Date of acceptance Grade

Instructor

Large-scale Multi-Label Text Classification for an Online News Monitoring System

Matthew Pierce

Helsinki November 24, 2015

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Matthew Pierce

Työn nimi — Arbetets titel — Title

Large-scale Multi-Label Text Classification for an Online News Monitoring System

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	November 24, 2015	77 pages + 0 appendices

Tiivistelmä — Referat — Abstract

This thesis provides a detailed exploration of numerous methods—some established and some novel—considered in the construction of a text-categorization system, for use in a large-scale, online news-monitoring system known as PULS. PULS is an information extraction (IE) system, consisting of a number of tools for automatically collecting named-entities from text. The system also has access to large training corpora in the business domain, where documents are annotated with associated industry-sectors. These assets are leveraged in the construction of a multi-label industry-sector classifier, the output of which is displayed on the web-based front-end of PULS, for new articles.

Through review of background literature and direct experimentation with each stage of development, we illuminate many major challenges of multi-label classification. These challenges include: working effectively in a real-world scenario that poses time and memory restrictions; organizing and processing semi-structured, pre-annotated text corpora; handling large-scale data sets and label sets with significant class imbalances; weighing the trade-offs of different learning algorithms and feature-selection methods with respect to end-user performance; and finding meaningful evaluations for each system component.

In addition to presenting the challenges associated with large-scale multi-label learning, this thesis presents a number of experiments and evaluations to determine methods which enhance overall performance. The major outcome of these experiments is a multi-stage, multi-label classifier that combines IE-based rote classification—with features extracted by the PULS system—with an array of balanced, statistical classifiers. Evaluation of this multi-stage system shows improvement over a baseline classifier and, for certain evaluations, over state-of-the-art performance from literature, when tested on a commonly-used corpus. Aspects of the classification method and their associated experimental results have also been published for international conference proceedings.

ACM Computing Classification System (CCS):
G.3 [Probability and Statistics],
I.2.1 [Applications and Expert Systems],
I.2.7 [Natural Language Processing]

Avainsanat — Nyckelord — Keywords
multi-label learning, text categorization, information extraction, NLP, online business news

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Multi-label learning is a burgeoning topic in the computer science research field of machine learning. Multi-label learning transforms the fundamental data classification task—predicting class membership of a data point for a single concept, represented by a *class label*—to predicting multiple class associations over a set of labels. Although conceptually simple, this transformation results in higher complexity when training, evaluating, and deploying learning systems. In adapting to the multi-label domain, novel methods have been developed and traditional methods have been altered or combined in various ways, however numerous difficult problems remain. While certain methods have produced encouraging results on specific problem sets, there are currently no optimal solutions in general. In particular, research is sparse into methods with scalable performance for large, unbalanced label-sets. Another drawback in current research is the specificity of problem settings, which may or may not be applicable to real-world applications with noisy data drawn from multiple sources.

This thesis tackles a number of problems in multi-label learning by exploring a variety of methods used to build a classification system from a large and continuously growing set of business news articles ($\sim$1.8M documents as of this writing). The articles are collected as part of an online information-extraction-based news monitoring system, known as PULS. The main classification task is to label new articles, drawn from a number of sources, with likely industry sectors. These sectors are displayed to PULS users for easy grouping and filtering.

Several experiments are outlined in this research that explore the myriad dimensions of the classification process (e.g., learning algorithms, training and test data collection, feature-engineering, etc.) in an attempt to learn about the specific advantages and drawbacks of each and, ultimately, to find configurations that perform optimally, both for the specific PULS setting and in general. Optimality, however, takes on different meaning depending on context, therefore a number of evaluation methods are also presented, both of the individual components and the overall system, in order to justify design choices. Some of the algorithms, schemes, and evaluations presented are based on standard approaches, while others are novel. The results of the experiments—many of which have been published for international conference proceedings—illuminate important characteristics and key tradeoffs inherent to multi-label classification, while also determining the best performing methods. A comparison of results alongside prior work also shows improvement on state-of-the-

art methods, for certain measures, when the system is tested on a corpus of news documents.

While the major focus of this thesis is on machine-learning methods and tools used in the performance of various classification tasks, it is also necessary to briefly introduce the field of natural language processing (NLP) in order to (a) outline the target setting for which this research and its associated programs are intended, and (b) define tasks that play key roles in the overall classification system, such as information extraction (specifically the sub-task of named-entity recognition), token lemmatization, and part-of-speech filtering. An overview of the machine-learning domain is also given in this introduction, with a focus on text-based classification. The problem setting, business-sector classification, is then described at the end of the introduction. The rest of the thesis is structured as follows: Section 2 discusses background research that this work starts from and attempts to expand on; Section 3 describes the PULS online news system, as well as the business news data used in PULS; Section 5 outlines methods used to organize the raw set of industry labels provided for the main data set, including statistical methods that aid in mapping them to a different label space, and the construction of a two-level hierarchy for improved classification; in Section 6, various ways of extracting, selecting, and representing lexical features of the data are explored; the entire multi-label classification process is described in Section 7, and the results of the system evaluations detailed in Section 8; finally, conclusions about the most important aspects of this research and where it should direct future work are provided in Section 10 and Section 11.

1.1 Natural Language Processing

NLP has been a major field of study within computer science for decades, tracing its history as far back as the 1940's, with the fundamental research on topics such as automata and information-theoretic modeling [JM00]. A wide-ranging discipline, main sub-topics include language and speech detection, information extraction, and machine translation. Some standard elements of NLP include syntax trees, a convenient data structure for parsing and storing natural language sentences; part-of-speech taggers, which attempt to find correct grammatical categories for each element of a sentence; and word-sense disambiguation tools, which return the most probable reading(s) for a word with multiple meanings, based on context [JM00]. More recently, machine-learning techniques have also been added to the NLP toolkit. For example, statistical classifiers may be used to automatically determine the rel-

evance of a text document to a specific domain, or clustering methods may be used to automatically learn semantic relationships between words. The machine-learning methods discussed in this thesis are similarly employed in an NLP setting, such as the industry sector classifiers outlined in Section 7.

The Stanford Natural Language Processing Group is a major researcher in NLP and provide a state-of-the-art toolkit for working on NLP problems [KM03]. Their Java toolkit is utilized in this research, in particular for part-of-speech tagging, which is essential to the lexical-feature engineering methods described in Section 6.

The main NLP task that will be discussed in this thesis is information extraction (IE). The goal of IE is to automatically obtain facts that conform to a pre-determined structure (or template), from machine-readable documents, such as natural language texts [CL96] [GW98]. Beginning from the late 1980's, research in IE was advanced through a series of seven Message Understanding Conferences (MUC), funded by the Defense Advanced Research Projects Agency (DARPA) [GS96] in the US. Each conference was structured as a competition to evaluate methods for extracting structured information from textual documents. For example, in MUC-3 attendees were required to fill out a template with facts regarding time, place, event type, participating agents, etc. (for a total of 18 slots) from a collection of reports about terrorist events in Central and South America [GS96].

Named-Entity recognition is a sub-task of IE, which was borne out of the MUC conferences. Initially conceived for the purpose of having a highly practical, accurate, and portable task for evaluating IE systems, the goal is to find names of all people, organizations (i.e., company names), geographic locations, and any other nominal items in a text [GS96]. The type of data extracted in the named-entity task plays a significant role in both the training and testing of the classification system described in Section 7.

1.2 Machine Learning and Text Categorization

A closely-related field to artificial intelligence, machine learning has a similarly long history, dating back to the 1960s and the invention of the first learning machines, such as F. Rosenblatt's Perceptron [Vap00]. The past two decades have seen an increase in interest and progress in the field. The result is the recent emergence of numerous standard algorithms and tools, which are now being applied in diverse domains, such as natural language processing, computer graphics, and biology.

Machine learning concerns the use of statistical models in attempting to replicate and automate cognitive learning processes. Although these statistical methods are currently not, and may never be, as advanced as human learning, a computer is able to efficiently process information on a much greater scale than is possible for any human, or collection of humans. Because of this, much work in machine learning focuses on the use of large, high-dimensional data for building sophisticated programs that automatically solve complex, dynamic problems. Most commonly, these programs work by applying statistical decision-making algorithms trained on large data sets, to previously unseen data from a similar domain [CMM83]. This task, known as classification, is one of the central tasks in machine learning and the main focus of this thesis.

There are several sub-categories of machine learning, with supervised learning, unsupervised learning, and reinforcement learning being principal among them [MRT12]. In this thesis, the focus is on supervised learning: the collection of algorithms and methods that use pre-existing knowledge of the data as input to the learning algorithms [MRT12]. This prior knowledge usually refers to the membership of a data instance to a specific, differentiable group within the data. Such a group is known as a class and instances belonging to a class are said to have that class's label. Examples of supervised learning algorithms that can predict class labels include support vector machines (SVMs) [SC08] and Bayesian inference [BT11].

A natural application for classifiers is text categorization, since digital texts are both abundant and easy to process and store. They are also categorizable by humans, and although this can be an expensive process, numerous sets of pre-labeled text data are readily available. Commonly, text classifiers are trained using properties of the words contained in the data. Statistical information about the occurrence of specific terms within a single document, a class of documents, or a data set's entire lexicon can all aid the classifier in discriminating between members and non-members of a class. In general, any item that is used by a classifier to help discriminate between classes is known as a feature, and in text categorization such words are known as lexical features. Characteristics of language such as synonymy and word ambiguity can confound what may be otherwise useful lexical features. For example, the word 'lead' may be a good descriptor, if it refers to the noun (i.e., the metal), but not if it refers to the verb. For this reason, more advanced techniques, such as lemmatization and feature selection have been developed to improve the reliability of the feature set. These techniques will be described further in Section 6.

Because text categorization is a common task, there are many software packages available for building and implementing classifiers. One package is the data-mining and machine-learning toolkit, Weka [HFH+09], which is used for building and running classifiers for the experiments in this thesis.

Early research in supervised learning focused on predicting class membership of a single label for a given data point. In many domains, however, an instance may belong to numerous classes. For example, the design of a system responsible for categorizing technical documents may require that a report on the harmful effects of smoking should belong to a class labeled "healthcare" and another class labeled "tobacco products." This type of problem is known as multi-label classification [ZZ07]. The problem of multi-label learning is becoming increasingly important, due to both the rapid rise of large, cheap, and efficient data storage and the ingress of machine-learning into domains that use multi-faceted taxonomies. Business is one such domain, and the classification of business texts into a set of industry sector labels comprises the main goal of this thesis.

1.3 Business Sector Classification

The primary motivation of the work presented in this thesis is to develop a high-performance multi-label classification system for augmenting the PULS online news-monitoring service (described in detail in Section 3) by supplying business sector tags to all incoming articles. Since PULS is a user-oriented service that operates in real-time, performance considerations must be given to the requirements of the customers. Imprecise classification will result in spurious articles being presented to users when filtering by sector label. On the other hand a classification system that returns few or no labels for most articles will cause users to miss important news for their sector(s) of interest. Additionally, time efficiency is important for classification, as new documents should be labelled as soon as possible.

Training for the PULS classification system is done using a data set consisting of articles which were annotated by our partners during an earlier joint project. This data set is referred to in this thesis as the PULS data set, and consists of approximately 1.8 million business news articles. The articles cover a range of news events such as company investments, corporate position nominations, product releases, bankruptcy filings, and political and environmental issues surrounding business dealings. The range of industries covered in the articles is even broader, consisting of over 700 unique labels.

Given the size and scope of the data and label set, building an accurate and efficient system for PULS is no trivial task. The data set used has significant imbalance with respect to the distribution of labels, which can lead to overfitting. The similarity of certain labels may create class ambiguity or completely unseparable classes. Furthermore, a large label set increases the chances of human annotation (labeling) errors, which results in inconsistent evaluation. Finally, the size presents significant efficiency challenges, both in terms of computation time and memory usage.

In addition, for meaningful comparison with other, prior work, we use a second data set that allows us to compare our methods against state-of-the-art methods on specific evaluations. The Reuters RCV1 data set [LYRL04] is used, as it is similar to the PULS data set, both in terms of content and size—specifically, label set size. Using the industry sector codes (as opposed to the more commonly used *topic* codes), a set of approximately 350,000 data instances, covering approximately 250 sector labels, can be used for training and testing purposes.

While the two aforementioned data sets are the only ones used in the experiments of Section 8, it is important to note that the data to be classified in the online PULS system is drawn from a range of sources, such as online collections, and RSS feeds. This means that the problem of overfitting to a specific corpus is a pertinent issue.

In general, the methods explored in the following sections and in the experiments are chosen with an eye towards use in a real-world setting, such as PULS, where realistic limitations must be taken into consideration.

2 Related Work

Prior research in multi-label learning is quite broad, with surveys found in [TKV10] and [Sor10]. This thesis focuses on the most closely-related research aspects, namely: supervised multi-label text categorization, hierarchical learning, feature selection, data imbalance, and work with large textual data sets.

Much work in supervised multi-label learning regards adapting single-label algorithms to multi-label tasks. Research provides two principal approaches: *problem transformation* and *algorithm adaptation* [TK07]. Problem transformation converts multi-label classification into a series of single-label sub-tasks while algorithm adaptation extends the underlying algorithms of traditional methods to work directly with multi-label data. A common problem-transformation method that has con-

sistently shown competitive performance is *binary relevance* (BR), where a single binary classifier is trained for each label [MKGD12]. Learning BR classifiers is often done by *cross-training* (also known as one-vs-rest) [BLSB04], in which each classifier uses instances tagged with the given label as positive examples, and *all* remaining instances as negative. This method assumes label independence, which is often untrue in practice. Correlation among labels may pose difficulties when combining binary classifiers [dCF09]. Another disadvantage of BR is that may exacerbate data imbalance, as the number of negative examples will likely significantly outweigh the positive [DK10].

In multi-label learning it is possible to leverage a hierarchical label structure, either natural or imposed, to improve classifier performance. Classification with labels organized in this way is referred to as multi-label *hierarchical* classification [GZ13, MKGD12, GCR09, CM07]. This thesis explores the use of a two-level hierarchy for both classification and evaluation.

A common *data representation* in text categorization is the "bag of words" (BOW) model, which ignores document structure and assumes word independence [KS97]. The BOW model can be extended by integrating n-grams [DVW$^+$12, ZYT11]. Experiments in this thesis use the BOW model with a combination of unigrams and bigrams.

In text data, the number of distinct word types often exceeds the number of training documents by an order of magnitude [For03]. This generally complicates learning and requires the use of *dimensionality reduction* in most approaches. While dimensionality reduction accelerates processing, it has also been shown to improve categorization performance [TB03, KS97] through avoidance of over-fitting [LLS09]. Reduction can be done either by *selection* of highly-relevant features or *grouping* (i.e., clustering) features [KS97]. This thesis explores feature selection based on comparing the discriminative power of a given word, relative to all other words in the feature set. Comparative studies of various feature selection methods can be found in [For03, YP97]. After comparing twelve feature selection methods, [For03] claims that, for highly-skewed data, Bi-Normal Separation (BNS) gives the best performance.

Feature selection can be done globally or locally [LLS09]. Local feature selection measures the usefulness of features with respect to one particular category, whereas global feature selection takes into account the feature distribution across all categories. In this thesis global and local feature-selection processes are combined.

Moreover, for global feature selection a novel measure is introduced, which, as far as the author is aware, has never been used before (see Section 6.2.3).

Information Extraction (IE) can be used to obtain additional features for classification [HR13, HVDY12]. As far as the author is aware, using features obtained via IE in *multi-label* text categorization has not been reported to date. In this thesis company names and descriptors, extracted from the text by the PULS named-entity recognition system, are used in building a baseline, Rote classifier (Section 7.1).

Text data sets are typically "naturally skewed," [LLS09], since topics differ both in frequency and importance, depending on their origin. Skew may also be introduced by annotator bias. Such imbalances pose a challenge for categorization, especially when the classes have a high degree of overlap [PBM04]. In the case of multi-label classification this means that for smaller classes (by positive-data size), classifiers usually perform worse than for larger classes [For04, Seb02] and, as a consequence, the macro-average results are usually lower than micro-average [DS05, LLS09]. This problem may be tackled on the *data* level or the *algorithmic* level [KKP+06]. The data-level approach is based on various re-sampling techniques [CBHK02]. Some re-sampling techniques for use in text classification are described in [EA13, DK10, Sta08]. Two approaches to re-sampling are *oversampling*, which puts more instances of the minor classes into the training set, and *under-sampling*, which reduces the number of instances of the major classes from training [JS02]. Over- and under-sampling can be either *random* or *focused* (i.e., *informed*). This thesis describes a random under-sampling approach that is used to build general classifiers, having good performance on data drawn from sources outside of the training source.

System performance will differ, depending on the data set. In general, the more skewed the data, the harder the classification task. A good example is seen in [DS05], who experiment with three of the most frequently used subsets of the Reuters-21578 [1] corpus: R(10), the set of 10 categories with the most positive training examples; R(90), the 90 categories with at least one positive training example and one test example; and R(115), the set of 115 categories with at least a single positive example. Their work demonstrates that the difficulty of text-classification increases as the number of categories grows. From their experiments, the mean difference in system performance between R(10) and R(90) was 25%, with respect to the macro-averaged F_1 (72% for R(10) and 47% for R(90)).

A number of large-scale corpora are available for use in classification. One, the

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

DMOZ Open Directory (www.dmoz.org) is an on-line catalogue of more than 4.5 million Web pages, labeled with approximately 700,000 hierarchical categories [Hen09]. The Reuters RCV1 (about.reuters.com/researchandstandards/corpus/) corpus contains over 800,000 news stories published by Reuters between 1996-1997, labeled semi-automatically with 103 Topic, 350 Industry and 296 Region codes.

3 The PULS IE System

The Pattern-Based Understanding and Learning System (PULS), is a document-retrieval and information-extraction system for collecting, processing, storing, and displaying news data from around the world [GHY03, YS09]. Currently, the system has three distinct domains: business, where we extract company activities such as bankruptcies, mergers, and product releases; security, where news about cross-border illegal activity, like smuggling and illegal entry is tracked [APvdGY11]; and medical, which follows the outbreak of infectious-disease epidemics [HNA+13]. Generally speaking, system functionality is the same, regardless of domain, and the major differences lie in the content of the retrieved data. This thesis will focus solely on the business domain. Business data for PULS is collected from multiple sources. One of the sources also includes industry sector labels, which allows for the use of supervised learning methods. This data set is described further in Section 4.

News articles are provided to PULS via a number of domain-relevant RSS feeds. Articles arrive in real-time as html documents, which are automatically downloaded to the system at regular intervals. The core of PULS is an IE system that automatically extracts useful information from the plaintext data. This is accomplished through a number of modules that, together, comprise the IE system. These modules are responsible for tasks such as text preprocessing, sentence parsing, semantic interpretation, word-sense disambiguation, and co-reference resolution. The output of the IE process is a template file that contains slots for each type of expected output, and corresponding facts for every slot that could be filled with a value [HYG02]. Some of these slots are filled using strings found in and extracted directly from the text, while others take values inferred by other methods such as supervised learning—as in the case of industry sectors—or cross-document aggregation, where values are drawn from similar and related documents [Yan06].

One of the IE modules is the Named-Entity Recognition (NER) module. The NER module is built from a cascade of low-level patterns, which find noun groups within

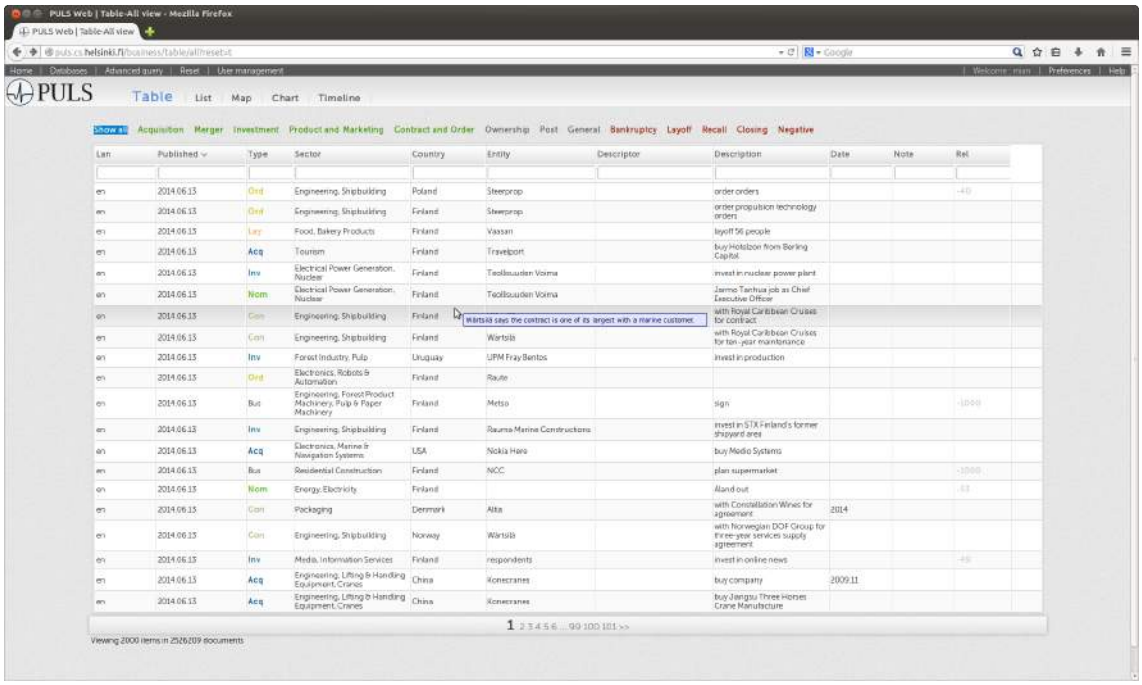


Figure 1: PULS table view

a text. This means that the module finds not only named entities, but also their *descriptors* (i.e., noun and adjective modifiers of a given name). For example, Apple can be described in the text as "computer maker" or "software giant". As seen in this example, a descriptor may consist of up to two main components (although one, the other, or both may be missing): *domain*, an area in which the company works (i.e., "computer", "software"), and *type*, a word that is synonymous with "company" (i.e., "maker", "giant"). A descriptor may also contain other components, such as a geographic marker (i.e., "English company", "Swedish company") or some additional information, (i.e., "big company", "local company", etc.). A descriptor may contain all of these components, or only some of them. PULS uses a short list of approximately 20 company words—such as "corporation", "firm", "manufacturer"—to determine the company type. When obtaining company domain, highly general words, such as "business," are filtered out. The company names and descriptors extracted by the NER module can be used in further tasks, such as sector classification (see Section 7).

The PULS front-end is a website (PULS-Web) that provides tools for sorting, filtering, and displaying news articles and various graphical views for visualizing related information. These views include: a table view, shown in Figure 1, which presents the newest articles along with advanced search options; a document view, seen in

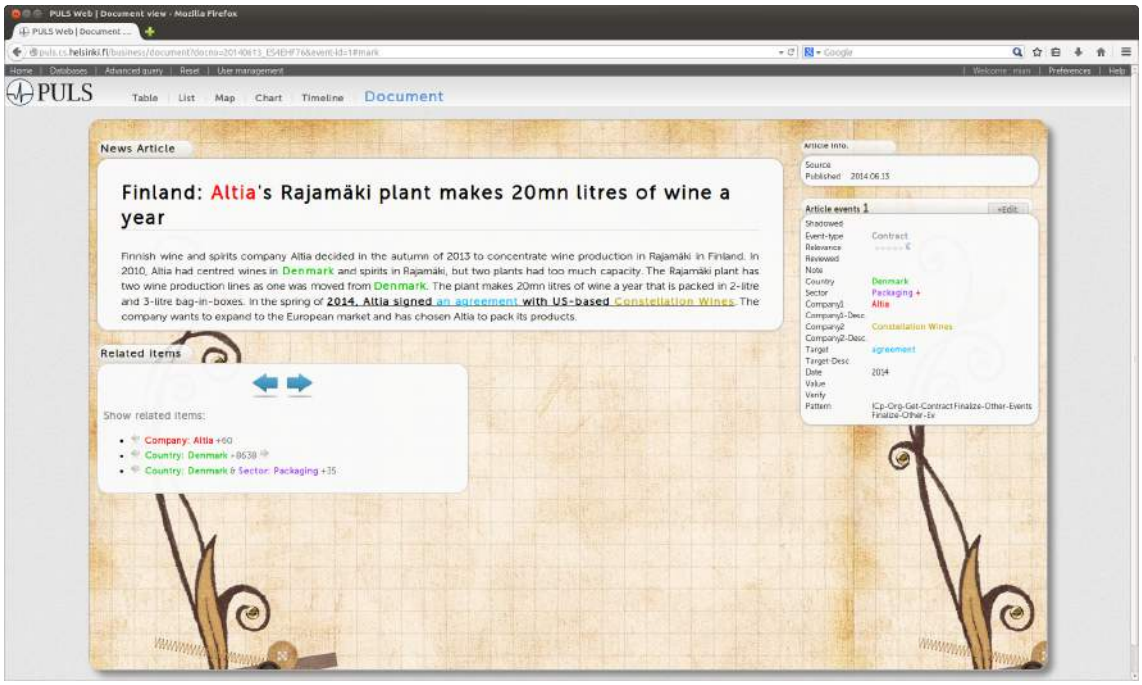


Figure 2: PULS document view

Figure 2, which allows the user to read the article and view the extracted elements; and a list view, which groups together stories concerning similar events. PULS-Web is also used to display auxiliary data related to articles, such as statistical information, document relevance, and relationships between extracted entities. It is also used to display the industry sectors for a business document, as shown in Figure 3. These sectors may be manually assigned, if the document was annotated manually, or it may be the output of the industry sector classification system that will be described in Section 7.

4 Online Business News

The two data sets used in the experiments and evaluations section of this thesis (Section 8) are the PULS business news corpus [HVDY12] and the RCV1 news corpus from Reuters [LYRL04].

The first set was compiled and annotated during a previous project, by our partners, and consists of approximately 1.8 million business news documents, collected daily from around the world, over the course of more than 7 years (2008-present). The documents have been manually tagged by domain experts from a set of over 700

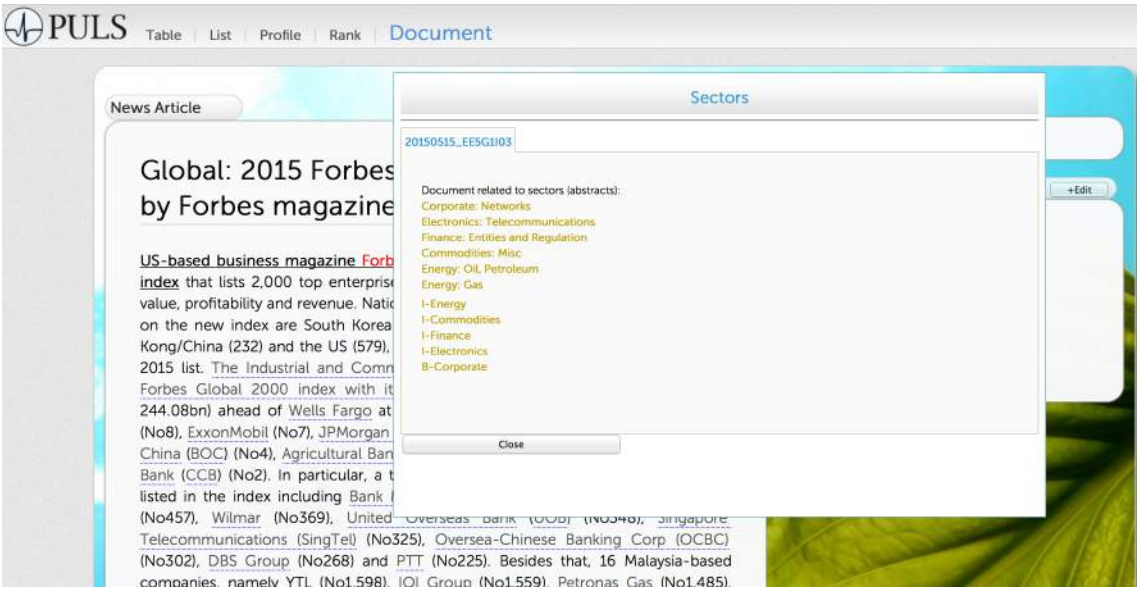


Figure 3: List of industry sectors in document view

distinct industry-sector labels, such as *Energy, Gas* and *Finance, Banking*. The documents are multiply-labelled, with no restriction on the number of labels per document. Due to various problems and inconsistencies in the original label set, these labels have been mapped to a smaller label set for use in the experiments of this thesis. Section 5 describes the motivation and procedure for mapping the original labels to this new set.

The second data set, RCV1, is a collection of Reuters news articles from 1996-1997. The data set has been used extensively in other research, however, these studies most commonly make use of the 103 *topic* labels with which the articles have been annotated. The experiments of this thesis instead make use of the data set's 245 (after some minor pre-processing) *industry* labels, as it corresponds much more directly to the given problem domain, both in terms of content, and size of label set. Industry labels in Reuters have the form of 5- or 7-digit codes; sectors having the same prefix are intended to be children of the same top-level sector. Others have used the first 3 digits as upper-level labels, however, these top-level labels are highly uneven: e.g., 170 distinct 3-digit codes appear in RCV1, but most of them (126) have only one child in the data. Thus classifying on the 5-digit and the 3-digit level is almost the same task.

Table 1 shows a comparison of the two original data sets. In addition to total number of instances and labels, the table includes the following measures as described in [Sor10]:

Table 1: Data and label set statistics for PULS and Reuters data sets

Data Set	**# Instances**	**# Labels**	**LCard**	**LDen**	**DL**	**PDL**
PULS	1815252	704	1.576	0.002	136443	0.075
RCV1	351810	245	1.395	0.006	16653	0.047

- *Label Cardinality* (LCard), the average number of labels per document;
- *Label Density* (LDen), the number of labels-per-document, divided by the total number of labels, averaged over all documents;
- *Distinct Label Set* (DL), the number of distinct label combinations found in the data set; and
- *Proportion of Distinct Label Set* (PDL), the DL measure normalized by the number of documents.

Working with so many distinct labels results in an increased likelihood of *class imbalance* in the data. The class imbalance problem has been explored in single-label classification, for example in [BPM04, LWZ09, DH$^+$03], however it poses an even greater challenge in multi-label learning, for instance in generating a set of training data that isn't dominated by frequenly occurring classes, since this will likely result in overfitting.

5 Sector Labels

In classification, a label is the representation of a specific concept, whose characteristic range of categories comprises the predictable class values for which a classifier may be trained. A good label should be a concise but accurate representation of the corresponding concept and its respective category range. For example, a binary classifier used to detect the presence of a beach scene in an image could be represented by the label *Beach*, whereas a multi-class classifier used for categorizing scene images into various natural environments (e.g., beach, mountain, forest, etc.) could use the broader label *Environment Type*.

In multi-label classification, label naming inherits an additional level of complexity, owing to the possibility of conceptual overlap. Consider the two industry sector labels: *Retailing* and *Jewelry*. The first label deals with the selling of items—any

items—in stores, while the other deals with one specific retail item. Though the two labels cover obviously distinct categories, there is also some content overlap. Nevertheless, a good multi-label classifier should treat the two as separate topics, and in the event that a text discusses a retail store that sells jewelry, hopefully predict both labels. On the other hand, the presence of a third label such as *Jewelry Stores* would be redundant and only serve to add unnecessary complexity to the classification. For this reason, the label set should, whenever possible, be carefully constructed to provide maximum coverage of pertinent concepts, while minimizing redundancy.

In many cases, however, it may be preferable or necessary to work with a data set annotated with a pre-existing set of labels that may or may not be optimized in the aforementioned way. In addition, pre-labeled data instances may be erroneously labeled, or have incomplete annotations. In these cases, preprocessing can be used to help organize the labels into a more manageable size and structure, to limit classification error later on.

The following subsections discuss two methods for preprocessing label sets, that have been applied to the original set of sector labels in the PULS domain. [2] Section 5.1 discusses the mapping of the original labels to a new label space, whose design is aided by both class-specific training-error analysis and by applying clustering techniques on positive-class data. Section 5.2 discusses the creation of a two-level sector-label hierarchy, that provides structure to the label set, and also allows for simple hierarchical classification.

5.1 Mapping Sector Labels

One of the biggest bottlenecks in constructing a supervised multi-label classification system is the process of manually annotating training instances. Given an (optimistic) projection that at least 100 document instances would be required to learn a predictive model with good performance, a domain with over 200 distinct labels, such as the ones described in this thesis, would still require on the order of 200,000 manually-labelled instances. [3] Furthermore, the content of each document must be considered with respect to not only one, but all possible labels, further extending

[2] Reuters labels are already concisely organized, and do not require extra preprocessing.

[3] With multiply-labelled data the actual number required may, of course, be smaller than this estimate, due to label-overlap. However, typical label-assignment statistics suggest that average label overlap is not usually high, so that this estimate remains appropriate.

the already tedious process.

Pre-annotated data may also present a number of challenges, however. One issue is whether the set of labels in the training data fully encompasses all concepts present in the data to be classified. Conversely, the range of labels provided by an external source may be too broad for one's own classification task. Similarly, label names may be too specific, too general, or too ambiguous for the intended audience. The most difficult issues, however, have to do with the problems of annotator bias and annotation error.

Annotator bias refers to characteristics of data annotation—e.g. the level of label-coverage across instances, the amount of skew in the distribution of labels, the amount of label overlap, etc.—introduced by the specific habits of the person(s) who annotated the data [AP05]. Annotator bias is a well-known issue in NLP that has been studied, for example, in [CS13] and [WBPB10]. One of the major drawbacks is the introduction of unwanted or hard-to-guess dependencies among the labels, or, conversely, the lack of a relationship between seemingly similar labels. As an example, consider a business news article that details the drilling of fossil fuels by a company that also provides residential heating services. This document could be labeled with an *Oil Drilling* label, a *Fossil-Fuel Energy Services* label, or both, depending on the annotator (or neither, although if no label is applied relating to fossil fuel, this should be considered annotation error). If the annotator's tendency is to label such documents with both tags, it may introduce hidden dependencies between labels that frequently occur with this kind of relationship, within the data. On the other hand, if the tendency is to only annotate with one or the other label, the learning model may be missing useful feature input.

Annotation error is another major concern when working with a pre-annotated data set, since it can significantly affect proper training and evaluation of learners. Annotation error refers to both erroneous application of a label to a document as well as the absence of labels that would cover a significant portion of the content contained in that document. As an example of the negative training effects of annotation error, consider the cross-training method (employed in the experiments of this thesis) in which all non-positive data instances are treated as negative by the learner. In this case, the learner for a label that reasonably should have been applied to the document will now use the instance as a negative exemplar instead, thus treating possibly useful positive features as negative. Evaluation will also impacted in a negative way due to spurious false-positives and false-negatives, leading to false

Table 2: Data and label set statistics for original vs. mapped PULS data sets

Data Set	**# Instances**	**# Labels**	**LCard**	**LDen**	**DL**	**PDL**
PULS (original labels)	1815252	704	1.576	0.002	136443	0.075
PULS (mapped)	1815252	454	1.54	0.003	116176	0.064

assumptions about the initial method.

In an effort to limit the negative performance-related aspects of the aforementioned issues, the original label set may be mapped to a new label space by either combining multiple labels into a single new label, or by splitting single labels into a number of sub-labels. Merging labels may, for example, reduce annotator bias by eliminating the possibly skewed distribution of conceptually similar labels. It may also reduce the effect of annotation error by providing larger training pools for labels, strengthening the impact of positive features. Two methods for label preprocessing are outlined in this thesis, which focus on combining labels, thus generating a smaller, more concise label set, with less overlap. The first method consists of a simple analysis of the training error—specifically the false-positive results—of original-label classifiers, while the second is a more complex analysis that explores the latent space of positive-class data by utilizing a method from linear algebra, single-value decomposition (SVD), in conjunction with hierarchical clustering.

Table 2 gives the statistical measures described for Table 1 for the mapped PULS data set (compared to the original set).

5.1.1 Training Error Analysis

Hypothetically, a full manual analysis of a label set could be done that would find natural pairs or groupings of labels whose conceptual overlap is high (i.e., so high as to be redundant in the output domain). By merging these multiple labels into a single label, not only would it remove ambiguity from the overall system, it would boost the available training data for the new label's class model. In a real-world scenario, however, there may not be enough time or resources to do such an analysis, especially given a label set with size greater than 200, such as those described in Section 4.

One simple automated method for determining closely related class labels is to first build classifiers for all original data labels, then analyze the training error rate of the classifiers. By collecting the training instances that are classified as *false positive*

(i.e., negative-class instances predicted as positive) from each class, two sets of label rankings can be determined for each label, m:

1. The ranking of labels that are most commonly mistaken as positive, by the classifier of m
2. The ranking of labels whose classifiers most commonly mistake positive instances of m as positive instances of their own class.

The scoring functions for producing the two rankings are therefore as follows:

$$r1(m,l) = \frac{|FP_{m,l}|}{|FP_m|} \qquad r2(m,l) = \frac{|FP_{l,m}|}{|FP_l|} \tag{1}$$

where FP_m is the set of false-positive training instances classified by the classifier of label m and $FP_{m,l}$ is the subset of FP_m that are also positive-class instances for label l.

While the two measures have merit on their own, intuitively, it makes sense to combine the scores, then compare all pairwise results, in order to find the most easily confusable pairs, overall. Using the geometric mean, this combined score is given by:

$$r(m,l) = \sqrt{r1(m,l) \cdot r2(m,l)} \tag{2}$$

and the complete set of pairwise false-positive scores is defined by:

$$FPS = \{r(l_i, l_j); i < j\} \tag{3}$$

Since scores are symmetrical, and order doesn't matter, its only necessary to calculate scores for pairs where $i < j$. Sorting this set will produce the ranking of the most highly-confusable label pairs, which can be used as an aide for targeting possible candidates for merging.

5.1.2 SVD, LSA, and Hierarchical Clustering

While the training-error analysis method for finding labels with highly-overlapping conceptual information can be useful, it can also be time- and resource-consuming, since it requires training classifiers for all original labels. Worse, reliability is compromised by all the typical pitfalls of learning algorithms, such as insufficient training data and overfitting.

One way to eliminate these drawbacks is to group labels based solely on their positive-class instance vectors. For this, one can adapt an NLP technique known as latent semantic analysis (LSA). LSA consists of two steps: 1) projecting data vectors in a term-document matrix to a lower-dimensional *latent feature space*, and 2) calculating pairwise distances (e.g., cosine distance, Euclidean distance, etc.) between vectors in this new space to find the nearest elements. The new, latent feature space is so-named because the projection of each feature in the new space is representative of a collection of highly correlated features from the original space. In other words, it finds the latent connections between the original features. For example, this technique is commonly used to find semantically related words, such as "tobacco" and "cigarette", based on document co-occurrence both with each other and with additional related terms, such as "nicotine."

The dimensionality reduction phase of LSA is done with a method from linear algebra, known as singular value decomposition (SVD). Briefly, SVD is a matrix factorization of the form [GR70]:

$$A = U\Sigma V^T, \text{ where} \tag{4}$$
$$U^TU = V^TV = VV^T = I_n \text{ and } \Sigma = diag(\sigma_1, ..., \sigma_n)$$

For latent-feature projection, the pertinent components of the above equation are the columns of matrices U and V^T, which hold n eigenvectors associated with the eigenvalues of AA^T and A^TA, respectively, and the diagonal values in Σ, which are the non-negative square roots of the eigenvalues—called the *singular values* of A. Importantly, the diagonal values of Σ can be sorted in descending order, and truncated below a certain threshold, as the original indices of these sorted values correspond to the most significant correlations in the initial data matrix A. This allows for a reconstruction of the data based only on the top k singular values, producing the reduced feature space.

Although typically done on a term-document matrix, in order to illuminate semantically-linked words or texts, SVD can easily be adapted to help group conceptually related class labels. If each label is represented as a single large text, consisting of all positive-class documents for that label, a term-label matrix can be constructed. The remaining steps of LSA are then normally executed.

The task of grouping labels based on similarity can be accomplished through the use of clustering techniques, such as k-means clustering, density-based methods, or hierarchical clustering. For this thesis, agglomerative hierarchical clustering was

performed, as it provides more detail and directly produces pairs and triplets of the most similar labels, making it easier to manually parse than, say, k-means clusters.

Agglomerative hierarchical clustering [DE84] is a bottom-up approach to clustering. The algorithm begins by assigning each observation to its own cluster of size 1. Every following step finds the two nearest clusters, according to a given distance metric, and merges them into a new cluster. This continues until a single cluster, containing all observations, has been formed. By retaining every cluster, along with their corresponding distances, a dendrogram can be built, with each node of the tree structure corresponding to a sub-cluster, and the final cluster serving as root node.

Common similarity metrics for determining cluster distances include Euclidean, city-block, Chebychev, and cosine distances. In order to compare clusters of size greater than 1, a method must also be specified to define the two items being compared. These methods include *single-linkage*, which finds the minimum distance between any two items in opposing clusters; *centroid linkage*, which computes the mean vector of all objects in each cluster and uses these centroids to determine the distance, and *Ward's linkage*, which finds the potential increase in the sum of squares of all distances within the new, would-be cluster [Mil80].

Figure 4 shows a portion of the dendrogram produced for the PULS label set, using agglomerative hierarchical clustering (with cosine distance, and the centroid linkage method) on the 100-dimensional sector label representation output by LSA. From the figure one can see several similar labels clustered together; many of which could be merged into a single label. Merging of labels can either be done manually, or automatically by, for example, setting a height threshold under which clustered labels will be merged into a new label.

5.2 The Sector Hierarchy

One of the most problematic aspects of working with a large label set, collected from an external source or sources, can be the lack of a cohesive structure amongst the labels. As an example, consider the two sector labels *Oil, Petroleum* and *Energy, Fossil Fuels*. Although seemingly similar, it is unclear without looking at the positive-class data, the extent to which the two labels overlap, conceptually. Does the first label concern oil drilling, refining, distribution, or all three? Which label, if any, covers infrastructure and corporate management of the companies working with fossil-fuel based energy? Furthermore, regardless the level of conceptual overlap be-

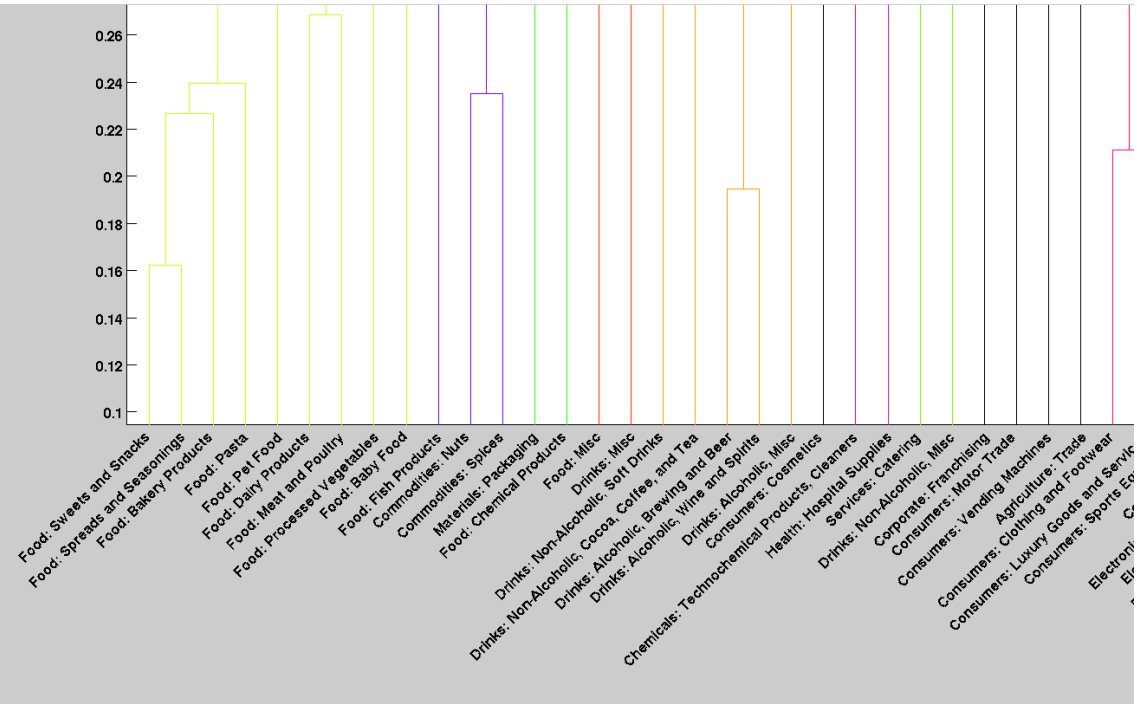
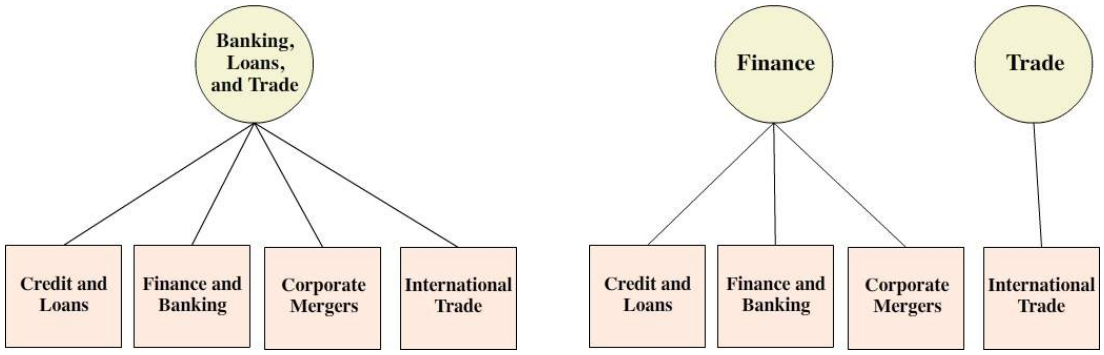


Figure 4: Portion of the sector-label dendrogram produced by LSA and agglomerative hierarchical clustering

tween the labels, it is likely that bag-of-words representations will produce many common features between the two, which could ultimately lead to misclassification between the two labels' classifiers.

Although there are advanced algorithmic approaches for predicting underlying hierarchical structures in data and applying them to training and classification, these methods require significant computation time and resources. A less-intensive method for bringing a measure of order to a large set of class labels is to create a simple two-level hierarchy by producing a set of general, super-class labels—referred to here as *level-I* labels—which serve as parent classes for all original labels—referred to in this context as *level-II* labels. Importantly, each level-II label would have only one parent level-I label, thereby sharpening the boundary between topics at this top level. This simple hierarchy can then be used in various ways in training, classification, and evaluation, to help eliminate some of the uncertainty brought on by closely-related or ambiguous labels in the original set. It is also a helpful tool for analyzing the general distribution and topic coverage present in the data, for example by comparing the number of child labels each level-I label is a parent of, after constructing the hierarchy.



(a) Non-optimal Mapping (b) Preferable Mapping

Figure 5: Two possible level-I mappings given four closely-related level-II sector labels

Methods such as latent semantic analysis, as discussed in Section 5.1.2, can also be employed to aid the construction of the level-I label set, by producing initial clusters of closely related topics. The most important aspect in the creation of upper-level labels is to ensure that all labels are less specific, conceptually, than any sub-class label, while at the same time on approximately the same level of generality as the other level-I labels.

To visualize how one might construct such a hierarchy, consider the following subset of original labels, that have been determined through their positive-class content (e.g., using LSA) to be closely related to one another: *Credit and Loans*, *Finance and Banking*, *Corporate Mergers*, and *International Trade*. Figure 5a shows a possible—but likely not optimal—mapping from level-II to level-I labels. One issue with this labeling is that the level-I label, *Banking, Loans, and Trade*, is too specific, as it is at the same level of specificity as both *Finance, Banking* and *International Trade*. It is also limiting, in the sense that it does not conceptually cover the label *Corporate Mergers*. Finally, it is important to avoid producing labels that are merely topic lists, since it is a somewhat unnatural representation to present for human consumption—particularly if the topics are not related that closely, such as finance and trade. Figure 5b shows a preferable mapping, consisting of two appropriately-general and user-friendly level-I labels that capture all original label concepts without forcing together any obviously distinct topics. Since constructing the hierarchy involves a number of subjective decisions, however, it is difficult to determine exactly how optimal a mapping is, and in this thesis the hierarchy is simply a best estimate, designed using the above principles.

Once constructed, there are a number of ways such a hierarchy could be employed, for example as a mapping for output labels or as additional feature data for training. In the experiments presented in Section 8, the sector hierarchy is used in a multi-stage classifier that leverages classifier models trained for both level-II and level-I sector labels.

6 Data Representation and Feature Engineering

Before a sector-classification system can be built for categorization, the input text must first be transformed into a meaningful representation. How informative, well-structured, and succinct the input is will strongly affect both the efficiency and accuracy of the classifiers. Different representations will afford different advantages. Additionally there are a number of data transformations for improving text classification, ranging from simplistic filters to complex probabilistic models. The experiments in this thesis explore various combinations of these representations and methods, in addition to some novel methods both for text extraction and feature selection.

6.1 Feature Extraction

To prepare articles for use in training, one must first map the raw text into an appropriate format for statistical learning algorithms. The traditional *bag of words* (BOW) representation transforms a text document into a feature vector of binary values—1 if a word appears in the document, 0 if it does not—in the space of all lexical features from a given data set (e.g., the training data). Word occurrence may be used in place of binary values, however this may introduce unwanted skew in data sets with highly variable document lengths. In the experiments presented in Section 8, data is represented using extended BOW features, where certain filters and transformations have been applied to enhance the quality of the input.

An important transformation to apply to BOW features is lemmatization. Lemmatization refers to the grouping of all differently-inflected forms of a word into a single, normalized form, known as a lemma [PLM+04]. For example, the words "run," "running," "ran," and "runs" are all reduced to the lemma "run." This immediately reduces dimensionality of the feature space, while also ensuring that all inflections of a term are directly correlated during statistical processing.

Infrequent words in a data set may be removed simply because they will not have a significant influence on the learning algorithms. The two common metrics to look at are term frequency—how often a word appears in the entire corpus—and term-document frequency—how many distinct documents a word occurs in. The experiments presented in this thesis use commonly recommended values of 3 and 2 for minimum term frequency and term-document frequency, respectively.

Certain parts of speech (POS), such as conjunctions and pronouns, consist mostly of words common to all texts; providing no discriminative value. These can be removed for a more compact representation. In the experiments, only nouns (excluding most proper nouns), adjectives, and verbs are retained in the feature set—as determined by the Stanford NLP POS-tagger [KM03]. Although proper nouns don't generalize well as predictors and are thus also candidates for filtering, there are certain exceptions that may be pertinent to keep. For example, the company name "Delta Airlines" would normally be filtered, even though "airlines" is a strong feature for a sector label such as "Air Transport." One method of ensuring this kind of proper name is retained for training is to keep all proper names that are also dictionary terms. This may be too lenient, however, as in the given example, this would also retain "delta" as a feature. In the experiments, therefore, a pre-computed list of *common endings* for company names—terms such as "Airline," "Power," "Financial Institute," etc.—is referenced when determining which parts of proper nouns to retain and which to remove.

In addition to filtering by part-of-speech, it is also recommended to remove certain words that may affect learning in undesirable ways. These *stop words* include tokens such as locations, dates, numerals, and common verbs like "have" and "do".

Traditional BOW features are limited by the fact that each feature represents a single, independent token, and ignores positional information. Commonly occuring word sequences may be more valuable indicators of a certain context than single words, for example "mobile phone," for telecommunications industry or "cable car" for transportation. One method for overcoming this limitation is to generate n-grams consisting of originally-extracted tokens that appear sequentially in the text. In the experiments, bigrams are generated from all original lemmas, and added to the feature space.

Figure 6 shows the lemmatization, filtering, and n-gram construction that is involved in the extraction process, for an example sentence.

Another limitation of BOWs is that ambiguous words, such as "card" or "plant", may

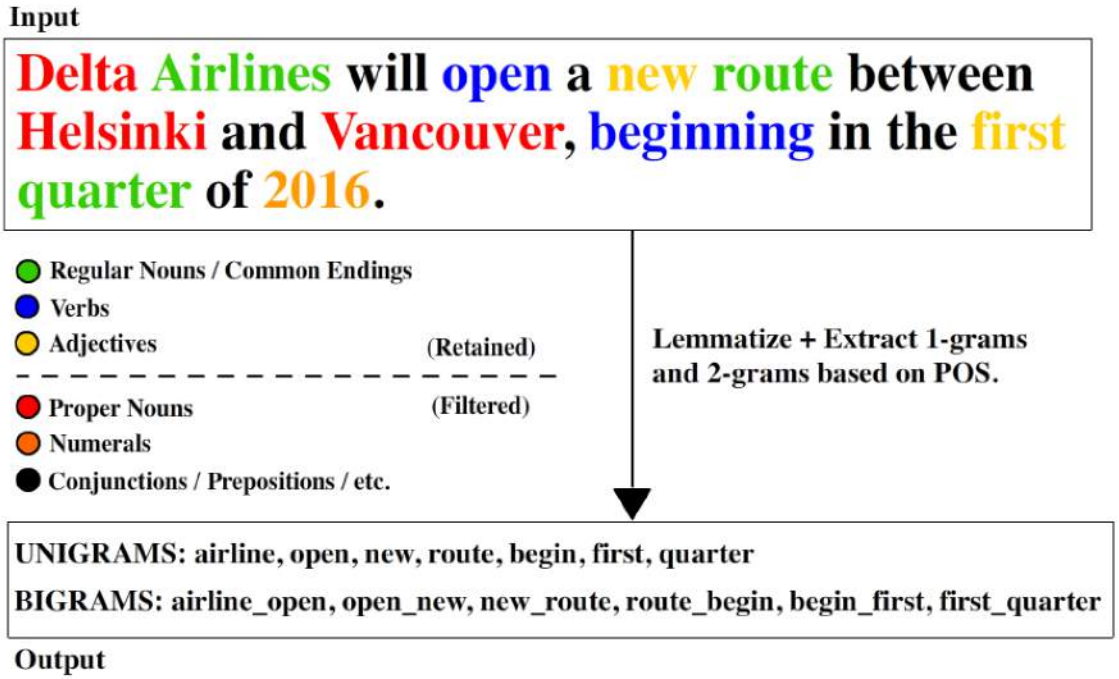


Figure 6: Lemmatization and feature extraction for a document sentence.

skew learning by appearing frequently in the training data of two or more unrelated classes. Bigram features can also be used to help overcome this: when indexing documents after feature selection (Section 6.2), a unigram is only kept as a feature if it appears *outside of any bigram features* extracted from that document. For example if the bigram "power plant" appears in a document, then "power" or "plant" are only considered as independent features for that instance if they also appear elsewhere in the document (and not in another extracted bigram). This helps resolve ambiguity to some extent; for example, documents containing the feature "SIM card," which may be relevant for *Telecommunications*, can now be distinguished from documents containing the feature "credit card," which is relevant for *Commercial Banking*.

Once feature vectors have been extracted from the text, there are a number of other transformation methods that may improve either the informativeness or efficiency of the data representation. Normalizing vectors to unit length can be effective when documents in the data set have highly variable length. Another common data normalization method is to use the *Term-Frequency Inverse-Document-Frequency* (TF-IDF) representation, where each feature is represented with the value given by:

$$\text{tfidf}(t, d, D) = f_{t,d} \cdot log \frac{N}{|d \in D : t \in d|} \tag{5}$$

where $f_{t,d}$ is the occurrence of term t in document d (this is basic term-frequency,

however, various weighted scores have also been proposed for this value), N is the total number of documents in the data set, and the denominator term is the number of documents in the data set in which the term appears.

Finally, dimensionality reduction is a very powerful way to maximize the effectiveness of textual features. One way to reduce dimensionality is feature selection, which is discussed in-depth in Section 6.2.

For an example of the feature-space that is produced by the extraction method outlined above, the training data collected from the RCV1 data set, consisting of 77,636 training instances (documents), generates 49,262 unique lexical features.

6.2 Feature Selection

The motivation of feature selection is two-fold: 1) to project the data to a lower-dimensional feature space that is more compact and therefore promotes efficient training and classification, and 2) to retain only those features that provide useful discriminative information for a particular class. The informativeness of a feature for a binary classifier may be either *positive* or *negative* depending on the class value it promotes. For example, the learner for the sector label "Farming and Livestock" may benefit highly from knowing the positive feature "cow" in determining if a data point is a member of its class, but it may benefit equally from knowing the negative feature "diamond," which suggests the data point is a non-class member. Conversely, a feature such as "profit" is too ambiguous to provide either positive or negative information for that label's learner, and should be discarded. The goal, then, of feature selection is to somehow rank the discriminative power of each feature present in the entire feature space, for a single class. There are a number of algorithms for feature selection that have shown to improve performance. In the experiments, two of these methods—*Information Gain* (IG) and *Bi-Normal Separation* (BNS)—are explored, in addition to a novel method based on feature entropy, that is referred to by the author as *Entropy Filtering*.

6.2.1 Information Gain

The information gain (IG) feature selection method [For03] is a rank-based method that utilizes the information gain criterion defined by:

$$IG(Y|X) = H(Y) - H(Y|X) \tag{6}$$

where $H(Y)$ is the initial entropy of probability distribution Y and $H(Y|X)$ is the conditional entropy of Y given the known distribution X.

When applied to feature selection, $H(Y)$ refers to the entropy of initial class distribution, C, of a given label. $H(Y|X)$, therefore, refers to the conditional entropy of the class distribution given the distribution of the feature, F, among distinct class values. The formula can therefore be rewritten as:

$$IG(C|F) = H(C) - H(C|F) \tag{7}$$

Intuitively, the IG score can be seen as an indicator of "information gained" about a class, when the value of a specific feature is known, regardless of the value of that feature. If information gain is 0—the minimum possible value, since conditional entropy can never be higher than initial entropy—then the natural distribution of the class is completely independent of that feature, which means that the feature provides no discriminative information about the value of that class.

For feature selection, the IG score for the given class is first determined for every feature in the original feature space. Once these values are obtained, the feature selection process is then concluded by ranking all features by their associated IG score and retaining either the subset of N top features, or by throwing away all features with a values below a tolerance threshold t. The optimal values for these parameters may be determined either experimentally, by comparing the performance of a classifier trained multiple times, over various values of N or t; or manually, by inspecting the feature sets collected for a classifier (i.e., the "eye test"). Tuning parameters such as these experimentally is always preferred, however, may be impractical or infeasible in reality, depending on data size, model complexity, or the availability of computing resources or time.

6.2.2 Bi-Normal Separation

Bi-Normal Separation (BNS) is another feature-ranking method, developed by Forman [For03], that produces a single score for each lexical feature, given an associated class. The general score function for BNS is given by:

$$BNS(f, C) = |\Phi^{-1}(tpr(f, C)) - \Phi^{-1}(fpr(f, C))| \tag{8}$$

where tpr and fpr are the true- and false-positive rates of feature f given class C, respectively—here, true-positive means the feature occurs with the positive class

of C and false-positive means it occurs with the negative class of C—and $\Phi^{-1}(x)$ is the inverse cumulative probability distribution function of the standard Normal distribution of a random variable described by x [4]. When parameterized with tpr the function returns the threshold for the quantile of the Normal distribution that is described by tpr. The final score for a feature, given a class, therefore corresponds to the distance on the Normal probability curve for the quantiles described by tpr and fpr, given a single class. If the difference is small, there is little separation between the expectation of a feature occurring in the positive class versus its expectation in the negative class, and the feature has low discriminative power. Conversely, if the difference is large, the feature is expected to occur more commonly in either the positive or negative class, and should be considered a good feature.

6.2.3 Entropy Filtering

One challenge of using cross-training with binary-relevance classifiers is in selecting those features that will be highly discriminative for a given class yet more unique than other features, with respect to *all* classes. Most popular feature-selection methods excel at solving the first problem, however, are unable to consider the second case, as positive data from other classes are grouped into a single negative class for training. The difficulty of ensuring unique terms as features is exacerbated by the presence of ambiguous words. For example, the word "plant" may be a strong indicator of the two sectors "Fruits and Vegetables" and "Electrical Power Generation", however, confusing these during training will have adverse effects on performance.

The novel feature-selection method Entropy Filtering (EF) aims to solve this by eliminating features that have high total entropy with respect to their frequency distribution over all classes. This is given by:

$$H(f, C) = -\sum_{c \in C} p(f, c) \cdot log_2(p(f, c)) \tag{9}$$

where C is the set of all possible class values, and $p(f, c)$ is the frequency of feature f occuring in documents having class value c.

A high entropy indicates a word is distributed evenly between classes, therefore, after removing all features having entropy greater than a threshold, t, the remaining

[4] Note that the value of $\Phi^{-1}(0)$ is undefined, and therefore set to $\Phi^{-1}(0.0005)$ to prevent algorithmic errors.

set will contain only the most unique terms.

It is important to make a distinction between entropy filtering and the information gain (IG) method described in Section 6.2.1, as both measures are based around entropy scores. Whereas IG ranks features for a *single* class by taking its prior entropy then subtracting its conditional entropy given a known feature value, EF finds, separately, the individual entropy scores of the features, over *all* classes. Features with high entropy across all positive class data for all labels are removed from the feature space prior to training, thus any individual classes that may have previously used those features (e.g., because of a high IG score) no longer have access to them.

EF can be used on its own or in conjunction with other feature selection methods. When used with another method, such as IG—which produces the combined feature-selection method IG+EF—the process works as follows:

1. Tabulate entropy scores for all features F across all classes C
2. Collect N features, F_i, for each class, c_i, with an initial feature-selection method (IG or BNS)
3. Remove all features from F_c, where $H(f, C) > t$
4. Train each classifier, c_i, on its respective set of final features

The threshold value t may be manually or experimentally determined for a given data set. As bigrams naturally occur more rarely than unigrams, their entropy scores will be lower on average. Using a single threshold will filter out fewer bigrams, even if the actual terms in the bigram feature are just as common (i.e., consider that the unigrams 'sale' and 'price' will have a higher entropy than the bigram 'sale price' but if we don't want the first two, we likely don't want the third). Individual thresholds could, therefore, be used for each. For example, in the experiments on PULS data, thresholds of $t_1 = 5.0$ for unigrams and $t_2 = 4.0$ for bigrams are found by manually appraising the quality of features from a sorted list and determining the point where feature quality begins to degrade significantly.

6.3 IE Features

Although BOWs is a good general-purpose feature representation, in the sense that it can be applied to any problem domain, one advantage of working with the PULS system is having access to sophisticated IE tools that can pull meaningful features

directly from the text. These IE features can be seen as informative without requiring the overhead of feature-selection, as they are a known entity at extraction time. The two IE features collected by the PULS system that are leveraged for use in industry-sector classification are *company name* and *company descriptor*. These features are used in a rote classifier, described in Section 7.1.

To parse and extract these elements from text, PULS uses a Named-Entity Recognition (NER) module. In brief, the NER module is a pipeline that first invokes a part-of-speech (POS) tagger, before passing the annotated text to a hierarchy of low-level patterns and dictionary terms for matching. Positive matches are then analyzed by a set of high-level heuristics that try to guess the correct entity-type. These entities are stored in a knowledge database alongside many other metadata, including, in the case of documents from the PULS data set, the annotated sector label. A more detailed description of this process, and the novel use of IE features in sector classification, is found in our prior publication [DPPY15].

7 Automated Sector Classification

7.1 Baseline: Rote Classifier

The PULS IE system (described in Section 3) is used, amongst other tasks, to build a knowledge base that contains sector distribution information for each company mentioned in an existing business corpus (i.e., Reuters or PULS). Part of this research investigates ways to use this information for text categorization, and one of the main ways it is utilized is in the construction of a *rote classifier*[DPPY15].

Using the NER module described in Section 6.3, the IE system finds mentions of companies in the corpus. It distinguishes company names from other proper nouns in the text, such as persons or locations. The NER module also merges variants of the same name that appear in the text, for example, “Apple,” “Apple Inc.,” “Apple Computer, Inc.,” etc., would all be merged into a single entry in the knowledge base.

In short, the knowledge base contains the following many-to-many relations:

- Document ↔ Sector
- Document ↔ Company
- Document ↔ Descriptor
- Company ↔ Descriptor

Table 3: Sector distribution for company "Apple"

Sector	**Freq**	**Prob**
Computer Systems and Software	549	0.61
Electronic Active Components	61	0.07
Datacommunications and Networking	36	0.04
Telecommunications	19	0.02
Electrical and Electronic Engineering	13	0.01

Where *Document* is a single news article from one of the business corpora, *Sector* is the manually-assigned industry label, *Company* is a proper noun that the IE system recognizes as a corporate entity, and *Descriptor* is a noun or adjective that describes or augments a company in the text (e.g., "manufacturer"). Various combinations of these entity-relationships can be used to build the rote classifier.

The assumption of the rote classifier is that each company has its own sector preferences: the set of industries in which it typically operates. Consequently, company names in the corpus should only co-occur frequently with particular sectors. For example, Table 3 shows the top sectors that co-occur with "Apple," in both frequency (number of co-occurring entries in the knowledge base) and proportion of that relationship over the company's entire sector distribution. It can be seen from the table that in 60% of cases Apple is mentioned in documents labeled with the *Computer Systems and Software* sector, thus it is natural to suggest that documents mentioning Apple will typically belong to this sector. However, as each document may belong to more than one sector, instead of only choosing the top-most frequent sector the classifier returns the entire sector distribution, which is calculated using evidence from all companies mentioned in the text. Thus the probability that document D belongs to sector S, in the simplest case, can be defined by the formula:

$$P(S|D)_{name} = \frac{1}{|C_D|} \cdot \sum_{c \in C_D} P(S|c) \tag{10}$$

where C_D is the set of companies mentioned in the document, and $P(S|c)$ is the proportion of times c co-occurs with S in the knowledge base; e.g.:

$$P(Computer\ Systems\ and\ Software|Apple) = 0.61 \tag{11}$$

(from Table 3). Note that although a company may be mentioned in a document several times, it is only counted once in the knowledge base.

This method performs reliably as long as the knowledge base contains sufficient evidence to associate the company with particular sectors. Therefore, it is important only to use company names that appear in the corpus frequently enough and throw away results for all companies below a so-called *garbage threshold*, for example, in the experiments of this thesis a garbage threshold of 3 is used. The result of using a garbage threshold is that if a document discusses a new (or little-known) company, the rote classifier would be unable to find a sector for the document. In this case descriptors could be used to label the document, as descriptors allow for the use of evidence gained from *other* companies in the corpus. For example, if company X is described in the text as "software company" one can assume that the sector distribution for this company would be similar to the sector distribution for "Apple". In this case the probability that document D belongs to sector S can be described by the formula:

$$P(S|D)_{name+desc} = \frac{\sum\limits_{c \in C_D} P(S|c) + \sum\limits_{d \in d_D} P(S|d)}{|C_D| + |d_D|} \tag{12}$$

where d_D is the set of all descriptors mentioned in the document. Note that $|C_D| \neq |d_D|$ because descriptors may be used for multiple different companies (or may not appear at all). Descriptors can also be used that do not co-occur with any particular name; e.g., if the document mentions "IT companies," but does not specify company names, this mention can still be used to help classify the document.

This estimate of $P(S|c)$ based on co-occurrence may be inaccurate: for rare companies, some sectors may dominate the distribution by mere chance. Moreover, sector overlap may lead to a situation where the company belonging to one sector frequently co-occurs with another. Descriptors, therefore, may sometimes be more reliable for predicting the sector. To check this assumption, a probability is defined for a company belonging to a particular sector as follows:

$$P(S|c) = \sum_{d \in d_C} P(d|C) \cdot P(S|d) \tag{13}$$

where d_C is the set of all descriptors associated with company c in the knowledge base. We then use (13) in (10) to obtain the final sector distribution for the document:

$$P(S|D)_{name \rightsquigarrow desc} = \frac{1}{|C_D|} \cdot \sum_{c \in C_D} \sum_{d \in d_C} P(d|C) \cdot P(S|d) \tag{14}$$

Note that in this case the company name is substituted by a set of descriptors; however it is possible to use the company name in combination with company de-

scriptors:

$$P(S|D)_{name+name \rightsquigarrow desc} = \frac{\sum_{c \in C_D} \sum_{d \in d_C} P(d|C) \cdot P(S|d) + \sum_{c \in C_D} P(S|c)}{2 \cdot |C_D|} \quad (15)$$

The performance of various combinations are evaluated and compared in Section 8.2.1.

7.2 Supervised Learning

Rote classification is useful in situations where the data set contains a strong correlation between a single piece of prior knowledge and specific class labels, such as the scenario described in Section 7.1 between company names (or descriptors) and sector labels. In practice, however, there will commonly be either no strong correlate in a data set or a large subset of data not containing an instance of the required knowledge. The latter is the case with both the PULS and Reuters data set, where the PULS IE system is unable to extract a company name or descriptor for a significant number of documents and would be therefore unable to make predictions for them using rote classification. In this case, more intelligent methods are required. This section will explore the application of statistical modelling and supervised learning, to the problem.

A method for performing multi-label classification with statistical classifiers, and the method adopted in this thesis, is *binary relevance*: building a single binary classifier for each unique label in the set and aggregating predictions from each to classify a test instance. More advanced methods involve extending traditional single-label methods to handle multiple labels, but require greater resources for learning and classification and are therefore less suitable for the online PULS system.

7.2.1 Training Data Selection

Collecting training data for supervised multi-label classification poses unique challenges. One significant one is the problem of data balancing. Under the cross-training scheme described in Section 2, if a particular sector S_1 is dominant in the training set, the negative features for other classifiers could become dominated by features drawn from S_1, which may hurt performance on some other sector, S^*, since it won't learn negative features from other, "minor" sectors (those having fewer documents in the corpus). If S_1 is also over-represented in the test set, there is a risk of over-fitting during evaluation. For these reasons one might want to keep the training

Table 4: Number of positive instances in the training pool, for the most frequent sectors (RCV1 data)

Sector	**Instances**	**Sector**	**Instances**
Diversified Holding Companies	3644	Electricity Production	1986
Commercial Banking	3153	Agriculture	1980
Petroleum and Natural Gas	2628	Computer Systems and Software	1805
Telecommunications	2145	Air Transport	1754
Metal Ore Extraction	2099	Passenger Cars	1713

data as balanced as possible across labels, and ensure that the test set will contain a sufficient number of instances for every binary classifier in the array.

In various experiments presented and evaluated in Section 8, training and testing data is generated both using traditional (random) data collection and a novel, specialized balancing procedure. This procedure, described here, is also explored in our previously published work [DPPY14]. The motivation for the procedure is to prevent learning skewed prior probabilities for the sector distribution based on the training corpus, since these probabilities are likely to change over time.

In cross-training, a single set, or *pool* of data are shared amongst all binary-relevance classifiers. In creating a *balanced* training pool, the aim is to provide all classifiers with a sufficient number of positive examples for each label. Creating a balanced test pool similarly ensures that there will be at least some data for which to test each individual classifier with. Because of this, both the training and testing pools can be generated simultaneously. Ranking the sectors by size, from 1 to N, data is collected into the pools from the sector, S_N, that has the smallest number of instances in the corpus. [5] Up to k positively-labeled instances (e.g., $k = 600$) for this smallest class are randomly selected and split into two subsets: 3/4 for the training pool and 1/4 for test. If there are not enough documents (e.g., < 600) for S_N, all available instances are collected, with the same training/test proportion.

Once data for the smallest sector is collected, the procedure then moves to the second smallest sector, S_{N-1}, and repeats the collection process with one additional

[5]Otherwise there is no guarantee that each sector will have a sufficient number of instances in the training and test pools. For example, if collection of the training and test data is done in random order and happen to start with the largest sectors, then by the time the smallest sectors are sampled all of their data may already be included in the training pool (due to multiple labeling of documents), leaving none for testing.

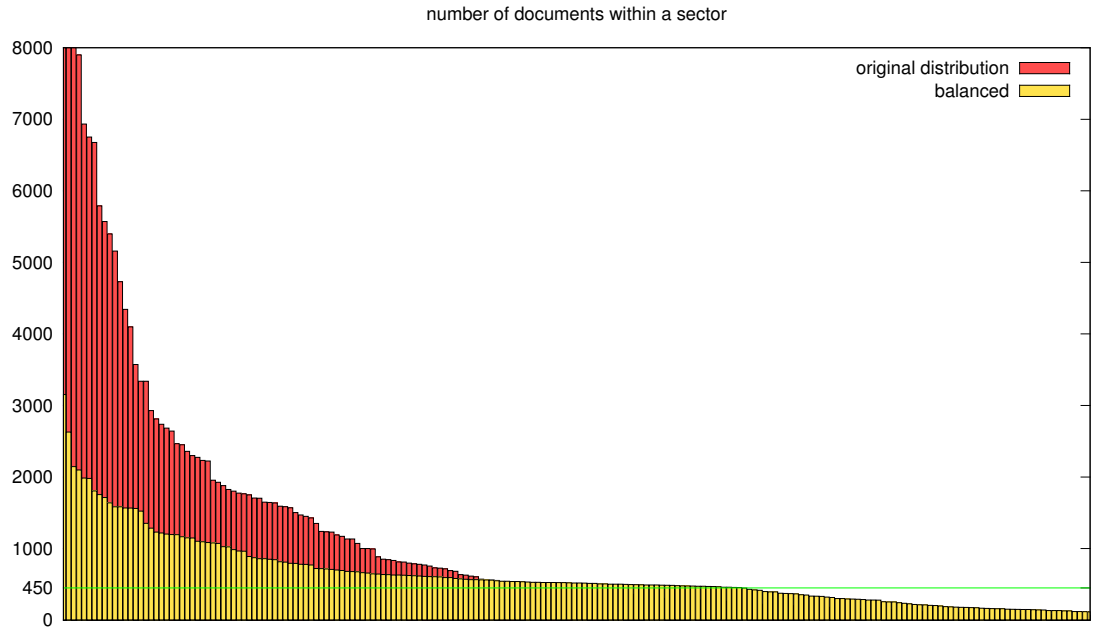


Figure 7: Document distribution among sectors in the original data (RCV1) versus the balanced training pool: aiming for approximately 450 documents per sector (horizontal line).

step: an initial check is made to determine how many documents labeled with S_{N-1} are *already present* in the training and test pools—which may happen due to label overlap. The number of documents collected for S_{N-1} at this step is reduced by the number already collected.

The collection process continues in this manner for all sector labels. Collection may be skipped for a label if it already has more than $\frac{3}{4}k$ documents in the training pool and $\frac{1}{4}k$ documents in the test pool (this happens for sectors with high label overlap). It is also possible that some sectors will have fewer than the expected number of documents for training, based on total availability. These are inherent limitations of the skew in the original corpus, and cannot be avoided.

Figure 7 shows the comparison between the balanced training pool collected for the experiments in this thesis, and the full corpus for the Reuters RCV1 collection. For these experiments, the value for k was set to 600 (i.e., 450 training and 150 test documents per label). The figure shows that although still skewed, the balanced data is much more evenly distributed compared to the original label distribution.

Table 4 shows the most frequent sectors in the balanced training pool. Note that although only 450 positive training instances were collected explicitly for the *Diversified Holding Companies* label, the procedure still obtains 3644 positive instances for the pool; the majority of which were picked up when collecting data for other sectors.

7.2.2 Statistical Classifiers

Statistical classification algorithms have been studied extensively, and an exhaustive survey is out of the scope of this thesis. Nevertheless, it is worthwhile to compare the performance of a few well-known algorithms, within the sector-classification framework, to gain insight into the relative merits and drawbacks of these methods. Additionally, it allows for easier comparison to other works, as these algorithms are the ones most commonly reported on.

Naive Bayes Naive Bayes classification is based on Bayes' Theorem, which defines the following conditional probability formula for random variables A and B:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{16}$$

where $P(A)$ and $P(B)$ are the prior probabilities of A and B, and $P(B|A)$ is the conditional probability of B given A.

This rule can be leveraged for class induction by setting the target variable A to be a given class variable C_k—where k is the number of possible class values—and taking the values of a given feature vector X as the evidence variable, B. The formula then becomes:

$$P(C_k|X = [x_1, x_2, ..., x_n]) = \frac{P(X|C_k)P(C_k)}{P(X)} \tag{17}$$

Intuitively this induction step requires prior knowledge—specifically, a set of training data for which the class values and feature vectors are known—in order to calculate all right-hand values. The larger the training set, the more reliable the estimates; provided the "true" labels are accurate and the feature representation is meaningful.

The term "naive" refers to the implicit assumption of the Bayes' Theorem that a probability can be fully described by completely independent observations of the evidence variable. In reality, the dimensions of a data vector may have any number of complex dependencies. An example from text categorization would be the presence of the words "coffee" and "tea" together within a document. Although these terms likely occur together frequently throughout the training data, the Naive Bayes classifier has no way of integrating this dependency information into it's calculation of a class value for a label such as "Food and Drinks."

Despite the naive assumption, however, Naive Bayes classifiers have shown strong performance in traditional classification settings [Ris01].

Support Vector Machine The Support Vector Machine (SVM) algorithm has become one of the most popular linear classification algorithms in recent years, due to its strong performance on traditional classification tasks, including text categorization; its lightweight implementation and storage requirements; and its tunability [ZZY+05].

SVM learners are built by finding the most appropriate $(n-1)$-dimensional hyperplane between n-dimensional data points of a binary class distribution. For example, given a set of 2-dimensional training data drawn from a random variable consisting of two classes, the SVM might produce a prediction boundary such as the solid line shown in Figure 8.

The hyperplane can be defined as the set of datapoints, X, that satisfy the equality:

$$w \cdot X - b = 0 \tag{18}$$

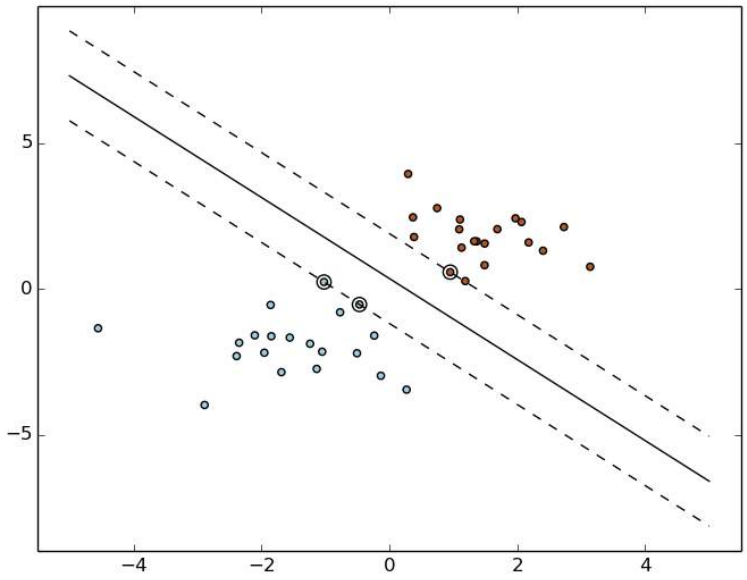


Figure 8: SVM hyperplane separating 2-dimensional data (Produced by Scikit-Learn [PVG+11]).

where w is the normal vector to the hyperplane and b determines the offset of the hyperplane from the origin along w.

The hyperplane is generated from training data by performing distance calculations between two other hyperplanes, constructed using opposing-class exemplars from the data that lie close to the boundary between the classes (the dashed lines from Figure 8). These exemplars are known as "support vectors," and are the only training data required by the algorithm, to build the hyperplane. The low space requirements of an SVM learner during testing is owing to this reliance on a limited subset of the training data.

Intuitively, the performance of the SVM classifier will depend on how well the margin is constructed to fit the underlying class distribution. Given a training set:

$$\{(x_i, y_i)\}_{1 \leq i \leq n},\ x_i \in \mathbb{R}^{\mathrm{d}},\ \mathrm{y_i} \in \{+1, -1\} \tag{19}$$

Constructing the best hyperplane is a quadratic programming optimization problem of the form:

$$\underset{(w,b)}{\arg\min} \frac{1}{2}||w||^2 \quad \text{s.t.} \quad y_i(w \cdot x_i + b) \geq 1 \quad \text{for} \quad i = 1, ..., N \tag{20}$$

The SVM algorithm was made more flexible by the introduction of a "soft margin" that allows data to be split as cleanly as possible in the event of linear inseparability. This alters the form of the optimization function to:

$$\arg\min_{(w,\xi,b)} \left\{ \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i \right\} \text{ s.t. } y_i(w \cdot x_i + b) \geq 1 - \xi_i; \quad \xi_i \geq 0 \tag{21}$$

where the ξ_i values are non-negative variables that will measure the misclassification error with the support vectors. The weight to which this factor affects the overall optimization is controlled by the complexity constant C, which can be experimentally tuned in order to further improve the prediction boundary for a data set.

Although SVM requires less memory than methods such as Naive Bayes, the time requirement of the traditional SVM learning algorithm is quadratic. One method that was developed to counter this drawback is the Sequential Minimal Optimization (SMO) algorithm [P+99a]. SMO divides the algorithm from a single large quadratic programming (QP) optimization problem into a set of minimally-sized QP problems that can be solved analytically—removing the need for an internal optimization loop and significantly reducing runtime. SMO is the training method utilized for building the SVM learners in the experiments of this thesis.

Finally, another drawback of the traditional SVM is its binary output, however, methods have also been developed to map linear SVM predictions to probabilistic confidence scores, allowing for further performance-based tuning. One method of achieving this is to train the SVM as a *kernel classifier*—a learner whose input includes both a data set and a transformation function to be applied to the data during training—with a logit link function and a maximum likelihood score, to produce class-conditional probability distributions. Another method, which is adopted in this thesis, is to fit logistic models to SVM outputs by first training the SVM, then training the parameters of a sigmoid mapping function, using maximum likelihood estimation on the training data [P+99b].

7.3 Multi-Stage Classification

Previous machine-learning research has shown that classification performance can be significantly improved by combining simpler methods into a larger classification system, as seen in techniques such as boosting and stacking [HE07]. Experiments in this thesis explore the combination of results from multiple classification stages. These stages may be totally independent classifications, such as rote classification

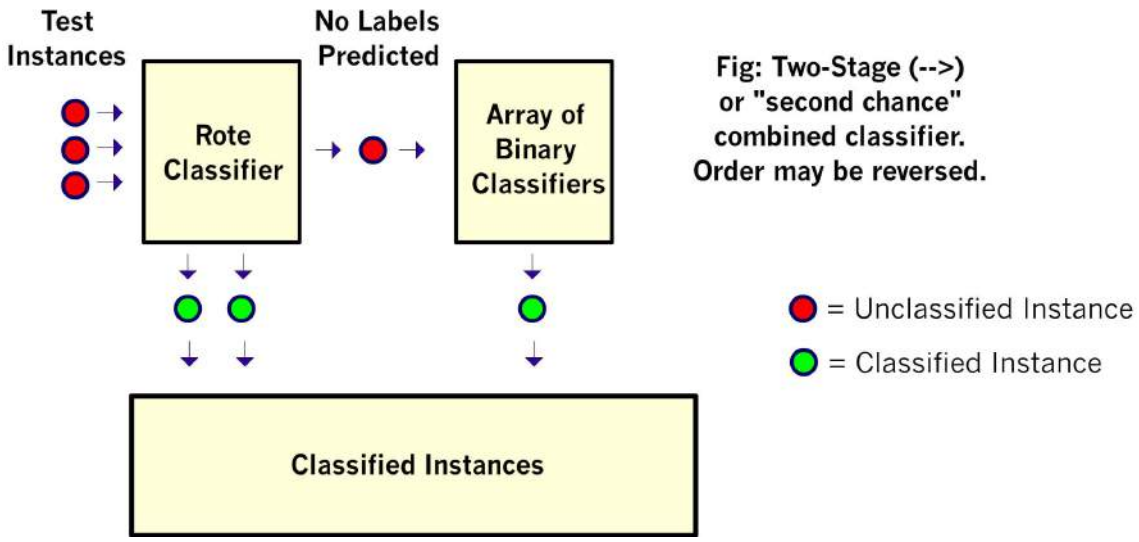


Figure 9: *Pipeline* Combined Classifier

and statistical classification, or be based on a multi-level hierarchy, like the two-level hierarchy described in Section 5.2. Specifically, two methods have been developed for combining the output labels of a multi-stage classification. These methods, and associated results, are also described in our prior publications [DPPY14, DPPY15].

The *pipeline* combined classifier is shown in Figure 9. The example from the figure shows how the system first returns any labels it can find using the rote classifier and, if no labels are found above the prediction threshold, proceeds with classification using an array of binary, statistical classifiers (e.g., binary-SVM classifiers). Predictions are therefore produced by either rote classification or statistical classification, but never both (and possibly neither). The idea behind the pipeline classifier is to give the system a "second chance" at classifying documents, since each method has different strengths that may allow it to classify documents the other cannot.

The two stages of the pipeline method can be reversed, so that statistical classification occurs first, followed by rote classification. This ordering may change the final predictions considerably if, for example, the rote classifier were to predict with high precision but low recall, while the statistical classifiers predict with high recall but low precision.

A variation on the example shown in the figure is to use the level-I statistical classifiers, as described in Section 5.2, in place of the rote classifier stage, thus leveraging the hierarchy structure. Depending on the order, this equates to either "backing off" from potentially too-specific classification or "drilling down" from too-general

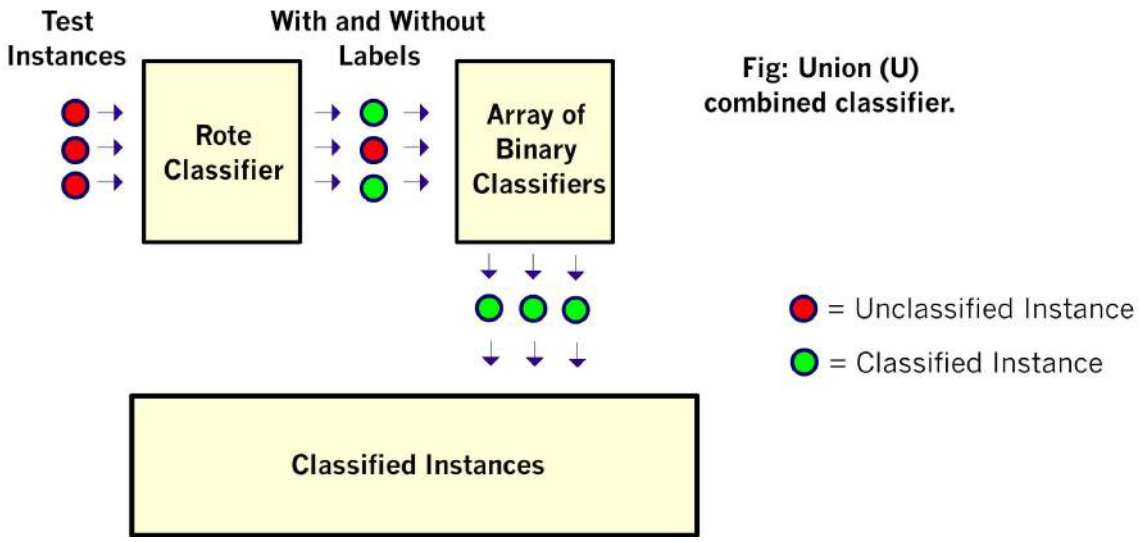


Figure 10: *Union* Combined Classifier

classification. Since this method involves classification directly at level-I it necessitates that evaluation also be done at level I, since the mapping of level-II to level-I labels is many-to-one, so it is impossible to evaluate predictions of level-I classifiers at level-II.

The second multi-stage classifier, the *union* combined classifier shown in Figure 10, is not dependent on the order of the classification stages, and simply returns—as the name would suggest—the union of label predictions from each stage. This mode of classification focuses on improving recall, with the hope that precision will not be negatively affected by pooling all results. As with the pipeline combined classifier, a "level-I" stage can be substituted for the rote stage, in an attempt to leverage the hierarchy information.

Finally, the two combined classifiers outlined above are both extendable to handle the general N-Stage case, given N distinct classification methods. Given the methods described in this thesis already, an obvious experiment would be to combine rote classification, level-II classification, and level-I classification, in a three-stage classifier.

The performance tradeoffs of each of the various permutations described here are explored in-depth in the experiments of Section 8.3.

7.4 Implementation

As outlined previously, multi-label classification involves the coordination of several distinct steps, each with different requirements that pose unique challenges. Using state-of-the art algorithms and implementations at each step is highly recommended. For this reason a number of publicly-available tools were utilized in the development of the sector classification system.

The initial extraction of lexical features from news articles is aided by the Stanford NLP group's software toolkit [KM03]. In particular, the part-of-speech tagger is used to identify and filter unnecessary tokens, thereby reducing initial data dimensionality (see Section 6.1).

Many learning algorithms, such as SVM, have multiple implementations, which may be better suited for particular scenarios. For example, some may trade off speed for accuracy, while others may replace internal calculations with approximations to reduce complexity. Efficiency is of paramount importance as factors such as large data sets, high data-dimensionality, large label sets, complex class dependencies, algorithmic complexity, and parameter tuning all increase the space and time requirements of learning and prediction tasks. Therefore, well-optimized, well-tested statistical software libraries are needed to improve the overall performance and speed of the system and related experiments. The WEKA toolbox [HFH$^+$09] is a specialized data-mining and machine-learning library that contains optimized implementations for a number of learning algorithms. In addition, it provides high-level tools for data-handling, feature-selection, parameter-tuning, and evaluation techniques, such as cross-validation. In this thesis, WEKA libraries are used for training and running all statistical classifiers, as described in Section 7.2.2, for the experiments presented in Section 8. It is used for information-gain feature selection as well as for directly interfacing with the ARFF-format data files that represent the various data sets.

Additional statistics-based algorithms, such as bi-normal separation and entropy-filtering feature selection methods, were implemented by the author, in Python, with reliance on the SciPy math and stats toolkit [JOP$^+$].

MathWorks' MATLAB software [MAT13] was also used for certain data transformations, in addition to producing calculations and figures used for verifying results and methodologies.

Beyond statistical algorithms, another important facet of optimizing the implementation is the use of distributed processes for parallel computing. In the majority of

the experiments presented, training-data size alone prohibits the handling of more than a few learners at once on a single machine. Given that the statistical classification involves hundreds of classifier models per scenario, it would take weeks or months to produce all results. Therefore, to expedite training and testing—allowing for relatively fast comparisons between the numerous scenarios—a high-performance cluster of approximately 200 nodes was used for concurrent processing, in a number of tasks. GNU Parallel [Tan11] is used to facilitate the distribution of these tasks. An example is shown in Figure 11, which gives a high-level view of feature extraction, done using distributed processing with GNU Parallel. Here, the master process generates the initial list of hundreds of thousands of training document IDs and splits it into k files of manageable length. GNU Parallel is then invoked to distributed the processing of these files to k different cluster nodes, which return their individual results to the master process, to be combined into the final feature set. In addition to extracting lexical features, this model is used for constructing data representations, performing feature selection, and building classifier models.

8 Experiments and Evaluations

8.1 Evaluation Measures

Common measures in text classification are accuracy, precision, recall, and F-measure. For a given class c, accuracy is defined as:

$$Acc_c = \frac{TP_c + TN_c}{TP_c + TN_c + FP_c + FN_c} \tag{22}$$

and precision, recall, and F-measure are given as:

$$Rec_c = \frac{TP_c}{TP_c + FN_c} \qquad Prec_c = \frac{TP_c}{TP_c + FP_c} \qquad F1_c = \frac{2 \cdot Rec \cdot Prec}{Rec + Prec} \tag{23}$$

where TP_c, TN_c, FP_c and FN_c are the number of true-positive, true-negative, false-positive, and false-negative classified instances for the class, respectively. The number of documents in the test pool labeled with the given class is referred to as $|c|$.

As can be noted from the formulae, accuracy takes into account both true-negative and true-positive instances, whereas F-measure focuses on true-positive instances only. If the data is heavily unbalanced, accuracy won't be too informative, since the negative instances significantly outnumber the positive instances [IKT05, BPM04]. For this reason, accuracy is not reported in any evaluations.

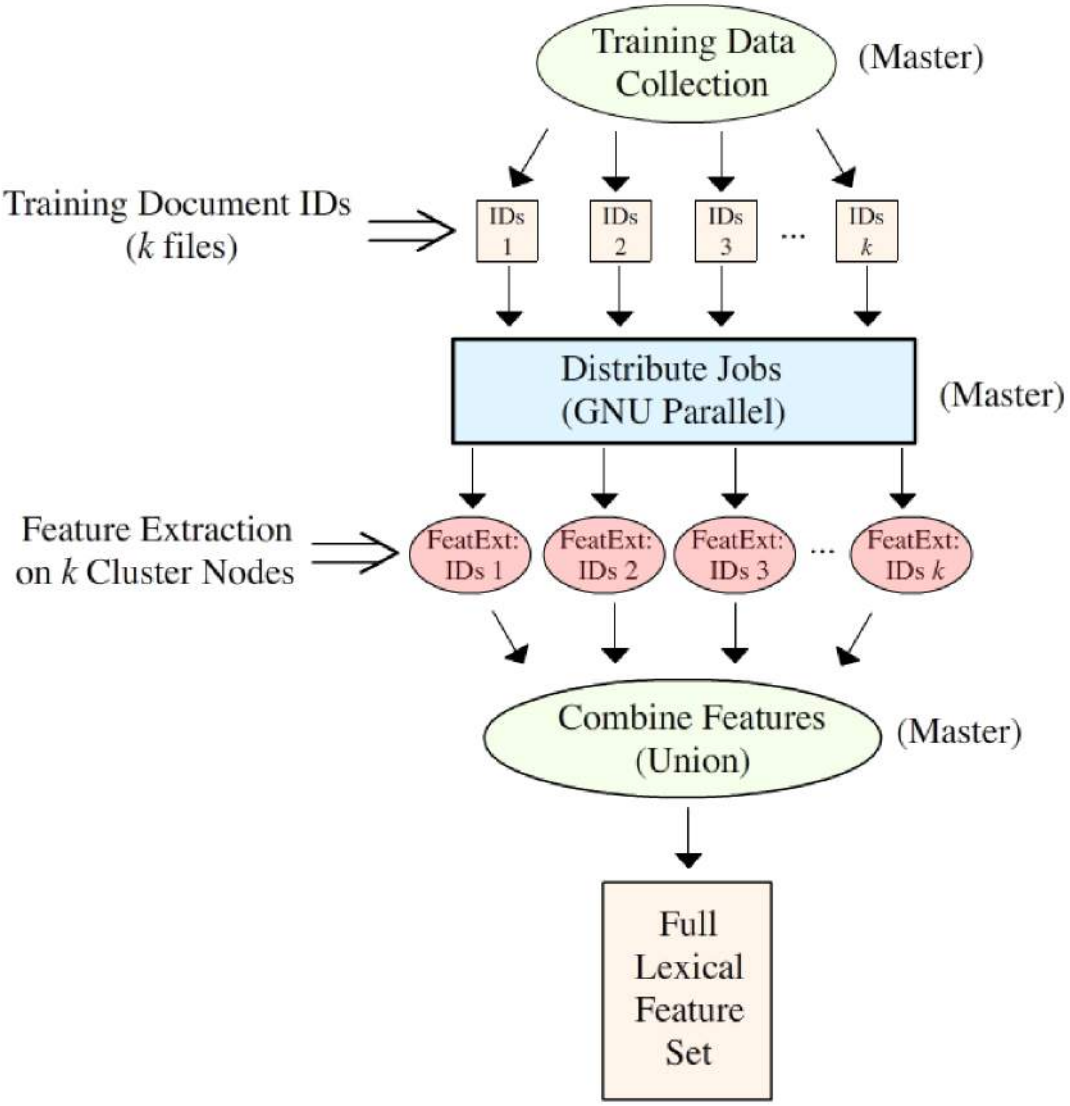


Figure 11: Distributed feature extraction using GNU Parallel

In evaluating multi-label classification, *macro-averages* and *micro-averages* are commonly reported [Yan99]. Micro-averaging takes the counts of all true- and false-positives, and true- and false-negatives for all instances in the test set, then uses the following calculations for recall, precision, and F-measure:

$$Rec_\mu = \frac{\Sigma_{i \in S} TP_i}{\Sigma_{i \in S}(TP_i + FN_i)} \qquad Prec_\mu = \frac{\Sigma_{i \in S} TP_i}{\Sigma_{i \in S}(TP_i + FP_i)}$$
$$\mu\text{-}F1 = \frac{2 \cdot Rec_\mu \cdot Prec_\mu}{Rec_\mu + Prec_\mu} \tag{24}$$

where S is the set of all classes. The micro-average score is a *per-instance* evaluation that does not penalize a multi-label system for bad performance on any one particular class label.

In the macro-average evaluation scheme, the standard measures are first calculated for each class *separately*, and then averaged across all classes:

$$Rec_M = \frac{\Sigma_{i \in S} Rec_i}{|S|} \qquad Prec_M = \frac{\Sigma_{i \in S} Prec_i}{|S|} \qquad M\text{-}F1 = \frac{\Sigma_{i \in S} F1_c}{|S|} \tag{25}$$

Macro-average is a *per-class* evaluation, therefore, for multi-label classification, individual classifiers that perform poorly will significantly influence the overall score. Henceforth the macro-averaged F-measure will be denoted by M-F1, and micro-averaged F-measure by μ-F1.

In this thesis both evaluation schemes will be reported, however, a particular focus is put on the macro-average scores, since they are less dependent on the particular distribution of labels in the training corpus. As the main goal is for the classification system to work on data instances drawn over long periods of time and from numerous sources, with varying label-distributions, macro-average should prove the more useful metric.

8.2 Single-Stage Evaluations

The following sections look at the individual performance of the two distinct types of multi-label classification explored in this thesis: rule-based rote classification, and binary-relevance statistical classification. Rote classifier evaluation compares the performance impact of using the different IE-generated descriptor types during learning, in various ways. Statistical classifier evaluation is done in two parts: first,

an evaluation of classifiers trained on balanced training data vs. unbalanced training data, and second, a comparison of the various learning algorithms and feature-selection methods.

Most evaluations are done for both the PULS and RCV1 data sets. Two exceptions are an evaluation that utilizes the two-level hierarchy, which can only be done using the PULS data set, and the evaluation of balanced vs. unbalanced training, which is only done on RCV1, for brevity.

8.2.1 Rote Classifiers

As outlined in Section 7.1, there are two different IE-based features that can be used for learning with the rote classifier: company names and company descriptors. These feature types can be used either on their own or combined in various ways. The experiment is therefore a comparison of rote classifiers trained using different descriptors and descriptor-combinations, from the knowledge base. As the simplistic rote classifier does not have the same susceptibility to overfitting as statistical learning methods do, the training data used in this experiment is the entire set of available documents, minus the held out testing data.

Table 5a shows the results for all possible feature configurations for the rote classifier, on Reuters data, while Table 5b shows the results for the PULS data set. The results show that the rote classifier type **name+desc** gives the best M-F1 and μ-F1 scores for Reuters data. The combination of two feature types boosts recall, making it the top performer for both macro and micro-average recall. At the same time it maintains a good precision rate, being the second-highest performer for that metric. For PULS data, using only company name is slightly better than combining it with the descriptor. At the other end of the spectrum, those classifiers that rely solely on descriptor features for making predictions perform significantly worse than those that use name features, showing name features to be much more reliable correlates.

8.2.2 Balanced vs. Unbalanced Training

Before completing the major evaluations of the statistical classification, an experiment is performed in order to justify the use of the balancing procedure described in Section 7.2.1—for generating training data with which to build the statistical classifiers. Using the RCV1 data set, two sets of training data are collected: an *unbalanced* training pool, which is simply half of the corpus, selected at random, and

Table 5: Comparison of different rote classifier types. Classifier names correspond to the following formulae from Section 7.1: **name** – (10), **name+desc** – (12), **name⇝desc** – (14), **name+name⇝desc** – (15)

(a) Reuters data set

	M-average			*μ-average*		
Classifier	**Rec**	**Pre**	**F1**	**Rec**	**Pre**	**F1**
Rote classifiers						
name	36.8±0.8	**65.2**±1.0	44.5±0.7	45.9±0.5	**60.5**±0.4	52.2±0.5
descriptor	8.8±0.3	38.4±1.2	11.6±0.3	16.4±0.2	29.0±0.3	20.9±0.4
name+desc	**39.4**±0.8	63.3±0.7	**46.2**±0.7	**48.5**±0.5	57.8±0.5	**52.8**±0.4
name⇝desc	11.9±0.2	48.0±0.9	16.0±0.3	20.6±0.4	39.0±0.4	27.0±0.4
name+name⇝desc	39.2±0.8	60.0±0.8	44.8±0.6	**48.5**±0.5	54.5±0.4	51.3±0.4

(b) PULS data set

	M-average			*μ-average*		
Classifier	**Rec**	**Pre**	**F1**	**Rec**	**Pre**	**F1**
Rote classifiers						
name	**20.7**±0.3	51.0±0.9	**23.8**±0.3	**28.8**±0.3	35.2±0.3	**31.7**±0.2
descriptor	4.6±0.2	**60.0**±1.1	7.1±0.2	7.3±0.2	31.4±0.5	11.8±0.2
name+desc	19.2±0.2	51.5±0.8	22.6±0.3	27.0±0.2	35.3±0.3	30.6±0.2
name⇝desc	10.4±0.2	58.7±0.9	12.6±0.2	17.2±0.2	31.3±0.3	22.2±0.2
name+name⇝desc	18.4±0.2	55.7±0.6	21.8±0.2	26.6±0.2	**36.5**±0.2	30.8±0.1

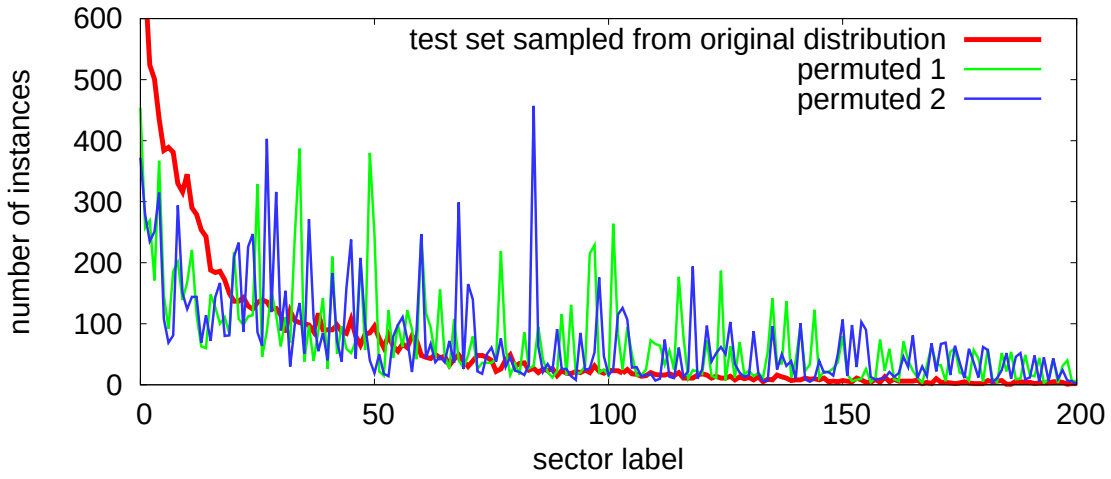


Figure 12: Label distributions of 1 original test set, and 2 permuted test-sets.

a *balanced* training pool generated using the new procedure. For the experiment, the procedure's k value is set to 600, meaning it attempts to collect 450 training documents per sector label. Distinct arrays of classifiers are generated using each of the two sets of training data (see Section 7.2.2). Both SVM and Naive Bayes classifiers are used in the comparison.

All data *outside* the balanced and unbalanced training pools—called the "test pool"—are available for the construction of test sets. The first part of the experiment attempts to evaluate how classifiers trained on each set of instances perform on data drawn from the same domain as the training instances (i.e., as one would see in a traditional machine-learning setting). 10 samples of 10,000 documents each are drawn from the test pool, using the original distribution in the corpus (i.e., random sampling).

One of these sample sets is used as a held-out *development* set for parameter tuning, and the remaining nine as test sets.

In the second part of the experiment, to simulate the effect of changing trends in news streams, 50 additional data sets are generated with *random* label distributions. To build these sets, the individual proportions of sector labels in the original distribution are calculated, then assigned to 50 *random permutations* of the sector labels. Approximately 10,000 documents (depending on availability) are sampled from the testing pool according to the new, permuted distributions. Each set among these 50 has its own label distribution, different from both the original and from each other. The distributions of labels in these random test sets will appear "naturally skewed,"

Table 6: Results for SVM+IG classifiers trained on balanced vs. unbalanced training sets, applied to originally-distributed and permuted test sets.

10 originally distributed test sets			
training	**Rec**	**Pre**	**F1**
	M-average		
balanced	**32.2**±0.6	66.2±1.4	**40.3**±0.7
unbalanced	24.3±0.9	**73.6**±1.3	31.8±0.9
	μ-average		
balanced	32.6±0.2	77.8±0.3	45.9±0.3
unbalanced	**36.8**±0.6	**79.5**±0.5	**50.3**±0.6

50 permuted test sets			
training	**Rec**	**Pre**	**F1**
	M-average		
balanced	**32.6**±0.8	71.0±1.4	**41.9**±0.9
unbalanced	23.5±0.9	**74.0**±1.5	31.4±0.8
	μ-average		
balanced	**34.5**±1.8	**78.7**±1.3	**47.9**±1.9
unbalanced	29.8±1.8	76.9±1.4	43.0± *2.1*

however, since each one mimics the original shape.

Three example test sets are shown in Figure 12, one "original," and two "permuted." As can be seen from the figure, the permuted distributions are still somewhat biased toward the largest classes in the original corpus. This is expected because some larger classes (such as *Diversified Holding Companies*) still have a high degree of overlap, and because the smallest sectors may not have enough data to dominate the permuted distribution. However, the distributions of the permuted test sets still look substantially different from the original distribution and contain more instances from small- and medium-sized sectors.

The averaged results obtained on both original and permuted test sets are presented in Table 6. Although all combinations of algorithms and feature-selection methods were evaluated, for brevity, only the top performer—SVM classifiers using Infogain feature selection (SVM+IG)—are presented. Regardless of the overall score, results for all combinations follow the same pattern: classifiers trained on the original distribution have higher μ-F1 on originally distributed test sets, but lower on the permuted test sets; the classifiers trained on the balanced training set yield higher M-F1 on *all test sets*, both original and permuted.

A comparison of balanced and unbalanced training is presented in Figure 13, where macro- and micro-averaged F-measure obtained by classifiers trained on balanced vs. unbalanced data for each *permuted* test set are plotted. As seen from the upper plot, the classifier trained on balanced data has significantly and consistently higher M-F1: for each test set, M-F1 is over 30% higher for the balanced classifiers. At the same time, the lower plot shows that μ-F1 for the balanced classifiers is always at least on par with (and often better than) μ-F1 results for unbalanced classifiers.

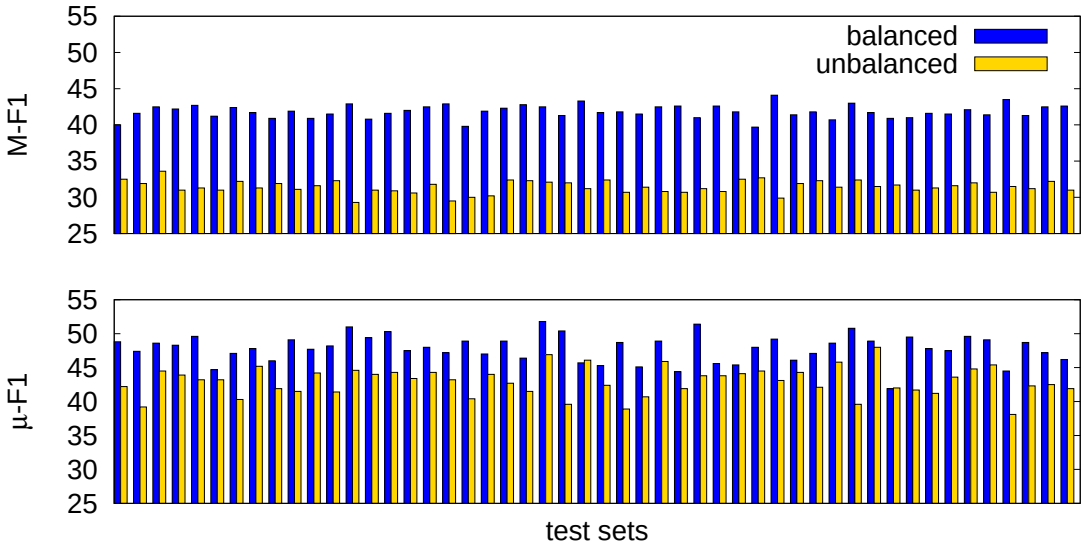


Figure 13: F-measure obtained by SVM+IG classifiers trained on balanced vs. unbalanced data, for all *permuted* test sets.

8.2.3 Supervised Learning Classifiers

The set of balanced training data collected using the procedure outlined in Section 7.2.1 and evaluated in Section 8.2.2 is again used in the experiments comparing statistical learning algorithms and feature-selection methods. For the experiments presented here, arrays of classifiers are trained and evaluated on test sets compiled in the same manner described in the previous section, averaging over 9 randomly-generated sets, with a single development set used to optimize prediction thresholds.

The first experiment compares precision, recall, M-F1, and μ-F1 across possible combinations of learning algorithms with feature-selection methods. The number of selected features remains constant, at 500, which was found to be generally optimal in terms of M-F1 by [For03], for both IG and BNS feature selection methods. Table 7a shows the results for the Reuters data set, while Table 7b shows the results for the PULS data set. From these we can see the best combination for the Reuters data set, for both M-F1 and μ-F1 is SVM classifiers with infogain feature-selection (SVM+IG). When testing on PULS data, however, SVM+BNS is the clear winner. For Naive Bayes classifiers, the best feature selection method for both data sets is the combined method that uses infogain with the novel entropy-filtering method (IG+EF).

The final experiment compares the level-II classifiers, trained on the PULS data set,

Table 7: Results obtained with different classifier and feature-selection combinations averaged among 9 random originally-distributed test sets. Optimal (M-F1) prediction thresholds tuned on a held-out development set.

(a) Reuters data set

M-average

Classifier	**Features**	**Rec**	**Pre**	**F1**
NB	IG	34.2±0.9	27.4±0.4	25.3±0.4
	IG+EF	37.0±0.5	**31.3**±0.4	**29.5**±0.5
	BNS	37.0±0.8	21.0±0.2	21.1±0.3
	BNS+EF	**37.7**±0.7	30.4±0.6	28.7±0.6
SVM	IG	32.2±0.6	66.2±1.4	**40.3**±0.7
	IG+EF	29.9±0.5	**66.4**±0.9	38.1±0.5
	BNS	**32.6**±0.5	62.0±0.8	39.4±0.5
	BNS+EF	28.2±0.5	66.2±1.6	35.8±0.6

μ-average

Classifier	**Features**	**Rec**	**Pre**	**F1**
NB	IG	36.1±0.4	30.4±0.3	33.0±0.3
	IG+EF	**38.5**±0.4	**39.8**±0.3	**39.1**±0.3
	BNS	38.1±0.4	19.0±0.3	25.4±0.3
	BNS+EF	37.7±0.5	37.0±0.3	37.3±0.3
SVM	IG	**32.6**±0.2	**77.8**±0.3	**45.9**±0.3
	IG+EF	31.0±0.3	77.5±0.4	44.3±0.3
	BNS	32.0±0.3	76.1±0.4	45.1±0.3
	BNS+EF	28.7±0.3	76.7±0.4	41.8±0.4

(b) PULS data set

M-average

Classifier	Features	Rec	Pre	F1
NB	IG	39.8±0.3	**39.8**±0.5	37.0±0.3
	IG+EF	**44.6**±0.3	37.3±0.4	**38.1**±0.3
	BNS	40.5±0.4	34.4±0.3	34.7±0.3
	BNS+EF	39.9±0.4	37.7±0.4	36.1±0.3
SVM	IG	35.8±0.4	47.2±0.3	39.8±0.3
	IG+EF	38.1±0.2	47.3±0.3	40.7±0.2
	BNS	**38.9**±0.4	51.4±0.4	**42.4**±0.3
	BNS+EF	32.0±0.3	**58.0**±0.5	37.9±0.2

μ-average

Classifier	Features	Rec	Pre	F1
NB	IG	41.3±0.3	**36.9**±0.2	39.0±0.2
	IG+EF	**47.8**±0.3	34.6±0.2	**40.2**±0.2
	BNS	40.7±0.2	32.7±0.2	36.2±0.2
	BNS+EF	41.0±0.2	35.8±0.2	38.3±0.2
SVM	IG	**35.7**±0.3	49.3±0.3	41.4±0.2
	IG+EF	34.4±0.1	50.1±0.3	40.8±0.2
	BNS	**35.7**±0.3	55.0±0.2	**43.3**±0.2
	BNS+EF	28.5±0.2	**61.4**±0.3	38.9±0.2

against a set of specially trained level-I PULS classifiers. Using the same balanced data collection procedure, a set of training data specifically targeting level-I labels is generated. The array of 26 level-I classifiers is then trained and evaluated on the same test data as prior experiments. This classification is necessarily evaluated "at level-I," meaning the confusion matrix is generated by using level-I labels both for predicted and true sectors. These results are compared to the best level-II predictions from before, after each prediction has been mapped to its level-I parent. Table 8 shows the outcome of these evaluations. Interestingly, the dedicated level-I classifiers are outperformed by the level-II classifiers, where the output has been mapped to level I. A possible explanation is that the level-II classifiers benefit from a more specific feature set (i.e., drawn from a more specialized set of instances)—which is supported by the higher precision score—and that the mapping of labels to level-I for evaluation gives an improved recall than evaluating at level-II, by not penalizing "close misses". Admittedly, however, we are unsure of the exact reason.

8.3 Multi-Stage Evaluations

After compiling results for both the baseline rote classifier and the statistical, binary-relevance classifiers at each level of the two-level hierarchy, the next step is to evaluate performance when the outputs of these single-stage classifiers are combined in a multi-stage, multi-label classifier.

The first experiment looks at various *two-stage* classifiers that utilize the two combination methods, *pipeline* and *union*, described in Section 7.3. The individual learners at each stage of the combined classifier are the exact same as those used in the previous experiments (i.e., trained on the exact same data), however, new prediction thresholds have been trained on the held out development set, which are chosen to optimize the *multi-stage* M-F1, as opposed to optimizing F-measure for the individual learners. Test data sets are also identical to those used for testing individual learners. All possible algorithm/feature-selection combinations, and two-stage orderings [6] of the two distinct learner types—rote and statistical—are tested, for both multi-stage combination methods. Table 9a shows the scores for the various combinations on the PULS data set. From this we see that [SVM+BNS→Rote] is the best combination (and ordering) for the pipeline-combined classifier, while [Rote ∪ SVM+BNS] is the best for the union-combined method, and best overall. Table 9b shows the corresponding level-II results for the Reuters data set, which has

[6] Recall, however, that for the union-combined multi-stage method, ordering is unimportant.

Table 8: Level-I results for all classifier/feature-selection combinations for PULS data set. Also compares the best level-II classifier combination evaluated at level-I.

M-average

Classifier	**Features**	**Rec**	**Pre**	**F1**
NB	IG	53.8±0.9	46.3±0.3	48.4±0.3
	IG+EF	**55.9**±0.9	**47.2**±0.3	**50.2**±0.4
	BNS	51.0±0.7	44.3±0.2	45.9±0.3
	BNS+EF	53.3±0.7	47.1±0.2	48.8±0.2
SVM	IG	**54.8**±0.9	54.6±0.4	**53.4**±0.4
	IG+EF	47.1±0.8	58.6±0.3	50.8±0.4
	BNS	52.4±0.8	58.0±0.4	53.3±0.3
	BNS+EF	47.9±0.5	**59.4**±0.2	51.0±0.2
SVM Level-II Mapped	BNS	48.9±0.4	**62.3**±0.2	**54.4**±0.2

μ-average

Classifier	**Features**	**Rec**	**Pre**	**F1**
NB	IG	56.5±0.4	52.0±0.3	54.1±0.3
	IG+EF	**59.2**±0.3	52.1±0.2	**55.4**±0.3
	BNS	52.9±0.4	50.2±0.3	51.5±0.3
	BNS+EF	55.6±0.3	**52.7**±0.2	54.1±0.2
SVM	IG	**57.5**±0.3	59.2±0.3	**58.3**±0.3
	IG+EF	47.5±0.3	**65.9**±0.4	55.2±0.3
	BNS	53.0±0.2	63.2±0.4	57.7±0.3
	BNS+EF	48.9±0.2	65.0±0.2	55.8±0.2
SVM Level-II Mapped	BNS	51.5±0.3	**68.0**±0.2	**58.6**±0.2

[Rote ∪ SVM+IG] as the top performer. [7]

The next experiment looks at two-stage classification utilizing the sector hierarchy. In this case one stage is the array of level-II statistical classifiers and the other stage is the level-I classifiers of the same model. Table 10 shows the evaluations—necessarily done at level-I— where the best combination and ordering (by M-F1) is [SVM+BNS-II→SVM+BNS-I]. This beats out either of the level-II or level-I classifiers, on their own, when evaluated at level-I.

The third experiment takes the next logical step for the combined classifier, given that there are three distinct multi-label learner types, which is to build a three-stage classifier. The output combination methods remain the same, there is simply an additional stage for each method (level-I → level-II classification was not included, because it always resulted in worse scores). As the level-I learners must be a part of each multi-stage classifier, evaluations for this experiment can only be done at level-I. Results on the PULS data set for the various configurations are shown in Table 11 and compared against the best two-stage methods, also evaluated at level-I. From this it can be seen that the best three-stage classifier is [SVM+BNS-II→Rote →SVM+BNS-I], and that the addition of the third stage actually does improve classification over the best two-stage performer. Therefore, three-stage classification appears to be optimal for the PULS data set.

The final experiment compares the best two-stage performer on the Reuters data set against state-of-the-art multi-label classifiers that have also been evaluated on the RCV1 corpus. The opposing methods were chosen because they were the ones found in the literature that have the same problem definition (i.e., multi-label industry-sector classification) that have also been tested on the Reuters data set. These experimenters used various learning algorithms, including SVM and Naive Bayes algorithms, in addition to an undiscussed, computationally-expensive method that utilizes Bloom Filters and class dependency information [CUAG13]. Table 12 shows these results compared with the best combination method for the Reuters data set from experiment 1, Rote∪SVM+IG, evaluated on a distinct 11th data set. From the table it can be seen that the M-F1 score for the two-stage combined classifier is better than any previously reported results, thus indicating it is a better *general* classifier, fit for a real-world scenario like PULS. While the μ-F1 score for the combined classifier is significantly lower than the highest performer, this can be attributed to the fact that we are not attempting to model the exact distribution of the RCV1

[7] Recall that there are no level-I learners for the Reuters data set.

Table 9: Results from all two-stage classifiers and feature selection methods, averaged across 9 test sets randomly sampled from original distribution; single classifiers on top, combined classifiers on bottom. For each classifier, the best threshold is trained on one random, originally-distributed development set; → and ∪ denote, respectively, pipeline and union combining methods, described in Section 7.3.

(a) PULS data set

	M-average			*μ-average*		
Classifier	**Rec**	**Pre**	**F1**	**Rec**	**Pre**	**F1**
Rote→NB+BNS	41.9±0.4	32.1±0.3	34.1±0.3	42.5±0.3	30.6±0.2	35.7±0.1
NB+BNS→Rote	32.8±0.4	49.1±0.5	36.9±0.4	35.3±0.2	46.1±0.3	40.0±0.2
Rote ∪ NB+BNS	35.6±0.4	47.3±0.4	38.0±0.4	39.8±0.3	43.1±0.2	41.4±0.2
Rote→NB+BNSEF	38.5±0.4	37.8±0.5	35.6±0.3	39.9±0.2	36.1±0.3	37.9±0.2
NB+BNSEF→Rote	40.8±0.3	42.2±0.6	38.8±0.4	43.6±0.2	40.1±0.3	41.8±0.2
Rote ∪ NB+BNSEF	43.8±0.4	40.6±0.5	39.5±0.4	48.3±0.2	38.0±0.2	42.6±0.2
Rote→NB+IG	39.4±0.3	39.3±0.5	36.7±0.3	41.1±0.3	36.5±0.2	38.7±0.2
NB+IG→Rote	42.7±0.4	41.4±0.4	39.6±0.4	45.8±0.3	38.9±0.2	42.0±0.2
Rote ∪ NB+IG	43.8±0.4	42.9±0.4	40.9±0.4	48.6±0.3	39.2±0.2	43.4±0.2
Rote→NB+IGEF	40.9±0.4	39.3±0.5	37.5±0.4	44.0±0.3	36.6±0.2	40.0±0.2
NB+IGEF→Rote	42.4±0.4	44.0±0.5	40.4±0.4	46.4±0.2	40.9±0.2	43.5±0.2
Rote ∪ NB+IGEF	**47.3**±0.4	40.5±0.4	41.0±0.4	**52.9**±0.3	37.0±0.2	43.6±0.2
Rote→SVM+BNS	39.0±0.4	48.9±0.3	41.5±0.3	36.2±0.3	52.2±0.2	42.7±0.2
SVM+BNS→Rote	40.6±0.4	51.1±0.3	43.9±0.3	38.7±0.2	54.6±0.2	45.3±0.2
Rote ∪ SVM+BNS	45.6±0.4	47.0±0.3	**45.3**±0.3	47.2±0.2	47.3±0.2	**47.2**±0.2
Rote→SVM+BNSEF	30.3±0.3	**58.5**±0.6	37.0±0.3	27.3±0.2	**62.1**±0.3	37.9±0.2
SVM+BNSEF→Rote	35.3±0.4	56.3±0.5	40.9±0.3	33.9±0.2	57.6±0.2	42.7±0.1
Rote ∪ SVM+BNSEF	40.1±0.4	51.8±0.5	42.8±0.3	41.9±0.3	49.4±0.2	45.3±0.1
Rote→SVM+IG	30.1±0.4	52.5±0.2	37.5±0.3	30.1±0.2	55.9±0.3	39.1±0.2
SVM+IG→Rote	34.4±0.4	51.6±0.3	40.6±0.4	35.7±0.3	53.6±0.3	42.9±0.2
Rote ∪ SVM+IG	39.4±0.4	48.2±0.3	42.2±0.3	43.7±0.2	46.6±0.3	45.1±0.1
Rote→SVM+IGEF	35.8±0.3	48.6±0.3	39.9±0.2	32.7±0.2	51.7±0.3	40.0±0.2
SVM+IGEF→Rote	39.7±0.3	47.9±0.3	42.3±0.2	37.4±0.2	50.6±0.3	43.0±0.2
Rote ∪ SVM+IGEF	44.9±0.3	44.6±0.3	44.0±0.2	46.3±0.2	44.7±0.3	45.5±0.1

(b) Reuters data set

	M-average			*μ-average*		
Classifier	**Rec**	**Pre**	**F1**	**Rec**	**Pre**	**F1**
Rote→NB+BNS	51.5±0.9	33.6±0.4	36.1±0.4	57.6±0.6	39.1±0.4	46.6±0.4
NB+BNS→Rote	49.7±1.0	24.0±0.2	26.9±0.3	53.3±0.4	23.7±0.3	32.8±0.3
Rote ∪ NB+BNS	**59.2**±0.9	25.4±0.3	30.7±0.3	**64.3**±0.5	26.2±0.3	37.2±0.3
Rote→NB+BNSEF	47.5±0.9	54.8±0.7	48.2±0.8	54.2±0.6	60.0±0.3	57.0±0.4
NB+BNSEF→Rote	48.7±0.9	48.1±0.8	44.4±0.7	52.4±0.6	54.4±0.4	53.4±0.5
Rote ∪ NB+BNSEF	52.9±0.8	47.5±0.7	46.3±0.6	58.7±0.5	53.8±0.3	56.1±0.4
Rote→NB+IG	51.8±0.9	39.8±0.6	41.5±0.6	59.1±0.5	47.3±0.4	52.5±0.4
NB+IG→Rote	48.7±1.0	31.5±0.5	33.4±0.4	53.0±0.5	36.3±0.3	43.1±0.3
Rote ∪ NB+IG	57.2±0.9	32.7±0.4	37.3±0.4	63.2±0.5	38.1±0.3	47.5±0.4
Rote→NB+IGEF	47.5±0.9	54.4±0.7	48.2±0.8	54.6±0.5	60.3±0.3	57.3±0.4
NB+IGEF→Rote	48.8±1.1	47.6±0.8	44.5±0.8	53.0±0.5	54.8±0.4	53.9±0.4
Rote ∪ NB+IGEF	53.0±0.9	46.9±0.6	46.4±0.6	59.3±0.5	54.0±0.3	56.5±0.3
Rote→SVM+BNS	48.2±1.0	67.5±1.0	54.7±0.9	53.7±0.5	70.1±0.3	60.8±0.4
SVM+BNS→Rote	48.0±1.1	63.0±1.0	52.6±1.0	50.2±0.4	70.8±0.4	58.7±0.4
Rote ∪ SVM+BNS	54.0±0.9	62.0±0.8	56.1±0.8	58.5±0.4	68.2±0.3	63.0±0.3
Rote→SVM+BNSEF	47.5±1.0	66.6±1.0	53.8±0.9	54.2±0.5	66.7±0.3	59.8±0.4
SVM+BNSEF→Rote	45.7±1.1	65.6±0.8	52.1±0.9	48.9±0.4	69.2±0.5	57.3±0.4
Rote ∪ SVM+BNSEF	52.5±1.0	62.9±0.7	55.6±0.8	58.4±0.5	65.3±0.3	61.7±0.4
Rote→SVM+IG	46.2±1.0	**73.7**±0.8	55.1±0.8	52.5±0.5	**75.9**±0.4	62.0±0.4
SVM+IG→Rote	47.0±1.2	67.7±0.9	53.7±1.1	49.9±0.3	73.9±0.3	59.6±0.3
Rote ∪ SVM+IG	52.2±1.1	66.3±0.8	**56.9**±0.9	57.7±0.4	71.1±0.3	**63.7**±0.4
Rote→SVM+IGEF	48.1±1.0	66.8±0.9	54.3±0.8	55.1±0.5	67.0±0.3	60.5±0.3
SVM+IGEF→Rote	46.6±1.0	66.0±0.7	53.0±0.9	50.3±0.3	69.9±0.5	58.5±0.3
Rote ∪ SVM+IGEF	53.4±1.0	63.2±0.7	56.4±0.7	59.6±0.4	65.7±0.3	62.5±0.3

Table 10: Comparison of two-stage classifications using the two-level sector hierarchy (PULS data set only). Level-II→Level-I pipeline classification denoted by →, reverse order denoted by ←, and union method denoted by ∪. Best single-stage results also shown at bottom for comparison.

		M-average			*μ-average*		
Classifier	**Method**	**Rec**	**Pre**	**F1**	**Rec**	**Pre**	**F1**
NB+BNS	→	54.9±0.4	49.8±0.5	51.3±0.4	58.3±0.4	54.9±0.3	56.5±0.3
	←	50.8±0.5	52.7±0.3	50.5±0.3	53.2±0.3	57.2±0.3	55.1±0.2
	∪	58.0±0.7	33.5±0.2	41.1±0.2	61.4±0.4	38.4±0.2	47.2±0.3
NB+BNSEF	→	57.0±0.4	52.2±0.4	54.2±0.3	60.8±0.3	57.0±0.3	58.8±0.2
	←	55.2±0.3	51.0±0.2	52.7±0.2	58.9±0.3	55.7±0.2	57.2±0.2
	∪	53.9±0.4	46.2±0.2	49.5±0.2	58.8±0.3	51.9±0.2	55.1±0.2
NB+IG	→	57.1±0.4	52.1±0.4	53.8±0.4	61.1±0.4	56.8±0.3	58.9±0.3
	←	55.6±0.6	51.8±0.3	52.7±0.4	58.9±0.4	56.0±0.2	57.4±0.3
	∪	**58.6**±0.7	37.0±0.3	44.2±0.4	**62.6**±0.4	42.6±0.4	50.7±0.4
NB+IGEF	→	54.1±0.5	57.5±0.7	55.5±0.5	58.6±0.4	62.1±0.3	60.3±0.3
	←	49.2±0.4	60.0±0.4	53.8±0.4	53.3±0.3	65.0±0.3	58.6±0.2
	∪	52.5±0.3	48.9±0.6	50.4±0.3	57.9±0.3	55.3±0.5	56.5±0.3
SVM+BNS	→	51.6±0.3	63.9±0.3	**56.4**±0.2	53.7±0.2	70.0±0.3	**60.7**±0.2
	←	51.5±0.3	61.6±0.3	54.8±0.3	52.4±0.3	67.9±0.4	59.1±0.3
	∪	53.3±0.6	50.1±0.6	51.0±0.5	56.9±0.4	56.1±0.5	56.5±0.4
SVM+BNSEF	→	45.3±0.3	**66.5**±0.3	53.4±0.3	48.3±0.2	**73.5**±0.3	58.3±0.2
	←	49.4±0.4	60.8±0.3	52.9±0.3	50.9±0.2	66.6±0.2	57.6±0.2
	∪	35.9±0.4	63.7±0.9	45.4±0.5	38.9±0.2	72.3±0.7	50.6±0.3
SVM+IG	→	47.1±0.4	63.4±0.4	53.8±0.4	51.1±0.3	69.4±0.3	58.8±0.3
	←	51.9±0.6	58.5±0.4	54.0±0.4	54.3±0.3	64.7±0.2	59.0±0.3
	∪	38.1±0.5	58.3±0.9	45.9±0.5	42.8±0.3	66.5±0.8	52.0±0.4
SVM+IGEF	→	49.4±0.4	60.7±0.4	54.0±0.4	52.1±0.3	66.7±0.3	58.5±0.3
	←	49.8±0.4	59.1±0.3	53.0±0.2	51.1±0.2	66.2±0.3	57.7±0.2
	∪	43.5±0.5	57.2±0.4	49.0±0.4	47.0±0.3	64.2±0.5	53.4±0.3
SVM+IG-I	- -	54.8±0.9	54.6±0.4	53.4±0.4	57.5±0.3	59.2±0.3	58.3±0.3
SVM+BNS-II (mapped)	- -	48.9±0.4	62.3±0.2	54.4±0.2	51.5±0.3	68.0±0.2	58.6±0.2

Table 11: Comparison of three-stage classification results, evaluated at level-I. Best two-stage results (mapped to level-I) are shown at the bottom for comparison.

		M-average			*μ-average*		
Classifier	**Method**	**Rec**	**Pre**	**F1**	**Rec**	**Pre**	**F1**
Rote &	Rote→II→I	53.6±0.6	50.3±0.5	51.0±0.5	57.1±0.4	55.4±0.3	56.3±0.3
NB+BNS	II→Rote→I	56.6±0.4	51.3±0.4	53.0±0.4	60.9±0.3	56.5±0.3	58.6±0.2
	∪	58.4±0.6	33.4±0.2	41.2±0.2	62.2±0.4	38.4±0.2	47.5±0.3
Rote &	Rote→II→I	55.5±0.5	52.5±0.5	53.6±0.4	59.5±0.3	57.3±0.3	58.4±0.2
NB+BNSEF	II→Rote→I	54.7±0.3	56.7±0.5	55.5±0.3	59.8±0.2	62.2±0.4	60.9±0.3
	∪	54.5±0.4	46.0±0.2	49.6±0.2	59.9±0.3	51.6±0.2	55.5±0.2
Rote &	Rote→II→I	55.6±0.6	52.5±0.5	53.3±0.4	59.8±0.4	57.3±0.3	58.5±0.3
NB+IG	II→Rote→I	54.8±0.4	56.9±0.5	55.3±0.4	60.0±0.3	62.2±0.3	61.1±0.3
	∪	**58.6**±0.7	37.0±0.3	44.2±0.4	**62.7**±0.4	42.6±0.4	50.7±0.4
Rote &	Rote→II→I	52.7±0.6	57.8±0.8	54.9±0.6	57.3±0.4	62.5±0.3	59.8±0.3
NB+IGEF	II→Rote→I	56.4±0.4	57.7±0.6	56.8±0.4	61.9±0.3	62.3±0.3	62.1±0.2
	∪	53.3±0.3	48.4±0.6	50.5±0.4	59.5±0.3	54.4±0.5	56.8±0.3
Rote &	Rote→II→I	49.2±0.4	65.4±0.3	55.5±0.3	51.4±0.2	71.6±0.2	59.8±0.2
SVM+BNS	II→Rote→I	50.7±0.3	66.4±0.4	**57.3**±0.3	53.9±0.1	72.3±0.2	**61.8**±0.1
	∪	53.8±0.6	49.8±0.6	51.1±0.5	58.0±0.4	55.7±0.5	56.8±0.4
Rote &	Rote→II→I	44.4±0.3	**66.6**±0.4	52.9±0.3	47.5±0.2	**73.4**±0.2	57.7±0.2
SVM+BNSEF	II→Rote→I	47.0±0.2	66.0±0.3	54.6±0.2	51.7±0.1	72.8±0.2	60.5±0.1
	∪	37.6±0.4	62.5±0.9	46.3±0.5	42.2±0.3	69.2±0.7	52.4±0.3
Rote &	Rote→II→I	46.1±0.5	63.4±0.4	53.1±0.4	50.2±0.3	69.4±0.3	58.3±0.3
SVM+IG	II→Rote→I	48.3±0.4	63.2±0.4	54.6±0.4	53.5±0.3	69.0±0.3	60.3±0.2
	∪	39.4±0.5	57.3±0.9	46.2±0.5	45.2±0.2	64.1±0.7	53.0±0.4
Rote &	Rote→II→I	48.2±0.4	60.9±0.4	53.5±0.4	51.1±0.3	66.9±0.3	58.0±0.3
SVM+IGEF	II→Rote→I	51.1±0.4	60.5±0.4	55.1±0.4	54.9±0.3	66.5±0.3	60.1±0.3
	∪	44.8±0.5	56.3±0.5	49.4±0.4	49.6±0.3	62.4±0.5	55.3±0.3
Best Two-Stage	SVM+BNS-II→I	51.6±0.3	63.9±0.3	56.4±0.2	53.7±0.2	70.0±0.3	60.7±0.2

Table 12: Classification results on RCV1 industry sectors, compared with state-of-the-art.

Reference	**Algorithm**	M-**F1**	μ-**F1**
[LYRL04]	SVM	29.7	51.3
[ZZY+05]	SVM	30.1	52.0
[Puu12]	Naive Bayes	-	70.5
[CUAG13]	Bloom Filters	47.8	**72.4**
Best two-stage results	Rote ∪ SVM+IG	**56.9**	63.7

data set, as they did.

8.4 Error Analysis

The macro- and micro-average evaluation schemes described in Section 8.1 can be considered as *strict evaluations*, due to classifiers being heavily penalized for returning sectors which are not in the set of original labels, but that may, nevertheless, be relevant to the document. Since there are a large number of sector labels and documents, there may be cases where correct labels were left out or extraneous labels incorrectly applied, through annotator error. These are known issues in multi-label learning and lead to uncertainty regarding to what extent a given evaluation method provides a fair assessment of perfomance.

To illuminate the extent of this issue, two types of error analysis are done using a set of labeled PULS documents: the first to determine the rate of documents that are "misclassified" under the strict evaluation scheme, but could be considered relevant based on document content; and the second, to determine the rate of originally-assigned labels that should not actually be inferred from the text.

For the first scenario, after classification, each false-positive sector label, and the text of the document it is applied to, is manually analyzed to determine whether the given label is: a) *highly relevant*, meaning the sector is related to the main focus of the article; b) *partially relevant*, meaning some details related to that sector are present, but it is not the primary focus; or c) *totally irrelevant*.

An example text is shown in Figure 14 along with the labels it is assigned by the multi-stage classifier. Strict evaluation for this document will return both precision

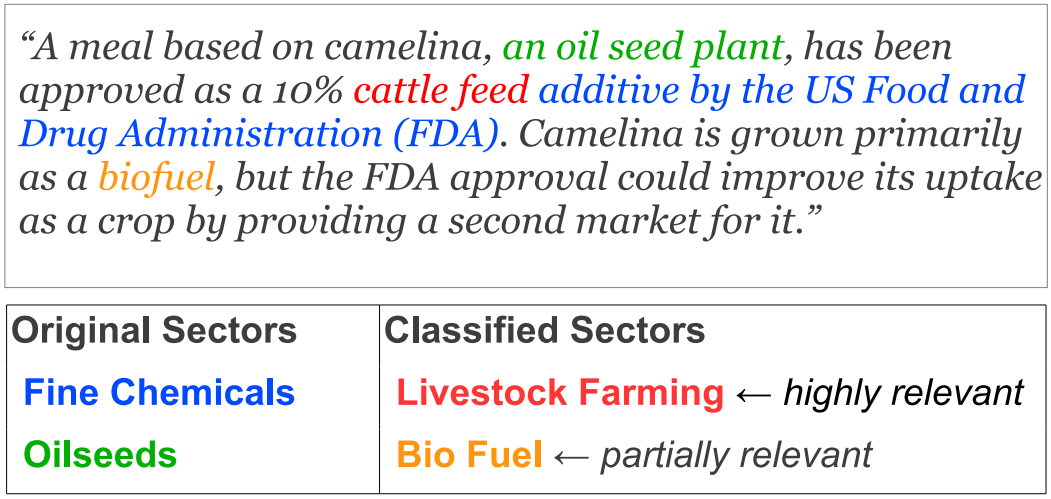


Figure 14: Misclassified analysis of false-positive labels.

and recall scores of 0, since none of the originally-assigned labels are present in the output, however, if the returned labels that are considered to be highly relevant are now assumed to be present in the set of original labels, the results change to: $pre = 50\%$ and $rec = 25\%$. Furthermore, if those labels that are considered partially relevant are also included in the original labels, the results become: $pre = 100\%$, $rec = 50\%$.

For 135 randomly selected test documents, classified by the best-performing two-stage classification system (SVM-IG→Rote), there are 191 false-positive labels returned. The error analysis shows that of these 191, 84 labels could be considered as highly relevant, for a rate of 44.0%. An additional 46 false positive predictions were partially relevant, which, when combined with the highly relevant labels, yields a rate of 68.1%.

The second error analysis focuses only on the original labels for each document, where each is adjudged to be either a) *highly irrelevant* or b) *partially irrelevant*, which may occur if the label appears to be included due to a minor mention in the article of some entity related to the sector (e.g., a company name). Using a different set of 170 random test documents, which contain 525 original sector labels, the analysis found that 75 labels (14.3%) were highly irrelevant to the documents they were assigned to, and another 56 were partially irrelevant, for a combined rate of 25.0%.

This simple analysis points to a large disparity between the evaluation results, and

what could be considered as the "true" performance of the classification system. This highlights a major difficulty in supervised learning for multi-label data.

9 Hierarchical Classification using PCA

In attempting to improve accuracy and efficiency of industry-sector classification we explore a method that leverages both the underlying structure of the training data and dependency relationships between classes. Although no concrete improvements have been found to date, the methods are worth discussing, as they highlight some alternative approaches to multi-label learning than the ones explored in our experiments, in addition to providing a promising base for future enhancements to PULS.

The feature-selection methods described in Section 6.2 are useful both because they improve performance and reduce model size. One drawback of those methods, however, is that they produce BOW representations, which do not take into account any inherent correlations (e.g., semantic links) between terms. Principal component analysis (PCA) is one method for projecting a feature vector to a lower-dimensional space containing only the most salient features of a data set [Jol02]. In our case, PCA is nearly identical to LSA, as described in Section 5.1.2; for example, both use SVD to reduce dimensionality and both sort the largest eigenvalues, before truncating and reconstructing the data. A main difference in methods is that PCA is applied to a term-covariance matrix rather than the term-document matrix directly, however, the output produced is again a projection of the data to N-dimensional space, using the largest N eigenvalues (here called principal components). PCA shows us the underlying structure of data by explaining in which dimensions data points vary the most from each other. This is similar to our goal using other feature-selection methods, only PCA operates at a conceptual level that BOWs is unable to achieve. Since the method also allows for the reduction of dimensionality, it fulfills both of our objectives.

The first idea involves replacing BOW features with PCA features in the binary-relevance classifiers as described in Section 7.2.2. The same set of balanced training data is used, and a projection to 50-dimensional space is produced. A visualization of the training data in 3-dimensional space is shown in Figure 15 to give an approximate view of the resulting data set. This figure highlights the main problem encountered with this method, which is that the data is densely clustered, with a

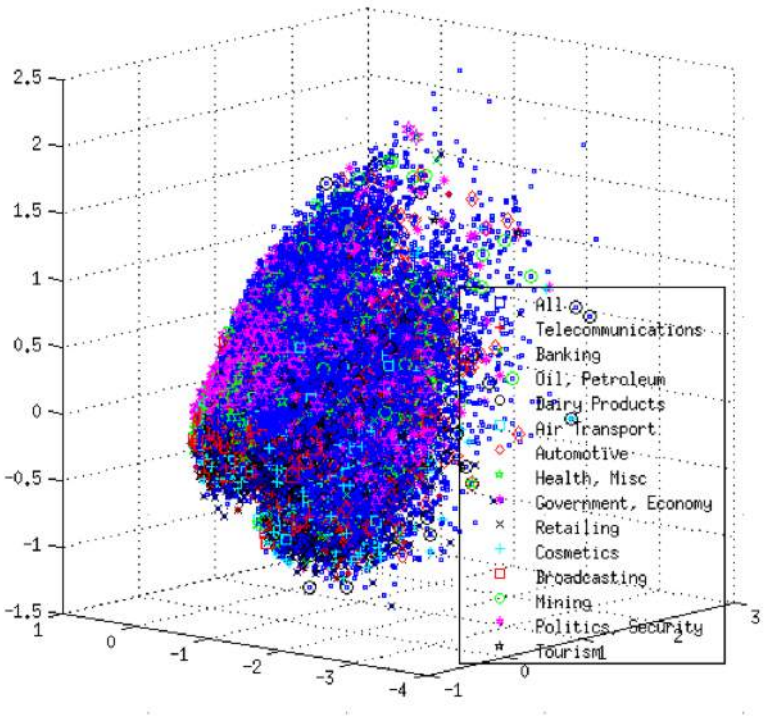


Figure 15: Visualization of all training data after applying PCA, in 3-dimensional space. Various sector-label exemplars are highlighted.

large degree of overlap amongst topics (even in 50-dimensional space), so that separating individual classes based on the distribution is difficult. Deeper analysis is required to understand the reason behind this noisy output.

Further experimentation finds that when the training data is reduced to include only exemplars of a small number of classes, separation between classes becomes more pronounced, implying classification may be possible by training on more-specific data. Figure 16 shows good separation between training data collected for two sectors related to fishing and two sectors related to insurance.

Importantly, separation also occurs if two or more of the retained class labels are conceptually similar. Figure 17 shows the 3-dimensional representation of the training data drawn from five different sectors related to energy. In this figure we see clustering starting to occur, even though there isn't full separation between classes. Going further, Figure 18 shows the projection of training data drawn from only two similar class labels *Beer* and *Wine*, where we once again see good separation. Given this, we could theoretically produce learners using pairs of class-label exemplars, to perform binary classification. However, producing classifiers for all possible sector

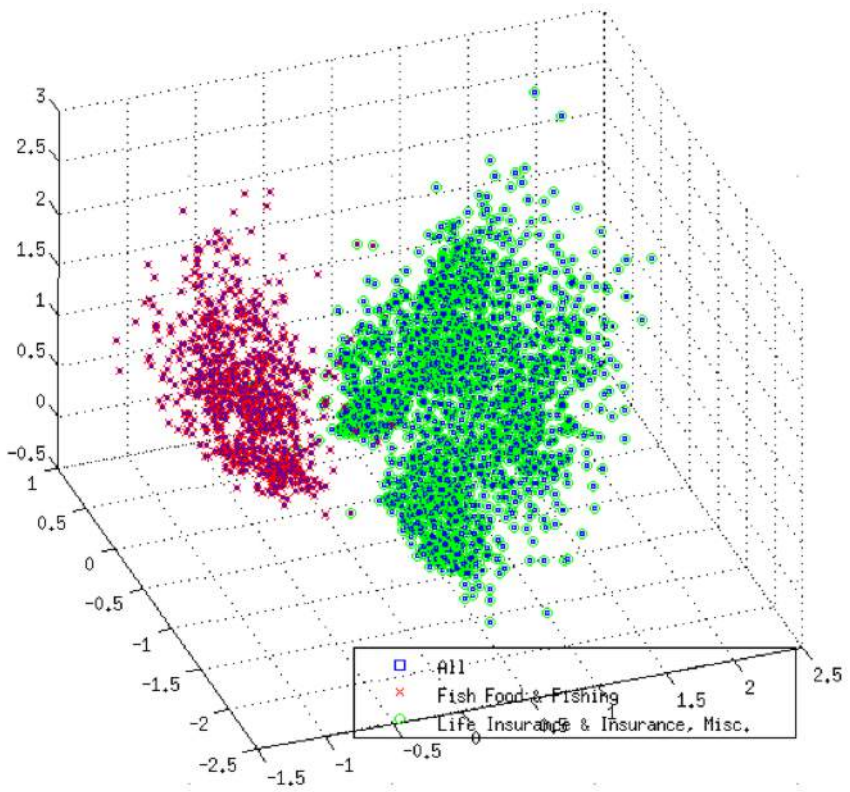


Figure 16: 3D visualization of training data for two industry sectors related to fishing vs. two related to insurance.

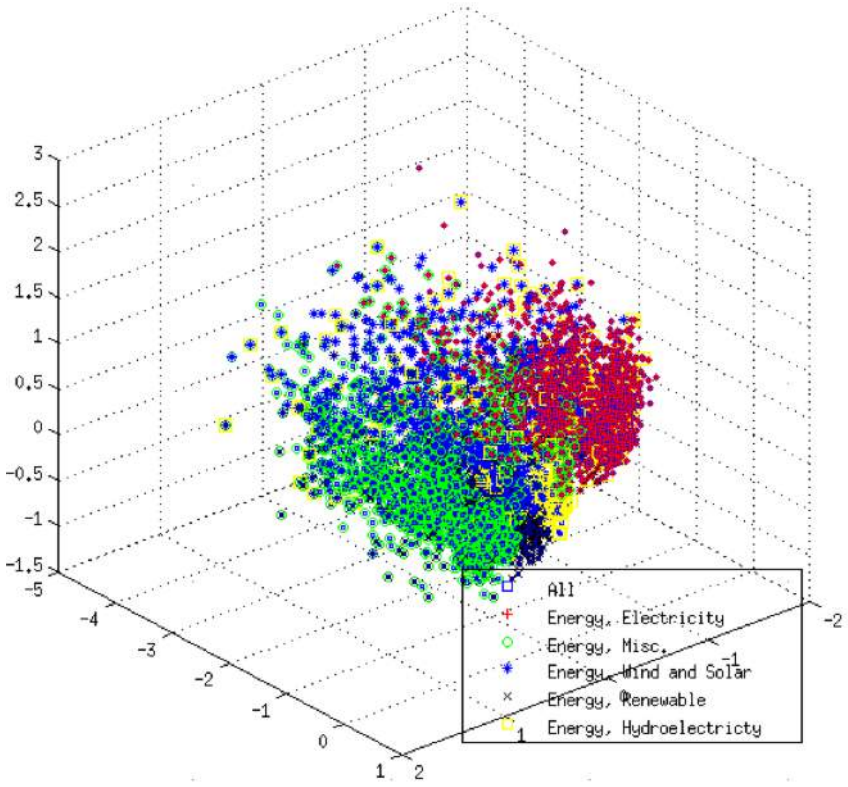


Figure 17: 3D visualization of training data for five sector labels related to energy.

pairs would require over 70,000 individual classifiers, and output would be hard to parse, since each label would have many classifiers associated with it.

In order to train classifiers based on PCA, therefore, we turn to hierarchical classification. Assuming we have a binary-tree hierarchy containing class labels at each leaf node, we could train a single model for each internal node of the tree, using training instances belonging to classes in the left subtree as positive-class instances and those in the right-subtree as negative-class. [8] The classifier of the root node would, in this case, use all training data, however, all sub-nodes would have progressively smaller and more specific training sets. Classification works as follows, given a new instance in the original feature space:

1. Project the instance to the reduced feature space of the root node's model, using the same eigenvector matrix produced during training.

2. Produce a probability distribution (i.e., positive-class/negative-class) for the

[8] If a data instance is an exemplar of class labels in both the left *and* right subtrees, either remove it from training or reassign it, where reassignment can either be random, or based on its distance from the positive- and negative-class centroids.

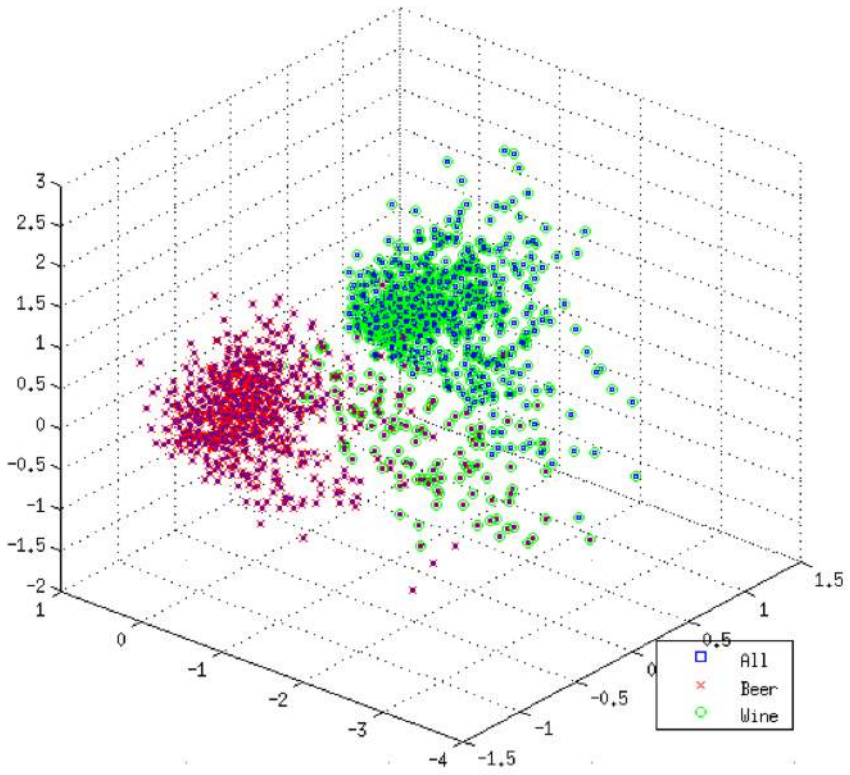


Figure 18: 3D visualization of training data for industry sectors 'Beer' and 'Wine'

instance.

3. If both the positive- and negative-class score is below a pre-defined threshold (e.g., 75%), continue classification on both the left and right subtrees, otherwise only follow the predicted path (i.e., the one greater than the threshold).

4. Repeat steps 1→3 for the root of each subtree to be followed.

5. If a leaf node is reached, add its associated sector label to the set of predicted labels.

6. Once the traversal is complete, return the full set of predicted labels.

Classifying in this way has the added benefit of reducing the total number of classifications required when a subtree is not followed. For this reason, however, it is important to structure the hierarchy in an appropriate way, so that dissimilar labels are further apart (e.g., we should be able to tell that an article about drilling for oil is not similar to an article about a new type of yoghurt early on in our traversal).

Therefore, to generate an appropriate hierarchy, we can use agglomerative hierarchical clustering, in the same way as was used for merging original sector labels, as described in Section 5.1.2.

Although this method has yet to be exhaustively tested, and no improvements have been found, further experimentation to aspects such as the number of features, hierarchical structure, and threshold values may still find it to be a superior method.

10 Conclusions

This thesis describes several possible approaches and techniques for tackling learning on large, highly-skewed, multi-label data sets in a real-world scenario: the online news-monitoring system, PULS. As the task is complex, the approach requires due attention be paid to each of the steps involved in the construction, optimization, and evaluation of the system. When broken down into these steps, one can compare and contrast the variety of existing methodologies—in addition to exploring novel techniques—to produce optimal results both for each individual component and for the final working classification system. The process of constructing the industry-sector classifier, as described here, highlights many of the major challenges involved in multi-label learning. Through detailed experimentation, the most suitable methods for overcoming these challenges are found.

One of the initial major obstacles in building the multi-label classifier is working with the large predefined label-set of the annotated PULS data. Overlapping label concepts, ambiguous naming conventions, and unclear hierarchical structure all contribute to incorrect or poorly defined output. One way to counteract the issue of ambiguity is to create a mapping of the original labels onto a smaller, more concise label space. This is achieved by doing hierarchical clustering on labels that are semantically linked—as determined through the use of latent semantic analysis—and manually merging those that cover the same conceptual space. For handling the disparity between the implicit hierarchical structure of label concepts and the flat nature of the pre-defined PULS labels, a two-level hierarchy is created that maps each level-II label to a single level-I label. By formulating the level-I labels to be at the same level of generality, this hierarchy can be utilized both in classification and evaluation to provide a "high-level" accuracy for difficult-to-classify test instances. An experiment comparing the test scores of level-II label classifiers against the level-I labels they map to shows that level-I performance and evaluation are more reliable,

thus even a simple hierarchy can improve system performance. While other methods exist for leveraging hierarchical relationships between class labels, they are difficult to implement in a real-world setting such as PULS.

In choosing how to build the multi-label learner itself, both performance and the problem setting must be considered. Efficient and accurate classification methods with low storage costs are necessary for the PULS sector classifier. A baseline rote classifier based on pre-extracted IE features provides an efficient and relatively accurate solution when PULS is able to extract the correct named-entities from documents. This unreliability, however, may be overcome using binary-relevance statistical classifiers, trained on extended bag-Of-word features, with part-of-speech-filtered unigrams and bigrams. While the storage requirements for using all available features would be prohibitive, dimensionality reduction via feature-selection provides a succinct representation to improve both speed and performance. In addition to concerns of efficiency, PULS must be able to handle data from numerous sources. The distribution of industry sectors covered by one source will likely be different from the distribution of others, thus overfitting to the training distribution is a major concern. A data-balancing procedure is used for training more general classifiers that can handle data from multiple sources and which might evolve and change over long time periods. Comparisons between two classifier models, SVM and Naive Bayes, and several feature-selection methods, including the novel Entropy-Filtering method, show that SVM classifiers with information gain feature-selection performs best on both the PULS and Reuters data sets. By also comparing macro-averaged F-measures for an array of classifiers trained on balanced data against the same classifiers trained on unbalanced data, it is shown that balanced training does indeed produce better "general" classifiers.

While both the IE-based rote classifier and binary-relevance statistical classifiers have their respective drawbacks, overall performance can be improved through the combination of these simpler methods. A multi-stage system for combining the output of each method is constructed, and operates in two distinct modes: *pipeline-combined*, which gives the system a "second-chance" at predicting missed instances from the first stage; and *union-combined*, which merges output from both sources. Both allow for better performance, without greatly increasing overhead by introducing highly-complex methods. Experiments on PULS and RCV1 data sets find the top performing two-stage classifier, Rote $\cup$ SVM+IG. When compared to other research performing multi-label classification on RCV1, this method improves on the state-of-the-art for macro-averaged F1 performance, indicating better performance

for a domain such as PULS. A three-stage classifier that integrates the two-level hierarchy is also constructed to further performance for PULS data. It is currently used in the PULS system.

In addition to producing a high-performance industry-sector classifier for use in PULS, the research in this thesis also contributes to overall research in both NLP and machine learning. The comparison of numerous techniques in IE and supervised learning highlight the appropriateness of each method for various tasks, especially with regards to real-world systems, which can guide future research in similar domains. In directly utilizing sophisticated PULS tools to generate IE-based features, such as company names and descriptors, the sector classifier successfully integrates IE with multi-label learning in a novel way. This method has been published in international conference proceedings [DPPY15]. A novel feature-selection method, Entropy-Filtering is defined, and produces comparable or better results when applied on top of other methods. The class-data balancing procedure used in training statistical learners provides improved performance over unbalanced training when evaluated for macro-average, providing a simple technique that helps reduce class overfitting for a specific domain, and can be useful in an environment where data is drawn from numerous sources. This balancing procedure has also been published [DPPY14]. Finally, as mentioned above, the multi-stage multi-label classifier shows improved performance over state-of-the-art results when evaluated for macro-average F-measure; another published result.

11 Future Work

The experiments outlined in this thesis are by no means comprehensive, and there are many directions to be explored in future work:

Learning Algorithms Due to time and resource constraints, only a limited number of learning algorithms are evaluated in the experiments of this thesis. In the future, a more comprehensive analysis should be done, including algorithms such as logistic regression, Gaussian process classification, and neural networks. Each algorithm should also be explored in greater depth by exhaustively comparing parameters settings and modalities of the different algorithms.

Feature Engineering Along with learning algorithms, feature-selection methods should also be expanded upon, to include, for example, filter methods, subset-selection methods, and methods that account for imbalance between positive- and negative-class features. Also, while features derived using principal component analysis (PCA) were discussed in the exploration of hierarchical classification (Section 9), a comprehensive comparison of results for classifiers trained on these features should be performed.

Threshold and Parameter Optimization Although the experiments make use of a development set to produce optimal global prediction thresholds for various classifiers, additional optimization can be done, for example to produce optimal local thresholds for individual binary classifiers in an array. Other parameters could also be optimized in future work, such as algorithm-specific parameters of classification methods (for instance the margin constant in the SVM algorithm), or the number of dimensions to use when reducing data for LSA-based sector clustering.

Multi-Label Schemes This thesis explores two main schemes for multi-label classification, namely binary-relevance and a simple rule-based multi-label classifier, in the rote classifier. Other schemes such as Label Powerset, Pruned Sets, and Pruned Problem Transformation (PPT) can also be applied to the problem setting. Future work should also explore multi-label adaptations of statistical algorithms, such as multi-label Naive Bayes.

Evaluation Metrics As outlined in Section 8.4, the current evaluation measures are limited in their ability to accurately represent overall system performance. A new evaluation scheme should be developed that accounts for annotator bias, conceptual ambiguity between labels, and the presence of "wrong-but-relevant" predictions. Future work could also incorporate results from additional measures such as *Hamming loss*, *zero-one loss*, and *Jaccard similarity*, to produce a clearer overall picture.

12 Acknowledgements

I would like to acknowledge and express my sincere gratitude for the numerous and significant contributions—both to this thesis in particular, and to our research efforts in general—of the PULS team: Lidia Pivovarova, Mian Du, Peter Von Etter,

Natalia Ostapuk, and Dr. Roman Yangarber. Instrumental guidance from Dr. Sotiris Tasoulis is also greatly appreciated. I would further like to recognize the financial support received from the PULS project at the University of Helsinki, and by the PULS/ARR Project of the Stockholm Environment Institute. Finally, I would like to thank my closest family and friends for—at countless times and in countless ways along the voyage—helping me to plug the leaks, bail out the water, and man the life raft. I'll sail with you lot any day.

References

AP05 Artstein, R. and Poesio, M., Bias decreases in proportion to the number of annotators. *Proceedings of FG-MoL*, pages 141–150.

APvdGY11 Atkinson, M., Piskorski, J., van der Goot, E. and Yangarber, R., Multilingual real-time event extraction for border security intelligence gathering. In *Counterterrorism and Open Source Intelligence*, Wiil, U. K., editor, Springer Lecture Notes in Social Networks, Vol. 2, 2011, pages 355–390.

BLSB04 Boutell, M. R., Luo, J., Shen, X. and Brown, C. M., Learning multi-label scene classification. *Pattern Recognition*, 37,9(2004).

BPM04 Batista, G. E., Prati, R. C. and Monard, M. C., A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6,1(2004), pages 20–29.

BT11 Box, G. E. and Tiao, G. C., *Bayesian inference in statistical analysis*, volume 40. John Wiley & Sons, 2011.

CBHK02 Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P., SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16,1(2002), pages 321–357.

CL96 Cowie, J. and Lehnert, W., Information extraction. *Communications of the ACM*, 39,1(1996), pages 80–91.

CM07 Ceci, M. and Malerba, D., Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28,1(2007), pages 37–78.

CMM83 Carbonell, J. G., Michalski, R. S. and Mitchell, T. M., Machine learning: a historical and methodological analysis. *AI Magazine*, 4,3(1983), page 69.

CS13 Cohn, T. and Specia, L., Modelling annotator bias with multi-task Gaussian processes: An application to machine translation quality estimation. *ACL (1)*. Citeseer, 2013, pages 32–42.

CUAG13 Cisse, M. M., Usunier, N., Arti, T. and Gallinari, P., Robust Bloom filters for large multilabel classification tasks. *Advances in Neural Information Processing Systems*, 2013, pages 1851–1859.

dCF09 de Carvalho, A. and Freitas, A., A tutorial on multi-label classification techniques. In *Foundations of Computational Intelligence Volume 5*, Springer, 2009, pages 177–195.

DE84 Day, W. H. and Edelsbrunner, H., Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1,1(1984), pages 7–24.

DH+03 Drummond, C., Holte, R. C. et al., C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. *Workshop on learning from imbalanced datasets II*, volume 11. Citeseer, 2003.

DK10 Dendamrongvit, S. and Kubat, M., Undersampling approach for imbalanced training sets and induction from multi-label text-categorization domains. In *New Frontiers in Applied Data Mining*, Springer, 2010, pages 40–52.

DPPY14 Du, M., Pierce, M., Pivovarova, L. and Yangarber, R., Supervised classification using balanced training. In *Statistical Language and Speech Processing*, Springer, 2014, pages 147–158.

DPPY15 Du, M., Pierce, M., Pivovarova, L. and Yangarber, R., Improving supervised classification using information extraction. In *Natural Language Processing and Information Systems*, Springer, 2015, pages 3–18.

DS05 Debole, F. and Sebastiani, F., An analysis of the relative hardness of Reuters-21578 subsets. *Journal of the American Society for Information Science and technology*, 56,6(2005), pages 584–596.

DVW+12 D'hondt, E., Verberne, S., Weber, N., Koster, C. and Boves, L., Using skipgrams and PoS-based feature selection for patent classification. *Computational Linguistics in the Netherlands*.

EA13 Erenel, Z. and Altınçay, H., Improving the precision-recall trade-off in undersampling-based binary text categorization using unanimity rule. *Neural Computing and Applications*, 22,1(2013), pages 83–100.

For03 Forman, G., An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3, pages 1289–1305.

For04 Forman, G., A pitfall and solution in multi-class feature selection for text classification. *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, New York, NY, USA, 2004, ACM, pages 38–45.

GCR09 Gauch, S., Chandramouli, A. and Ranganathan, S., Training a hierarchical classifier using inter-document relationships. *Journal of the American Society for Information Science and Technology*, 60,1(2009), pages 47–58.

GHY03 Grishman, R., Huttunen, S. and Yangarber, R., Information extraction for enhanced access to disease outbreak reports. *Journal of Biomedical Informatics*, **35**,4(2003), pages 236–246.

GR70 Golub, G. H. and Reinsch, C., Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14,5(1970), pages 403–420.

GS96 Grishman, R. and Sundheim, B., Message Understanding Conference-6: A brief history. *COLING*, volume 96, 1996, pages 466–471.

GW98 Gaizauskas, R. and Wilks, Y., Information extraction: Beyond document retrieval, 1998.

GZ13 Gao, W. and Zhou, Z.-H., On the consistency of multi-label learning. *Artificial Intelligence*, 199–200,0(2013), pages 22–44.

HE07 Hatami, N. and Ebrahimpour, R., Combining multiple classifiers: diversify with boosting and combining by stacking. *Int. J. Comput. Sci. Network Security*, 7,1(2007), pages 127–31.

Hen09 Henderson, L., Automated text classification in the DMOZ hierarchy, 2009.

HFH+09 Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H., The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11,1(2009), pages 10–18.

HNA+13 Hartley, D. M., Nelson, N. P., Arthur, R., Barboza, P., Collier, N., Lightfoot, N., Linge, J., Goot, E., Mawudeku, A., Madoff, L. et al., An overview of Internet biosurveillance. *Clinical Microbiology and Infection*, 19,11(2013), pages 1006–1013.

HR13 Huang, R. and Riloff, E., Classifying message board posts with an extracted lexicon of patient attributes. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pages 1557–1562.

HTF09 Hastie, T., Tibshirani, R. and Friedman, J., *The elements of statistical learning*, volume 2. Springer, 2009.

HVDY12 Huttunen, S., Vihavainen, A., Du, M. and Yangarber, R., Predicting relevance of event extraction for the end user. In *Multi-source, Multilingual Information Extraction and Summarization*, Poibeau, T., Saggion, H., Piskorski, J. and Yangarber, R., editors, Theory and Applications of Natural Language Processing, Springer Berlin, 2012, pages 163–176.

HYG02 Huttunen, S., Yangarber, R. and Grishman, R., Diversity of scenarios in information extraction. *LREC*, 2002.

IKT05 Ikonomakis, M., Kotsiantis, S. and Tampakas, V., Text classification using machine learning techniques. *WSEAS Transactions on Computers*, 4,8(2005), pages 966–974.

JM00 Jurafsky, D. and Martin, J. H., *Speech & Language Processing*. Pearson Education India, 2000.

Jol02 Jolliffe, I., *Principal component analysis*. Wiley Online Library, 2002.

JOP+ Jones, E., Oliphant, T., Peterson, P. et al., SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed 2015-03-04].

JS02 Japkowicz, N. and Stephen, S., The class imbalance problem: A systematic study. *Intelligent data analysis*, 6,5(2002), pages 429–449.

KKP+06 Kotsiantis, S., Kanellopoulos, D., Pintelas, P. et al., Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30,1(2006), pages 25–36.

KM03 Klein, D. and Manning, C. D., Accurate unlexicalized parsing. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2003, pages 423–430.

KS97 Koller, D. and Sahami, M., Hierarchically classifying documents using very few words. Technical Report 1997–75, Stanford InfoLab, February 1997.

LLS09 Liu, Y., Loh, H. T. and Sun, A., Imbalanced text classification: a term weighting approach. *Expert Systems with Applications*, 36,1(2009), pages 690–701.

LWZ09 Liu, X.-Y., Wu, J. and Zhou, Z.-H., Exploratory undersampling for class-imbalance learning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39,2(2009), pages 539–550.

LYRL04 Lewis, D. D., Yang, Y., Rose, T. G. and Li, F., RCV1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5, pages 361–397.

MAT13 MATLAB, *version 8.2 (R2013b)*. The MathWorks Inc., Natick, Massachusetts, 2013.

Mil80 Milligan, G. W., An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika*, 45,3(1980), pages 325–342.

MKGD12 Madjarov, G., Kocev, D., Gjorgjevikj, D. and Džeroski, S., An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45,9(2012), pages 3084–3104. Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011).

MRT12 Mohri, M., Rostamizadeh, A. and Talwalkar, A., *Foundations of machine learning*. MIT Press, 2012.

P+99a Platt, J. et al., Fast training of Support Vector Machines using sequential minimal optimization. *Advances in kernel methods–support vector learning*, 3.

P+99b Platt, J. et al., Probabilistic outputs for Support Vector Machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10,3(1999), pages 61–74.

PBM04 Prati, R. C., Batista, G. E. and Monard, M. C., Class imbalances versus class overlapping: An analysis of a learning system behavior. *MICAI 2004: Advances in Artificial Intelligence: Third Mexican International Conference on Artificial Intelligence, Mexico City, Mexico, April 26-30, 2004, Proceedings*, volume 3. Springer, 2004, pages 312–321.

PLM+04 Plisson, J., Lavrac, N., Mladenic, D. et al., A rule based approach to word lemmatization. *Proceedings of IS-2004*, pages 83–86.

Puu12 Puurula, A., Scalable text classification with sparse generative modeling. In *PRICAI 2012: Trends in Artificial Intelligence*, Anthony, P., Ishizuka, M. and Lukose, D., editors, volume 7458 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2012, pages 458–469.

PVG+11 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, pages 2825–2830.

Ris01 Rish, I., An empirical study of the Naive Bayes classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3. IBM New York, 2001, pages 41–46.

RLC+09 Ren, J., Lee, S. D., Chen, X., Kao, B., Cheng, R. and Cheung, D., Naive Bayes classification of uncertain data. *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, 2009, pages 944–949.

SC08 Steinwart, I. and Christmann, A., *Support Vector Machines*. Springer, 2008.

Seb02 Sebastiani, F., Machine learning in automated text categorization. *ACM computing surveys*, 34,1(2002), pages 1–47.

Sor10 Sorower, M. S., A literature survey on algorithms for multi-label learning. Technical Report, Oregon State University, Corvallis, OR, USA (December 2010), 2010.

Sta08 Stamatatos, E., Author identification: Using text sampling to handle the class imbalance problem. *Information Processing & Management*, 44,2(2008), pages 790–799.

Tan11 Tange, O., Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36,1(2011), pages 42–47. URL `http://www.gnu.org/s/parallel`.

TB03 Tikk, D. and Biró, G., Experiments with multi-label text classifier on the Reuters collection. *Proceedings of the International Conference on Computational Cybernetics (ICCC 03)*, 2003, pages 33–38.

TK07 Tsoumakas, G. and Katakis, I., Multi-label classification: an overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3,3(2007), pages 1–13.

TKV10 Tsoumakas, G., Katakis, I. and Vlahavas, I., Mining multi-label data. In *Data mining and knowledge discovery handbook*, Springer, 2010, pages 667–685.

Vap00 Vapnik, V., *The nature of statistical learning theory*. springer, 2000.

WBPB10 Welinder, P., Branson, S., Perona, P. and Belongie, S. J., The multidimensional wisdom of crowds. *Advances in neural information processing systems*, 2010, pages 2424–2432.

Yan99 Yang, Y., An evaluation of statistical approaches to text categorization. *Information retrieval*, 1,1-2(1999), pages 69–90.

Yan06 Yangarber, R., Verification of facts across document boundaries. *Proc. International Workshop on Intelligent Information Access*, 2006, pages 6–8.

YP97 Yang, Y. and Pedersen, J. O., A comparative study on feature selection in text categorization. *ICML*, volume 97, 1997, pages 412–420.

YS09 Yangarber, R. and Steinberger, R., Automatic epidemiological surveillance from on-line news in MedISys and PULS. *Proceedings of IMED-2009: International Meeting on Emerging Diseases and Surveillance*, Vienna, Austria, 2009.

ZYT11 Zhang, W., Yoshida, T. and Tang, X., A comparative study of TF*IDF, LSI and multi-words for text classification. *Expert Systems with Applications*, 38,3(2011), pages 2758 – 2765.

ZZ07 Zhang, M.-L. and Zhou, Z.-H., Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40,7(2007), pages 2038–2048.

ZZY+05 Zhuang, D., Zhang, B., Yang, Q., Yan, J., Chen, Z. and Chen, Y., Efficient text classification by weighted proximal SVM. *Fifth IEEE International Conference on Data Mining*, 2005.