditional Kolmogorov complexity, denoted as $K(x|y)$ for strings x and y is the length of the shortest program that generates x as output, while receiving y as input.

Kolmogorov complexity has been shown to be uncomputable. A corollary is that compression lengths attained by running universal compression algorithms on x are upper bounds to $K(x)$ [18].

2.3 The Minimum Description Length Principle

The Minimum Description Length (MDL) principle, states that given a set of models $\mathcal{H}$ capable of encoding a dataset $\mathcal{D}$, the best model $H \in \mathcal{H}$ is the one that minimizes:

$$L(H) + L(\mathcal{D}|H)$$

where:

- $L(H)$ is the length in bits of the description of H.
- $L(\mathcal{D}|H)$ is the length of the description of the data when encoded using H.

MDL aims to obtain an approximation to the Kolgomorov complexity of the data: the length of the shortest program which produces $\mathcal{D}$ as output [8]. MDL adjusts to the definition in the sense that the same program can be used to generate any database given its compressed representation and its code table, ie. its length is a fixed constant. In doing so, the formulation seeks a balance between the complexity of the model and the length of the data, a natural safeguard against the problem of over-fitting. This allows MDL-based models to possibly be parameter-free.

2.4 The KRIMP algorithm

The KRIMP algorithm is the product of applying the MDL principle to itemset databases. The original formulation of MDL given above, known as crude MDL [8], was applied to itemset databases by Vreeken et al. [41], with the objective of finding a set of interesting non-redundant itemset that compresses the database well.

2.4.1 Basic KRIMP

The proposed approach operates on an itemset database $\mathcal{D}$ over an alphabet $\mathcal{I}$. A hypothesis $H \in \mathcal{H}$ is defined as a code table: a mapping of itemsets to codes. Concretely, a code table CT is a two-column table where the first column contains an itemset $X \in \mathcal{I}$ and the second column contains a unique code from a codeword set $\mathcal{C}$. In KRIMP, since data is not begin transmitted, the specific code used is irrelevant. Only the length of the code in bits is considered. A KRIMP code table contains at least the set of singleton itemsets.

A code table is used to cover the database transactions such that no items are left uncovered and no overlaps are allowed. Formally, a cover function is a function $\mathit{cover} : \mathcal{CT} \times \mathcal{P}(\mathcal{I}) \Rightarrow \mathcal{P}(\mathcal{P}(\mathcal{I}))$ such that:

- If $X \in \mathit{cover}(CT, t)$ then $X \in CT$, that is, the cover function covers a transaction using itemsets in CT.
- If $X, Y \in \mathit{cover}(CT, t)$ then either $X = Y$ or $X \cap Y = \emptyset$ (itemsets covering t cannot overlap).
- $\bigcup_{X \in \mathit{cover}(CT,t)} X = t$ (t is completely covered).

The code lengths, in general, will be chosen such that more common itemsets correspond to shorter codes, since this leads to the best compression. The usage count of an itemset is the number of times it is used to cover a transaction:

$$\mathit{usage}_{\mathcal{D}}(X) = |\{X \in \mathcal{D} | X \in \mathit{cover}(CT, t)\}|$$

which implies a probability distribution given by:

$$P(X|\mathcal{D}) = \frac{\mathit{usage}_{\mathcal{D}}(X)}{\sum_{Y \in CT} \mathit{usage}_{\mathcal{D}}(Y)}$$

The code lengths are chosen optimally in accordance to this probability distribution:

$$L(\mathit{code}_{CT}(X)) = |\mathit{code}_{CT}(X)| = -\log(P(X|\mathcal{D}))$$

The compressed size of an itemset, given a cover function *cover* and a code table CT is then obtained by adding the code lengths of the itemsets $X \in \mathit{cover}(CT, t)$:

$$L(t|CT) = \sum_{X \in \mathit{cover}(CT,t)} L(\mathit{code}_{CT}(X))$$

where, as said before, the covering function does not allow overlaps. The compressed size of the database is the sum of the compressed sizes of its itemsets:

$$L(\mathcal{D}|CT) = \sum_{t \in \mathcal{D}} L(t|CT)$$

In accordance to crude MDL, the best code table is the one that minimizes the sum of the lengths of data and model:

$$L(\mathcal{D}, CT) = L(\mathcal{D}|CT) + L(CT|\mathcal{D})$$

The length of a code table $L(CT|\mathcal{D})$ must then be defined. This is pointed out as being a source of ambiguity in the crude MDL approach. In KRIMP, this length is defined with respect to $\mathcal{D}$ itself. Specifically, the length of the first column (description of $X \in CT$) is the length of the items in X encoded using a code table containing only the singleton itemsets, known as the Standard Code Table, which simply associates items with a code length proportional to the inverse of their support. The second column of CT contains codes, or code lengths, and then its encoded length is known. This results in the expression:

$$L(CT|\mathcal{D}) = \sum_{X \in CT: usage_{\mathcal{D}}(X) \neq 0} L(code_{ST}(X)) + L(code_{CT}(X))$$

KRIMP makes use of a set of candidate itemsets $\mathcal{F}$, which may be used in the code table, and are obtained using, for example, frequent set mining algorithms like Apriori.

The Minimal Coding Set Problem that KRIMP solves [41] is then defined as: given a database $\mathcal{D}$ over $\mathcal{I}$, and a candidate set $\mathcal{F}$ of itemsets, find the coding set $CS \in \mathcal{F}$ with no unused non-singleton itemsets such that for the corresponding code table CT the total compressed size $L(\mathcal{D}, CT)$ is minimal.

The KRIMP algorithm is a heuristic that provides an approximate solution to this problem, and works as follows:

1. Start with the standard code table containing only single items.
2. Add candidate itemsets from $\mathcal{F}$ to the code table, one by one. Keep the ones that lead to a better compression of the database. Otherwise discard the set.
3. Repeat until all candidates have been considered.

Two parts of the algorithm remain to be defined: the ordering of the candidate itemsets $\mathcal{F}$ and the covering function. For the ordering of $\mathcal{F}$, KRIMP considers itemsets first in order of frequency, then in order of length, and then lexicographically. As covering function KRIMP defines a heuristic to cover a transaction t: traverse the code table CT in a fixed order, and include an itemset in the cover of t if it is contained in t. Then remove the covered itemsets and proceed until t is covered. Since CT contains the singleton itemsets initially, t will be fully covered. This means that the ordering of the code table and the order of the candidates is relevant. KRIMP choses to order the code table itemsets first by decreasing cardinality, then by decreasing support, and then lexicographically. The candidates are considered such that itemsets with probably shorter codes have preference: first by decreasing support, then by cardinality, and finally lexicographically. Then, as the algorithm progresses shorter itemsets are replaced by longer itemsets in the code table. This leaves itemsets in the code table which may have very low or even zero usage counts. This itemsets may harm the compression, and are then removed by applying post-acceptance pruning: once a candidate is accepted, the algorithm considers for removal all code table itemsets whose usage count has been reduced by the new element. If removing the itemset leads to better compression, it is permanently eliminated from the code table.

2.4.2 Complexity

The worst case time complexity of KRIMP is shown in [41] to be:

$$O(|\mathcal{F}| \log |\mathcal{F}| + |\mathcal{F}|^2 \times (|\mathcal{D}| \times |\mathcal{F}| \times |\mathcal{I}| \times |\mathcal{F}|))$$

However, considering that the code tables obtained by KRIMP are in practice very small, a more accurate complexity is:

$$O(|\mathcal{F}| \log |\mathcal{F}| + |\mathcal{D}| \times |\mathcal{F}| \times |\mathcal{I}|)$$

The $|\mathcal{I}|$ term is also very loose, since itemsets are normally much shorter than $|\mathcal{I}|$.

2.4.3 KRIMP code tables

The models obtained by the KRIMP algorithm (code tables) have important advantages in quantifying information thanks to the foundation on information theory of the algorithm. The compression size resulting from KRIMP is a quantity in bits that approximates the Kolmogorov complexity of the data.

An important property of the algorithm is that running the KRIMP algorithm with an empty candidate set $\mathcal{F}$ results in a compression size for the database (excluding the code table size) equal to the entropy of the data. This is because the code table will compress the database using only alphabet elements–itemset division becomes irrelevant. Code lengths are assigned using optimal coding, just like in an entropy calculation.

When frequent candidate itemsets are added, the database will be compressed to a size $L(\mathcal{D}|CT)$ normally lower than the entropy of $\mathcal{D}$, thanks to its ability to combine alphabet items into itemsets. These two compressed sizes are comparable as both are bit quantities. The ratio between the two gives important information about how compressible a database is. The lower the ratio, the greater the information content being compressed. The ratio between compressed size by KRIMP and entropy will be termed CR_H to differentiate from the compression ratio used in the KRIMP papers Vreeken et al. [41] explained next.

$$CR_H(\mathcal{D}) = \frac{L(\mathcal{D}|CT)}{H(\mathcal{D})}$$

Although this is an attractive measure due to its relation with entropy, the compression ratio used in the KRIMP paper Vreeken et al. [41] takes into account the size of code table, true to the MDL theory behind KRIMP. This alternative compression ratio is defined as:

$$CR(\mathcal{D}) = \frac{CT(\mathcal{D}, CT)}{CT_{STD}(\mathcal{D}, CT)}$$

where $CT(\mathcal{D}, CT)$ is the total compressed size of database and code table. In practice, for large databases and relatively small alphabets both compression ratios are expected to be similar.

2.5 A dissimilarity measure for databases

The Minimum Coding Set problem, and KRIMP as an approximate solution, naturally define a dissimilarity measure for databases [40]. The central idea is that an optimal code table CT_X for database $\mathcal{D}_X$ is able to compress $\mathcal{D}_X$ better than any other code table CT_Y learned from a different database $\mathcal{D}_Y$. In general, the more different $\mathcal{D}_X$ and $\mathcal{D}_Y$ the greater the value of:

$$D_{MDL}(\mathcal{D}_X, \mathcal{D}_Y) = CT_Y(\mathcal{D}_X) - CT_X(\mathcal{D}_X) \tag{2.1}$$

Note that this measure is not symmetric, meaning $D_{MDL}(X, Y) \neq D_{MDL}(Y, X)$ in general. The MDL dissimilarity as formulated above is expect to be non-negative, however, it is not provably so since the code tables are obtained by minimizing the combined length of compressed database and code table.

It is however simple to formulate a version of the dissimilarity measure that is provably non-negative:

$$D_{MDL,NN}(\mathcal{D}_X, \mathcal{D}_Y) = CT_Y(\mathcal{D}_X) + L_X(CT_Y) - CT_X(\mathcal{D}_X) - L_X(CT_X)$$

where $L_X(CT_Y)$ is the length of the code table learned from $\mathcal{D}_Y$ but calculated with respect to the standard code table of X. The proof follows by contradiction from the fact that $CT_X(\mathcal{D}_X) + L_X(CT_X)$ is minimal for $\mathcal{D}_X$. If $CT_Y(\mathcal{D}_X) + L_X(CT_Y) < CT_X(\mathcal{D}_X) + L_X(CT_X)$ then code table CT_Y is a better choice and CT_X would not be optimal. This is assuming that the candidates sets used are the same (eg. all possible subsets of $\mathcal{I}$ (ie. $\mathcal{F} = \mathcal{P}(\mathcal{I})$). This is normally not the case in practice. However, the analysis is being made for ideal MDL, for which KRIMP is only a heuristic.

None of the measures satisfy the identity property $D_{MDL}(\mathcal{D}_X, \mathcal{D}_Y) = 0 \Leftrightarrow \mathcal{D}_X = \mathcal{D}_Y$, which is a property both of distance measures and divergences like the KL-divergence. Although any code table over the same coding set for $\mathcal{D}_X$ is worse than CT_X, it is the case that multiple databases might generate the same code table, meaning that $D_{MDL}(\mathcal{D}_X, \mathcal{D}_Y) = 0$ does not imply $\mathcal{D}_X = \mathcal{D}_Y$. However, it can be argued that the set of databases equivalent to $\mathcal{D}_X$:

$$\{\mathcal{D}|CT_{\mathcal{D}} = CT_X\}$$

is restricted to databases in a neighborhood of $\mathcal{D}_X$, specifically databases which have the same $P(X|(D))$ distribution when compressed, databases which can be covered optimally using exactly the same itemsets and the same number of them.

Hence, the ideal MDL dissimilarity operates very much like a divergence for multi-variate distributions (restricted to the discrete distributions that can be represented by a database), which takes into account the most important dependencies between itemsets. In fact, it can be seen that the empirical Kullback-Leibler divergence is a special case for the first formulation of the MDL dissimilarity (ignoring code table length), for databases of single items. The measure must be normalized by dividing by the length of $\mathcal{D}_1$ to exactly match the empirical KL-divergence.

A dissimilarity measure for databases which combines the two ways of the MDL dissimilarity has been presented in [40] as:

$$DS(\mathcal{D}_X, \mathcal{D}_Y) = \max\left\{\frac{CT_Y(\mathcal{D}_Y) - CT_X(\mathcal{D}_X)}{CT_X(\mathcal{D}_X)}, \frac{CT_X(\mathcal{D}_Y) - CT_Y(\mathcal{D}_Y)}{CT_Y(\mathcal{D}_Y)}\right\}$$

This measure has the symmetry property and normalizes for the size of the database, which may also be done by dividing by the length of the databases, to obtain an average difference in the compressed size of a transaction, in bits.

2.6 KRIMP in practice

KRIMP compression and cross-compression have been used in a variety of Machine Learning applications:

- In classification [37], where a compressor (code table) is learned for each class in the training set, and a test instance t is classified to the class of the compressor which compresses it best. The results show a performance at least as good as many classifiers, despite KRIMP not being designed for classification.

- In clustering [38], where two approaches are followed. In the first one (model-driven), for a given number of clusters k, the algorithm starts with k copies of the best code table of the entire database. Next, each element in each code table is eliminated, and the database compressed, using the k code tables, where each transaction is compressed using the code table which compresses it best. The code tables are pruned in this way until no elimination of an element leads to reducing the total compressed size. This induces a clustering by assigning elements to the code table which compresses them best.

  In data-driven clustering, the database is initially randomly partitioned, and a code table obtained for each partition. Then iteratively, every database element is re-assigned to the partition whose code-table compresses it best. This is iterated until the assignment converges, delivering a clustering of the data.

Both approaches are shown in [38] to be viable clustering approaches, able to separate classes in itemset databases by using differences in structure.

- As a similarity measure for databases [40]. Databases were learned for the different classes of a database. The results show that given the elements of a class from a test itemset databases, code tables obtained from other classes are worse at compressing those elements than its own class code table. It is also shown that the magnitude of the dissimilarity measure has correspondence with the classification error between two classes, as shown on a confusion matrix. Furthermore, it is shown that the resulting cross-compression code tables and covers give a way to interpret the dissimilarity between two databases, unlike other approaches based on compression which make use of general compressors.

Chapter 3

Anomalous Database Detection

3.1 Introduction

An important problem in data mining is that of identifying exceptional distributions, those which deviate substantially from other known distributions over the same set, or a similar set of variables. This can also be seen as the problem of identifying outlying distributions. Consider, for example, the problem of identifying businesses that place anomalous orders to their suppliers, from a set of businesses with similar customers. It is expected that businesses with similar customers will place similar orders, and hence the problem can be formulated as that of finding outliers among the distributions/databases representing their orders. In some settings such anomalies might indicate, for example, that fraud is being committed in a systematic way. In the natural language domain one might be interested in finding authors, editorials, or news outlets that treat a topic in a way that is out of the ordinary, or possibly biased. More simple applications simply seek to find mistakes in a corpus, documents that do not belong for a reason that is unknown beforehand. In the networking setting, nodes with very particular traffic characteristics over a long period of time can be identified based on databases of recorded packets. These may signal fraudulent activity which requires further attention by security software or the network administrator. In all these cases there is a notion of anomaly that cannot be defined precisely, but refers to anything that do not fits the norm.

In this chapter, the problem of finding anomalous databases is addressed by making use of the KRIMP algorithm. The objective is to device an algorithm that learns the most important structure present in the data, and signals out databases that do not conform to this notion of normality. The use of the KRIMP algorithm means that the algorithm is specially aimed at databases with a significant amount of structure, and is able to detect differences in arbitrary-order co-occurrence relations between itemsets. Furthermore, KRIMP code tables ensure that results are relatively easy to interpret and have a basis on information theory. The chapter begins with a definition of the problem and a survey of related work, and proceeds to the proposal and justification of an algorithm for the task. The algorithm is tested making use of standard itemset databases and a more interpretable experiment is performed on text data.

3.2 The Problem

According to Chandola et al. [4], anomalies are "patterns in data that do not conform to a well defined notion of normal behavior". In other words, defining anomalous behavior implies a definition of normal behavior, and vice-versa. Since in most cases we only have a dataset at our disposal and do not know hard truths about what constitutes an anomaly, the first assumption that is normally made is also a straightforward one: most of the transactions in the dataset constitute normal behavior, and the dataset exhibits structure. The user is then most probably interested in subsets of the data which do not exhibit this structure. There are different ways of capturing the structure present in the data. For example, the dataset or database may be considered as a whole or clustered and anomalies defined with respect to those clusters.

The Anomalous Database Detection problem will be defined here as: given a set of n itemset databases $\mathcal{S} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$, find an anomaly score s_i for each database $\mathcal{D}_i$ which captures the likelihood of such database not being generated by the data-generating distribution that generated most of the data in $\mathcal{S}$.

By solving the previous problem we clearly also solve the problem of returning the k top anomalous databases in $\mathcal{S}$. Since the definition of what constitutes an anomaly is ultimately subjective: data has a potentially infinite number of characteristics which could be used to define anomalous behavior; it makes little sense to attempt to define the problem absolutely. In this work, anomalous behavior is defined in terms of the dissimilarity measure induced by a KRIMP code table [41]. The KRIMP algorithm can be used to, given a database $\mathcal{D}_1$ over domain $\mathcal{X}$, find a code table CT_1 which encodes $\mathcal{D}_1$ in a nearly-optimal way (since KRIMP is a heuristic) according to the Minimum Description Length principle. This code table can also be used to encode any other database $\mathcal{D}_2$ over the same domain X, resulting in a compressed size $CT_1(D_2)$ which is analogous to a cross-entropy between probability distributions, but captures dependencies between items. A measure analogous to a Kullback-Leibler divergence can be defined by subtracting the optimal compressed size of database $\mathcal{D}_2$, which can be seen as a measure of its entropy. This dissimilarity measure, assuming the code tables are optimal, shares several properties with the KL-divergence, such as being non-negative, being zero only if the databases are equal, and asymmetry. Both the cross-compression sizes and MDL dissimilarity measure are analyzed theoretically as ways to define anomalous behavior based on KRIMP.

3.3 Related work

This section is a review of the fields most closely related to the problem. To the best of our knowledge the problem of finding anomalous itemset databases, as opposed to that of finding anomalous data records or itemsets, has not been treated directly.

3.3.1 Subgroup discovery

In subgroup discovery every data point is identified by a set of variables and a target variable. The goal of subgroup discovery is to find subgroups of the data that have a distribution for the target variable that is very different from that of the complete data. [11]. These subgroups must be identified by a rule, which takes the form of a conjunction of attribute-value pairs or inequalities (for numerical variables), which describe the subset of the data with this potentially interesting distribution of the target value. Normally we are interested in rules which have a big enough support and are interesting according to some measure. Multiple measures and approaches to the problem have been proposed.

Apriori-SD [15], for example, mines frequent itemsets using the Apriori algorithm, and then discovers rules containing only the target variable on the right hand side. The rules are also subject to meeting a given minimal support and confidence. Then the algorithm selects the most interesting rules by a measure which favors rules with high support and which induce a big change in the probability of the target variable's value:

$$WRAcc(X \Rightarrow Y) = P(X)(P(Y|X) - P(Y))$$

The second multiplicand can be seen then as quantifying the interestingness of the distribution of Y among the subgroup. Subgroup discovery can be seen as a search for distributions that are interesting, but with respect to a single target variable.

Multiple models have been proposed for the subgroup discovery problem, most considering categorical datasets, although some handling numerical variables (both target and left hand side) as well. Most approaches can be divided into extensions of classification algorithms, extensions of frequent rule mining algorithms (like Apriori-SD) and genetic algorithms employing fuzzy rules to describe the subgroups. [11]

Exceptional Model Mining (EMM), a generalization of Subgroup Discovery is concerned with finding not a subgroup with an exceptional distribution with respect to a single target variable, but an exceptional multi-variate distribution, with respect to multiple categorical target variables. This is achieved through training of a model on the complete data and candidate subgroups and using a dissimilarity measure based on such model. The goal is to find subsets of the data, identified by a rule, which are both big enough (it is easy to find exceptional distributions from only a few samples) and whose model has maximum dissimilarity with the model trained on the complete data.

The use of Bayessian Networks as models has been presented, to capture the most important dependencies between target variables. A dissimilarity measure for Bayessian Networks is used to quantify the dissimilarity between the two models [6].

EMM is similar to the problem presented in that it must compare multi-variate models. However, there are many important differences. In EMM, as in subgroup discovery, the focus is on finding subgroups that can be described by rules, and which are as large as possible, and anomalous with respect to a fixed set of variables. In Anomalous Database Detection we are concerned with anomalies with respect to all the variables, and the databases are assumed to be known (ie. the search space is the set of databases as opposed the set of possible subgroups of the data described by a rule).

3.3.2 Data exploration

A similar problem has been treated in the context of data exploration in databases. In this case the user is browsing through a database in search of subsets of the data with interesting characteristics, but the user doesn't know exactly what he is looking for. Datamaps [34] have been proposed as a way to guide the user in the exploration. A simple datamap is a representation of a partitioning of a subset of the data, which supports zooms, projections and rollbacks. It is desirable for this partitioning to be represented by simple rules over the variables. To achieve this the subset under consideration is first clustered, and then a decision tree is trained using the clusters as labels (some information may be lost in the process). Datamaps allow users to find outliers, which will manifest as small isolated clusters. The technique operates in a similar way to some anomaly detection techniques, which assume that anomalies correspond to small clusters.

Being a data exploration technique, datamaps are focused on exposing this to the explorer, and not in ranking the interestingness of these clusters. A version of the technique for high dimensional spaces first partitions the data vertically by finding sets of columns which are mutually dependent, with respect to the variation of information:

$$VI(X,Y) = H(X) + H(Y) - 2I(X,Y)$$

A low Variation of Information indicates variables which describe roughly the same aspect of the data. To do this the diameter of a subset of variables is defined as the maximum Variation of Information between two variables in the set. Since smaller subsets (more partitions) will lead to lower diameters, a two-objective optimization problem is formulated to balance cardinality and diameter. This is solved by letting the user pick a maximum number of partitions MAX_p and finding a suitable partitioning using a hierarchical clustering algorithm, which starts with one cluster per variable and iteratively joins the two closest clusters, as measured by their diameter. Once a partitioning of the variables is found, a simple map is obtained as explained before, by clustering and learning a decision tree. This technique outputs several data maps that are comprehensive, and allows identifying interesting and outlying clusters with respect to subgroups of the variables which are probably mutually dependent.

3.3.3 Anomaly detection

Anomaly Detection is the field of data mining most closely related to the problem. In fact, Anomalous Database detection is can be seen as a generalization of Anomaly Detection to databases of any length. Anomaly Detection, in general, refers to the problem of finding patterns in data which do not conform to expected behavior. The way expected behavior is characterized varies among approaches, and in general specific formulations of the problem are solved, according to the kind of data considered (eg. categorical vs numerical), availability of labeled data and the type of anomalies detected, which is normally a consequence of the models used to describe normal/anomalous data.

Anomaly detection has many applications, in situations in which an anomaly in data requires attention. Anomalies in a transaction database (eg. credit card transactions) could, for example, correspond to fraudulent transactions. These could be anomalous, for example, because of a high amount, a very rare sender/receiver combination, the time or date of the transaction, or a rare combination of all previous factors. Furthermore, anomalies could also be defined in terms of sets of transactions. A series of multiple transactions between two particular credit card holders could constitute an anomaly, while a single transaction between them occurs normally. In surveillance data, anomalies could indicate undesirable events, such as an emergency worth of attention. Anomaly detection can also be applied to text data to detect the emergence of new topics.

Most anomaly detection techniques deal with univariate of multivariate data points, which are not related to one another. According to [4] can be classified in:

Point Anomalies . The most studied case. Anomalies are single data points that are rare with respect to the rest of the dataset.

Contextual Anomalies . A data point is anomalous only in a specific context. For example, in spatial data, it is normally desired to find anomalous points with respect to the points closest to them.

Collective Anomalies . Anomalies are sets of data points. Individual points may not be anomalous, but their occurrence together could be.

To this list we should add our anomalous database problem, which is analogous to a point anomaly, but an instance is represented by a complete database rather than a single data record.

According to the availability of labeled data, anomaly detection problems can be supervised, semi-supervised and unsupervised. Supervised problems assume labeled normal and anomalous data, which can be used to learn a classifier. Finding such labeled data, is, however, very difficult in practice. Semi-supervised instances assume only labels for the normal class, which is modeled, and anything deviating from this model of normal behavior is classified as an anomaly. We are more concerned about unsupervised detection, which operates under the critical assumption mentioned before: most data points corresponds to normal behavior, and the data exhibits structure.

3.3.3.1 Classification-based anomaly detection

Classification-based anomaly detection techniques operate under the assumption [4]:

> A classifier that can distinguish between normal and anomalous classes can be learned from the data.

In general, these techniques learn a single-class or multi-class classifier using labeled data (requirement). In the multi-class case a classifier is trained for every class. It learns to distinguish the class from the rest of the data. A data point is classified as anomaly if no classifier classifies it as normal. Multiple approaches have been presented, making use of different classifiers:

Neural Networks . A network is trained to recognize the normal classes. The single output simply indicates whether or not the record is considered normal.

Bayesian Networks . For the univariate case, a naive Bayessian network is trained on the labeled data, with the normal classes and anomalous class as nodes. An observation is assigned to the class with the largest posterior probability.

Support Vector Machines . For the one-class case: an SVM learns the region that contains the normal instances.

Rule-based Techniques . For the multi-class case, rules are learned on the training set, with the class on the right hand side of the rule. For each test example, the rule with highest confidence which captures the test example is found. The inverse of the confidence of such best rule is used as anomaly score for the point.

Unsupervised rule-based algorithms have also been proposed. One simple approach is to mine frequent itemsets from the database, and every point is scored with a value equal to the sum of the supports of the frequent itemsets it contains. Anomalous itemsets are then the ones with the lowest scores [10].

3.3.3.2 Clustering-based anomaly detection

In contrast with classification-base approaches, clustering-based techniques are primarily used in unsupervised anomaly detection. They may operate based on different assumptions:

- Normal data instances belong to a cluster, while anomalies do not. Techniques based on this assumption use clustering algorithms which do not assign every point to a cluster.
- Anomalies are far away from their closest cluster centroid. These techniques cluster the data (possibly forcing every data point into a cluster) and then rank the points by distance based on this assumption. These techniques, however, fail if the anomalous points form a cluster.
- Small or sparse clusters are anomalous. This assumption solves the problem of anomalies forming clusters by setting a threshold for the cluster size or density, below which its data points are considered anomalies.

Note that clustering-based techniques may also be applicable to the anomalous database problem, the requirement being a suitable (dis)similarity measure between a cluster of databases and a single database. Although we follow a different approach, the MDL dissimilarity measure could also be adapted for this purpose.

3.3.3.3 Statistics-based anomaly detection

Although it may not be clear how statistical anomaly detection techniques are applicable to our problem, the concepts used in statistical techniques are very general and potentially applicable to the problem of identifying anomalies based on compression sizes scores, which define probability distribution which can be captured by a continuous model.

Parametric statistical techniques, for continuous variables, assume that the normal data is generated by a distribution $f(x, \theta)$, where the anomaly score of a datum x is $1/f(x, \theta)$. In other words, the parameters θ of a probability distribution are learned from the training data, and the probability of a test instance corresponds to the value of the PDF for that instance. The Gaussian case is a very simple one, where values are distributed normally. Hypothesis tests can be used to offer guarantees. Normally, the null hypothesis H_0 is that the data instance has been generated using the estimated distribution. If the statistical test rejects H_0, the data point is an anomaly. The test statistic provides an anomaly score for the data point.

In Gaussian models, a common arbitrary rule is to declare as anomalies all points that are more than 3σ away from the mean. Also for normally distributed variables, Grubb's test is able to determine if the most extreme point in the data is an outlier. The hypotheses are, then:

H_0: there are no outliers in the dataset

H_1 is: there is at least one outlier in the dataset

If H_0 is rejected, the most extreme value is considered an outlier. Repeated application of the test to find more outliers can be biased due to masking effects, although it is sometimes used in practice. The Generalized Extreme Studentized Deviate (ESD) Test or Rosner test is the correct way to test for multiple outliers [31]. Grubb's test statistic is:

$$z = \frac{|x - \bar{x}|}{s}$$

A test instance is anomalous if:

$$z > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/(2N),N-2}}{N-2+t^2_{\alpha/(2N),N-2}}}$$

where N is the sample size, $t^2_{\alpha/(2N),N-2}$ is the value of a t-distribution at significance level $\alpha/2N$ with $N-2$ degrees of freedom.

The Generalization of the Grubb's test to multiple anomalies, also under the assumption that normal data is normally distributed, is the Rosner test or Generalized ESD [31]. Given a maximum number of anomalies k to test for, a sample of size n and a confidence α the test consists in:

1. Compute the R_i and critical values λ_i for $i = 1, 2, ..., k$:

   $$R_i = \frac{max_i|x_i - \bar{x}|}{s}$$

   where x_i is the value farthest away from the mean $\bar{x}$ and s is the sample standard deviation. The mean and standard deviation are recomputed on every iteration (for every i) after removing the most outlying element. The corresponding critical values are:

   $$\lambda_i = \frac{(n-i)t_{p,n-i-1}}{\sqrt{(n-i-1+t^2_{p,n-i-1})(n-i+1)}}$$

   where $t_{p,d}$ is the $p-$percentile of a t-distribution with d degrees of freedom, and:

   $$p = 1 - \frac{\alpha}{2(n-i+1)}$$

2. The number of outliers corresponds to the largest i such that $R_i > \lambda_i$.

Other techniques include mixtures of parametric distributions, where, provided labeled data, normal and anomalous distributions are modeled separately. An observation is classified into one of the two by applying Grubb's test and comparing the statistics and thresholds.

Statistical techniques require the parameter estimation step to be robust against anomalies in the data, in order to be useful in unsupervised detection. The underlying assumption that the data is generated from such distribution should also be as accurate as possible.

3.3.3.4 Information-theoretic anomaly detection

Of special interest for this work are anomaly detection techniques which have a basis on information theory. Techniques in this area rely on quantifying the relative complexity of the data. Several measures of complexity have been proposed, including relative entropy (Kullback Leibler divergence), and approximations to Kolmogorov Complexity using standard compression algorithms which [4].

Some of these techniques are based on the idea that anomalies greatly increase the complexity of a dataset. In other words, if those anomalies are removed the complexity of the dataset should decrease substantially. One way of formulating the problem is a two-objective optimization problem: find a subset I of the dataset D such that $C(D) - C(D \setminus I)$ is maximized and $|I|$ is minimized, where C is some measure of complexity of the dataset. Clearly both objectives compete since larger subsets will lead to larger complexity reductions. Pareto-optimal solutions can then be obtained for this front. Heuristics are necessary due to the size of the search space (all subsets of the database).

Keogh et al. [18] present an algorithm to detect anomalous sections of a database. The algorithm is based on Kolmogorov complexity (see 2.2) and consists on a divided and conquer approach: recursively divide the database, picking every time the most anomalous section, to be divided next. The anomaly of a section d of the database is measured by using the Compression-based Dissimilarity Measure (CDM), with respect to the complete database $\mathcal{D}$:

$$CDM(d, \mathcal{D}) = \frac{C(d\mathcal{D})}{C(d) + C(\mathcal{D})}$$

where C is a universal compressor and $C(d\mathcal{D})$ is the compressed size of the concatenation of d and $\mathcal{D}$. The more similar d and $\mathcal{D}$ are, the lower their CMD.

No work has been proposed which uses KRIMP in the way studied here, although it has previously been used in two papers on anomaly detection. In of them KRIMP was used to detect a special kind of anomaly: the unexpected co-occurrence of itemsets. To do it, Bertens et al. [2] propose using the number of bits needed to explain the most unlikely item co-occurrence in a transaction. A transaction t is scored as:

$$score(t) = \max_{X,Y \in \mathcal{S} | X,Y \subseteq t} -log(P(XY)) + log(P(X)P(Y))$$

where $\mathcal{S}$ is a set of patterns which can in principle be chosen. It is argued in favor of mining $\mathcal{S}$ using KRIMP, to obtain a reduced set of non-redundant itemsets. $P(X)$ is simply the relative support of item X in the data, and $P(XY)$ the support of the itemset, which is obtained in the mining step [2].

In a second paper KRIMP is used as benchmark for an anomaly detection algorithm for categorical data [1]. The CompreX algorithm is based on MDL, but compresses the database using an optimal set of code tables, rather than a single one. The algorithm seeks to exploit dependencies and independencies between categorical values by creating a vertical (column) split of the dataset, with each feature set containing variables with strong dependencies. To do it, every variable is first placed in its own feature set. All possible joins between pairs of feature sets are ranked by their information gain:

$$IG(F_i, F_j) = H(F_i) + H(F_j) - H(F_i, F_j) \geq 0$$

Joins are performed using a heuristic, and kept if they reduce the total compressed size of database and code tables (ie. according to MDL). The process is repeated until no merge reduces the compressed size.

Compression by multiple code tables is shown to outperform KRIMP in identifying single point anomalies. The algorithm also has the advantage of being faster due to its ability to separate independent sets of variables and avoid the mining of corresponding itemsets.

3.3.3.5 Spectral anomaly detection

Spectral techniques project a database into a lower dimensional subspace, with the objective of separating normal data and anomalies in this subspace. Principal Component Analysis (PCA), for example, allows identifying an orthonormal basis of the space, formed by a set of principal vectors. The first principal vector corresponds to the direction of greatest variance in the data, and the rest are orthogonal to it (and to each other), obtained in order of variance. The idea of some of these techniques is that data points can be projected into a principal vector, and analyzed with respect to the variance present in that vector. A data point that deviates from the structure present in the data will have a large projection value [4].

Further approaches to Anomaly Detection can be found which are not so relevant to the anomalous database problem. For example, Nearest Neighbor-based Techniques use the distance from an observation to its k nearest neighbors as anomaly score. Others estimate the density of the neighborhood of a data point to set its anomaly score [4].

3.3.3.6 Anomaly detection and anomalous database detection

Note that techniques for finding Point Anomalies that deliver an anomaly score based on the complete data, like many clustering and spectral-based techniques could be generalized to the anomalous database problem, by following the same procedure on the conjunction of the databases, and then scoring each database with the average anomaly score of its data points:

$$AS_{DB}(db) = \frac{\sum_{x \in db} AS_P(x)}{N}$$

where AS_P is the point anomaly score defined by the technique. However, this simple technique clearly misses relations between items: a database formed by instances of only one common itemset will be categorized as normal, while being anomalous.

One particular technique of interest, due to also being based on compression, but which does not fall naturally into any of the anomaly detection categories presented, is the Replicator Neural Network technique [9]. In a Replicator Neural Network the input variables are equal to the output variables (the dataset variables), and the network is trained to reproduce its inputs. In other words, when trained on a dataset, the neural network learns an approximation of it. It can also be said that the neural network compresses the database into its hidden units. Concretely, a Replicator Neural Network is composed of three hidden layers (layers 2,3 and 4), between the input and output layers of the same size. The activation function for the outer hidden layers (layers 2 and 4) is:

$$S_k(\theta) = tanh(a_k\theta)$$

and θ is the standard weighed sum of the outputs of the previous layer. For the middle layer (3) a staircase function is used. The activation function is divided into N levels (eg. $N = 4$ in the article), inducing a clustering of the data in the hidden layer, where for L_3 hidden units, there are N^{L_3} possible clusters. Each data point can be seen as been assigned to one of them depending on where the input/output of the middle layer falls when presented the point. The staircase function is given by:

$$S_3(\theta) = \frac{1}{2} + \frac{1}{2(k+1)}\sum_{j=1}^{N-1} tanh[a_3(\theta - \frac{j}{N})]$$

For the output layer (5), either the sum of its weighted inputs (θ) or the sigmoid function on θ are proposed as activation functions. An adaptive rate is used to learn the neural network, making use of the mean squared error of the output. The network is applied to outlier detection by scoring an observation with the Outlier Factor, the squared error, over all the variables:

$$OF_i = \frac{1}{n}\sum_{j=1}^{n}(x_{ij} - o_{ij})^2$$

where n is the number of variables, x_{ij} input variable j for observation i and o_{ij} output variable j for the same observation. In other words, OF_i is just the normalized squared L_2 distance between input and output.

3.4 Anomalous Database Detection

The MDL dissimilarity measure (see section 2.5) provides a way to summarize the difference between two databases into single real numbers: the database compression and cross-compression sizes $CT_X(\mathcal{D}_Y)$, $CT_Y(\mathcal{D}_Y)$, $CT_Y(\mathcal{D}_X)$ and $CT_X(\mathcal{D}_X)$. It will be argued that these values behave in a way analogous to the single variate entropy, and cross-entropy and both parts of the MDL dissimilarity measure behave very much like a Kullback-Leibler divergence, but taking into account arbitrary order relations between itemsets.

The proposed approach rests on the assumptions that:

- The data has some degree of structure. Since the KRIMP algorithm is built to capture the structure in itemset databases, it is necessary that normal or anomalous databases exhibit some form of structure to obtain any meaningful results.

- Most input databases are not anomalous. Hence, the joint database (joining all input databases, normal and anomalous) is a good representation of the non-anomalous distribution.

- The ordering of the itemsets within a database is irrelevant (ie. all itemsets are equivalent).

Since the joint database is assumed to be a good representation of non-anomalous data, its code table is expected to be a good succinct representation of it. Compressing normal data using this joint database should lead to consistent compression sizes, while compressing anomalous data should lead to compression sizes that are outliers with respect to the distribution of compressed sizes of the normal data. Algorithm 1 shows the general approach to anomalous database detection using KRIMP.

Input databases are first merged into a joint database, from which a joint code table is extracted. Since we assume most of the itemsets in this joint database to be non-anomalous, the joint code table is expected to capture most of the relations present in normal databases, and thus to be able to compress normal databases better than anomalous databases. The compressed size of database i, divided by its number of items (ie. the compressed size per item) is denoted $D(\mathcal{D}_i, joint)$. The compressed sizes of individual databases form the set: $\{D(\mathcal{D}_i, joint) | \forall i \in \{1, 2, ..., N\}\}$. The reciprocal is also done: code tables obtained from every individual database are used to compress the joint database to obtain the set $\{D(joint, \mathcal{D}_i) | \forall i \in \{1, 2, ..., N\}\}$. The resulting vectors define a distribution of compressed sizes, where larger sizes indicate greater dissimilarity. Outliers are detected in these distributions using rules or a statistical test for outliers. Multiple approaches may be valid here depending on the resulting probability distribution. When the number of outliers is known, simple rules like taking the top n compressed sizes from each distribution might be a better approach.

The reasons for taking the cross-compression size alone without subtracting the complexity of the database are explained in section 3.4.1.

```
Data: A set of $N$ databases $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_N\}$
Result: Bits $a_1, a_2, ..., a_N$ indicating whether database $i$ is considered anomalous.
create joint database $\mathcal{D}_{joint} = \cup_{i=1}^{N} \mathcal{D}_i$;
compress joint database: $CT_{joint} \leftarrow KRIMP(\mathcal{D}_{joint})$;
for $i \leftarrow 1$ to $N$ do
    compress $\mathcal{D}_i$: $CT_i \leftarrow KRIMP(\mathcal{D}_i)$;
    calculate cross-compression distances:
    $D(\mathcal{D}_i, joint) = CT_{joint}(\mathcal{D}_i)/nitems(\mathcal{D}_i)$;
    $D(joint, \mathcal{D}_i) = CT_i(\mathcal{D}_{joint})/nitems(\mathcal{D}_{joint})$;
end
perform a test $TS1$ for outliers in $D(\mathcal{D}_i, joint)$;
perform a test $TS2$ for outliers in $D(joint, \mathcal{D}_i)$;
for $i \leftarrow 1$ to $N$ do
    if $TS1_i == 1$ or $TS2_i == 1$ then
        $a_i$ = true
    end
    else
        $a_i$ = false
    end
end
```

Algorithm 1: The general Anomalous Database detection algorithm, which calculates cross-compression sizes between the provided individual databases and the joint database, and identifies anomalies in the distribution of compression sizes.

3.4.1 KRIMP and the KL-divergence

Consider the Kullback-Leibler divergence defined as:

$$D_{KL}(X, Y) = -\mathbb{E}_{x\sim X}[logP_Y(x)] + \mathbb{E}_{x\sim X}[logP_X(x)]$$

And the MDL dissimilarity defined as:

$$D_{MDL}(\mathcal{D}_X, \mathcal{D}_Y) = CT_Y(\mathcal{D}_X) - CT_X(\mathcal{D}_X)$$

Just as $D_{KL}(X, Y)$ captures how well X can be compressed using a code optimized for Q, so does $D_{MDL}(\mathcal{D}_X, \mathcal{D}_Y)$ capture how well $\mathcal{D}_X$ can be compressed using a code optimized to compress $\mathcal{D}_Y$, where the way items are compressed is determined by the covering function of KRIMP. The main distinction between the two is that KRIMP takes into account the compressed size of the code table when obtaining the best code table, which means that even assuming optimal compression, the previous measure cannot be guaranteed to be positive, since a larger code table (in bits) could be able to compress the database better. However, this is rarely the case when compressing real data across classes as shown by Vreeken et al. [40]. Even when compressing individual transactions, compressed sizes attained by code tables from different classes are consistently worse.

A real distribution P that concentrates most of its mass in the most dense areas of Q, but with much lower variance may have a low $D_{KL}(P, Q)$ (although always greater than zero), while $D_{KL}(Q, P)$ would be much larger due to the effect of the logarithm. In a discrete distribution this can be seen as most of Q's elements being compressed by very large codes, since they correspond to low probability in P. The same can be said about the MDL dissimilarity. Both directions of compression reveal different ways in which the databases can be considered dissimilar.

There are various reasons for detecting outliers using the distribution of compressed sizes rather than the dissimilarity measure. Consider the MDL dissimilarity between databases X and Y given above. First, the self-compression term $CT_X(\mathcal{D}_X)$, which depends only on the database being compressed, is very much necessary when using MDL dissimilarity as a similarity measure. This is because it subtracts the complexity of the database; it adjusts for the fact that some databases are very simple, and thus cross-compressed to a small size, without being similar. However, in the anomalous database problem, all normal databases are expected to have a reasonably similar complexity, and thus a similar self-compression size. Databases that are more complex than the norm will necessarily have a greater cross-compression size than the norm, and, if significantly more complex, will show up as anomalies in the distribution of $CT_Y(\mathcal{D}_X)$. Databases that are less complex than the norm will have a smaller cross-compression size, possibly lower than the norm. This might appear problematic but it means that such databases are expected to compress the joint database (more representative of a normal database) badly. These databases will then likely show up as anomalies in the distribution of $CT_X(\mathcal{D}_Y)$.

A second reason is that this second term in the MDL dissimilarity is expected to have a very high variance when the individual databases are small. The term is also expected to be reasonably independent of the first term, since the code tables may be very different for small databases, which causes the variance of both terms to add up, obscuring the results simply because of the size of the databases. The rationale for detecting outliers in both distributions separately is then that both distributions may capture different types of anomalies, and aggregating them may obscure results that are otherwise clearly anomalous.

One extra consideration in the anomalous database problem is that the joint database and the individual databases are in most cases of very different lengths. Even though the dissimilarity scores can be normalized by dividing by the self-compressed size of the databases that is being compressed as proposed in [40], the effect of generating code tables from comparatively smaller databases is that the variability expected in these code tables is much larger. Since the code sizes follow the logarithm of the probability, much higher variance is expected in the distribution of compressed sizes of the joint database using the individual database code tables than the other way around. Extreme variance can reduce the accuracy of the method. Where this situation is identified it may be preferable to consider only the compressed sizes of individual databases by the joint code table. This is the case, for example, when the algorithm is reduced to perform anomaly detection of single itemsets.

3.5 Compressed size distribution of synthetic data

The idea behind the algorithm was first tested using synthetic data, to characterize the distribution of compressed sizes obtained by the algorithm on data with a single mode.

A set of 200 normal (non-anomalous) databases of size 200 were obtained by sampling itemsets from a categorical, or multinoulli distribution of size 40. The categorical distribution parameters were uniformly sampled. The lengths of the itemsets were sampled from a Poisson distribution with mean $\lambda = 5$. This resulted in an itemset database of size 40000, with an alphabet of size 40, generated from a multinomial distribution.

The database was joined and compressed, and the cross-compression sizes $D(\mathcal{D}_i, joint)$ and $D(joint, \mathcal{D}_i)$ were obtained for the 200 databases. Figure 3.1 shows the histogram of the obtained compressed sizes. The experiment was repeated multiple times, and for different alphabet sizes with similar results. This leads to conclude that a simple distribution of cross-compressed sizes can be expected for this kind of simple unimodal data.

Because the total cross-compression size is the average of the cross-compression sizes of the itemsets in the database, it is suitable to be analyzed in the light of the Central Limit Theorem which establishes that the average of independent identically distributed variables approaches a normal distribution. First consider the total compressed size of the database, which is the sum of the compressed sizes of the itemsets. Under the assumption that observations (itemsets) in normal databases are independently sampled from a single underlying distribution and databases have the same length, the Central Limit Theorem applies to cross-compression of databases using KRIMP. Independence comes from the fact that itemsets are sampled independently and KRIMP also covers every itemset independently. Sizes are identically distributed because all itemsets in the database are sampled from the same underlying distribution. When taking the average size per item independence does not hold because KRIMP covers complete itemsets. However, in this case experimental results indicate that the distribution is very close to normal.

In a real application scenario, however, normal databases are not necessarily sampled from exactly the same underlying distribution, and itemsets are not independently sampled. In fact, in applications such as text, dependencies between itemsets are likely strong. The degree to which normality holds for real data must be determined experimentally.

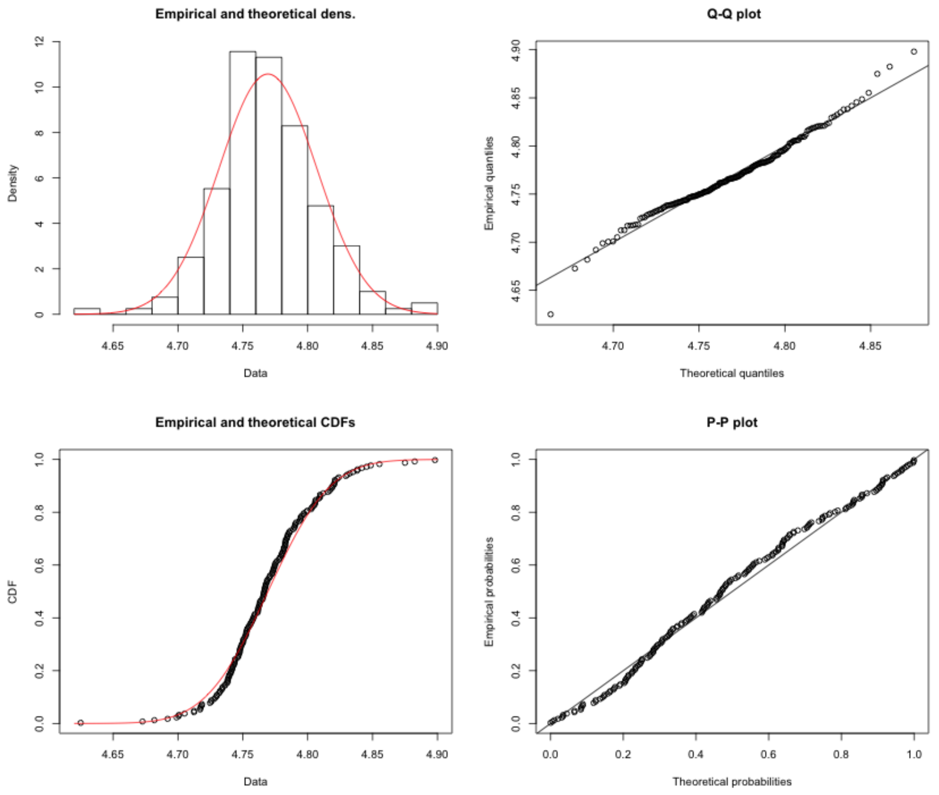

FIGURE 3.1. Distribution of normal $D(\mathcal{D}_i, joint)$, for a synthetic database generated from a multinomial distribution. The red plot is the MLE fit of a normal distribution.

3.5.1 KRIMP requires structure

It is important to note that the KRIMP algorithm requires significant structure in the data for the MDL dissimilarity to be meaningful. Datasets with no dependencies within itemsets like the ones generated from categorical, or multinoulli distributions tend to generate code tables which do not work well to define a dissimilarity measure. This is because less common items will be kept in the code table as single items, without forming part of multi-item itemsets in the code table. This happens because more common items already form itemsets with other common items, and less common items are not common enough for the 2-itemset that they form to improve the code table. This means that the less common single items will be covered most of the time by the single alphabet item. These single items, although uncommon, might then have a relatively small code. In some cases these might be among the smallest codes in the code table, although the item is relatively uncommon. When a database containing a much bigger proportion of this item is covered using such a code table, its compressed size may be small, a misleading result given that the item distributions are very different.

In the case of data generated from a categorical distribution, one is better off using the

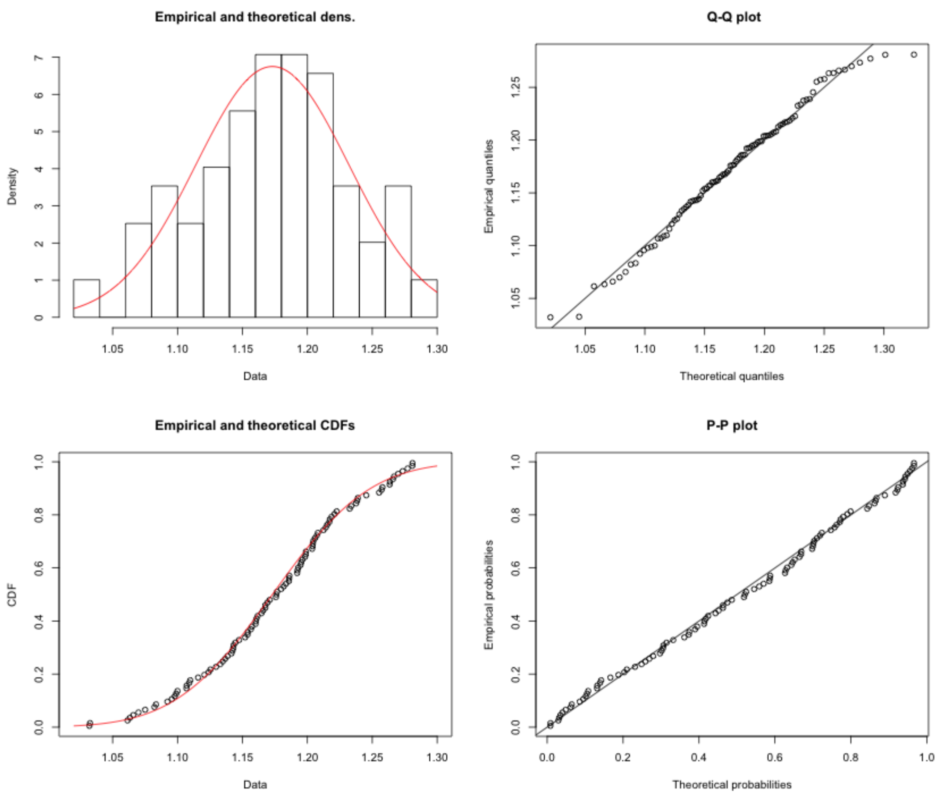

FIGURE 3.2. Distribution of normal $D(joint, \mathcal{D}_i)$ for a synthetic database generated from a multinomial distribution. The red plot is the MLE fit of a normal distribution.

KL-divergence as dissimilarity measure, since in facts it captures *all the difference* between both datasets. In short, KRIMP works better the more structure the data has, and items which have little or no dependencies with other items will likely pose a challenge. In general, these cases can be identified by inspecting the database code tables. Preliminary experiments made this clear and this is the reason why the algorithm was not tested further using categorical distributions. The following section performs experiments making use of real data, with much more significant structure.

3.6 Experiments with classification databases

To assess the ability of KRIMP to detect anomalies in real data, several UCI datasets used in pattern set mining were used to test the algorithm. These databases had to fill a few requirements, namely having at least a class with enough observations to use as normal data. The number and size of the normal databases was adjusted to use the vast majority of the observations in the majority class. Samples were drawn from the rest of the classes repetitively to form possibly anomalous databases, termed foreign databases (to differentiate from real anomalies). Table 3.1 shows the datasets used and some general information. The goal was to characterize the trade-offs made by the algorithm in detecting anomalous databases based on their structure, as well as the sensibility to the size of the databases and the proportion of foreign data. Note that unsupervised anomaly detection is different from binary classification and accuracy / recall values should not be taken as benchmarking the algorithm.

3.6.1 Experiment

One iteration of the experiment with classed itemset databases can be outlined as:

1. Sample N normal databases of size s from a single class (ie. the normal class).
2. Sample n foreign databases of size s from the rest of the dataset.
3. Run algorithm 1 on the conjunction of the databases.

Table 3.1 shows the databases used, as well as the number of classes used as normal classes, for each database. These always correspond to the largest classes in the database. The maximum class dissimilarity, as reported in [40] is included for reference. This are the minimum and maximum dissimilarities between pairs of classes in the database, per the aggregate measure proposed in [40] and presented in section 2.5. This should give a measure of how distinguishable the classes in the database are from one another.

Dataset	Rows	Alphabet	Classes used	Dissimilarity	minsup
adult	48842	95	2	0.6 - 0.6	0.001
mushroom	8124	117	2	8.24 - 8.24	0.02
nursery	12960	27	3	1.26 - 10.12	1
pendigits	10992	76	10	1.33 - 4.43	0.002

Table 3.1. UCI datasets used for the anomaly detection experiments. The minimum support listed is the one used for all experiments involving the database.

3.6.2 Detecting anomalies

In this first experiment, all the classes from table 3.1 are used as normal class, with the number of databases fixed at 100 for all classes. The number of foreign databases is fixed at 10. Each experiment is repeated 30 times. The goal is to measure the behavior of the algorithm on the different classes the data set contains, which vary in size and dissimilarity.

The precision and recall of the algorithm in detecting the foreign databases were measured based on three different methods of selecting anomalies from the resulting compressed sizes:

Top N. Pick the top N compressed sizes from each distribution as anomalies (ie. a maximum of $2N$). N is set to the number of foreign databases introduced in the dataset unless specified otherwise. This is to give a better idea of the results achieved when the number of anomalies is known. It is not suggested as an anomaly detection method unless the number of anomalies is known beforehand.

3SD rule. Regard as anomalies all compressed sizes further than 3 standard deviations away from the mean.

Rosner test. Assuming an approximate normal distribution of non-anomalous compressed sizes (the validity of this assumptions is investigated further), the test by Rosner [31] is used to detect anomalies in the compressed sizes. The Rosner test is run for a maximum of $2a$ anomalies, where a is the number of introduced foreign databases, and $\alpha = 0.05$. Note that the validity of the t-distribution approximation to the Rosner critical values is untested for $k > 10$ [31]. This is not a limitation of the Rosner test itself. Although the approximation is used here to test for more than 10 anomalies, in most experiments this number is not exceeded greatly in order to stay withing reasonable bounds. Furthermore, experiments appear to indicate that the approximation remains useful even when the number of anomalies greatly exceeds 10, as shown in section 3.6.3.

Preliminary tests showed that outliers on the left side of the compressed size distributions where unlikely when running algorithm 1 on real and synthetic data. This is expected since, as analyzed in section 3.4.1 these correspond to a very special kind of anomalies: databases which tend to have a more simple structure while also sharing the most common itemsets in the code table. Having observed this it was decided to use the single-tailed test for anomalies in the right side of the distributions. When this is not the case, however, it is recommended to use the two-tailed test. Anomalies found in the left hand side can either be ignored (since they are likely to show up as anomalies in the right hand side in the second distribution as analyzed in section 3.4.1) or directly detected as anomalies.

Both compressed size distributions are considered separately and then joined as in: a database i is considered anomalous if $D(\mathcal{D}_i, joint)$ is anomalous in the distribution of single database compressed sizes *or* $D(joint, \mathcal{D}_i)$ is anomalous in the distribution of joint database compressed sizes.

Table 3.2 shows the results obtained, along with the database size used for each class. Since the database size is simply $s = floor(classSize/100)$, and the classes have very dissimilar sizes, the database sizes are vary variable, ranging from 10 to 371.

Results indicate that the 3SD Rule tends to favor accuracy over recall, while the Top N and Rosner test criteria favor recall, but also display very high accuracy. It stands out that the Rosner test is able in some cases to outperform the top N approach in accuracy even though it has less information. Interestingly, the most challenging classes are also the largest ones.

Table 3.2. Results of performing anomaly detection for multiple normal classes. s gives the database size. *anomalous/normal* indicates the ratio of the average compressed sizes using normal and foreign databases, where $D(\mathcal{D}_i, joint)$ and $D(joint, \mathcal{D}_i)$ are the two distributions.

Class	s	Top N		3SD rule		Rosner test		foreign / normal	
		Acc	Rec	Acc	Rec	Acc	Rec	$D(\mathcal{D}_i, joint)$	$D(joint, \mathcal{D}_i)$
adult1	371	0.94	1.00	1.00	0.64	0.97	1.00	1.24	1.19
adult2	116	0.97	1.00	1.00	0.51	0.88	1.00	1.25	1.47
mushroom1	42	1.00	1.00	1.00	0.93	0.92	1.00	1.73	1.88
mushroom2	39	1.00	1.00	1.00	0.90	0.82	1.00	1.76	2.18
nursery1	43	1.00	1.00	1.00	0.78	0.99	1.00	1.36	1.35
nursery2	42	0.89	1.00	1.00	0.58	0.97	1.00	1.23	1.12
nursery3	40	0.94	1.00	1.00	0.64	0.99	1.00	1.34	1.14
pendigits0	11	0.97	1.00	1.00	0.48	0.97	1.00	2.30	1.15
pendigits1	11	0.97	1.00	1.00	0.52	0.99	1.00	2.29	1.16
pendigits2	11	0.97	1.00	1.00	0.40	0.98	1.00	2.16	1.16
pendigits3	11	0.99	1.00	1.00	0.48	0.98	1.00	2.62	1.25
pendigits4	11	0.89	1.00	0.97	0.44	0.99	1.00	1.92	1.08
pendigits5	10	0.99	1.00	1.00	0.45	0.99	1.00	2.27	1.19
pendigits6	10	0.81	1.00	0.79	0.38	0.99	1.00	1.72	1.04
pendigits7	10	0.84	1.00	0.95	0.36	0.99	1.00	1.74	1.06
pendigits8	10	0.94	1.00	1.00	0.39	0.98	1.00	1.99	1.17
pendigits9	10	0.96	1.00	0.99	0.38	0.99	1.00	2.00	1.15

This is however due to the biggest databases having classes more similar to one another. The experiments with *pendigits* use only 10/11 itemsets per database but these are enough to capture the common structure.

3.6.2.1 Normality

It was verified to what degree the normal class distributions obtained from the FIMI databases resemble a normal distribution, in this case with real data, since this is the assumption underlying the use of Rosner's test. Figures 3.3 and 3.4 show the histogram, fitted normal distribution (MLE), Q-Q and P-P plots for the 100 normal databases used in one run of the experiment using the minority class of the *adult* database.

In general it was found that for most of the classes the distribution of compressed sizes can be approximated reasonably well by a normal distribution. The distribution plots obtained for the rest of the databases are shown in Appendix A. The minority class was used for these plots because it is the one for which the smallest databases were used, and thus the one with highest expected variance.

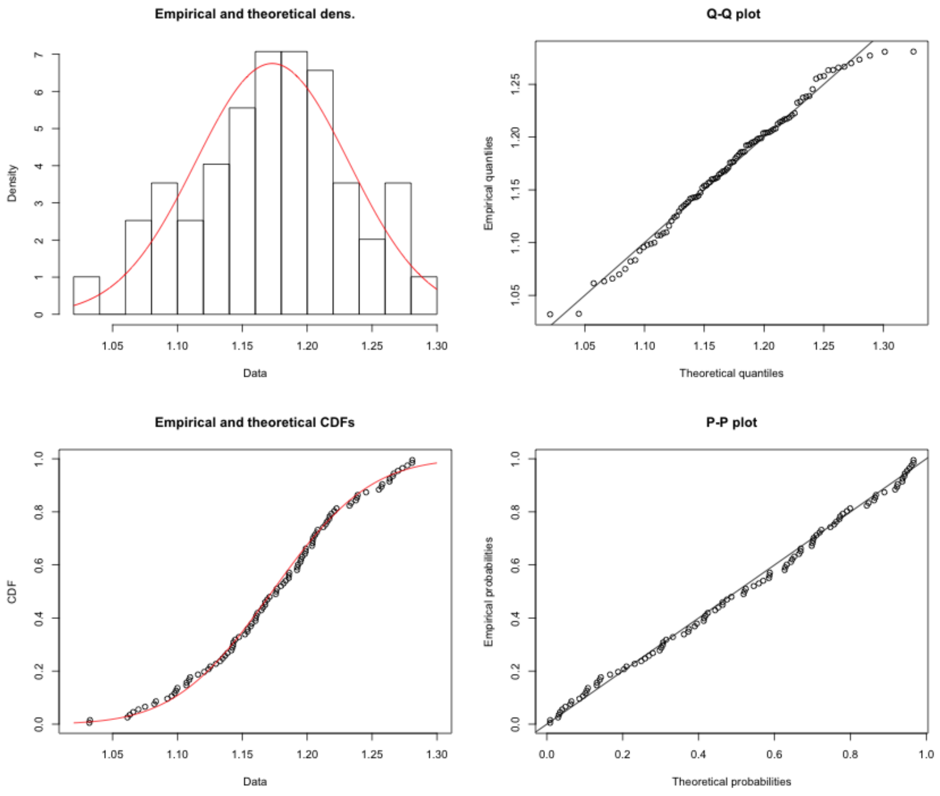

FIGURE 3.3. Distribution of normal $D(\mathcal{D}_i, joint)$, for the minority class of the *adult* database, in one run of the experiment of table 3.2.

3.6.3 Adding anomalies

A natural question is how the algorithm behaves in the presence of more anomalies. The next experiment makes use of the *nursery* database due to it being one of the most challenging databases in terms of dissimilarity between classes. Now the number of foreign databases is varied in increments of 4. The number of normal databases is kept fixed at 100 and the size of the databases is also kept fixed at 43 (in order to almost completely use the normal class' rows).

Table 3.3 shows the results for this class. As would be expected, as the number of foreign databases increases, the ratio between the compressed size of foreign databases and the compressed size of normal databases decreases (last two columns), making them harder to distinguish. Note however that even for 48% of foreign classes, the *topN* and Rosner test criteria show almost 100% accuracy and recall. This is explained by the very low variance observed in the compressed size distributions. Note however that in this experiment the Rosner test is being used for numbers of anomalies far beyond those cited in the original paper [31], and t-distribution approximation might not be exact for these values.

It stands out that the experiments with a lower number of anomalies do not have accuracy

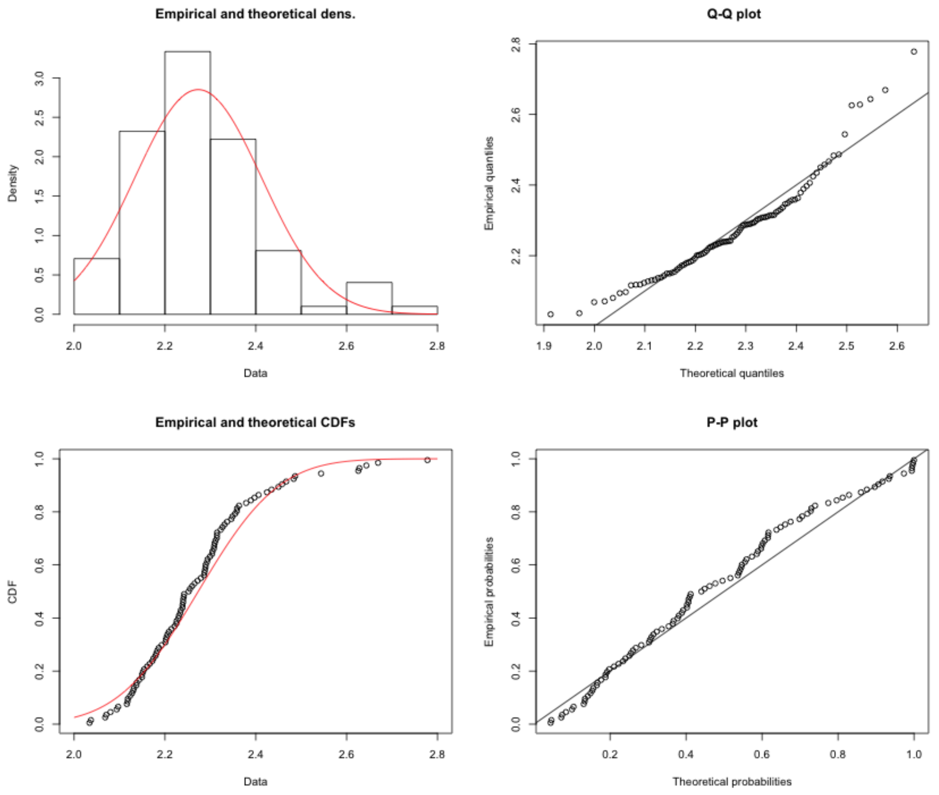

FIGURE 3.4. Distribution of normal $D(joint, \mathcal{D}_i)$, for the minority class of the adult database, in one run of the experiment of table 3.2.

of 1 for the Rosner test. This is due to a particularly anomalous normal database that is being picked up by the Rosner test, but not by 3SD rule (ie. a nuance of the data).

The rest of the classes in the *nursery* database displayed a very similar behavior. Figures 3.5 and 3.6 show the average ratio between compression sizes of anomalous databases / code tables and the normal ones for the three largest classes in the *nursery* database. These tend to indicate that class 1 is the most easily distinguishable and class 2 the hardest. For this database, the distribution of compressed sizes of individual databases (first figure) appears to better expose anomalies in most cases, since the ratios tend to be higher. Given the very significant amount of structure in this database it is likely that compressing in both directions is mostly redundant.

Table 3.4 presents the results of the same experiment on the majority class of the *adult* database. This is the most challenging of the tested databases despite being the largest and, as expected, accuracy and recall decrease with the addition of foreign databases. It is still remarkable that the Rosner test algorithm is able to detect with 95% accuracy and recall up to 18 anomalies. These results, and the normality plots are indications that the normality assumption is indeed appropriate for itemset databases with sufficient structure.

TABLE 3.3. Results of performing anomaly detection on the majority class of the *nursery* database for different numbers of anomalies. *anomalous/normal* indicates the ratio of the average compressed sizes using normal and foreign databases, where $D(\mathcal{D}_i, joint)$ and $D(joint, \mathcal{D}_i)$ are the two distributions. The database size was kept fixed at $s = 43$ and the number of normal databases at 100.

Foreign DBs	Top N		3SD rule		Rosner		foreign / normal	
	Acc	Rec	Acc	Rec	Acc	Rec	$D(\mathcal{D}_i, joint)$	$D(joint, \mathcal{D}_i)$
4	1.00	1.00	1.00	1.00	0.97	1.00	1.67	1.40
8	1.00	1.00	1.00	0.99	0.99	1.00	1.45	1.37
12	1.00	1.00	0.90	0.26	0.98	1.00	1.30	1.34
16	1.00	1.00	0.27	0.02	0.99	1.00	1.28	1.31
20	1.00	1.00	0.00	0.00	0.99	1.00	1.27	1.28
24	1.00	1.00	0.00	0.00	0.99	1.00	1.25	1.26
28	1.00	1.00	0.00	0.00	0.99	1.00	1.22	1.24
32	1.00	1.00	0.00	0.00	0.99	1.00	1.22	1.22
36	1.00	1.00	0.00	0.00	0.99	1.00	1.20	1.20
40	1.00	1.00	0.00	0.00	1.00	1.00	1.20	1.18
44	1.00	1.00	0.00	0.00	1.00	1.00	1.18	1.16
48	0.99	1.00	0.00	0.00	1.00	1.00	1.18	1.15

TABLE 3.4. Results of performing anomaly detection on the majority class of the *adult* database for different numbers of anomalies. *anomalous/normal* indicates the ratio of the average compressed sizes using normal and foreign databases, where $D(\mathcal{D}_i, joint)$ and $D(joint, \mathcal{D}_i)$ are the two distributions. The database size was kept fixed at $s = 116$ and the number of normal databases at 100.

Foreign DBs	Top N		3SD rule		Rosner		foreign / normal	
	Acc	Rec	Acc	Rec	Acc	Rec	$D(\mathcal{D}_i, joint)$	$D(joint, \mathcal{D}_i)$
14	0.92	1.00	0.90	0.16	0.94	0.98	1.21	1.43
18	0.89	1.00	0.67	0.06	0.95	0.95	1.19	1.39
22	0.87	1.00	0.57	0.04	0.90	0.84	1.17	1.37
26	0.81	0.99	0.33	0.01	0.64	0.60	1.13	1.34
30	0.76	0.98	0.42	0.02	0.25	0.22	1.11	1.32

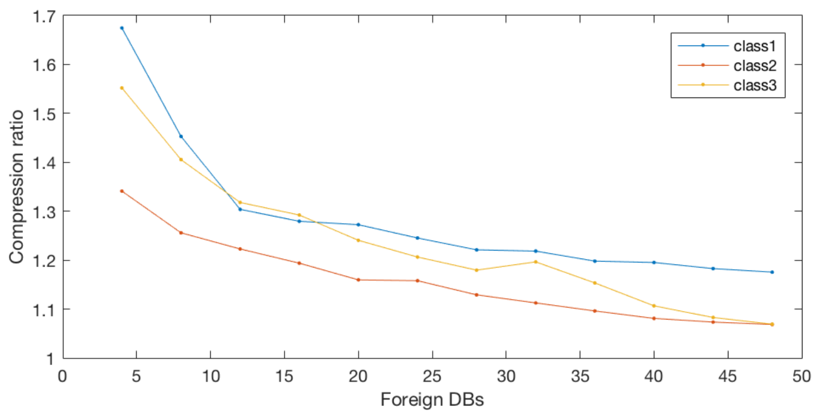

FIGURE 3.5. Ratio between the average compressed size per item of foreign databases and average compressed size per item of normal databases (ie. ratio of $D(\mathcal{D}_i, joint)$ between foreign and normal databases) for the three largest classes in the *nursery* database, as a function of the number of anomalies.

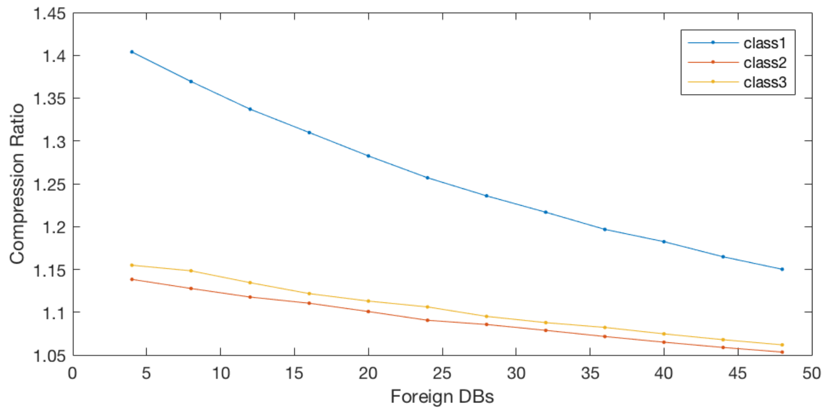

FIGURE 3.6. Ratio between the average compressed size per item of the full database using code tables from foreign databases and using code tables from normal class databases (ie. ratio of $D(joint, \mathcal{D}_i)$ between foreign and normal databases) for the three largest classes in the *nursery* database, as a function of the number of anomalies.

3.6.4 Effect of database size

The *adult* database was used to asses the effect of database size on the algorithm. The smallest of the two classes in the database was used as normal class, and the database size was varied from 10 to 100 itemsets, while keeping the number of normal databases (100) and number of foreign databases (10) fixed. A relative minimum support of 0.001 was used for frequent itemset mining and the experiments were repeated 10 times.

Table 3.5 shows that, as expected, the algorithm consistently takes advantage of having more data at its disposal. Note how the Rosner test and 3SD rule tend to favor accuracy over recall for smaller databases, and are able to identify more databases as anomalous as the amount of data increases. This is a result of the reduction of variance made possible by the extra data, which causes the compressed sizes to get closer to bimodal. It is only with 100 itemsets per database that the Rosner test identifies the 10 foreign databases almost perfectly.

TABLE 3.5. Results of performing anomaly detection on the minority class of the *adult* database for different database sizes. *anomalous/normal* indicates the ratio of the average compressed sizes using normal and foreign databases, where $D(\mathcal{D}_i, joint)$ and $D(joint, \mathcal{D}_i)$ are the two distributions. The number of anomalies was kept fixed at 10 and the number of normal databases at 100.

s	Top N		3SD rule		Rosner		foreign / normal	
	Acc	Rec	Acc	Rec	Acc	Rec	$D(\mathcal{D}_i, joint)$	$D(joint, \mathcal{D}_i)$
10	0.33	0.55	0.83	0.14	0.67	0.10	1.19	1.18
20	0.45	0.76	0.90	0.21	0.68	0.19	1.21	1.19
30	0.56	0.88	0.90	0.19	0.90	0.22	1.24	1.22
40	0.61	0.91	1.00	0.16	0.90	0.16	1.24	1.24
50	0.78	0.99	1.00	0.20	0.80	0.27	1.25	1.29
60	0.84	0.99	1.00	0.24	0.90	0.36	1.25	1.33
70	0.85	1.00	1.00	0.33	0.96	0.85	1.25	1.35
80	0.91	1.00	1.00	0.30	0.95	0.99	1.24	1.38
90	0.92	1.00	1.00	0.34	0.97	0.99	1.25	1.42
100	0.96	1.00	1.00	0.46	0.87	1.00	1.25	1.48

Figure 3.7 gives a better idea of the variability in the 10 runs of the experiment. Note that the first two ratio plots have a different y axis scale. The average ratios between anomalous and normal data are very stable, however, up to $s = 60$ some runs of the experiment report no outliers. It is until $n = 70$ that results start being consistent. Note also that in this experiment, the ratios in $D(joint, \mathcal{D}_i)$ are greater than those in $D(\mathcal{D}_i, joint)$. Accordingly, the results indicate that it was in this distributions where most outliers were detected, contrary to the experiments with the *nursery* database. This can only be attributed to the structure of the databases and shows the importance of detecting outliers in both distributions.

3.7 An interpretable experiment

In order to better illustrate the idea behind the algorithm, a simple experiment was set up making use of highly interpretable data: natural language. It was decided to make use of freely available Wikipedia articles. A set of normal databases was created by harvesting the raw text of the Wikipedia articles of the top 40 highest-grossing artists and bands in history, as per the relevant Wikipedia ranking at the time of writing. Data was filtered only by removing punctuation and stopwords from a list of 152 English stopwords and lemmatizing. One itemset

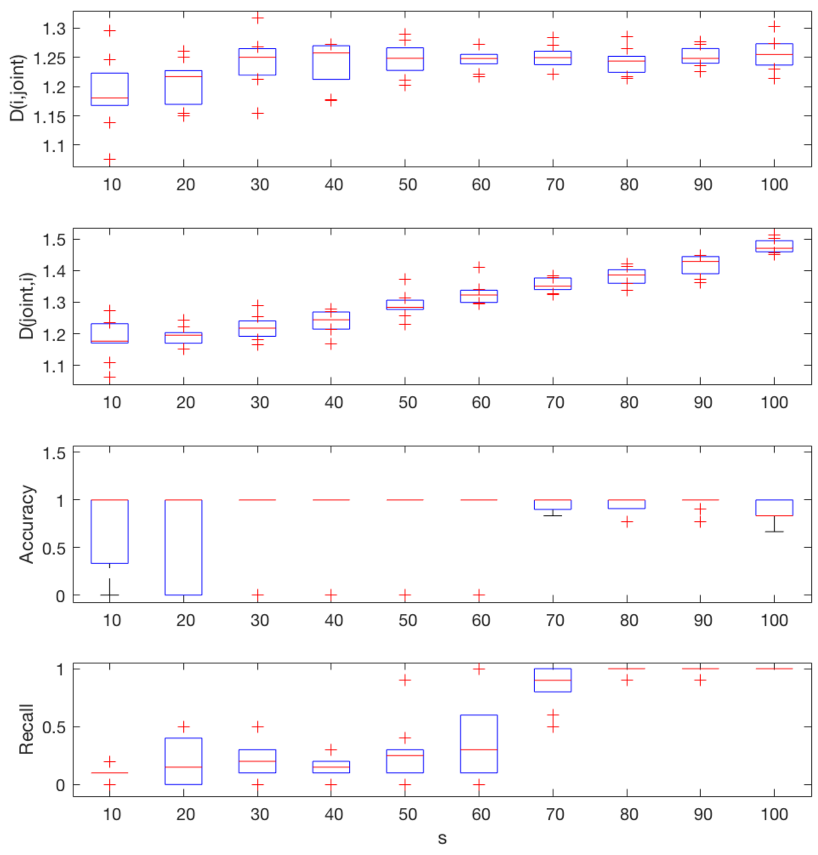

FIGURE 3.7. Box plots for the *foreign/normal* ratios and Accuracy and Recall when using the Rosner test to select anomalies in the variable database size experiment. Boxes span from the 0.25 percentile to the 0.75 percentile for the 10 runs of the experiment (3d and 8th values) and values outside are drawn as an x.

was created per sentence. The resulting itemsets were then filtered to remove the less common words on each article using a relative threshold, for each document separately. No further pre-processing was done, as the goal of the experiment was only illustrative.

Articles in the normal set tended to contain biographies, career descriptions, lists of awards and descriptions of historical and personal happenings. Three unrelated articles were separately added as anomalies, one for a fruit (mango), one for a historical figure (George Washington) and one for a famous football player (Lionel Messi), the one intuitively more similar, but anomalous in any case. The anomaly detection algorithm was run, compressing with a minimum threshold of $\theta = 0.005$. Figure 3.8 shows the distribution of compressed sizes. The red line indicates the corresponding compressed size for the anomalous text. The three anomalies are easily identifiable in at least one of the plots. The article on the football player is the hardest to distinguish as it is not anomalous in the forward distribution. This illustrates the fact that the two ways of compression capture different aspects of dissimilarity. In this case the article is relatively easy to compress using the joint code table, as it contains similar vocabulary, most certainly also describing life, career and awards. The variance in the normal data is enough to obscure it as an anomaly. However, normal articles contain music-specific vocabulary that is not easy to compress in the backward direction, and this leads to the article being clearly identified as anomalous in this distribution.

The left histograms look very much alike, and in fact these forward distributions were almost identical despite the fact that a different anomaly, and of different length, was added every time.

Besides illustrating the idea behind the algorithm, this simple experiment shows that KRIMP is able to find obvious anomalies in text data. An important question is whether KRIMP would be able to capture much more subtle relations in text in a way that is useful for classification. The following chapter explores that possibility in the context of authorship attribution.

3.8 Generalization to multi-modal databases

One drawback of the anomaly detection algorithm comes from the weakness of the dissimilarity measure when the normal database is heavily multi-modal. For example, consider an input with 100 normal databases and 10 databases with a very different structure. If these 10 databases are similar to one another in structure, one may wish to consider them normal, while considering them as anomalous if every individual database has a different structure. The proposed algorithm only partially behaves in this way since 10 databases with similar structure will tend to have lower compressed sizes than if each has a different structure, as the common structure tends to be aggregated in the joint database. However, this may not be enough to offset the effect when the complexity of the group of database is significantly greater than that of the main mode of normal data.

An algorithm which clusters the data (for example, as proposed in [38]) before performing anomaly detection on the clusters would provide a solution for this problem. The extension of the algorithm to deal with this situation can be outlined as:

- Cluster the databases using the KRIMP clusterer [38], or some other appropriate clustering algorithm that considers structure.
- Apply the anomaly detection algorithm to each cluster. Because clustering algorithms will assign elements to the cluster with which they share the most structure, if an element is anomalous within a cluster, it can be expected to be anomalous within any other cluster.
- Determine if some of the clusters are made up of anomalous data. These are clusters with little common structure within their itemsets. A simple approach would consider the

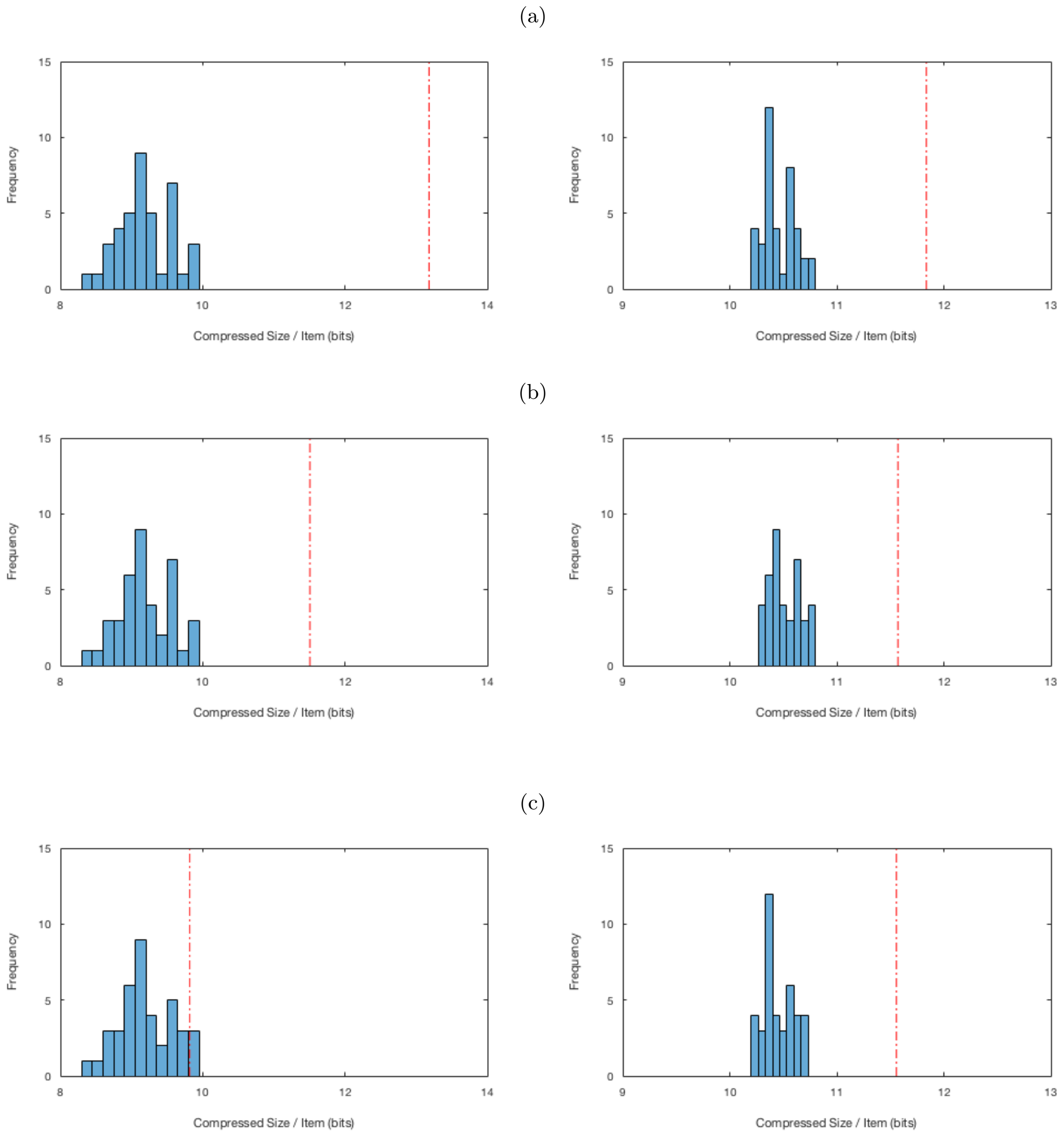

FIGURE 3.8. Histograms of compressed sizes $D(\mathcal{D}_i, joint)$ (left) and $D(joint, \mathcal{D}_i)$ (right), for the experiment using Wikipedia articles. The inserted anomalies are the articles for: (a) Mango fruit. (b) George Washington. (c) Lionel Messi. The red line indicates the cross-compression size corresponding to the anomalous document.

clusters in order of variance of the distribution of compressed sizes of their elements, along with a threshold or rule for selecting top elements.

3.9 Conclusions

The most important conclusions derived from the experiments are:

- The proposed anomaly detection algorithm behaves in a consistent way and is able to find subjective anomalies that depend on the context defined by the normal data. Whether or not a particular database is classified as an anomaly depends on several factors:
  - Its cross-compression sizes, which are a measure of how anomalous the database is.
  - The number of databases with similar compression sizes. The Rosner test was used to govern this decision under the normality assumption.
  - The structure of the normal data.
- Cross-compression sizes are able to consistently rank the data that "does not belong" to a normal class. When the number of anomalies is known, simply considering the largest cross-compression sizes as anomalies gives very high accuracies even for small databases. The Rosner and 3SD tests assume less knowledge.
- The assumption of normality of the distribution of compressed sizes is justified for databases that are relatively single-modal. The Rosner criterion performs remarkably well given enough data. Code tables tend to capture the behavior of the normal data well as long as most of the data is normal.
- A limitation of the approach comes from the fact that dissimilarity is summarized into two single real numbers. The algorithm does not distinguish between databases with the same compressed sizes. This particularly affects multi-modal databases, where a group of similar databases that is harder to compress than the norm might be signaled out as anomalous, despite it not being particularly rare.
- A limitation specific to the use of the Rosner test is the need of data to validate the normality assumption. However, other selection rules, presented here or not, are able to provide useful results given the anomaly rankings that are output from the process.

Chapter 4

Information-theoretic authorship attribution

4.1 Introduction

One area of study in which one wants to quantify the similarities and differences between documents, which can be regarded as itemset databases, is that of natural language processing. The traditional bag-of-words assumption used in many algorithms disregards the ordering of the words within documents, as well as all structural information. Even when word ordering and sentence division are clearly important to us, computers have shown that given enough documents, reducing the representation to word or n-gram counts per document is enough to perform many tasks in the field, such as topic extraction, as in the Probabilistic Latent Semantic Analysis [12] and Latent Dirichlet Allocation techniques; and text categorization, using a variety of classification algorithms.

Another problem that typically makes use of such aggregated representations is algorithmic authorship attribution, in which one wishes to determine the most likely author of a text whose authorship is disputed, but there are likely candidates. Other than the disputed text, the input data consists of a set of collection of texts (eg. books) from the candidate authors. Authorship attribution has a long history, with the most famous example in computational attribution being that of the Federalist Papers [13]. The statistical analysis of text productions, known as *stylometry*, however, predates computers by at least half a century, dating back to the late 19th century, with work by Thomas Mendenhall on word length spectra [26].

Such techniques have not only been applied to historical texts, but also to forensics [23], in tracking down criminals based on their writings [13], plagiarism detection, and even in attempts to de-anonymize important figures [14]. Target documents have expanded from literary works to text written at online forums and even computer source code [3].

Authorship attribution algorithms typically define a (dis-)similarity measure between texts and assign likelihood scores to each author having created the disputed text. In this chapter, the bag-of-words assumption is questioned by using the KRIMP algorithm to capture structure in text. Several ways of representing text documents as itemset databases are tested, in order to determine if taking into account arbitrary order co-occurrence relations between itemsets (words or n-grams) is able to provide an improvement to the accuracy of authorship attribution algorithms, with respect to traditional approaches based on dissimilarity measures over the vector of element counts.

The KRIMP-based classifier used for the experiments was proposed by van Leeuwen et al. [37]. The algorithm was adapted from classifying single itemsets to classifying complete databases. This change is trivial, since the compressed size of the database is the sum of the compressed size of its itemsets. Its application to literature is however novel, and brings important challenges, namely the preprocessing of the data and the choice of data representation becomes central to the accuracy of the algorithm. This involves mainly the selection of an alphabet and a way of partitioning the text into itemsets with relevant common structure that can be exploited by the algorithm. At the same time, alphabet elements must ideally have dependencies that are relevant to the attribution task, in order to benefit from the MDL dissimilarity measure.

The chapter begins by presenting the algorithm used: the KRIMP classifier. Experiments were performed on two datasets with very different characteristics: a large corpus of books from Victorian novelists, and a smaller corpus of songs. Emphasis was placed on carefully detailing the data preprocessing steps, as these are very important for the interpretation of the results using KRIMP. The objective is to determine whether or not using KRIMP to find arbitrary order intra-item relations offers an advantage over much less expensive measures like the Kullback-Leibler Divergence, frequently used in authorship attribution and text classification. The two datasets are a good fit for this task since novels and poems exhibit very different kinds of structure.

4.2 Authorship Attribution

Authorship attribution is a form of text classification with very specific needs. Depending on the type of data at hand and objective of the algorithm, authorship attribution algorithms might attempt to capture stylistic or content-dependent information to differentiate among writings from different authors. The core algorithm consists normally in a generic classification algorithm (eg. SVM) or a dissimilarity measure like the Kullback-Leibler divergence. The main differences with, for example, document classification by topic are normally in the preprocessing and feature extraction steps, which must be decided based on the knowledge of the problem.

4.2.1 Feature extraction for Authorship Attribution

As explained further in the next section, not all authorship attribution techniques make use of explicit features. However, most classification-based algorithms do define a feature set, attempting in most cases to capture author style and choice. The most common authorship attribution features present in the literature are:

Words . The bag-of-words representation is a common one in the text domain, in problems like document classification, and topic extraction [12]. However, when it is not desired to take topics into account for attribution, the word profile of a particular author is reduced, normally by considering only function words, specific lists of common words, or the set of most frequently occurring words [33]. While in topic extraction / classification many of these words would be removed from the document representation due to being stopwords or having a low tf-idf score, in authorship attribution the distribution of these common words has shown to be able to distinguish between up to 20 authors with accuracies greater than 50% [7].

Character n-grams . A character n-gram representation of a text is obtained by applying a sliding a window of length n to the text, normally including punctuation. Character n-grams are an attempt to capture more stylistic information in features that are easy to obtain from the text. n-grams capture punctuation and word choice, as well as other aspects such as spelling mistakes and have also been shown to be effective [7, 32]. However, n-gram profiles have the problem of being heavily biased when texts have been edited. N-grams also produce an explosion in dimensionality relative to word profiles. This last problem, however, is not as important for n-grams of size 2 or 3, for languages using the Latin alphabet. n-gram sizes in the range $[2, 4]$ have been shown to be the most effective, [7], with accuracy dropping for very large n-grams due most certainly to the resulting sparseness.

When using n-grams as features small n-gram sizes do not necessarily increment the number of features, relative to the use of words. However, when representing documents as itemsets (eg. for compression using KRIMP) using n-grams means an explosion in the number of itemsets per item, relative to the use of words. A natural question is whether all n-grams are important for the attribution task. Sapkota et al. [32] present a classification of n-grams into 10 categories:

Affix n-grams .

Prefix. Covers the first n characters of a word at least $n + 1$ characters long.

Suffix. Covers the last n characters of a word at least $n + 1$ characters long.

Space-prefix. Begins with a space.

Space-suffix. Ends with a space

Words n-grams .

Whole-word. Covers a word of exactly n characters.

Mid-word. Covers a part of a word. Does not include the first or last characters.

Multi-word. Spans multiple words.

Punctuation n-grams .

Begin punctuation. Its first character is punctuation but middle characters are not.

Mid-punctuation. Contains a punctuation character that is not its first or last character.

End punctuation. Its last character is punctuation but middle characters are not.

Sapkota et al. [32] perform experiments using only n-grams of every one of the 10 subcategories on two English datasets, a single domain and a cross-domain dataset. Results indicate that *affix* and *punctuation* n-grams tend to be the most effective, specially in the cross-domain setting. When the corpus comes from a single domain *mid-word* features become relevant. Experiments combining categories showed that *affix* and *punctuation* features alone perform better than the three categories combined.

Semantic features . Features that attempt to capture the semantics, as opposed to the writing style of a text, through, for example the use of a POS tagger to include some degree of semantic information. These features might also consider synonymity and semantic dependencies [35].

Derived features . Other features proposed in the literature include words and sentence length distributions and aggregated vocabulary richness measures [7]. These features are not used in this work, since these are calculated from the complete document, and translation to itemsets is not straightforward. Their use is thus mostly limited to general classification algorithms.

Yet another class is that used by many compression algorithms, which do not explicitly define a feature set, and are thus normally not n-gram nor word features, but variable-length character-level substrings which form the compressor's dictionary or model.

Features can also be subdivided based on the role they play on putting together a piece of writing, as [23]:

Lexical features These are related to the extension and choice of vocabulary of the author. Derived features such as vocabulary richness also fall into this category.

Syntactic features Include punctuation and function words, and are related to habits and choices when putting sentences together.

Structural features capture even higher-level structural choices, when putting together paragraphs and complete documents. Some are paragraph length, indentation, and use of signature.

Content-specific features These are similar to lexical features but are topic-specific. Even when writing about the same topic, authors tend to differ in their choice of words.

Li et al. [23] argue that these feature classes together form a *writeprint*, applied unconsciously when putting together a document. Other authorship attribution studies support these ideas Grieve [7], as different features in these categories have independently shown to encode information relevant for determining authorship. It is only reasonable to assume that, when aggregated correctly, the discriminative power of the features would increase.

4.2.2 Algorithms

Although in principle any general classification algorithm can be used for the authorship distribution problem, the fact that n-gram and word representations result in a probability distribution vector makes it straightforward to use dissimilarity measures for distributions. Algorithms making use of the word and n-grams representations can roughly be divided in two categories: simple distance measures and general classification algorithms. A third class of algorithms are compression algorithms, which generally do not make use of the features presented before, but of the lower level byte/character representation of the original text. Yet another class of methods that does not make use of explicit features is that based on language models.

4.2.2.1 Distance measures

Some distance measures proposed in the literature for the attribution task are:

Chi-Squared distance . This is a classic distance used in attribution studies. The Chi-Squared statistic is used to determine if a sample could have been drawn from a population. The Chi-Squared statistic is applied to the attribution problem as:

$$\chi(Q, A_j) = \sum_x \frac{(q(x) - a_j(x))^2}{a_j(x)}$$

where Q represents the query text, A_j a particular author, and the sum is over the union of the features from both texts. The author with the smallest statistic is attributed the text. The problem of having zero values in the denominator is solved by Grieve [7] through a simple feature selection rule: only features that appear in at least n of the documents for all authors are considered.

Kullback-Leibler(KL) Divergence . The KL divergence is a dissimilarity measure for probability distributions defined as:

$$D_{KL}(Q, A_j) = -\sum_x q(x) log a_j(x) + \sum_x q(x) log q(x) = H(P, Q) - H(P)$$

where $H(P, Q)$ is the cross-entropy of P and Q and $H(P)$ entropy. Classification by KL divergence normally takes into account only one side of the divergence: $D_{KL}(joint, \mathcal{D})$ where *joint* represents the probability distribution from an author's corpus and $\mathcal{D}$ that of the disputed text. This is because the complementary divergence normally has a larger variance due to the comparatively small size of disputed texts. The text is attributed to the author with the smallest divergence. Note that the entropy term is the same for all comparisons, as it only depends on the disputed text. Hence, the minimum KL divergence decision rule is equivalent to the minimum cross-entropy decision rule. As explained in depth in chapter 3, this is analogous to classification by compression size when using the KRIMP classifier.

The comparison of feature vectors based on the KL divergence has been tested in authorship search in large document collections with more than 100 authors [42], delivering very reasonable accuracies. This KL-divergence-based algorithm is similar to algorithm 2, but makes use of Dirichlet smoothing using a background model. Note that applying this kind of smoothing to a KRIMP-based algorithm is hardly practical.

Being a special case of the KRIMP classifier, using the KL divergence will allow for a more straightforward interpretation of the results, regarding the effect on accuracy of taking structure into account. For the purpose of comparing both approaches on equal ground, a Laplace correction is applied in this work, just as it is applied to the KRIMP code table.

4.2.2.2 Classification algorithms

Some of the most common algorithms proposed in the literature are:

Naive Bayes is a common algorithm in text classification due to the acceptable results it delivers for its simplicity. Note, however, that a Naive Bayes classifier is exactly equivalent to the KL divergence or cross-entropy classifier presented before, under a uniform prior. In Naive Bayes classification, one wishes to classify a document (represented by the associated probability distribution) to the author with the highest posterior probability given the document:

$$p(A_j|Q) = \frac{p(A_j)p(Q|A_j)}{p(Q)}$$

Assuming a uniform prior and setting $p(Q) = 1$:

$$p(A_j|Q) \propto p(Q|A_j)$$

Given the Naive Bayes conditional independence assumption:

$$p(A_j|Q) = C \prod_{i=1}^{n} p(w_i|A_j)$$

where the product is over all the words in Q and C is some constant. Since the logarithm is a monotonically increasing function, assigning to the largest $p(A_j|Q)$ is equivalent to assigning to the largest $logp(A_j|Q)$:

$$logp(A_j|Q) = \sum_{i=1}^{n} \log p(w_i|A_j) + log(C)$$

Since C is a constant only the first term is relevant in classification. By summing over the set of features in Q (rather than the words), and diving by the number of words in the document (also a constant given $\mathcal{D}$), the first term corresponds exactly to the cross-entropy $H(Q, A_j)$.

$$logp(A_j|Q) = -\sum_{x} q(x)loga_j(x) + log(C) = H(Q, A_j) + C$$

Maximizing the probability under the Naive Bayes assumption is exactly equivalent to minimizing the KL divergence, and both classifiers are equivalent.

Support Vector Machines have been proposed Diederich et al. [5] as a method that does not require feature selection, due to their ability to handle large feature sets with relatively low training times. SVMS were used by the authors to detect the author of a text, identifying documents from a given author from a set composed mainly of documents by other authors. The algorithm performed with nearly perfect accuracy and varying recall both on the word representation of the complete text, and a derived feature set combining word length counts, tagged words and bigrams of tagged words. The algorithm, however, was not tested against more simple approaches like Naive Bayes.

Neural Networks were tested by J. Tweedie et al. [13] for stylometric analysis of the Federalist Papers, with results consisted with previous work on this dataset. Neural Networks are attractive due to their ability to generalize and fault tolerance. The authors used a predefined set of only 11 English function words for the study: any, from, an, may, upon, can, his, do, there, on, every. These words are part of a set proposed by Mosteller and Wallace [27]-authors of the historical paper that settled the case of the Federalist Papers-as being relevant for this particular problem. A neural network with a single hidden layer was trained on the word counts. Results were not compared with more simple techniques.

4.2.2.3 Compression algorithms

Applying compression to authorship attribution seems a reasonable and straightforward idea. Should a compressor's dictionary or code table computed from the works of author A be able to compress documents written by author A better than documents from other authors? The KRIMP algorithm is in fact not the first compression algorithm to be applied to the task. In general, compression-based approaches compress the author corpora using a particular compression algorithm that builds a model or dictionary from the text. The disputed text is then compressed by each of the models and classified to the one yielding the best compression ratio. From an information-theoretic perspective, the compression rate can be seen as an approximation of the cross-entropy between the documents. In contrast to most other attribution methods, compression methods are normally character based.

Nagaprasad et al. [28] and Oliveira et al. [29] test different combinations of compression distance measures and compression algorithms. These are measures defined for general compressors. Some of the most common in the literature are:

Normalized Compression Distance [39]

$$NCD(x,y) = \frac{C(xy) - min(C(x), C(y))}{max(C(x), C(y))}$$

where $C(xy)$ denotes the compressed size of the concatenation of x and y, and lower values indicate higher similarity.

Compression-based Dissimilarity Measure [17]

$$CDM(x,y) = \frac{C(xy)}{C(x) + C(y)}$$

where $C(xy)$ denotes the compressed size of the concatenation of x and y, and the CDM is close to one if x and y are not related and lower otherwise.

Conditional Complexity of Compression [29]

$$CCC(y|x) = |S_c| - |X_c|$$

where $|S_c|$ is the compressed size of xy in that order and $|X_c|$ the compressed size of x. This quantity measures how the compressor adapts to y after compressing x and is an approximation to conditional Kolmogorov complexity.

Well known compression algorithms are used: ZIP, BZIP, GZIP, LWZ, PPM and PPMd. These algorithms are very diverse in the way they build dictionaries or models of the data. LWZ, for example, builds the dictionary in place, considering the input document as a single string, while BZIP compresses data in fixed size blocks. They are also applied at the character level. The word level and other transformations of the input are not considered directly, although compressors might very well pick words in their dictionary. In none of the papers, however, results achieved by the compressors are directly compared against simpler distance measures or classifiers like Naive Bayes. Oliveira et al. [29] do compare against an SVM approach on the same dataset, with an advantage for the compressors. They conclude that the performance of compression distances are highly dependent on the attribution method: profile-based or instance-based. In the profile-based approach documents by an author are aggregated into an author's corpus, and disputed texts are assigned to the author with the greatest similarity. In the instance-based approach the disputed document is compared with each document in the trainings set, and dissimilarities are aggregated by using a rule such as, for example, voting.

Khmelev and Teahan [21] present the R-measure, a repetition-based measure consisting of "a normalized sum of lengths of all substrings of the document that are repeated in other documents of the collection". An approximation of the R-measure is calculated efficiently using the suffix array data structure. Although not precisely a compression approach, the experiments benchmark the R-measure against several compressors in an authorship attribution task. Although it delivers competitive results, superior to SVMs (in a one-vs-rest approach), the RAR compressor was found to be the best-performing algorithm, outperforming other compressors like BZIP2 and GZIP by a large margin, and SVMs by a smaller margin. Experiment were run on a corpus of news data with 50 authors. This is consistent with results by V. Kukushkina et al. [36], where the RAR and specially RARW algorithms delivered the best accuracy among 14 other general compression algorithms. In this case the algorithms were applied to a corpus from 82 Russian writers. The RAR algorithm has also been shown to be the one delivering the best results for general text categorization [25].

4.3 The Algorithm

Algorithm 2, used for the authorship attribution problem is virtually identical to the KRIMP classifier proposed by van Leeuwen et al. [37]. The process consists in compressing every one of the classes (author corpora) to obtain a code table per corpus. Disputed texts are then compressed with every one of the code tables. The lower the compressed size, the greater the likelihood of the author corresponding to the code table being also the author of the disputed text. The ranked list of compressed sizes is returned as result.

As explained in section 3.4.1, cross-compression sizes can be regarded as generalizing the concept of cross-entropy. The compression term, analogous to entropy, is not calculated as it is constant for a given disputed text (ie. it depends only on the database). Note that running the KRIMP classifier with no candidate itemsets (a code table with only the alphabet) results in the Kullback-Leibler divergence classifier, which is in turn equivalent to a Naive Bayes classifier.

Classifying based on the compressed size of the author corpora using the code table from the disputed text is also a valid way to classify the database. This is however disregarded as the variance of this value tends to be much larger, due to generating a code table from a normally much smaller database.

```
Data: A set of $N$ author databases $\{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_N\}$ and a disputed text database $\mathcal{D}$
Result: List of dissimilarities $\{d_1, d_2, ..., d_N\}$, where the lowest score indicates the most
        likely author.
for $i \leftarrow 1$ to $N$ do
    compress author corpus: $CT_i = KRIMP(A_i) \forall i \in 1, 2, ..., N$;
    encode disputed text with $CT_i$: $d_i = CT_i(\mathcal{D})$;
end
```

Algorithm 2: The KRIMP database classification algorithm used for Authorship Attribution algorithm. It assigns a text to its most likely author, according to its one-way cross-compression size.

4.4 Attributing prose: Victorian authors

The objective of a first experiment was to determine whether there is useful structure in prose that can be exploited by the KRIMP algorithm for the attribution task. It was decided to use data which is freely available to test the algorithm's performance. However, data for authorship attribution studies has to meet certain requirements in order to provide useful results:

- The corpus must be large enough and contain enough authors to be challenging.
- Every document must be attributable to a single author. Documents which have multiple authors or are heavily edited, like Wikipedia pages, are not suitable.
- The type of the works must be uniform. It makes little sense to compare poems to novels.
- All documents must ideally be written around the same time. Books written in different periods can certainly be compared but there may be big biases due to how languages evolve over time, and these are not representative of most real attribution problems.
- Ideally all documents must treat the same topic, in order to avoid biases due to topic-specific vocabulary.

Given that it was not possible to find standard datasets for the problem which are freely accessible and meet these requirements, it was decided to use books part of the Gutenberg Project, as done by Ke Selj et al. [16]. A corpus of 289 novels and short stories from 19 authors from the Victorian period was harvested from the Gutenberg dataset [22]. Books where chosen spanning the complete extent of the authors' careers.

Since the KRIMP algorithm has no parameters that must be tuned, a test set is not always a requirement. However, given that manual tests had to be performed regarding the best preprocessing steps, the data was partitioned, separating a small set with less authors for these experiments. Given that the number of books per author was highly dissimilar, some additional books were removed randomly from authors having more than 12 books in the corpus. Two authors were removed due to their corpus being too small. The details of the resulting test with 156 books from 17 authors are shown on table 4.1.

Table 4.1. Characteristics of the text corpus used for the Victorian novelist attribution experiment. Paragraph and sentence counts include paragraphs/sentences within quotes. Token counts include only words and punctuation characters not within quotes.

Author	Books	Paragraphs	Sentences	Tokens
Benjamin Disraeli	12	24980	79961	1097775
Bram Stoker	4	5150	22538	294020
Charles Dickens	12	44438	115750	1606135
Charles Kingsley	8	19121	52953	801464
Charlotte Bronte	4	13819	38085	521092
Charlotte Mary Yonge	12	27073	66970	1008466
George Bernard Shaw	4	7085	21470	152705
George Eliot	8	12987	43649	854121
Henry James	12	22653	70234	824482
Henry Rider Haggard	12	19269	54284	822761
Herbert George Wells	12	18503	62581	764728
Joseph Conrad	12	15843	75446	834425
Lewis Carroll	4	2367	5973	73034
Robert Louis Stevenson	12	14591	44809	657645
Rudyard Kipling	12	13677	46282	500121
Sir Arthur Conan Doyle	12	16290	53126	606940
William Makepeace Thackeray	4	5736	25123	465791

4.4.1 Preprocessing and data representations

In preprocessing, the complete set of books was first preprocessed as follows:

- Titles were removed, as these do not constitute sentences, which are tested later on to define itemsets.
- Glossaries, indexes, prefaces, appendices, citations, bibliography, and non-prose blocks of text were removed. The beginning and ending of every book was manually checked for these sections.

- Quote and dash use was normalized, as there are differences among the original books in the dataset in the use of single and double quotes and the em-dash.
- Text between quotes was removed, as it does not in most cases reflect the regular writing style of the author. The quote character was left, as the use of quotes is considered part of the writing style of the author.

After this step the documents consist of prose text, preserving paragraphs and sentence division. The Stanford CoreNLP natural language software [24] was used to accurately identify sentence boundaries, as not every period constitutes a sentence boundary. No lemmatization or tagging was done at this stage.

One important question is of course how to form itemsets from prose text. It was decided that division in sentences would constitute a good initial hypothesis. However, preliminary compression tests showed that the itemsets generated from sentences are too long to be able to compress at a reasonably low minimum support when allowing duplicate tokens, that is, when successive appearances of a token are encoded as different itemsets. This was true specially for the n-gram representation, but also for the word representation, even when allowing only function words, since many of these tend to appear several times per sentence. The KRIMP algorithm was however able to mine at a reasonable support when ignoring multiple appearances of a token in the same sentence (ie. only keeping one item).

Sentence separation is feasible but has two extreme characteristics: a very large alphabet and very long itemsets. Some sentences in the text are more than 200 tokens long, and the complete alphabet consisted of more than 40000 words. To test the performance of KRIMP under radically different circumstances, that would at the same time allow to mine at lower minimum supports, it was decided to perform a second experiment, reducing both the size of alphabet and the length of the itemsets by partitioning sentences into multiple itemsets based on punctuation. The further details and preprocessing steps particular to each of the two experiments are explained in the next sections.

4.4.2 Looking for structure in sentences

In this first experiment, one itemset is created for each sentence, containing all the words and punctuation in the sentence. If a word appears multiple times in one sentence only one item was kept. Only personal pronouns were removed, since, it was reasoned, the relatively small number of books from some authors can cause biases due to the choices of voice or narrator (eg. first person vs. third person). All proper nouns were replaced by a single token to avoid biases by choice of specific names. In this way the use of proper nouns as a stylistic choice is not lost. Numerals were replaced in the same way. Large author corpora were sampled down to a maximum of 20000 lines in an attempt to avoid extreme differences in the size of the corpora.

4.4.2.1 Results

Tables 4.2 and 4.3 contain the results of the sentence compression experiment. KRIMP performs slightly worse than the KL divergence algorithm, and no clear trend can be established when varying the database size. The confusion matrix indicates small differences in the algorithms' decisions. Two of the authors do not have assignments. The particularity of these authors is that they had small corpora. Even though the largest corpora were being sampled down to a maximum of 20000 lines, the differences, probably combined with particularities in the distribution of the authors, produced this effect. The large alphabet and the use of Laplace smoothing could have contributed to the situation.

Table 4.2. Results of the experiment of sentence compression on the Victorian dataset. Results show slightly worse performance from KRIMP on the authorship task. The minimum frequent itemset mining support for this experiment was set at $\theta = 0.0025$.

Itemsets	Documents	Items/doc	KLD	KRIMP
2000	171	34655	0.9591	0.9298
1750	208	30558	0.9567	0.9471
1500	255	26182	0.9647	0.9451
1250	323	21816	0.9659	0.9474
1000	417	17344	0.9496	0.9353
750	585	13025	0.9487	0.9402
500	917	8658	0.9433	0.9378
250	1908	4324	0.9340	0.9245

Table 4.3. Confusion matrix for the sentence compression experiment on the Victorian dataset, for sub-documents of 250 sentences. Shown are classifications by the KRIMP (left) and KLD / Naive Bayes algorithm (right).

	BD	BS	CD	CK	CB	CMY	GB	GE	HJ	HRH	HGW	JC	LC	RLS	RK	SACD	WMT
BD	177|177	0|0	0|0	1|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|1
BS	0|0	29|27	3|4	2|2	0|0	1|1	0|0	2|4	0|0	10|12	0|0	0|0	0|0	0|0	2|2	5|2	0|0
CD	0|0	0|0	246|248	0|0	0|0	1|0	0|0	0|0	0|0	1|2	0|0	0|0	0|0	1|1	0|0	1|1	3|1
CK	0|0	0|0	0|0	102|102	0|0	0|0	0|0	1|1	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0
CB	0|0	0|0	0|0	0|0	76|78	0|0	0|0	2|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	1|1
CMY	0|0	0|0	0|0	0|0	0|0	130|130	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0
GB	0|0	0|0	18|16	0|0	0|0	7|9	0|0	4|3	0|1	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0
GE	0|0	0|0	0|0	0|0	0|0	0|0	0|0	108|108	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0
HJ	0|0	0|0	0|0	0|0	0|0	0|2	0|0	0|0	137|135	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0
HRH	0|0	0|0	0|0	0|0	0|0	0|0	0|0	2|2	0|0	123|123	0|0	0|0	0|0	0|0	0|0	0|0	0|0
HGW	0|0	0|0	2|0	1|0	0|0	0|0	0|0	1|1	0|0	2|1	143|148	1|2	0|0	1|0	1|0	1|1	0|0
JC	0|0	0|0	0|0	0|0	0|0	0|0	0|0	2|3	0|0	0|0	0|0	163|162	0|0	0|0	0|0	0|0	0|0
LC	0|0	0|0	10|11	1|0	0|0	0|0	0|0	1|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	1|2
RLS	0|0	0|0	2|3	1|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	107|107	0|0	0|0	0|0
RK	0|0	0|0	0|1	3|2	0|0	0|0	0|0	1|1	0|0	1|1	0|0	0|0	0|0	0|0	94|91	1|0	2|6
SACD	0|0	0|0	2|3	0|0	0|0	1|0	0|0	0|1	0|0	1|0	0|0	0|0	0|0	1|0	0|0	99|101	2|1
WMT	0|0	0|0	13|6	2|0	0|0	8|12	0|0	8|0	0|0	0|0	0|0	0|0	0|0	0|0	2|0	0|0	30|45

An important advantage of using compression is that the compression ratio can be used as an indication of how much structure is being captured by KRIMP, which is a good indication of the degree of structure in the database. The average compression ratio for the author databases during the 10 folds of the experiment was 0.867 ± 0.017. This is a high but not insignificant ratio. It is higher than the ratios observed when compressing itemset databases used for research, as reported in [40], but it must be kept in mind that a minimum support of 0.0025 is a high, though also necessary one.

One of the advantages of the KRIMP algorithm is its interpretability, since the model generated is a succinct code table, with normally only hundreds or a few thousand itemsets, besides the alphabet. Looking at code tables can help acquire an idea of the kind of structure that KRIMP is compressing in the code table. Table 4.4 shows the top 15 itemsets by area and by usage count in the code table of one of the authors during one fold of the experiment. Note however that the information that is compressed the best is not necessarily the most important for classification. In this case it can be seen that segments containing function words and proper nouns are being compressed well. These are most likely descriptions of dialogs, which are very common in general in the dataset and have a significant amount of repetition. As expected many singletons appear among the top 15 itemsets by usage. No dependency between these singletons is strong enough to appear before them in the code table. However, many common combinations of function words appeared deeper in the code table.

TABLE 4.4. Top 15 itemsets ordered by area (left) and usage count (right) of a code table for author George Bernard Shaw during the sentence compression runs. Percentages are with respect to the total area and the sum of code table itemset usage counts respectively. PPN is the substitution token for proper nouns.

Area-ordered		Usage-ordered	
, NNP “ said	1.73%	NNP “ said	0.65%
NNP “ said	1.38%	, NNP “ said	0.61%
, “ said	0.68%	in	0.47%
, NNP the and to a of in	0.50%	was	0.47%
, NNP the and to a of with	0.44%	for	0.41%
, NNP the and to a of in was that	0.44%	NNP	0.40%
, NNP the and to a of	0.43%	as	0.37%
“ said	0.38%	with	0.37%
, NNP the and a of in	0.36%	by	0.33%
in	0.34%	, “ said	0.32%
, NNP the “ said	0.33%	at	0.32%
was	0.33%	a	0.31%
, NNP the and to of	0.32%	the	0.31%
, NNP the and to of in	0.31%	that	0.30%

4.4.3 Compressing word relations

The next experiment on the Victorian authors dataset made use of function word items only. Furthermore, sentences were divided at punctuation characters in order to get shorter itemsets. Itemsets consisted then on the function words between two punctuation characters. It was hypothesized that these short sentence fragments could contain enough differentiating structure for KRIMP to leverage into improved results. Function words were identified by part of speech using the POS tagger [24], rather than using a list or a reference distribution. All pronouns, nouns, verbs, adjectives and adverbs were removed, as were punctuation, symbols and numerals.

4.4.3.1 Data representation

The documents, after being preprocessed as explained in section 4.4.1, were split at the punctuation characters: ",",";",":",".","—","?","!". Any resulting itemsets of length 2 or shorter were removed from the database. Note that the delimiting punctuation characters themselves were not included in the itemsets, nor any differentiation is made in this regard. Large author corpora were sampled down to a maximum of 20000 lines.

4.4.3.2 Results

Table 4.5 shows the results for several document lengths (after all pre-processing). These appear to indicate that KRIMP does not perform any better than the KL divergence when using this technique. The KL divergence delivers surprising results for such a simple measure. KRIMP was specially sensitive to the smallest document sizes.

Table 4.5. Results for the attribution experiment in sub-documents of different lengths extracted from books written by Victorian authors. Itemsets were created by splitting at punctuation characters. Itemsets were mined for a relative support of $\theta = 0.0001$.

Itemsets	Documents	Items/doc	KLD	KRIMP
2000	208	10369.28	0.7483	0.7587
1750	246	9147.45	0.7343	0.7238
1500	299	7792.09	0.7413	0.7378
1250	373	6584.52	0.7552	0.7413
1000	490	5201.51	0.7343	0.7343
750	679	3893.91	0.7343	0.7063
500	1056	2535.79	0.6573	0.6224
250	2187	1315.82	0.6049	0.4895

Table 4.6 presents the confusion matrix for the experiment with databases of length 1000, for both KRIMP and the KL divergence. The results indicate that using KRIMP causes the

classifier to completely mis-classify documents from some of the authors. Taking itemsets into account reduced the discrimination power of the classifier.

TABLE 4.6. Confusion matrix for the Victorian authors attribution experiment using function words and itemsets defined at punctuation characters, for sub-documents of length 1000. Shown are classifications by the KRIMP (left) and KL divergence algorithm (right).

	BD	BS	CD	CK	CB	CMY	GB	GE	HJ	HRH	HGW	JC	LC	RLS	RK	SACD	WMT
BD	58|55	0|0	0|0	0|0	0|1	1|1	0|2	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	1|1
BS	1|1	7|16	0|0	0|0	1|0	0|0	0|0	1|1	0|0	3|0	0|0	0|0	0|0	1|0	0|0	4|0	0|0
CD	10|15	1|1	54|40	1|6	14|19	2|3	0|1	0|2	0|0	3|0	6|1	0|0	0|0	2|1	0|0	0|1	0|3
CK	0|0	0|0	0|0	42|39	0|0	1|2	0|0	0|2	0|0	4|2	0|0	0|0	0|0	0|0	0|0	0|2	0|0
CB	1|0	0|0	1|1	0|0	16|21	3|1	0|0	3|6	0|0	0|0	0|0	0|0	0|0	5|1	0|0	0|0	1|0
CMY	4|5	0|3	2|0	0|1	0|3	42|40	0|0	0|0	0|0	4|0	0|0	0|0	0|0	2|1	0|0	0|0	0|1
GB	0|0	0|0	1|0	0|0	0|0	4|0	0|6	1|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0
GE	5|2	0|1	0|1	0|0	0|0	0|1	0|0	41|41	0|0	2|1	0|0	0|0	0|0	1|0	0|0	0|0	0|2
HJ	2|0	1|4	0|0	0|0	1|1	2|3	0|0	1|1	29|30	0|0	0|0	0|0	0|0	2|0	0|0	1|0	0|0
HRH	2|6	0|1	0|0	0|2	0|0	3|4	0|0	0|0	0|0	40|31	0|0	0|0	0|0	0|1	0|0	1|1	0|0
HGW	0|1	0|1	0|0	0|0	0|1	1|0	0|5	0|0	3|0	4|1	27|25	0|0	0|0	2|4	0|0	1|0	0|0
JC	0|0	0|0	0|0	0|1	0|3	0|0	0|0	0|1	1|1	0|0	3|0	36|32	0|0	2|3	0|1	1|1	0|0
LC	0|0	0|0	0|0	1|1	0|0	2|0	0|0	0|0	0|0	0|0	0|0	0|0	0|2	0|0	0|0	0|0	0|0
RLS	2|3	0|0	2|1	2|2	0|4	1|0	0|0	0|1	0|0	0|0	3|3	0|0	0|0	25|20	0|2	1|0	0|0
RK	0|0	0|0	0|0	3|3	0|0	0|0	0|0	0|0	0|0	1|0	2|0	0|0	0|0	1|1	14|19	2|0	0|0
SACD	0|2	0|1	1|0	0|2	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	1|0	0|1	31|27	0|0
WMT	8|1	0|0	4|0	2|0	0|0	2|0	0|0	5|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	5|25

The compression ratio of the author corpora for this experiment was 0.84 ± 0.02, only slightly lower than that of the previous experiment. This indicates that slightly more structure is indeed being used in the classification process. However, this is again higher than compression ratios reported in [41] for classification databases. In other words, there is only little compressible structure when splitting sentences and reducing the alphabet to function words. It is an indication that the amount of structure present at the word level, even in heavily preprocessed text containing only function words, is relatively low. Table 4.7 shows the top 15 code table itemsets by area and by usage.

4.4.4 The effect of smoothing

Given the significantly better results when using complete alphabets, in this section the comparison is taken further by analyzing a different way of applying smoothing. An interesting finding was that applying the Laplace correction or smoothing in a different way yielded the best results observed for this experiment. In this experiment, instead of defining a global alphabet that is used for all databases and applying correction to this alphabet (or the full code table in the case of KRIMP), the alphabets of the individual author databases are considered separately. The alphabet is completed by inserting only the elements needed to compress the disputed text and then applying a Laplace correction. In other words, the correction is a applied on a case-by-case basis.

TABLE 4.7. [Top 15 itemsets ordered by area (left) and usage count (right) of a code table for author George Bernard Shaw during the function word experiment runs. The number to the left of the bar indicates the occurrence. For example, "0|on" represents the first occurrence of "on" in the itemset; "1|on" the second one; and so on. Percentages are with respect to the total area and the sum of code table itemset usage counts respectively.

Area-ordered		Usage-ordered	
0|on	1.30%	0|on	2.08%
0|by	1.24%	0|by	1.99%
0|as	1.22%	0|as	1.95%
1|a	1.18%	1|a	1.89%
0|the 0|of 0|and	1.18%	1|of	1.88%
1|of	1.18%	0|an	1.86%
0|an	1.16%	0|at	1.75%
0|the 0|of 0|to 0|for	1.12%	0|which	1.66%
0|at	1.10%	0|with	1.62%
0|the 0|to 0|for	1.07%	0|from	1.56%
0|which	1.04%	1|the	1.51%
0|with	1.02%	0|who	1.38%
0|the 0|and	0.98%	1|and	1.32%
0|from	0.98%	0|but	1.31%

Although this approach is not the one used normally for Naive Bayes classifiers, it is arguably valid from the viewpoint of information theory. If the alphabet is considered to be unknown, this constitutes a less biased measure of cross-entropy, as elements not appearing in any of the two databases do not increase the code sizes for the rest (ie. the correction is less dramatic, specially when the global alphabet is large). General compression algorithms are generally applied on a case by case basis in the literature, without correcting for a global alphabet [25]. This is not necessary for some of them because the models used by compression algorithms are generally built while the algorithm runs, and can be simply completed when new symbols appear. KRIMP has a clearly defined training phase, after which its model is final.

Applying the algorithm in this way does not come without its problems. Concisely, the opposite effect may appear: the alphabet of small author corpora containing general terms and few uncommon vocabulary will be completed with few new terms, and is thus likely to compress databases from other authors very well. Table 4.8 shows the results obtained when compressing in this way. These were globally the best results obtained for the Victorian dataset. The results revealed that most mis-classifications by the KL divergence algorithm were to the author with the smallest corpus. Books from this author also happened to contain an above-average amount of dialog, and sections with significant dialog from other books were thus mis-classified. KRIMP was less susceptible to this because, as shown by the analysis of the code tables, sentences containing dialog are in general compressed very well. This is due to the repetition of phrases and punctuation that introduce dialog, and the common use of quotes.

Table 4.8. Results of the sentence compression experiment on the Victorian dataset when alphabets are completed on a case by case basis. Results show better performance from KRIMP on the authorship task. The minimum frequent itemset mining support for this experiment was set at $\theta = 0.0025$.

Itemsets	Documents	Items/doc	KLD	KRIMP
2000	171	34655	0.9649	0.9942
1750	208	30558	0.9375	0.9952
1500	255	26182	0.9255	0.9961
1250	323	21816	0.9195	0.9969
1000	417	17344	0.8968	0.9928
750	585	13025	0.8598	0.9915
500	917	8658	0.8037	0.9858
250	1908	4324	0.7264	0.9738

4.4.5 Using less data

A final experiment was performed making use of less training data. The objective was both to determine the performance of KRIMP with significantly less information, and to perform an experiment where the sizes of the author corpora were completely balanced, to eliminate the adverse effect of smoothing on the Naive Bayes classifier seen in the last experiment. To do it each author's corpus was limited to 2500 lines, sampled without replacement from their original corpus. For the 4 folds of the experiment every original corpus contained more than 2500 lines, meaning that every sampled corpus had exactly 2500 lines. The Laplace correction was applied case-by-case, as in the previous experiment.

Table 4.9. Results of the sentence compression experiment on the Victorian dataset when author corpora are sampled to 2500 lines. Results show better performance from KRIMP, confirming that KRIMP indeed provides an advantage. The minimum frequent itemset mining support for this experiment was set at $\theta = 0.0025$.

itemsets	Chunks	Items	KLD	KRIMP
2000	171	34655	0.9766	0.9883
1750	208	30558	0.9663	0.9856
1500	255	26182	0.9725	0.9843
1250	323	21816	0.9655	0.9876
1000	417	17344	0.9664	0.9832
750	585	13025	0.9504	0.9761
500	917	8658	0.9258	0.9727
250	1908	4324	0.8732	0.9418

Table 4.9 shows the results. From it is clear that KRIMP is superior in accuracy when considering the complete alphabet. Naive Bayes does improves with less data, confirming that the use of Laplace smoothing was indeed affecting its results. However, KRIMP still performs about 7% better for the smallest documents. The reduction in the performance for KRIMP is relatively minor with respect to the previous experiment with more data.

The previous results are an indication that KRIMP is capturing relevant structure information. One explanation as to why KRIMP is an improvement comes from the analysis of code tables in the light of the *writeprint* hypothesis [23], which states that lexical and syntactic features (among others not so prominent in this particular dataset) are of utmost importance in capturing a personal style of writing. This idea is not new but stems from years of research in the field.

Syntactic features are mainly function words and punctuation, and it is these features that KRIMP joins into code table elements. In other words, KRIMP compresses our preferred structural choices when putting together sentences. What the results indicate is that the combination of these choices at the sentence level are more telling than assuming independence.

Lexical features, understood as more uncommon words that characterize the breadth of an author's vocabulary, are less likely to form itemsets, due to their sparseness, and hold power in isolation. KRIMP code tables can then be regarded as a combination of high-level syntactic features in the form of itemsets that capture common choices of the author in structuring sentence or sentence fragments, and individual lexical features. Their weights (code sizes) are balanced in a way that is governed by the MDL principle.

Table 4.10 shows the corresponding confusion matrix for document lengths of 250. Many mis-classifications by the KLD algorithm are to the corpus of Lewis Carroll, the one containing the greatest amount of dialogue. The algorithm is susceptible even when the size of the training sets are balanced. KRIMP incurs less frequently in this type of error, due to its ability to combine items into itemsets more characteristic of author style.

Table 4.11 shows a review of all the experiments performed for a document length of 250 and the complete alphabet. In previous experiments, the same features / items were input to both algorithms. To test whether counting only first occurrences of items in sentences was affecting the accuracy of Naive Bayes, the standard approach of counting every item was tested as well (indicated by KLD*). Interestingly, both accuracies are practically the same.

4.5 Attributing songs

The second experiment was run on data with, *a priori*, much more structure: songs. The objective was to test whether there exists significant structure in the author's choice of function words, or in the n-gram representation of the song, that KRIMP can utilize to improve attribution accuracy.

Table 4.10. Confusion matrix for the Victorian authors attribution experiment with one itemset per sentence and corpora of length 2500, for sub-documents of length 250. Shown are classifications by the KRIMP (left) and KL divergence algorithm (right).

	BD	BS	CD	CK	CB	CMY	GB	GE	HJ	HRH	HGW	JC	LC	RLS	RK	SACD	WMT
BD	172|173	2|1	1|0	0|0	0|0	0|0	1|2	0|0	0|0	0|0	0|0	0|0	0|0	1|2	0|0	0|0	1|0
BS	0|0	52|52	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	1|1	0|0	1|1
CD	0|0	2|3	216|204	0|0	0|0	1|1	0|5	3|0	7|10	6|5	0|0	0|0	0|20	10|3	1|0	1|1	6|1
CK	0|0	1|4	0|0	100|84	0|0	0|0	0|0	0|0	0|0	0|4	0|0	1|1	0|4	0|0	0|4	1|2	0|0
CB	0|0	3|4	0|1	0|0	73|64	0|0	0|0	2|1	0|7	0|0	0|0	1|0	0|0	0|1	0|0	0|0	0|1
CMY	0|0	0|1	0|0	0|0	0|0	130|124	0|1	0|0	0|0	0|1	0|0	0|0	0|1	0|2	0|0	0|0	0|0
GB	0|0	0|0	0|0	0|0	0|0	0|0	28|29	1|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0
GE	0|0	2|8	0|0	0|0	0|0	1|0	0|0	105|91	0|7	0|0	0|0	0|0	0|1	0|0	0|1	0|0	0|0
HJ	0|0	0|0	0|0	0|0	0|0	0|0	0|1	0|0	137|134	0|0	0|0	0|0	0|2	0|0	0|0	0|0	0|0
HRH	0|0	3|4	0|0	0|0	0|0	0|0	0|0	0|0	0|0	122|121	0|0	0|0	0|0	0|0	0|0	0|0	0|0
HGW	0|0	6|14	0|0	0|0	0|0	0|0	1|4	0|0	2|10	0|1	132|100	3|2	3|19	0|0	3|0	3|3	0|0
JC	0|0	1|5	0|0	0|0	0|0	0|0	0|0	0|0	0|13	0|0	0|0	162|143	0|1	0|0	2|3	0|0	0|0
LC	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	11|11	0|0	2|2	0|0	0|0
RLS	0|0	0|0	0|0	0|0	0|0	0|0	0|3	0|0	0|0	0|0	0|0	0|0	0|2	110|105	0|0	0|0	0|0
RK	0|0	2|1	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|1	0|0	0|0	3|17	0|0	90|82	0|0	7|1
SACD	0|0	6|9	0|2	0|0	0|0	0|0	0|1	0|0	1|0	1|1	0|0	0|0	0|1	0|0	2|4	95|88	1|0
WMT	0|0	0|0	0|2	0|0	0|0	1|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	0|0	62|61

Table 4.11. Summary of experiments on the Victorian authors dataset for a document length of 250. KLD* corresponds to a Naive Bayes classifier trained using the complete word counts (as normally) rather than counting only the first occurrence in a sentence.

Algorithm	Max corpus length	Global alphabet	Accuracy
KLD	20000	yes	0.9340
KRIMP	20000	yes	0.9245
KLD	20000	no	0.7285
KRIMP	20000	no	0.9738
KLD*	2500	no	0.8732
KLD	2500	no	0.8721
KRIMP	2500	no	0.9418

4.5.1 The data

The corpus of poems used in this experiment was made publicly available by the authors of the book *Van wie is het Wilhelmus?* (Dutch for *Who wrote the Wilhelmus?*) [20], which presents the results of a computational analysis on the authorship of the national anthem of The Netherlands.

The corpus consists of songs from seven different authors, some of which are considered historical candidates for the authorship of the anthem. The songs were written in the 16th century—close to the time of the first known appearance of the anthem—in the *Geuzenliedboek*, a book compiling songs printed in 1577 or 1578. Most of the songs in the book are unattributed. The data contains unprocessed transcriptions of the original books in which songs were published. Some of these songs are also available in lemmatized and POS-tagged form. POS-tags for this subset were manually corrected by the book's authors for mistakes made by the tagger. Statistics for the original and lemmatized and tagged datasets are given in the following sections.

4.5.2 Looking for differentiating structure in n-gram itemsets.

The first experiment on song data made use of n-grams. This experiment was performed to determine the ability of KRIMP to capture structure present in the n-gram representation. Given that the songs have rhyme and a more uniform structure than prose, and also generate more constant itemset lengths by simply partitioning per verse, the n-gram representation seemed like a good fit.

Specifically, n-grams forming part of common words are expected to be joined in the KRIMP code tables, while others with less strong dependencies, possibly expressing more general style choices of the author in word choice, rhyme and punctuation, would be left alone.

Note however that the n-gram representation would not be used in an actual attribution study for these works; there are spelling differences among the books from which the authors' songs were extracted that are of unknown origin. These could have been introduced by editors rather than by the original song writer. However, for the purpose of determining whether KRIMP provides an advantage when using the n-gram representation, the differences were considered acceptable.

For this experiment, it was decided to use 3-grams, as this size has been found to be among the most effective [7] and it keeps the alphabet size in check. To further reduce itemset size, 3-grams where filtered by removing *whole-word*, *mid-word*, and *multi-word* 3-grams (see section 4.2.1), which where found in Sapkota et al. [32] to decrease accuracy in authorship attribution, while accounting for a large portion of the total n-grams.

Table 4.12 shows the details of the original and processed n-gram dataset, which totaled 827 songs from 6 authors. The reduction in item count after removing non-word n-grams was 35% on average.

TABLE 4.12. Characteristics of the dataset used for the n-gram experiment on the Dutch songwriter corpus. Values are totals for each author's corpus. The token count corresponds to splitting the original text on the space character. The n-gram count includes all n-grams (approximately one per character) while *non-word n-grams* counts only those used for the experiment.

	Songs	Lines	Lines/song	Tokens	n-grams	non-word n-grams
coornhert	108	4859	44.99 ± 39.19	30563	179396	114508
datheen	155	9071	58.52 ± 51.04	55443	324424	205961
fruytiers	132	5031	38.11 ± 19.90	30032	171192	109487
haecht	221	11,057	50.03 ± 39.87	61206	357122	244998
heere	42	3010	71.67 ± 147.79	20960	121593	77908
marnix	169	10329	61.12 ± 51.05	64337	371486	240806

4.5.2.1 Data Preprocessing and Representation

The data was preprocessed as follows:

- Songs were extracted from the original transcriptions, preserving capitalization and line breaks but removing accents (to normalize editing differences in accent use).
- A line break character was added at the end of each line of the song. No character was used to separate stanzas.
- Character 3-grams were extracted per line, by applying a sliding window through the song. The 3-grams containing the line break character were kept. These characters were added to better capture the rhyme present in some songs, as well as the choices for starting verses.

4.5.2.2 Results

The authorship attribution experiment consists in measuring the classification accuracy of the algorithm on the corpus of text. All songs were concatenated into author corpora. Preliminary experiments showed that attributing complete songs was not challenging enough, probably in big part due to spelling and edition differences. The classifier on complete songs had 100% accuracy. It was then decided to take shorter song segments, using 10-fold cross-validation. All the songs were first concatenated into author corpora. In one fold of the experiment, the corresponding 10% of the lines are separated for each author and used as test set, after splitting them into documents of length l (number of lines). A document might then contain text from two different songs of the same author.

Table 4.13 shows results of the attribution experiment for different lengths l of the disputed text. Very high accuracies, even for sections of only 20 lines, indicate that author databases are highly dissimilar. This is expected given that every author database comes from a different source, and edition and spelling differences are most certainly playing an important role.

In analyzing the results, it was found that practically all miss-classifications occurred for sections of the corpus by Filips van Marnix, which contains a good amount of French language, in contrast with the other authors. Had the corpus been cleansed of these sections, the accuracy would have been even closer to 100%. It is notable however that the KRIMP is more resilient to this circumstance. It comes as no surprise that documents containing mainly French are assigned to him. However, also many documents containing only Dutch language are correctly assigned to Filips van Marnix, even though his corpus contains a significant amount of French. The KL divergence tends to assign these documents to other authors.

Given the very high accuracies, nothing can be said regarding an advantage of disadvantage from the use of KRIMP. Using even smaller database sizes is both unrealistic (virtually all songs in the dataset had a size of at least 20) and likely to favor the KL divergence due to the reduced variance expected from compressing the same small database using more elements.

Table 4.13. Accuracies of the n-gram experiment on the Dutch song corpus, for different sizes of the disputed documents.

Length (l)	Songs	KLD	KRIMP
20	2380	0.9651	0.9748
30	1584	0.9735	0.9823
40	1184	0.9704	0.9882
50	948	0.9821	0.9873

4.5.3 An authorship attribution experiment

Given that the n-gram representation is not suitable for heavily edited data due to the loss of information and potential bias, a proper authorship attribution experiment in this scenario must either manually normalize spelling and edition (if viable) or make use of lemmatization to extract semantic features like lemma-tag pairs.

Normalizing the spelling is an option when differences stem from mistakes and it is in fact desirable given that it preserves the most stylistic information. However, the lack of a standard Dutch language in the 16th century (a common problem in attribution of old documents), made normalization of this dataset a difficult and questionable choice, as explained by Kestemont et al. [20]. The data was instead lemmatized and POS-tagged, a process which maps words to their modern Dutch base form and part of speech.

In Kestemont et al. [20], lemma-tag pairs were manually corrected after being generated by a memory-based tagger trained on a Middle Dutch dataset. However, not enough data was available in corrected form for an experiment using KRIMP. Given that many of the tagging errors where expected to be random and that an inspection of the tagged data revealed that most of the errors affected uncommon words, it was decided to perform an experiment using the uncorrected tagged data, which formed a significantly larger corpus. Table 4.14 shows the author corpora used along with some statistics about each author's corpus. These values are for the lemmatized and tagged corpus with no feature selection.

4.5.3.1 Data Preprocessing and Representation

Lemma-tag pairs did not have to be extracted for this experiment as the data was tagged for the study by Kestemont et al. [20] and made public in this form. The main question pertained how to select the relevant lemma-tags (alphabet) for the experiment. Some form of frequency based-selection method seemed to be the best option since the tagger produced mistakes specially for less frequent lemma-tag pairs.

It was decided to perform two experiments with different alphabet selection procedures:

TABLE 4.14. Characteristics of the lemmatized and tagged corpus. Itemset and item counts are per song. The number of itemsets corresponds with the number of lines in a song–one itemset was created per line. The number of words corresponds roughly with the number of words in the original poems after removing punctuation.

Author	Songs	Itemsets	Items
coornhert	14	74.79 ± 7.55	584.71 ± 11.52
datheen	35	80.31 ± 13.63	586.94 ± 15.61
fruytiers	33	66.24 ± 24.40	460.36 ± 156.95
haecht	35	95.43 ± 26.55	632.43 ± 136.06
heere	23	68.52 ± 9.60	571.04 ± 9.76
marnix	46	78.43 ± 14.75	577.61 ± 15.50

Top 300 words . The top 300 tag-lemma pairs from the reference corpus are taken as the alphabet. These words are mainly function words, but also include some nouns, adjectives and verbs, which are not always taken as function words.

Top 200 function words . Here the term *function word* is used loosely. The top 300 lemma-tag pairs where filtered to remove nouns, adjectives and verbs, before taking the top 200 pairs as alphabet. This was done in an attempt to eliminate biases caused by nouns, adjectives and verbs linked to the particular theme of the song.

The reference corpus used consisted of the author corpora, along with other songs from time, from the *Geuzenliedboek* and other sources.

4.5.3.2 Results

Tables 4.15 and 4.16 show the confusion matrices for both of the experiments. The complete dataset was used as test set, as there were no model or pre-processing parameterers to adjust. The 10-fold cross-validated results indicate high accuracies for both classifiers, and higher accuracies for the feature selection method taking the top 300 words. The KL divergence algorithm outperforms KRIMP by 2 correct classifications in this first experiment, while in the second one both classifiers have exactly the same accuracy.

Table 4.17 shows a summary of the accuracies obtained. Despite the short length of the songs, both algorithms are able to differentiate the authors with high accuracy. Although it can be said that there is no advantage to using KRIMP, a more expensive algorithm, for this experiment, it is hard to say anything regarding experiments with more candidates. It is possible that the aggregation of items into itemsets in the KRIMP code table increases the differentiation power of the algorithm.

TABLE 4.15. Confusion matrix for the 10-fold cross-validated authorship attribution experiment using as alphabet the top 300 lemma-tag pairs in the reference distribution. The numbers shown are classifications using KRIMP (left) and KL-divergence (right). Both algorithms classify in a very similar way. The accuracy for KRIMP algorithm was 0.935 and for the KL-divergence classifier 0.946. The KRIMP algorithm was run with a relative minimum support of $\theta = 0.002$.

	coornhert	datheen	fruytiers	haecht	heere	marnix
coornhert	14 | 14	0 | 0	0 | 0	0 | 0	0 | 0	0 | 0
datheen	0 | 0	35 | 34	0 | 0	0 | 1	0 | 0	0 | 0
fruytiers	0 | 0	0 | 0	31 | 31	1 | 2	0 | 0	1 | 0
haecht	0 | 0	8 | 4	2 | 2	25 | 29	0 | 0	0 | 0
heere	0 | 0	0 | 0	0 | 0	0 | 0	23 | 23	0 | 0
marnix	0 | 0	0 | 0	0 | 1	0 | 0	0 | 0	46 | 45

TABLE 4.16. Confusion matrix for the 10-fold cross-validated authorship attribution experiment using as alphabet the top 200 lemma-tag pairs corresponding to function words in the reference distribution. The numbers shown are classifications using KRIMP (left) and KL-divergence (right). Both algorithms coincide in most classifications. The accuracy for both algorithms is 0.925. The KRIMP algorithm was run with a relative minimum support of $\theta = 0.002$.

	coornhert	datheen	fruytiers	haecht	heere	marnix
coornhert	13 | 14	0 | 0	0 | 0	0 | 0	0 | 0	1 | 0
datheen	0 | 0	34 | 34	0 | 0	1 | 1	0 | 0	0 | 0
fruytiers	0 | 0	1 | 1	31 | 30	1 | 1	0 | 1	0 | 0
haecht	0 | 0	5 | 4	3 | 4	26 | 26	1 | 1	0 | 0
heere	1 | 0	0 | 0	0 | 0	0 | 0	22 | 23	0 | 0
marnix	0 | 0	0 | 0	0 | 1	0 | 0	0 | 0	46 | 45

TABLE 4.17. Accuracies of the two experiments performed on the lemmatized and tag Dutch songwriter corpus. Accuracies are very high despite the short length of the songs. Taking all words into account, rather than only function words, gives better results for both algorithms. *Items* indicates the average number of items per song, after pre-processing.

Features	Songs	Items	KLD	KRIMP
Top 300 words	186	322.63 ± 59.49	0.9462	0.9355
Top 200 fn words	186	231.08 ± 42.50	0.9247	0.9247

4.5.4 Attributing the *Wilhelmus*

The text of the *Wilhelmus* was attributed using the classifiers by inserting it into the cross-validation set, in such a way that is was classified 10 times.

For the classifier trained on the 300 most common words, in the ten folds, the *Wilhelmus* was attributed to Pietrus Datheen, both by KRIMP and by the KL divergence algorithm, coinciding with the results by Kestemont et al. [20]. This is no surprise considering that the data used for the classifier and the preprocessing steps are very similar (even though the songs included are not exactly the same).

When using the top 200 function words, however, KRIMP and the KL-divergence disagree. While the KL divergence algorithm attributed the text to Pietrus Datheen in all 10 folds, KRIMP attributed the text to Willem van Haecht in 8 out of the 10 folds. The fact that there is a disagreement between these classifiers with very high accuracies, and which agree in the vast majority of classifications hints that the *Wilhelmus* may not be very easily identifiable as creation of a particular author. This is explored further in the next section.

4.5.5 Back to Anomaly Detection

One of the conclusions of the work by [20] was that Pietrus Datheen, a candidate overlooked lately, was the most probable writer of the *Wilhelmus*, based on a computational analysis [19], and historical evidence.

In many attribution studies like the the one on the *Wilhelmus* it is not known with complete certainty whether the real author of a text is within the candidate authors. This is the reason why a different approach to attribution attempts to rather answer the question: how likely is it that author X is the writer of the disputed text? Insight into how normal a particular production is in the work of different candidates may also help validate the assumption made when approaching the attribution problem as a classification problem: the author is within the candidates. Algorithms answering this question may find that a particular work is anomalous in all of the author corpora, shedding doubt into the assumption. This is, however, not always the case as the ability to tell apart a document from those in a corpus certainly depends on the degree of uniformity in the style of the author and the amount of data available.

In this section, this idea is explored using the insights from chapter 3, aiming to get a clearer picture of how well the text of the *Wilhelmus* compares with the writings of the most likely candidates, in isolation. An anomaly detection experiment was run using the complete corpus from these candidates. Such an anomaly detection experiment, as opposed to a classifier, answers a very different question: how normal would the given text be if written by author X?

The experiment was run for the three most probable candidates according to Kestemont et al. [20]: Pietrus Datheen, Willem van Haecht and Filips van Marnix. The objective is not to automatically detect anomalies–the small size of the corpus and of the individual songs rendered this unlikely to work–but to visualize how well the *Wilhelmus* compresses when placed in an

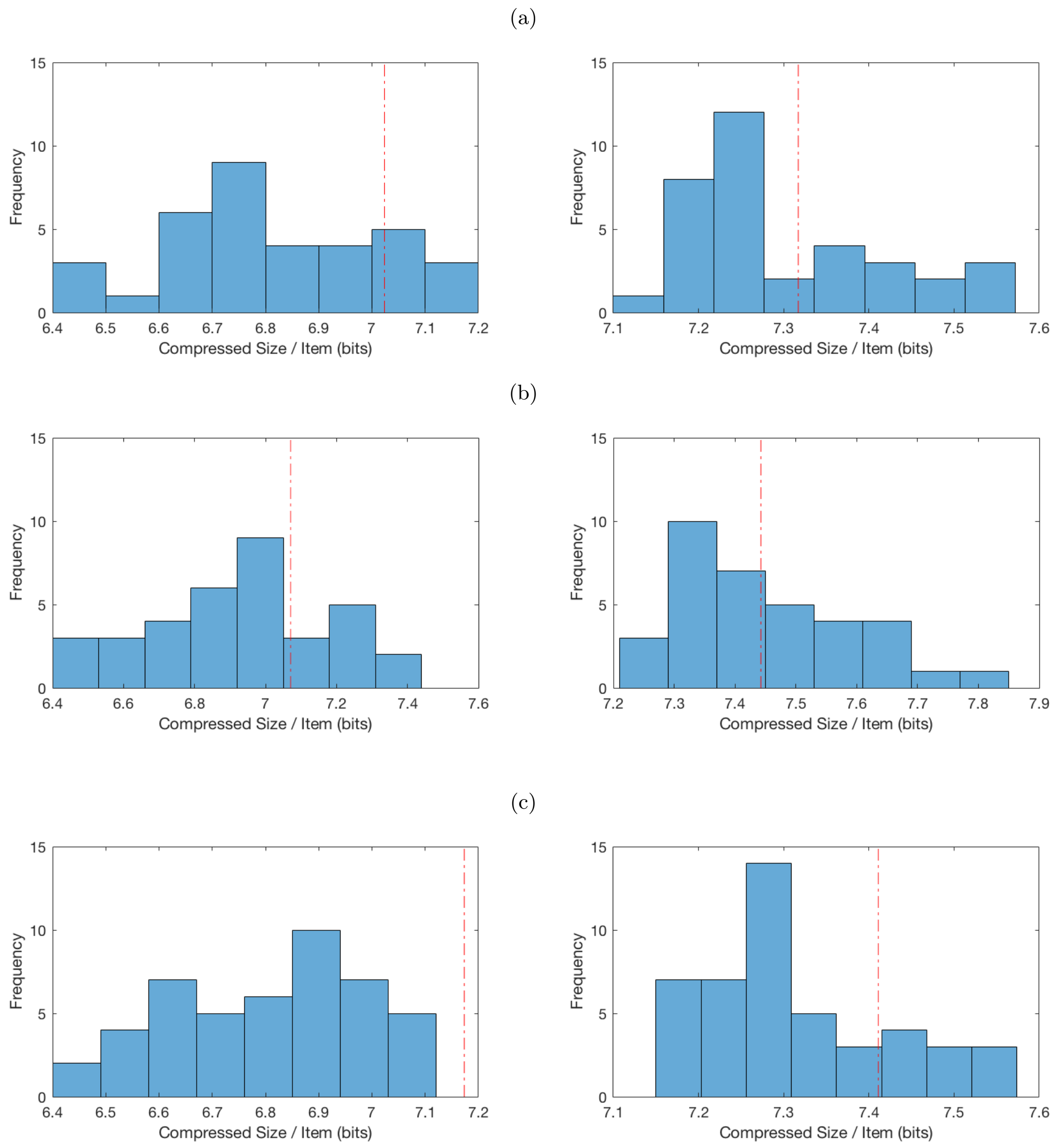

FIGURE 4.1. Histograms of compressed sizes $D(\mathcal{D}_i, joint)$ (left) and $D(joint, \mathcal{D}_i)$ (right), for three authors in the Dutch corpus: (a) Datheen. (b) Haecht. (c) Marnix.

author's corpus, in comparison with other songs from the author. This can be interpreted as visualizing how anomalous the *Wilhelmus* would be if written by a particular author.

The top 300 words in the reference distribution were used for this experiment, as per the first classifier. The KL Divergence was used as distance measure to reduce variance of the compressed sizes as much as possible, given the very small size of the database. Figure 4.1 plots histograms for both distributions of compressed sizes. In this case the observed distribution of results is not close to normal, possibly because of the different structure present in text, the small size of the databases, the small number of observations or a combination thereof.

Results from figure 4.1 tend to support the idea that Filips van Marnix is not the author of the *Wilhelmus*, mainly because its compressed size is the largest among 46 other songs when compressed using the joint distribution–an indication of anomalous behavior. The distribution of $D(\mathcal{D}_i, joint)$ (left) is expected to have a lower variance due to the small size of the individual databases in comparison with the joint database. Although not the most extreme point, the song is also among the ones that compress the joint Marnix corpus the worst.

However, van Haecht's results tend to indicate that the song would not be anomalous within his corpus, as is also the case with Datheen. Not only is the *Wilhelmus* not anomalous in these distributions, but it is also close to average, both when compressing the joint database using the code table obtained from the *Wilhelmus*, as the other way around. These results tend to further indicate that the *Wilhelmus* is not clearly characteristic of any one of these two authors when using the preprocessing and feature selection techniques that were used in this study.

4.6 Conclusions

Experiments using KRIMP for authorship attribution lead to the following conclusions:

- KRIMP captures significant punctuation and function word co-occurrences in its code tables, as indicated by compression ratios and the code tables themselves. Experiments on the Victorian authors dataset indicate strongly that the structure being captured by KRIMP is important for attribution of prose text, as indicated by a consistent advantage over the Naive Bayes classifier. The very similar theoretical grounds of both classifiers result in a very solid comparison. Naive Bayes performed slightly better when smoothing was performed on a global dataset alphabet. However, KRIMP outperformed Naive Bayes consistently when smoothing was applied on a case-by-case basis, with big and with smaller training sets, and for different document lengths tested. This second approach resulted in the globally best results for prose data. The improvement in accuracy from using KRIMP justifies its use in literature over Naive Bayes when running time is not important.

- Code tables reveal that KRIMP is compressing high-level structural choices into itemsets, capturing, in a sense, preferred ways of structuring sentences, or describing dialogs. These high-level structural choices tend to be compressed in itemsets, while rare words are not. Results are compatible with the idea that there is an individual grammar that encodes important information about authorship.

- The KRIMP classifier displayed more robustness with respect to differences in the size and composition of the corpus of different authors. In fact, the best classifier obtained for the Victorian authors dataset made use of KRIMP with heavily imbalanced author corpora. This is due to KRIMP's capacity to combine items which are not particularly characteristic into itemsets which are much more characteristic of a particular author. The same effect was made evident in the Dutch songs experiment when an author's corpus contained songs in both Dutch and French. KRIMP was able to attribute many Dutch songs correctly despite the bias in the corpus. The Naive Bayes classifier struggled in both of these cases.

- Creating one itemset per sentence is a superior approach to splitting sentences at punctuation characters. In fact, looking for structure in the use of function words in sentence fragments resulted in worse performance than simply applying the Naive Bayes classifier. One possible reason for this is that the structure imposed by language grammar itself is too constraining; the freedom left for author choice in putting together sentence fragments, as opposed to full sentences, is limited.

- The function word profile, very prevalent in the literature, falls short for a corpus of the size used. For both KRIMP and Naive Bayes, the use of the complete alphabet delivered clearly superior results.

- When dealing with unseen vocabulary and large alphabets, the way of application of Laplace smoothing severely impacts accuracy of Naive Bayes, but KRIMP showed to be more resilient to this choice even with highly dissimilar sizes of the training corpora. Completing alphabets on a case-by-case basis, as suggested by information-theory-based approaches, rather than defining a global alphabet resulted in better accuracy when using KRIMP.

- Ignoring second occurrences of a token within a sentence does not affect the accuracy of Naive Bayes, and does not severely affect KRIMP either, as indicated by the very high accuracies obtained. This is necessary when compressing literature using KRIMP in order to avoid an explosion in the number of frequent patterns.

- Regarding the attribution of the *Wilhelmus*, the experiments tend to agree with previous work by Kestemont et al. [20] in attributing the text to Pietrus Datheen in most cases. However, KRIMP attributed the anthem to Willem van Haecht when a different feature selection method was used. The anomaly experiments performed on the dataset tend to indicate that the text would not be anomalous in the corpus of any of these two authors. The same experiment showed that the anthem would be much more anomalous in the corpus of Filips van Marnix, the second most likely candidate according to Kestemont et al. [20].

4.7 Future work

Some possible areas for future work identified during the development this thesis are:

- Experiments with songs. The use of KRIMP on songs in chapter 4 did not improve the results for the Dutch songwriter dataset. However, the algorithm remains untested in two regards:
  - Unedited n-gram data. Songs with rhyme seem a natural fit for KRIMP given the high degree of structure of the data. In the experiments, it achieved a high accuracy in distinguishing text from different sources. Even when this was not a proper authorship attribution experiment, the almost perfect accuracy on n-gram data leaves the question open of whether the algorithm would offer an advantage with n-gram data from an unedited source.
  - Full vocabularies. Both of the experiments on song data made use of a reduced set of function words. If the results for prose are an indication, using the complete vocabulary should significantly improve accuracy.
- Comparison with language-model-based approaches, which in general disregard sentence division but capture local dependencies. Some of these approaches also have a Bayesian interpretation [30].
- Experiments using KRIMP for text categorization. Even though many of the algorithms used are the same, this is a fundamentally different problem, mainly because the preprocessing steps are very different, and thus also the data characteristics. While in authorship attribution function words, punctuation and their combination are important, in text categorization mainly less common words and in particular their meanings are of importance. This is an advantage for KRIMP because the removal of function words and punctuation is likely to enable compression at lower minimum supports. It is possible that the combination of words into itemsets increases the discrimination power of a classifier.

APPENDIX A

Appendix A

This appendix includes the histograms, normal distribution fits, Q-Q and P-P plots of the distributions obtained during the class database experiments in chapter 3. The histograms for the *adult* database are included in chapter 3, and are then not included here.

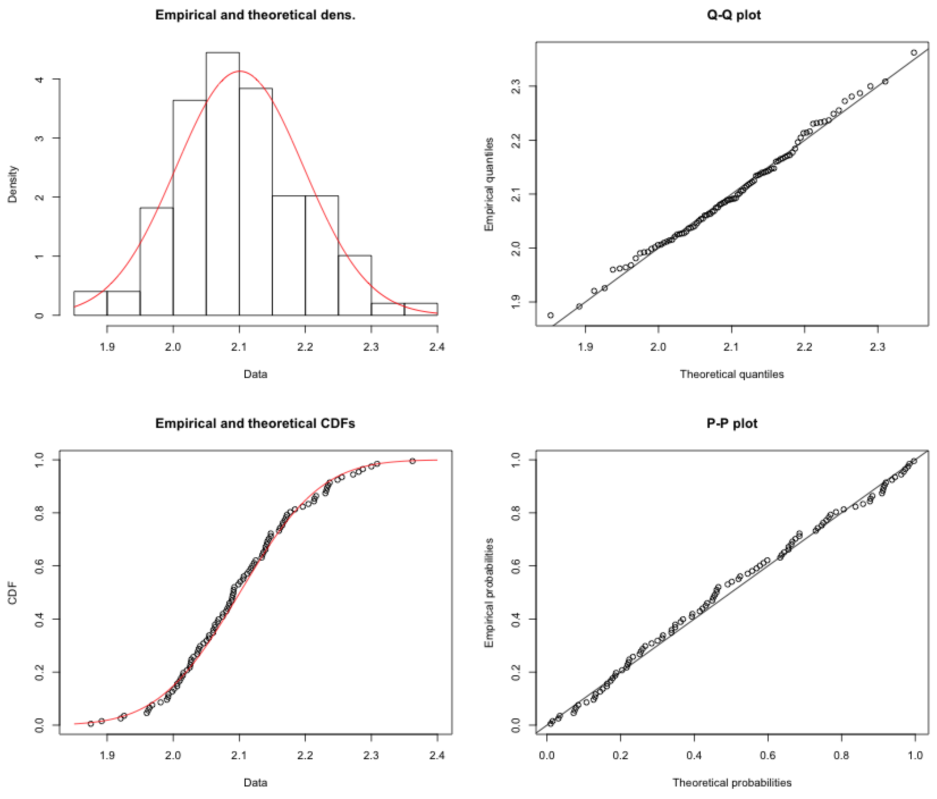

FIGURE A.1. Distribution of normal $D(\mathcal{D}_i, joint)$, for the minority class of the *mushroom* database, in one run of the experiment of table 3.2.

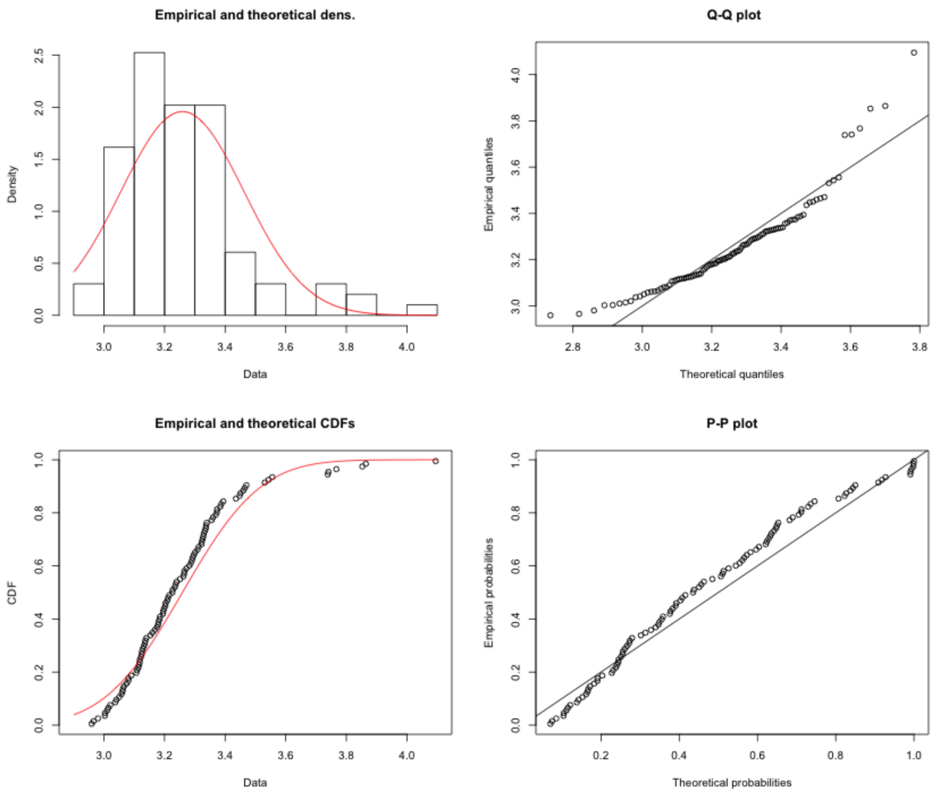

FIGURE A.2. Distribution of normal $D(joint, \mathcal{D}_i)$, for the minority class of the *mushroom* database, in one run of the experiment of table 3.2.

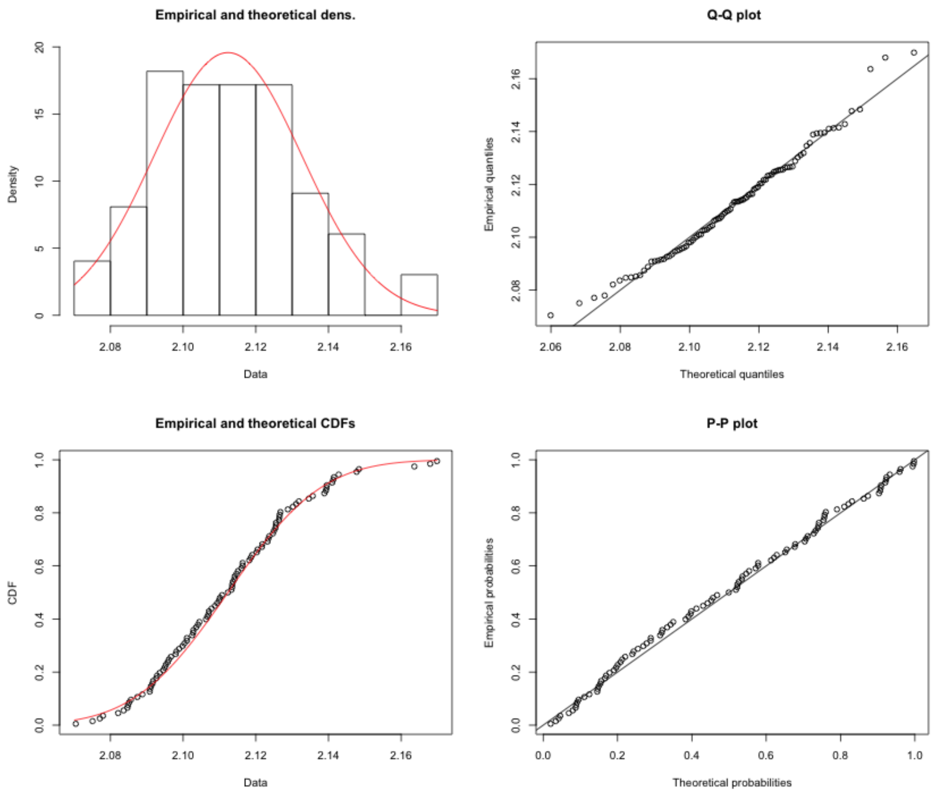

FIGURE A.3. Distribution of normal $D(\mathcal{D}_i, joint)$, for the minority class of the *nursery* database, in one run of the experiment of table 3.2.

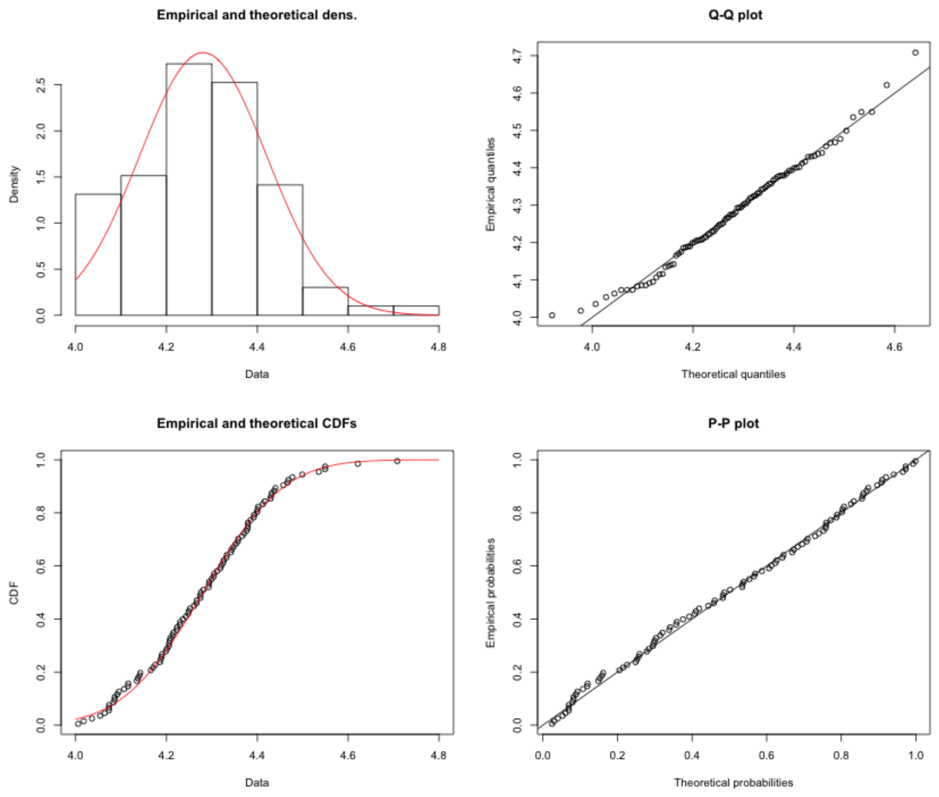

FIGURE A.4. Distribution of normal $D(joint, \mathcal{D}_i)$, for the minority class of the *nursery* database, in one run of the experiment of table 3.2.

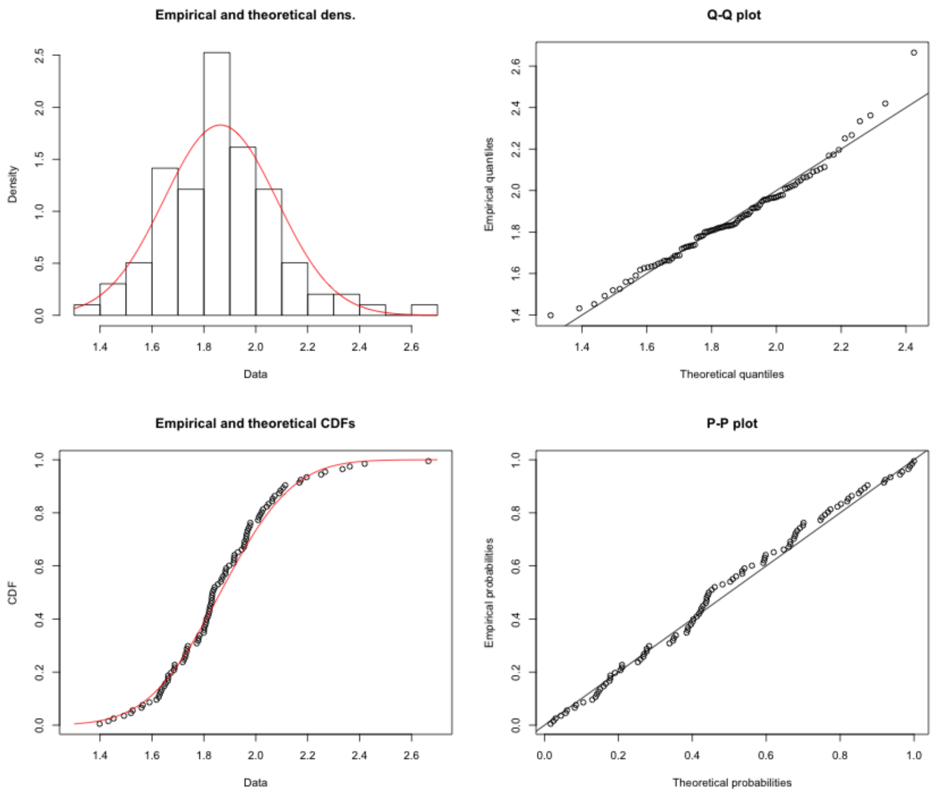

FIGURE A.5. Distribution of normal $D(\mathcal{D}_i, joint)$, for the minority class of the *pendigits* database, in one run of the experiment of table 3.2.

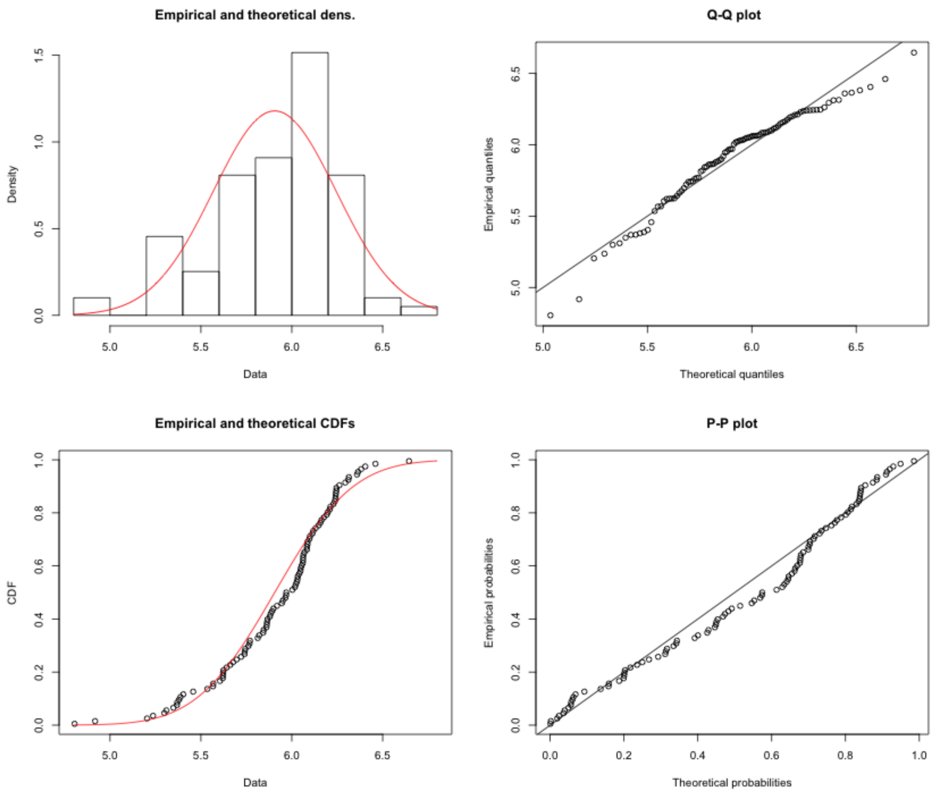

FIGURE A.6. Distribution of normal $D(joint, \mathcal{D}_i)$, for the minority class of the *pendigits* database, in one run of the experiment of table 3.2.

Bibliography

[1] Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos.
Fast and reliable anomaly detection in categorical data.
In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 415–424, New York, NY, USA, 2012. ACM.
ISBN 978-1-4503-1156-4.
doi: 10.1145/2396761.2396816.
URL `http://doi.acm.org/10.1145/2396761.2396816`.

[2] Roel Bertens, Jilles Vreeken, and Arno Siebes.
Beauty and brains: Detecting anomalous pattern co-occurrences.
CoRR, abs/1512.07048, 2015.
URL `http://arxiv.org/abs/1512.07048`.

[3] Steven Burrows and S.M.M. Tahaghoghi.
Source code authorship attribution using n-grams.
01 2007.

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar.
Anomaly detection: A survey.
ACM Comput. Surv., 41(3):15:1–15:58, July 2009.
ISSN 0360-0300.
doi: 10.1145/1541880.1541882.
URL `http://doi.acm.org/10.1145/1541880.1541882`.

[5] Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass.
Authorship attribution with support vector machines.
Applied Intelligence, 19(1):109–123, Jul 2003.
ISSN 1573-7497.
doi: 10.1023/A:1023824908771.
URL `https://doi.org/10.1023/A:1023824908771`.

[6] W. Duivesteijn, A. Knobbe, A. Feelders, and M. van Leeuwen.
Subgroup discovery meets bayesian networks – an exceptional model mining approach.

In *2010 IEEE International Conference on Data Mining*, pages 158–167, Dec 2010.
doi: 10.1109/ICDM.2010.53.

[7] Jack Grieve.
Quantitative authorship attribution: An evaluation of techniques.
22, 05 2007.

[8] Peter Grunwald.
A tutorial introduction to the minimum description length principle, June 2004.
URL `arXiv:math/0406077`.

[9] Simon Hawkins, Hongxing He, Graham J. Williams, and Rohan A. Baxter.
Outlier detection using replicator neural networks.
In *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, DaWaK 2000, pages 170–180, London, UK, UK, 2002. Springer-Verlag.
ISBN 3-540-44123-9.
URL `http://dl.acm.org/citation.cfm?id=646111.679466`.

[10] Zengyou He, Xiaofei Xu, Joshua Zhexue Huang, and Shengchun Deng.
A Frequent Pattern Discovery Method for Outlier Detection, pages 726–732.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
ISBN 978-3-540-27772-9.
doi: 10.1007/978-3-540-27772-9_80.
URL `https://doi.org/10.1007/978-3-540-27772-9_80`.

[11] Francisco Herrera, Cristóbal José Carmona, Pedro González, and María José del Jesus.
An overview on subgroup discovery: foundations and applications.
Knowledge and Information Systems, 29(3):495–525, Dec 2011.
ISSN 0219-3116.
doi: 10.1007/s10115-010-0356-2.
URL `http://dx.doi.org/10.1007/s10115-010-0356-2`.

[12] Thomas Hofmann.
Probabilistic latent semantic indexing.
In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.
ISBN 1-58113-096-1.
doi: 10.1145/312624.312649.
URL `http://doi.acm.org/10.1145/312624.312649`.

[13] F J. Tweedie, S Singh, and David Holmes.

Neural network applications in stylometry: The federalist papers.
30:1–10, 01 1996.

[14] Patrick Juola and George Mikros.
Cross-linguistic stylometric features : A preliminary investigation.
JADT 2016 : 13ème Journées internationales d'Analyse statistique des Données Textuelle, 2017.

[15] Branko Kavšek, Nada Lavrač, and Viktor Jovanoski.
APRIORI-SD: Adapting Association Rule Learning to Subgroup Discovery, pages 230–241.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
ISBN 978-3-540-45231-7.
doi: 10.1007/978-3-540-45231-7_22.
URL `http://dx.doi.org/10.1007/978-3-540-45231-7_22`.

[16] Vlado Ke Selj, Fuchun Peng, Nick Cercone, and Calvin Thomas.
N-gram-based author profiles for authorship attribution.
09 2003.

[17] Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana.
Towards parameter-free data mining.
In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 206–215, New York, NY, USA, 2004. ACM.
ISBN 1-58113-888-1.
doi: 10.1145/1014052.1014077.
URL `http://doi.acm.org/10.1145/1014052.1014077`.

[18] Eamonn Keogh, Stefano Lonardi, Chotirat Ann Ratanamahatana, Li Wei, Sang-Hee Lee, and John Handley.
Compression-based data mining of sequential data.
Data Mining and Knowledge Discovery, 14(1):99–129, Feb 2007.
ISSN 1573-756X.
doi: 10.1007/s10618-006-0049-3.
URL `https://doi.org/10.1007/s10618-006-0049-3`.

[19] Mike Kestemont, Justin Stover, Moshe Koppel, Folgert Karsdorp, and Walter Daelemans.
Authenticating the writings of julius caesar.
Expert Syst. Appl., 63(C):86–96, November 2016.
ISSN 0957-4174.
doi: 10.1016/j.eswa.2016.06.029.
URL `https://doi.org/10.1016/j.eswa.2016.06.029`.

[20] Mike Kestemont, Els Stronks, Martine De Bruin, and Tim De Winkel.
Van wie is het Wilhelmus? Auteurskwesties rond het Nederlandse volkslied met de computer onderzocht.
Amsterdam University Press, Amsterdam, 2017.

[21] Dmitry V. Khmelev and William J. Teahan.
A repetition based measure for verification of text collections and for text categorization.
In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 104–110, New York, NY, USA, 2003. ACM.
ISBN 1-58113-646-3.
doi: 10.1145/860435.860456.
URL `http://doi.acm.org/10.1145/860435.860456`.

[22] Shibamouli Lahiri.
Complexity of Word Collocation Networks: A Preliminary Structural Analysis.
In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.
URL `http://www.aclweb.org/anthology/E14-3011`.

[23] Jiexun Li, Rong Zheng, and Hsinchun Chen.
From fingerprint to writeprint.
Commun. ACM, 49(4):76–82, April 2006.
ISSN 0001-0782.
doi: 10.1145/1121949.1121951.
URL `http://doi.acm.org/10.1145/1121949.1121951`.

[24] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky.
The stanford corenlp natural language processing toolkit.
In *ACL*, 2014.

[25] Yuval Marton, Ning Wu, and Lisa Hellerstein.
On Compression-Based Text Classification, pages 300–314.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
ISBN 978-3-540-31865-1.
doi: 10.1007/978-3-540-31865-1_22.
URL `https://doi.org/10.1007/978-3-540-31865-1_22`.

[26] T. C. Mendenhall.

The characteristic curves of composition.
Science, 9(214):237–249, 1887.
ISSN 00368075, 10959203.
URL `http://www.jstor.org/stable/1764604`.

[27] F. Mosteller and D.L. Wallace.
Applied Bayesian and Classical Inference: The Case of The Federalist Papers.
Springer Series in Statistics. Springer New York, 2012.
ISBN 9781461252566.
URL `https://books.google.nl/books?id=LJXaBwAAQBAJ`.

[28] S Nagaprasad, Vijayapal Reddy, and A.Vinaya Babu.
Authorship attribution based on data compression for telugu text.
110:1–5, 01 2015.

[29] W. Oliveira, E. Justino, and L.S. Oliveira.
Comparing compression models for authorship attribution.
Forensic Science International, 228(1):100 – 104, 2013.
ISSN 0379-0738.
doi: https://doi.org/10.1016/j.forsciint.2013.02.025.
URL `http://www.sciencedirect.com/science/article/pii/S0379073813000923`.

[30] Fuchun Peng and Dale Schuurmans.
Combining naive bayes and n-gram language models for text classification.
In *Proceedings of the 25th European Conference on IR Research*, ECIR'03, pages 335–350, Berlin, Heidelberg, 2003. Springer-Verlag.
ISBN 3-540-01274-5.
URL `http://dl.acm.org/citation.cfm?id=1757788.1757820`.

[31] Bernard Rosner.
Percentage points for a generalized esd many-outlier procedure.
Technometrics, 25(2):165–172, 1983.
ISSN 00401706.
URL `http://www.jstor.org/stable/1268549`.

[32] Upendra Sapkota, Steven Bethard, Manuel Montes y Gómez, and Thamar Solorio.
Not all character n-grams are created equal: A study in authorship attribution.
In *HLT-NAACL*, 2015.

[33] Jacques Savoy.
Authorship attribution based on specific vocabulary.
ACM Trans. Inf. Syst., 30(2):12:1–12:30, May 2012.

ISSN 1046-8188.
doi: 10.1145/2180868.2180874.
URL `http://doi.acm.org/10.1145/2180868.2180874`.

[34] Thibault Sellam and Martin Kersten.
Cluster-driven navigation of the query space.
IEEE Trans. on Knowl. and Data Eng., 28(5):1118–1131, May 2016.
ISSN 1041-4347.
doi: 10.1109/TKDE.2016.2515590.
URL `http://dx.doi.org/10.1109/TKDE.2016.2515590`.

[35] Efstathios Stamatatos.
A survey of modern authorship attribution methods.
60:538–556, 03 2009.

[36] O V. Kukushkina, A A. Polikarpov, and D V. Khmelev.
Using literal and grammatical statistics for authorship attribution.
37, 06 2002.

[37] Matthijs van Leeuwen, Jilles Vreeken, and Arno Siebes.
Compression Picks Item Sets That Matter, pages 585–592.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
ISBN 978-3-540-46048-0.
doi: 10.1007/11871637_59.
URL `http://dx.doi.org/10.1007/11871637_59`.

[38] Matthijs van Leeuwen, Jilles Vreeken, and Arno Siebes.
Identifying the components.
Data Mining and Knowledge Discovery, 19(2):176–193, Oct 2009.
ISSN 1573-756X.
doi: 10.1007/s10618-009-0137-2.
URL `http://dx.doi.org/10.1007/s10618-009-0137-2`.

[39] Paul M. B. Vitányi, Frank J. Balbach, Rudi L. Cilibrasi, and Ming Li.
Normalized Information Distance, pages 45–82.
Springer US, Boston, MA, 2009.
ISBN 978-0-387-84816-7.
doi: 10.1007/978-0-387-84816-7_3.
URL `https://doi.org/10.1007/978-0-387-84816-7_3`.

[40] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes.
Characterising the difference.

In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 765–774, New York, NY, USA, 2007. ACM.
ISBN 978-1-59593-609-7.
doi: 10.1145/1281192.1281274.
URL `http://doi.acm.org/10.1145/1281192.1281274`.

[41] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes.
Krimp: mining itemsets that compress.
Data Mining and Knowledge Discovery, 23(1):169–214, Jul 2011.
ISSN 1573-756X.
doi: 10.1007/s10618-010-0202-x.
URL `http://dx.doi.org/10.1007/s10618-010-0202-x`.

[42] Ying Zhao and Justin Zobel.
Entropy-based authorship search in large document collections.
In *Proceedings of the 29th European Conference on IR Research*, ECIR'07, pages 381–392, Berlin, Heidelberg, 2007. Springer-Verlag.
ISBN 978-3-540-71494-1.
URL `http://dl.acm.org/citation.cfm?id=1763653.1763698`.