Multilingual Text Classification using Ontologies

Gerard de Melo
J.W. Goethe-Universität Frankfurt am Main

December 2006

Advisors:
Prof. Rüdiger Brause, Johann Wolfgang Goethe-Universität, Frankfurt am Main
Dr. Stefan Siersdorfer, Max-Planck-Institut für Informatik, Saarbrücken

Statement: I hereby declare that this submission is my own work, and that any sources or resources used are explicitly acknowledged in the thesis.

Erklärung: Ich versichere, dass ich diese Diplomarbeit selbstständig verfasst, und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Frankfurt am Main, 21. Dezember 2006

Abstract

This thesis investigates strategies for automatically classifying documents provided in different languages thematically, geographically, or according to other criteria, and presents Ontology Region Mapping, a novel linguistically motivated text representation method that can be used with adaptive machine learning algorithms in order to learn classifications from pre-classified examples and then automatically classify documents that might be given in completely different languages. Ontology Region Mapping associates terms occurring in a text with concepts represented in formal ontologies and lexical resources, thereby, however, going beyond a simple mapping from terms to concepts by fully exploiting the external knowledge manifested in the resources and mapping to entire regions of concepts. In order to do so, a graph traversal algorithm is used that explores further related concepts that might be relevant. Extensive testing has shown that this method leads to significant improvements compared to existing approaches.

Zusammenfassung

Die vorliegende Arbeit untersucht Strategien zur automatischen Klassifizierung von Dokumenten, die in unterschiedlichen Sprachen vorliegen, wobei die Zuordnung nach thematischen, geographischen oder anderen Kriterien erfolgen kann. Vorgestellt wird ein neues linguistisch motiviertes Textrepräsentationsverfahren namens Ontology Region Mapping, mit dem eine Klassifikation anhand bereits markierter Beispieldokumente mit adaptiven Algorithmen gelernt werden kann, um sodann ein Klassifizieren neuer Dokumente zu ermöglichen, selbst wenn diese in unterschiedlichen Sprachen vorliegen. Ontology Region Mapping verknüpft Begriffe aus den Dokumenten mit Konzepten, die von formalen Ontologien oder lexikalischen Ressourcen spezifiziert werden, geht dabei jedoch über ein direktes Abbilden dieser Terme auf Konzepte hinaus, in dem es das in derartigen Ressourcen manifestierte Wissen effektiver ausnutzt und gesamte Regionen von Konzepten berücksichtigt. Zu diesem Zweck wird ein graphentheoretisches Verfahren zur Entdeckung weiterer potenziell relevanter Konzepte eingesetzt. Umfangreiche Tests haben ergeben, dass diese Methode im Vergleich zu existierenden Ansätzen signifikante Verbesserungen erzielt.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Language is the vehicle of thought and the fundamental tool that has allowed us to develop human culture, accumulate knowledge, and communicate with other human beings. At the same time, language today is still one of the prime reasons whenever there is a lack of communication, as language barriers represent significant obstacles to those wishing to communicate with other cultures or access information.

The need of having to organize knowledge in multiple languages has been an important issue for a very long time now, considering that even the ancient Library of Alexandria is thought to have held hundreds of thousands of scrolls written in different languages. The digital age has significantly improved our prospects of successfully tackling such tasks and has enabled us to make information a lot more accessible, yet at the same time the volume of available data has increased exponentially, meaning that good principles of organization are highly essential.

Automatic text classification (also known as text categorization) is the process of automatically associating text documents with the categories or classes considered most appropriate, thereby distinguishing, for instance, texts from the field of particle physics from those dealing with optical physics (cf. Figure 1.1). Although research in this area has progressed to a point where certain techniques have become well-established with successful application in many production settings, nearly all known approaches presuppose that every single document is provided in the same single language. A multilingual approach to text classification, in contrast, would allow for adding new documents in several different languages to a collection without the time-consuming need for someone to manually decide to which categories or classes these documents belong, and people

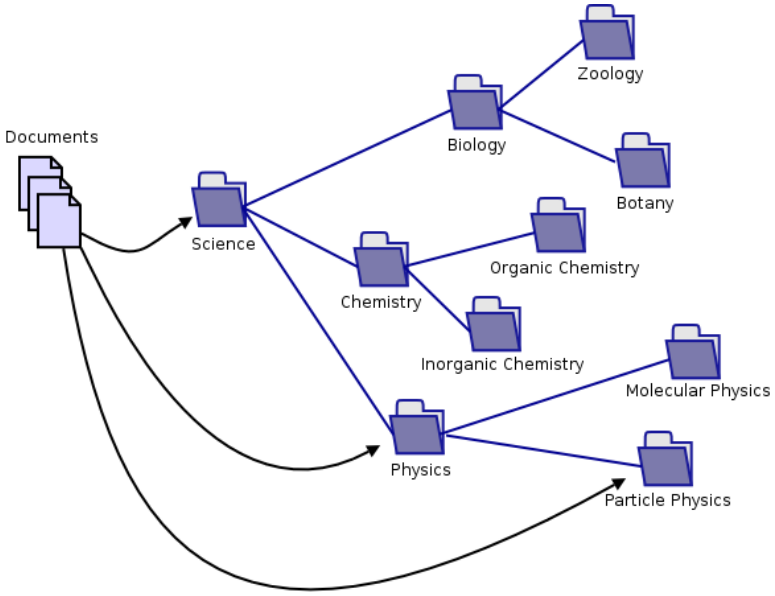

Figure 1.1: Classification: For each document we need to determine which class labels are appropriate.

seeking information could then find relevant documents written in languages other that the ones they normally use for searching and browsing. Once it is known which documents are relevant, if necessary, automatic or manual translations can be performed, or if the user has a very basic understanding of the language, the document can be read using a multilingual comprehension assistant (see Grefenstette and Segond, 2003, pp. 708ff).

This, however, is just one main motivation for dealing with text classification. It should be considered that apart from truly multilingual environments in places such as India and large parts of Africa, people all over the world are getting used to speaking a *lingua franca* such as English, Spanish, and Swahili in addition to their native local languages. It has been estimated that at least 50% of Africa's population is multilingual (Wolff, 2000), and far less than 10% of the world's population speak English as their native language, so it is not surprising that most applications of monolingual text classification also turn out to be potential applications for multilingual text classification, including the following.

- News wire filtering: The large quantity of news articles received at news agencies and editorial departments needs to be classified so that recipients only receive those

articles that suit their specific profiles of interest (Sebastiani, 2006). For instance, articles about the latest developments of oil prices in Venezuela might not be of interest to subscribers of Italian sports journals. International news agencies deal with articles coming from all parts of the world, presented in different languages.

- Traditional libraries and digital libraries: The vast resources available to us in our Information Age need to be well-organized for them to be of any practical use to us. Even within single issues of journals, one can often find articles written in different languages.

- Web page and web site classification: Text classification assists in organizing World Wide Web pages and sites with respect to a given taxonomy. This might also occur on-the-fly in the context of an information retrieval request from a search engine or in the context of content filtering systems.

- E-mail: In many countries users receive messages in different languages on a daily basis. Multilingual text classification allows incoming messages to be automatically placed into different categories, separating work-related matters, private correspondence, and unsolicited e-mails. These techniques can also be used in order to automatically forward messages to the most appropriate people in organizations.

- Surveillance technologies: Due to the high volume of documents (e.g. e-mails, conversation logs) that need to be scanned when gathering intelligence, it is necessary to automatically select a small amount of suspicious documents for further review from the millions of documents that typically pass through surveillance systems.

- Document management systems: Many organizations and companies nowadays operate in several different countries or in multilingual regions and need to store and organize documents in various languages.

1.2 Contribution

In this thesis we analyse the problem of automatic multilingual text classification, evaluating several strategies for classifying texts from multilingual document sets. We provide linguistic arguments against some existing approaches to the problem and devise a new approach, called Ontology Region Mapping, that exploits background knowledge from ontologies and lexical resources by first mapping the terms occurring in a document to concepts associated with an ontology, disambiguating between different possible word

senses, and then exploiting the background knowledge manifested in such ontologies by weighting not only immediately connected concept identifiers but also multiple levels of related concepts using a graph traversal algorithm. We propose strategies for setting the parameters and additionally introduce several combined text representations for decreasing the error rate even further. Although the model proposed is designed to cope particularly well with several phenomena inherent to multilingual documents, it likewise offers a viable alternative approach to monolingual text classification.

1.3 Overview

The remainder of this document is organized as follows. We begin with a basic introduction to some fundamental concepts and techniques used in automatic text classification and natural language processing in Chapter 2, followed by a presentation of some alternative approaches to text classification related to our work in Chapter 3. Based on an analysis of the problem of multilingual text classification, Chapter 4 will attempt to identify weaknesses of those alternative approaches, while Chapter 5 will proceed with the presentation of a novel approach called Ontology Region Mapping. Additional considerations to be made when implementing this approach are described in Chapter 6, followed by an experimental evaluation of it in Chapter 7. Finally, the concluding Chapter 8 will outline some of the implications for continued research in this area.

Chapter 2

Background

This chapter provides an introduction to how monolingual automatic text classification is conventionally performed and presents some well-known techniques and resources from the domain of natural language processing. It will serve as a prerequisite for a proper understanding of the new techniques presented later on. The first section will introduce the basics of learning classifications automatically by means of machine learning algorithms, while the second section will then describe how these algorithms can be of use in the case of monolingual text documents. A final third section will provide information on certain additional natural language processing techniques and resources that will be of use later on when more sophisticated means of using such learning algorithms on text documents will be presented.

2.1 Learning Classifications

Automatic classification is the process of establishing and deploying *classifiers* $\widehat{\phi}$ that approximate classifications ϕ made by human experts.

Definition 2.1.1. A *classification* is an assignment of class labels $C \in \mathcal{C}$ to objects $D \in \mathcal{D}$ in the form of a function $\phi : \mathcal{D} \times \mathcal{C} \longrightarrow \{\top, \bot\}$ where $\top$ indicates that object $D \in \mathcal{D}$ is assigned class label $C \in \mathcal{C}$, and $\bot$ indicates that this is not the case (Sebastiani, 2002).

Definition 2.1.2. A text classification problem $\mathcal{T}$ consists in finding a classifier $\widehat{\phi}$ for documents $D \in \mathcal{D}$ needing to be categorized using fixed class labels $C \in \mathcal{C}$ based on their contents such that a given classification ϕ is approximated with a high level of effectiveness, perhaps in terms of low error rates or other evaluation measures (see Section 7.2.4).

In the context of this thesis, $\mathcal{D}$ will be a set of documents and each class label $C \in \mathcal{C}$ will correspond to a category such a document might belong to. For instance, there could be categories for nuclear physics, astrophysics, and optics, among others.

Additional conditions and constraints may apply to classification tasks. One may demand that the set of class labels of an object D, $\mathit{classes}(D) := \{C \in \mathcal{C} \mid \phi(D, C)\}$, always have a cardinality of 1, meaning that every object must be assigned exactly one class label $C \in \mathcal{C}$. This case is called *single-label classification*, whereas in *multi-label classification* every object can have zero, one, or multiple class labels.

Binary classification is a special case of single-label classification that imposes an additional constraint, this time on the cardinality of $\mathcal{C}$, requiring that $|\mathcal{C}| = 2$. This implies that every object is assigned to exactly one of two possible classes. Frequently, the second class involved is simply the complement $\overline{C}$ of the first class C. In contrast, if no constraints are made regarding the cardinality of $\mathcal{C}$, we have a *multi-class classification*. When this is the case, there might additionally also be dependencies between classes in the sense that certain classes are sub-classes of other ones, a special case called *hierarchical classification*.

In the past, rules for classifying objects frequently used to be handcrafted by human experts and then the classification process could run automatically by following those rules. In order to perform classifications in a truly automated manner without the need for human experts to program the system, one may now rely on techniques from machine learning, a branch of Artificial Intelligence that deals with automated methods enabling a machine to acquire knowledge inductively from examples. Such examples are much easier to produce than handcrafted rules for an expert system. In text classification, the examples are pre-classified documents, so the general type of learning process is called *supervised* learning. The respective algorithms attempt to learn an existing classification ϕ using pre-classified example objects, creating a classifier $\widehat{\phi}$ which is hoped to be able to classify new, hitherto unclassified objects correctly. In many cases, a sufficient number of classified examples already exist from the time before the transition to a new automatic classification solution (Sebastiani, 2006). For example, if a system is supposed to be able to automatically distinguish articles about nuclear physics from articles about astrophysics, one could perhaps use 100 existing articles about nuclear physics and further 100 existing articles about astrophysics as pre-classified examples. The machine learning algorithm would then study those examples and attempt to construct a model that could be used to make the distinction for new, unseen documents. The set of training objects is called the *training set* $\mathcal{D}_{\mathrm{tr}} \subseteq \mathcal{D}$. In research settings, another set of

pre-classified objects, called the *test set* $\mathcal{D}_{\text{tst}} \subseteq \mathcal{D}$ ($\mathcal{D}_{\text{tr}} \cap \mathcal{D}_{\text{tst}} = \emptyset$), is used to test how well the learnt classifier approximates the original classification.

An adequate representation of the objects in $\mathcal{D}$ is an important prerequisite for being able to apply learning algorithms. It is normally necessary to identify certain attributes of the objects that are considered relevant, and represent them as discrete or continuous *features*. The algorithms referred to in the context of this thesis assume that the features are represented by real numbers. When categorizing automobiles one could use features representing engine power, weight, and maximum speed, among others. Section 2.2 will describe a common way of representing text documents using real-valued features that is mainly based on how often particular terms occur within a document.

For each learning process, a fixed set of features is finalized as well as a fixed order of the features, enabling us to define a vector space in which each object $D \in \mathcal{D}$ may be represented by a so-called *feature vector*, an n-dimensional vector $\mathbf{d} = (d_1, \ldots, d_n)$ of feature values in the Euclidean space $\mathbb{R}^n$. Each feature corresponds to a separate dimension of this vector space, so in the case of automobiles one dimension could represent the engine power, another the weight, and so on.

A number of algorithms exist for learning classifications by analysing a series of labelled feature vectors representing objects in the training set $\mathcal{D}_{\text{tr}}$ and then inductively constructing a classifier $\widehat{\phi}$ that can then be applied to new objects such as the ones in the test set $\mathcal{D}_{\text{tst}}$. Three algorithms that have successfully been used for classifying text documents will now be presented.

2.1.1 Artificial Neural Networks

Inspired by models of the functioning of neurons in the human brain, Artificial Neural Networks (ANNs) are networks of interconnected units, where each unit takes one or more real numbers as input and produces a single real number as output. In order to classify an object, each component value of the object's feature vector $\mathbf{d}$ is passed to a separate *input unit*. As shown in Figure 2.1, the activation of such units is propagated through the network, and the output of a designated set of *output units* determines the classification prediction made by the network.

The most basic type of Artificial Neural Network is composed of a single layer of units called *Perceptrons* (Rosenblatt, 1958). A single layer in this context implies that the

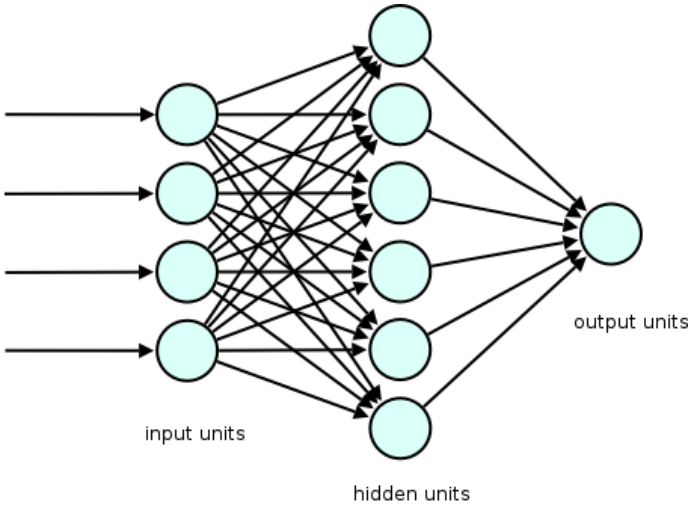

Figure 2.1: Artificial Neural Networks

input units directly pass their activation to the output units without any additional *hidden units* existing. The output of a Perceptron for input $\mathbf{d} = (d_1, \ldots, d_n)$ is

$$o(\mathbf{d}) = \begin{cases} 1 & w_0 + w_1 d_1 + w_2 d_2 + \ldots + w_n d_n > 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.1}$$

for some weight vector $\mathbf{w} \in \mathbb{R}^n$ and an additional threshold constant $w_0 \in \mathbb{R}$. A Perceptron therefore implements a linear discriminant which in turn represents a hyperplane decision surface $\mathbf{w} \cdot \mathbf{x} + w_0 = 0$ in the feature space (Figure 2.2). The output is 1 for instances lying on one side of the feature space and -1 for instances lying on the other, allowing for binary classification problems to be solved where examples from one class (*positive examples*) need to be separated from examples associated with another class (*negative examples*). Problems allowing for a separation of positive from negative instances by means of such a hyperplane decision surface are called *linearly separable* problems.

In order to learn a classification, the weights are adapted to a set of training instances $\mathbf{d}_1, \ldots, \mathbf{d}_m$ (each $\mathbf{d}_i = (d_{i,1}, \ldots, d_{i,n})$), possibly passed to the network multiple times (in several *epochs*). One way of doing this is to initially set all w_j to some default value or to random values and then iteratively apply the Perceptron to the training instances $\mathbf{d}_i$, modifying the w_j as

$$w_j \leftarrow w_j + \eta(c - p)d_{i,j} \tag{2.2}$$

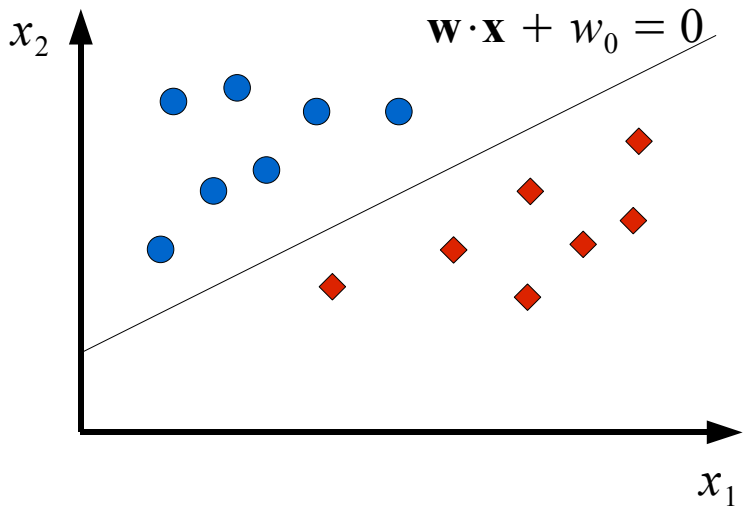

Figure 2.2: Linearly separable problem

where we imagine extended document vectors $\mathbf{d}_i$ by defining $d_{i,0}$ as 1 in order to simultaneously account for w_0 being updated (Mitchell, 1997, ch.4). Here, $c \in \{\pm 1\}$ represents the true classification as the desired target output for $\mathbf{d}_i$, $p \in \{\pm 1\}$ is the actual output received as a prediction, and $\eta > 0$ is a value called the *learning rate* which determines to what degree the weights are adapted (η can also be changed over time). If the output matches the original classification, the Perceptron is left unchanged because $c - p = 0$. In the case of a misclassification, however, the weight vector is adapted in accordance with $\mathbf{d}_i$. It has been shown that this method is guaranteed to converge if a sufficiently small learning rate is used and the training instances are indeed linearly separable (Mitchell, 1997, p.89). Since it might not always be possible and desirable to continue iterating up to the point of convergence, one may for example use the Voted-Perceptron algorithm by Freund and Schapire (1998), an extension of this approach that combines the update rule mentioned above with a leave-one-out-strategy: When classifying an unseen example $\mathbf{d}$, rather than just looking at the final $o(\mathbf{d})$ using the weight vector $\mathbf{w}$ received at the end of the training process, a separate prediction $o(\mathbf{d})$ is made after each single training instance. The final prediction for $\mathbf{d}$ is then determined in a majority voting process.

For problems that are not linearly separable, more complex neural networks with multiple layers or non-linear transfer functions as in the case of radial basis function neural networks can be used, and the weights can be updated using back-propagation algorithms

(see Brause, 1999). While such techniques have proven useful for tasks such as image recognition, in the context of text classification the use of hidden units has not shown significant improvements over the single-layer Perceptron model (Sebastiani, 2002). We assume that the reason for this lies in the fact that, as will be explained later on, text documents are usually represented in very high-dimensional feature spaces, obviating the need to capture nonlinear relationships between single features.

2.1.2 Support Vector Machines

As explained earlier on, in the case of linearly separable binary decision problems there exist some hyperplanes in the feature space that separate the positive from the negative examples. The basic idea behind Support Vector Machines (SVM), introduced by Vapnik (1995), is to use the hyperplane that maximizes the distance to the closest positive and the closest negative examples by solving a constrained optimization problem, assuming that this minimizes the risk of confusing positive and negative instances when classifying (see Figure 2.3). These closest examples are called the *support vectors*, the sum of the two minimal distances, one for each class, between them and the hyperplane is also known as the *margin*, and hence the hyperplane chosen by a support vector machine is called the *maximum-margin hyperplane*. New instances can then be evaluated by computing the distances to the support vectors and determining in which half-space with respect to the maximum-margin hyperplane the vector lies.

Formally, a hyperplane can be characterized as $\mathbf{w} \cdot \mathbf{x} = w_0$ for some weight vector $\mathbf{w} \in \mathbb{R}^n$ and some scalar w_0, allowing $\widehat{\phi}(D, C)$ to be determined by a feature vector $\mathbf{d} \in \mathbb{R}$ for D and a binary decision rule for C characterized by $o(\mathbf{d}) = sgn(\mathbf{w} \cdot \mathbf{d} + w_0)$ with $o(\mathbf{d}) = 1$ indicating a positive and $o(\mathbf{d}) = -1$ indicating a negative classification, as in the case of the Perceptron algorithm. Let us assume that the training documents in $\mathcal{D}_{\text{tr}}$ are represented by $\mathbf{d}_1, \ldots, \mathbf{d}_m$ and their respective classifications are represented by $c_1, \ldots, c_m \in \{\pm 1\}$ ($c_i = 1$ for positive examples, -1 for negative ones). Since it is possible to rescale $\mathbf{w}$ and w_0 such that the points closest to the hyperplane satisfy $|\mathbf{w} \cdot \mathbf{d}_i + w_0| = 1$, one may assume that the maximum-margin hyperplane satisfies $c_i(\mathbf{w} \cdot \mathbf{d}_i + w_0) \geq 1$ for all $i \in 1, \ldots, m$, as illustrated in Figure 2.4.

For two points $\mathbf{d}_+$, $\mathbf{d}_-$ closest to this hyperplane on either side we then have $(\mathbf{w} \cdot \mathbf{d}_+) - (\mathbf{w} \cdot \mathbf{d}_-) = 2$, so the margin becomes:

$$\left(\frac{\mathbf{w}}{||\mathbf{w}||} \cdot \mathbf{d}_+ - \frac{\mathbf{w}}{||\mathbf{w}||} \cdot \mathbf{d}_-\right) = \frac{2}{||\mathbf{w}||} \tag{2.3}$$

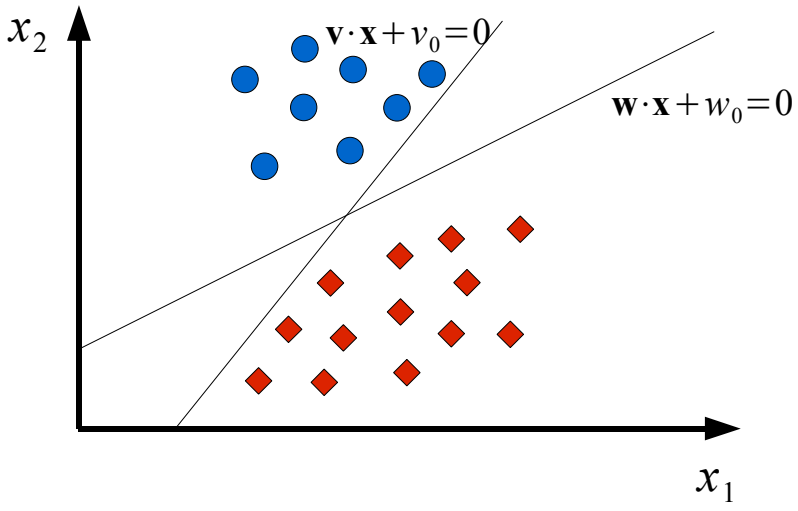

Figure 2.3: Multiple separating hyperplanes: Since $\mathbf{v}\cdot\mathbf{x}+v_0 = 0$ is closer to the training instances than $\mathbf{w}\cdot\mathbf{x}+w_0 = 0$, the latter is considered less likely to lead to misclassifications of new examples.

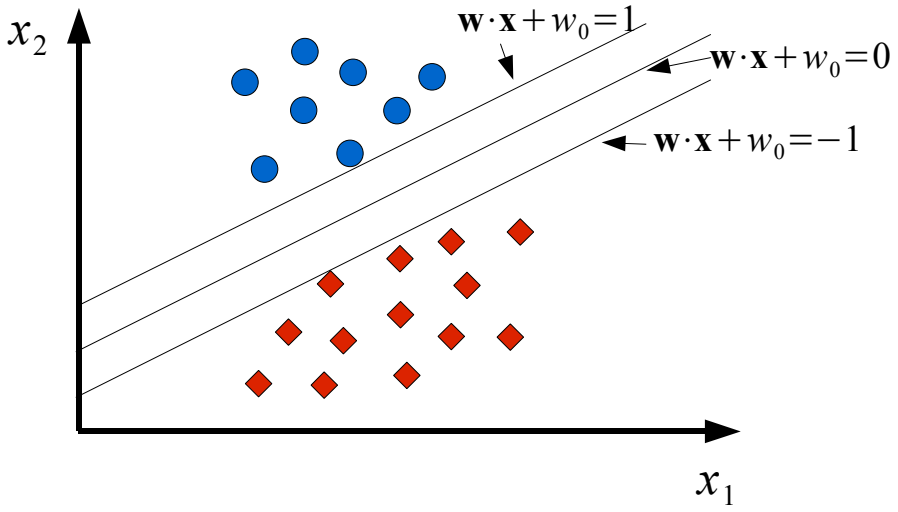

Figure 2.4: The maximum-margin hyperplane must satisfy $c_i(\mathbf{w}\cdot\mathbf{d}_i+w_0)\geq 1$ for all i. We can imagine two parallel hyperplanes marking this minimal distance.

Maximizing this margin is thus equivalent to solving the following quadratic optimization problem:

$$\begin{aligned} \text{minimize:} \quad & \frac{1}{2}\mathbf{w} \cdot \mathbf{w} \\ \text{subject to:} \quad & \forall i \in \{1, \ldots, m\} : \quad c_i[\mathbf{w} \cdot \mathbf{d}_i + w_0] \geq 1 \end{aligned} \tag{2.4}$$

Cortes and Vapnik (1995) introduce the soft margin hyperplane which extends this basic idea to cases when no hyperplane exists that perfectly separates *all* positive examples from all negative ones by allowing slight deviations ξ_i:

$$\begin{aligned} \text{minimize:} \quad & \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + \kappa \sum_{i=1}^{m} \xi_i \\ \text{subject to:} \quad & \forall i \in \{1, \ldots, m\} : \quad c_i[\mathbf{w} \cdot \mathbf{d}_i + w_0] \geq 1 - \xi_i \\ & \forall i \in 1, \ldots, m : \quad \xi_i \geq 0 \end{aligned} \tag{2.5}$$

where κ is a constant.

Support Vector Machines can also be employed for problems with truly non-linear decision boundaries by relying on the so-called *kernel trick*, which involves mapping vectors that are not linearly separable to a higher-dimensional space where this becomes the case in such a way that the computation of the formulae remains efficient. As explained earlier, however, non-linear separation is not deemed necessary for conventional representations of text documents.

SVMs work particularly well for text classification because the runtime is linear with respect to the number of attributes and because the method is able to cope rather well with irrelevant attributes. Support Vector Machines were used for text classification by Joachims (1998) and it is now generally accepted that they outperform Neural Networks when conventional text representation methods are used, such as the one presented in the following section (Yang and Liu, 1999; Sebastiani, 2002). The use of the maximum-margin hyperplane implies that SVMs have the property of being fairly resistant to the problem of overfitting, i.e. arriving at models that fail to generalize to new examples (Sebastiani, 2002).

2.1.3 Adaptive Boosting

Adaptive Boosting (AdaBoost Freund and Schapire, 1997) is a meta-learning algorithm that uses other learners as its basis. Often, the base classifiers are simple decision

stump functions that merely determine whether a single feature value surpasses some fixed threshold. The power of AdaBoost derives from its ability to find several low accuracy base classifiers that are then suitably combined to form a single high-accuracy classifier.

The classifying committee, in this case, comprises base classifiers that are all learnt by the same base learner, albeit with different weightings of the training instances. For every training instance labelling there is an associated *importance weight* value that indicates to the base learner how important correctly making the respective classification is.

In each iteration, the base learner generates a new classifier based on the weighted training examples paying respect to the fact that misclassifying training instances is considered more severe for labellings with a high importance weight than for those that are associated with a lower one. The weighted error that is then obtained for this classifier on the training examples later determines to what extent the classifier will contribute to the final meta-classifier. In the first round, all importance weights are equal. Subsequently, however, whenever a base classifier fails to guess the correct label of a training instance, the respective labelling will be weighted more in the next round, whereas all other labellings have their weight decreased. Hence, the misclassified instances will presumably be handled more appropriately by the next classifier to be learnt, as it is expected to avoid the errors made by its predecessor.

In the end, when all base classifiers have been learnt, a meta-classifier is constructed by linearly combining them using coefficients computed from the respective weighted error rates.

Adaptive Boosting has exhibited very high levels of effectiveness in text classification tasks (Sebastiani, 2002). More information about this technique can be found in Freund and Schapire (1999).

2.1.4 Multi-Label and Multi-Class Classification

Algorithms such as the Perceptron and Support Vector Machine normally do not support multi-label and multi-class problems directly. In such cases, rather than resorting to algorithms without such restrictions, one may use *class binarization* or *decomposition* methods that reduce the problem to multiple binary problems that can be addressed separately.

A popular approach to decomposition is the *one-against-all* strategy where each class $C \in \mathcal{C}$ is examined using a separate binary classifier that has been trained to discriminate objects in the respective class C from all other objects. The predictions made by these binary discriminators can trivially be combined to determine which classes a document is associated with in the multi-label case. For single-label problems, the classifiers should be able to provide a confidence rating of their prediction so that the class with highest score can be chosen as the one to associate the document with.

2.2 Conventional Monolingual Text Classification

In the previous sections we introduced classification using learning algorithms for arbitrary objects that are described by feature vectors with real-valued feature values. In the case of text classification, the objects to be classified are text documents, and although some related tasks such as authorship attribution and classification by genre exist, the classes correspond to categories based on topics that the documents may or may not pertain to. It is not obvious how a text document's content can adequately be represented numerically in a Euclidean vector space such that the feature vectors allow for such thematic characterizations. In what follows, we shall present the most popular solution to this problem for monolingual documents, an approach relying on counting how often terms occur in the respective documents. All steps involved in arriving at feature vectors for such documents will be described in detail.

2.2.1 Import and Lexical Analysis

For our purposes, a document is simply a unit of text that might be a journal article, a web page, or even an entire book. We will assume that we are dealing with purely textual data taking the form of a sequence of symbols, uncoupled from any embedded images or advanced formatting information. When creating a feature vector representation for a document, the first required step is thus that the text be imported from the respective document data sources. These might be local files in various formats, e.g. Hypertext Markup Language (HTML), Extensible Markup Language (XML), or OASIS OpenDocument files. They could as well be text objects stored in a relational database. When importing, formatting information codes that might indicate font changes etc. need to be removed. This process might also involve merging supplementary information such as the document title and subtitle to the main text body, whereas other types

of meta-information usually are discarded. In some cases, the actual full text may not be available and a surrogate is used, e.g. an abstract.

Formally, given a set Σ of elementary symbols consisting e.g. of the letters of the alphabet and punctuation characters, a *text* can be modelled as a finite sequence $\mathbf{s} = (s_1, \ldots, s_n)$ of symbols $s_i \in \Sigma$ with some length n such that $\mathbf{s}$ adheres to the rules of a *language*. A *language* is a highly complex, structured system of signs with rules that govern how a set of fundamental symbols are combined in a linear fashion in order to arrive at higher-level units such as morphemes, words, sentences, and texts.

The next elementary task is thus the *lexical analysis* step, performed by a so-called *tokenizer*, which involves segmenting a text into so-called *tokens*, while ignoring the whitespace and punctuation between them. A token is defined as a short finite sequence of symbols corresponding to a word or a similar unit (numbers, formulae, etc.). Despite the fact that the segmentation is mainly derived from occurrences of whitespace, tokens may contain whitespace characters, as in the case of the term '*European Union*'. One might also need to deal with issues such as removing hyphenation and line breaks, recognizing abbreviations, and of course dealing with the general problem that there usually is no clear definition of what a word exactly is, so one might need to simply establish certain conventions based on the requirements of the task (for more information about these issues, see Grefenstette and Tapanainen, 1994). Formally, the tokenization step transforms the finite sequence of symbols $\mathbf{s} = (s_1, \ldots, s_n)$ into a finite sequence of tokens.

2.2.2 Term Processing

The terms in the finite sequence of tokens are then processed in several operations. The first operation normalizes words, e.g. by ensuring that all letters are converted to a lower-case form. For text classification purposes, it normally does not make much sense to distinguish different capitalizations of a word (e.g. '*LASER*', '*Laser*', and '*laser*')

In the next step, the sequence of terms is filtered such that certain words considered topic-neutral due to their frequent occurrence in the language are removed. These include function words such as articles, prepositions, and conjunctions that regularly can be found in all kinds of texts and thus are not valuable for the process of learning classifications because their presence does not speak for any particular class. This process, first employed in information retrieval systems, is called *stop word removal* and is performed

using predetermined lists of words (Luhn, 1958), containing among others the words '*although*', '*enough*', '*get*', '*not*', '*probably*', and '*they*', for example.

Since the goal is to establish similarities between documents, it makes a lot of sense to abstract from specific word forms rather than regard forms such as '*child*', '*children*', '*childishly*' as completely distinct. It is thus very common to perform a step called *stemming* where the different possible derivative forms a word may take are conflated to one single word stem or similar normalized form. The normalized forms need not correspond to morphologically correct words of the language, e.g. English '*chastity*' could be reduced to the string '*chast*' and French '*éclaircirent*' could be shortened to '*éclairc*'. Most stemming algorithms used in text classification and information retrieval, including the one devised by Porter (1980), which is the most commonly used one, are limited to removing suffixes using general pattern-based rules. *Lemmatization*, explained in Section 2.3, is a related, more linguistically motivated process, which, however, is rarely used in text classification.

2.2.3 Term Weighting

In order to represent the resulting sequences of terms as feature vectors as required by the learning algorithms, one generally adopts the vector space model originally developed for information retrieval purposes (Baeza-Yates and Ribeiro-Neto, 1999, p. 27ff), which associates each distinct term with a separate vector dimension. A vector space is constructed where each dimension corresponds to a feature associated with a particular term such that all terms received as output after stemming from any of the training documents are covered. For every document, one can then construct a feature vector consisting of feature values that are based on the number of occurrences of the respective term in the document (after preprocessing) and are intended to represent the importance or *weight* of that term in the document with respect to our particular goal of classifying the document using a given classification scheme.

In the course of such a transition from finite sequences of terms to feature vectors, one thus loses all information about where particular words occurred within the document, which is why this representational model is often called the *bag-of-words model*. Put differently, however, this process results in an abstraction from details that highly simplifies the learning process, yet works very well in practice. The most commonly used *term weighting* schemes are:

- **Term Counts (TC):** Term Count values simply indicate the absolute number of occurrences of a term.
  Definition 2.2.1. If $\mathbf{t} = \{t_1, \ldots, t_k\}$ is the processed token sequence for document $D \in \mathcal{D}$, then
  $$tc(\mathbf{t}, t) := \sum_{i=1}^{k} I_t(t_i) \tag{2.6}$$
  for a term t and an indicator function I_t that evaluates to 1 if its argument equals t and to 0 otherwise.

  The underlying assumption is that the higher the number of occurrences, the more important the term is for our purposes.

- **Term Frequency (TF):** TF is similar to TC except that each feature vector is normalized such that the feature values characterize the relative importance of terms within the document. Although variations exist, the most common choice is to normalize with respect to the sum of all Term Counts values:
  Definition 2.2.2. The term frequency (TF) is defined as
  $$tf(\mathbf{t}, t) := \begin{cases} \dfrac{tc(\mathbf{t}, t)}{\sum_{t'} tc(\mathbf{t}, t')} & tc(\mathbf{t}, t) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$
  where t' iterates over the set of all distinct terms occurring in $\mathbf{t}$.

- **TF-IDF**: Considering that terms that occur frequently in a wide range of documents bear less discriminatory information, it is common practice to give them less weight by multiplying all TF values with the so-called inverse document frequency (IDF) introduced by Spärck Jones (1972).
  Definition 2.2.3. The inverse document frequency (IDF) is defined as
  $$idf(t) := \begin{cases} \log \dfrac{1}{df(t)} & df(t) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$
  where $df(t)$, the document frequency, is defined as
  $$df(t) := \frac{|\{\mathbf{t}_i \in D_{\text{tr}} \mid tc(\mathbf{t}_i, t) > 0\}|}{m} \tag{2.9}$$
  and provides information about which fraction of all $m = |D_{\text{tr}}|$ training documents ($m > 0$) with processed token sequences $\mathbf{t}_i \in D_{\text{tr}}$ contains a particular term (D_{tr} is taken to be the set of processed token sequences resulting from the set of training documents $\mathcal{D}_{\text{tr}}$).

Terms that occur in only few documents thus have a higher IDF value than terms occurring in many documents. This finally leads to the following formula based on seminal work in information retrieval by Salton and Buckley (1988):

Definition 2.2.4. TF-IDF term weights are defined as

$$tfidf(\mathbf{t}, t) := tf(\mathbf{t}_i, t) \cdot \log \frac{1}{df(t)} \tag{2.10}$$

for a term t, a term sequence $\mathbf{t}$ for some document (that need not be from the training set), and term sequences $\mathbf{t}_i \in D_{\text{tr}}$ corresponding to the training documents.

It should be pointed out that some systems use slightly deviating formulae (e.g. Singhal et al., 1996).

The term weights are numeric values that can then be used as feature values in a feature space constructed by assigning every processed term occurring in some term sequence $\mathbf{t}$ a separate feature dimension.

2.2.4 Feature Selection and Normalization

Obviously, using term weights for all terms as separate feature values results in a tremendously high-dimensional feature space. Fortunately, most of the components of any given vectors will have the value 0, and such *sparse* vectors can be stored and processed rather efficiently.

Nevertheless, many learning algorithms benefit from only receiving feature vectors with a limited number of highly relevant features rather than all available information, as this not only has a desirable effect on runtime performance but also often induces an increased accuracy. In order to reduce the dimensionality of the feature space, a commonly used approach is to employ so-called *feature selection* algorithms. Given a set of training instances, these algorithms determine which vector dimensions may be considered extraneous and therefore are not taken into account when learning. One can then construct a new feature vector space that is lower-dimensional than the original feature space and map the training and test instances accordingly. A simple yet effective heuristic is to use the document frequency as described above as an indication of how important a feature is, as many terms typically just occur in one or two documents (cf. Sebastiani, 2002). Such terms with an extremely low document frequency are unlikely to contribute substantially to the constructed classifier because no general decision rules can be inferred without additional knowledge. A more sophisticated measure for evaluating the usefulness of particular features is the Information Gain (IG) measure, defined

as the number of bits of information gained for class prediction by knowing whether the event F of some specific feature being nonzero is given (YANG and PEDERSEN, 1997).

$$IG(F) = -(\sum_{C \in \mathcal{C}} P(C) \log P(C)) \tag{2.11}$$
$$+ P(F)(\sum_{C \in \mathcal{C}} P(C|F) \log P(C|F)) + P(\overline{F})(\sum_{C \in \mathcal{C}} P(C|\overline{F}) \log P(C|\overline{F}))$$

Here, $P(C)$ is the probability of a document being in a class $C \in classes$, $P(F)$ and $P(\overline{F})$ are the probabilities of the feature value being nonzero or zero, respectively, and $P(C|F)$ and $P(C|\overline{F})$ are the corresponding conditional probabilities for C. These probabilities are estimated based on the training set and then only the l highest-ranking features are maintained (for some predetermined value l), or alternatively some threshold Information Gain is specified.

Finally, one can optionally normalize the vectors to have a Euclidean norm of 1.0, facilitating the learning algorithm's task of comparing different vectors.

Definition 2.2.5. The L_2 norm of a vector $\mathbf{d} = (d_1, \ldots, d_n)$ is

$$norm(\mathbf{d}) := \frac{1}{||\mathbf{d}||}\mathbf{d} = \frac{1}{\sqrt{d_1^2 + \ldots + d_n^2}}\mathbf{d} \tag{2.12}$$

The resulting feature vectors can then be used for learning and deploying classifiers that are able to distinguish text document categories.

2.3 Other Techniques and Resources for Natural Language Processing

In the following chapters, alternatives to the conventional approach to text classification described so far will be introduced. First, however, this section will provide a brief overview of some additional techniques and resources from natural language processing that will be used for this purpose.

2.3.1 Morphological Analysis

Morphological analysis is a process that is used to obtain information about the lexical category of a term as well as its lemma. The words of a language can be associated

with different lexical categories (traditionally called *parts of speech*). In general, for instance, *nouns* denote various types of things (including abstract entities, classes of entities, etc.), *verbs* denote events, *adjectives* modify nouns, and *adverbs* modify words from other lexical categories. A morphological analyser uses rules or statistical methods in order to determine the most appropriate category for a term. At the same time, most analysers are able to perform a process called *lemmatization*. Words such as '*give*', '*gives*', '*gave*' all belong to the same *lexeme*. The canonical form used to refer to such a lexeme, in our example the form '*give*', is called the *lemma* or *citation form*, and such lemmata correspond to the headwords of dictionaries. For verbs this is by convention often the infinitive form, though sometimes other conventions may be used, as in the case of Arabic, where the masculine third-person singular past tense form of verbs is chosen. Lemmatization will serve as an alternative to conventional stemming in our approach, because word stems often are not morphologically correct words. It should be pointed out that morphological analysis may require disambiguating between multiple possibilities with respect to the local context, e.g. the prefixes and suffixes of '*unionizable*' could be analysed as either '*un-ion-izable*' or '*union-izable*' (Russell and Norvig, 1995, p. 703).

2.3.2 Word Sense Disambiguation

Word sense disambiguation (WSD) algorithms attempt to choose the most appropriate or probable from a set of potential meanings that a term might have in particular context. Disambiguating word senses allows for more accuracy in text classification. The word '*case*', for instance, can be used to refer to a container for objects, an instance or example, a legal case, the distinction between majuscule or minuscule forms of letters, a grammatical case (e.g. genitive case), or a showcase. Nevertheless, in the phrase '*The judge dismissed the case*', it is fairly clear to a human reader that what is being referred to is a legal case. It is not, however, evident how such inferences could be made algorithmically. Most existing approaches look at co-occurrences of words, exploiting the fact that the words '*judge*' and '*dismissed*' are more likely to occur when the phrase is referring to a legal case. One approach is to regard the terms surrounding a word in the text document as the word's local context string, and then attempt to construct a similar context string for the candidate senses. Theobald et al. (2003) make use of the fact that the lexical resource WordNet (see Section 2.3.4) provides English-language descriptions for each sense that it lists. They create context strings for the candidate senses by concatenating the respective descriptions with additional descriptions of closely

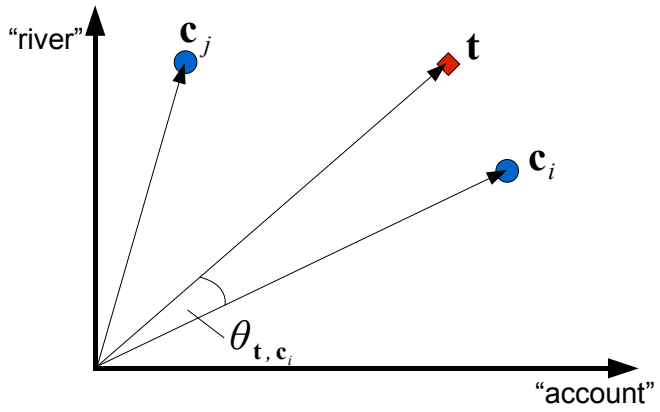

Figure 2.5: Cosine similarity measure: $\mathbf{c}_i$ is closer to $\mathbf{t}$ than $\mathbf{c}_j$ because the term '*internet*' occurs more often in the respective context. This proximity is characterized by $\theta_{\mathbf{t},\mathbf{c}_i}$. Note that in reality, word stems are used rather than full words.

related senses. For all context strings, TF-IDF feature vectors as described in Section 2.2 are then constructed. Given two feature vectors $\mathbf{t}$ and $\mathbf{c}_i$ representing the contexts for a human language term and a candidate sense, respectively, one can then determine their similarity using the so-called cosine measure which assumes that the angle $\theta_{\mathbf{t},\mathbf{c}_i}$ between the two vectors characterizes their similarity (Figure 2.5), for vectors that point in similar directions are based on context strings containing similar terms. The angle can be computed using the inner product:

$$cossim(\mathbf{t}, \mathbf{c}_i) := \cos\ \theta_{\mathbf{t},\mathbf{c}_i} = \frac{\langle \mathbf{t}, \mathbf{c}_i \rangle}{||\mathbf{t}|| \cdot ||\mathbf{c}_i||} \tag{2.13}$$

Instead of directly using $\theta_{\mathbf{t},\mathbf{c}_i}$, one uses the cosine because it is easier to compute, and, since all vector components are nonnegative, we obtain $-\frac{\pi}{2} \leq \theta_{\mathbf{t},\mathbf{c}_i} \leq \frac{\pi}{2}$ and thus receive cosine values in the range $[0, 1]$. The candidate sense with the highest corresponding cosine value is taken to be the most likely sense of the word.

2.3.3 Machine Translation

Machine translation (MT) is the automatic translation of texts from one language to another language. Proper machine translation normally involves some kind of parsing of the original source text, a dictionary lookup step where words and more complex expressions in the source language are translated to the destination language or to some intermediate form, and the final output according to the grammar of the destination

language. The translation process thus is a very complex one, and the current state of the art technology still is highly inferior to human translation work. Some of the challenges that make machine translation particularly difficult are problems such as ambiguity (e.g. synonyms), the complexity of the grammatical structures in the source language, and the fact that different languages structure the world differently making one-to-one translations of terms difficult. Consider for example that an English text might not contain enough inherent information in order to decide whether an English '*you*' should be translated as French '*vous*' or rather as the informal form '*tu*' (Russell and Norvig, 1995, p. 693). This example shows that what would be required in order to attain the level of quality reached by human translators would indeed be an understanding of both cultures involved as well as additional intelligence. However, despite the imperfections of current translation software, such systems do have useful applications, for example for translations in very limited domains such as weather reports, or when someone desires to quickly arrive at a basic understanding of the meaning of a text without waiting for costly professional human translations to be performed. In Section 3.2 we will describe approaches that make use of translations for solving multilingual text classification problems.

2.3.4 Thesauri and Ontologies

Later on we will attempt to use multilingual lexical resources and so-called ontologies in order to establish a common form of representation for documents written in different languages.

A *thesaurus*, according to the ANSI/NISO standard Z39.19, is 'a controlled vocabulary arranged in a known order and structured so that the various relationships among terms are displayed clearly and identified by standardized relationship indicators' (ANSI/NIZO, 2005). Synonyms are grouped together, whereas homonyms are distinguished. *Roget's Thesaurus*, first published in the 19th century, is the most famous such resource for the English language. Table 2.1 provides an example of what a thesaurus might list under the headword '*scholarly*', though a number of other relationship types are possible, some of which are presented in Table 2.2.

An *ontology*, from ancient Greek '-λογία' (science) and 'ὄντος' (of being), is a theory of what possesses *being* in the world or in a limited domain. In knowledge representation, such theories are heavily formalized in a formal language such as the *Knowledge Interchange Format* (KIF) or *Web Ontology Language* (OWL) and their objective

Table 2.1: Example thesaurus entry (source: KIPFER (2006))

Entry:	*scholarly*
Synonyms:	*bookish*, *cultured*, *educated*, *erudite*, *intellectual*, *learned*, *lettered*, *literate*, *long-hair*, *scholastic*, *schooled*, *studious*, *taught*, *trained*, *well-read*
Antonyms:	*illiterate*, *uneducated*

Table 2.2: Relationships captured by thesauri

relation type	**inverse relation type**	**description**
synonymy	synonymy	nearly identical meaning
similarity	similarity	similar meaning
antonymy	antonymy	opposite meanings
hypernymy	hyponymy	more general term / less general term ('*a kind of*')
holonymy	meronymy	whole/part-relations (further distinction possible: part, location, member, '*made of*', or portion holonymy and meronymy)
causes	is caused by	Physical causality
involves agent	has role of agent	e.g. '*hunting*' involves agent '*hunter*'

is to support the sharing and reuse of knowledge by different applications. GRUBER (1993) defines an ontology as an 'explicit specification of a conceptualization' and refers to GENESERETH and NILSSON (1987) who define a conceptualization as 'the objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold among them'. There is a considerable amount of dissent, however, on what exactly constitutes an ontology. For example, according to many definitions, it suffices for ontologies to simply describe relations holding between entities rather than axiomatically defining the entities, so lexical resources such as thesauri, too, are sometimes construed as implying weakly formalized ontologies. In Section 5.2 we thus provide our own definition that corresponds very well to the specific needs of our approach to multilingual text classification.

Princeton WordNet is a lexical database for the English language that organizes terms into so-called synonym sets or *synsets*, which in turn are linked by different relations such as hypernymy, hyponymy, and meronymy (FELLBAUM, 1998), as shown in Figure 2.6. While WordNet is frequently regarded as a thesaurus, it is also often perceived as defining an ontology. The original WordNet later inspired the creation of similar resources for other languages. Some of these are strictly aligned to the original, while others require the use of mappings in order to establish cross-lingual correspondences, e.g. in the case of the EuroWordNet project an interlingual index was created for this purpose.

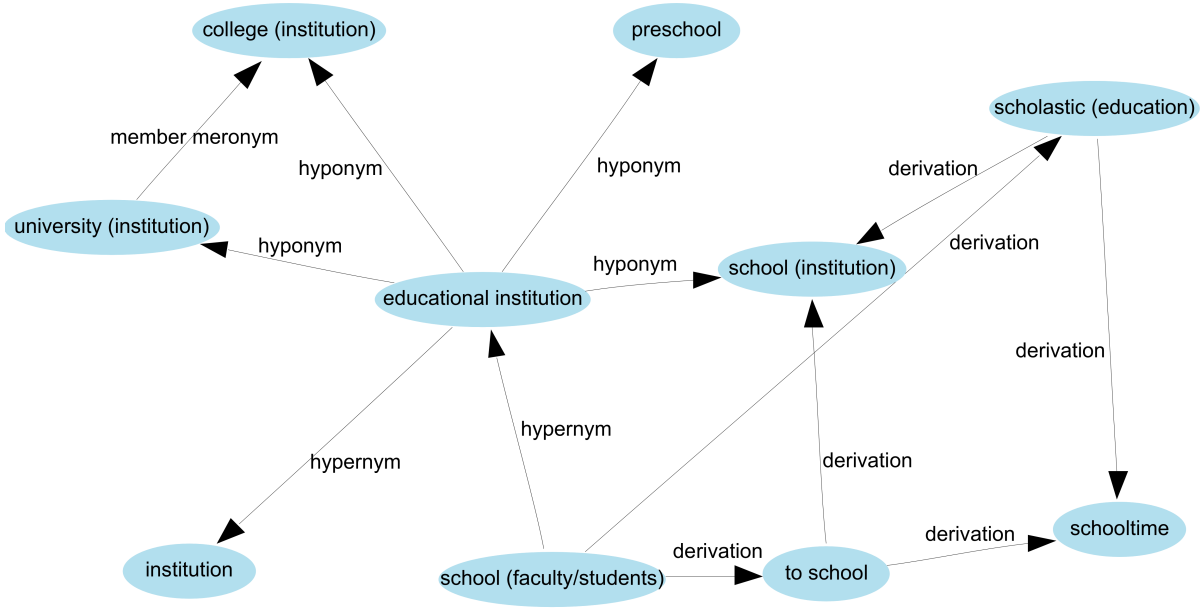

Figure 2.6: Sample of information stored in Princeton WordNet

Chapter 3

Related Work

While Chapter 2 described the standard way of performing monolingual text classification, relying on the TF-IDF bag-of-words model, we will now provide an overview of certain alternative approaches that are related to the path pursued in our work.

3.1 Semantic Text Representation Methods

The conventional monolingual bag-of-words-based approach presented in Chapter 2 fails to properly uncover the meaning of terms, so synonyms such as '*car*' and '*automobile*' are treated as distinct. With varying levels of success, several authors thus have sought to use ontologies and thesauri to develop text representation models that purport to be semantic rather than remaining at the surface level, for instance de Buenaga Rodríguez et al. (1997), Moschitti and Basili (2004), Rosso et al. (2004), and Ifrim et al. (2005). Most authors retain the bag-of-words-based features, choosing to augment them with additional features rather than replacing them. These additional features are constructed by mapping the original terms occurring in the document to ontology concepts or to thesaurus entries. The additional features are hence based on the meanings of terms rather than on the terms themselves, so '*car*' and '*automobile*' will normally be represented by one single feature rather than by two distinct ones.

While, in theory, such methods should be able to outperform term-based approaches, these endeavours often produced rather discouraging results. A lack of precise word sense disambiguation methods is frequently cited as the main inhibiting factor for an increased accuracy. Our results confirmed that word sense disambiguation is important for distinguishing homographs, i.e. words sharing the same spelling yet differing in meaning, however, we will also attempt to show that overly fine-grained discrimination between

word senses is not desirable when attempting to represent the general topic of a document. In Section 4.3.2 an extensive list of arguments against a simple concept mapping approach will be presented. A few authors have experimented with the use of hypernyms for increased efficiency (e.g. Scott and Matwin, 1998; Bloehdorn and Hotho, 2004). Ramakrishnanan and Bhattacharyya (2003) additionally attempted to use certain other direct neighbours, following e.g. pertainym relationships, which link adjectives such as '*educational*' to words they pertain to ('*education*' in this case), however, even in monolingual text classification there apparently has not been any research on using an activation spread model as employed by our Ontology Region Mapping algorithm.

3.2 Multilingual Solutions

In our work, we attempt to devise a text representation scheme that is not only semantic but also works with documents provided in different languages. It should be noted that, although language identification can be a part of multilingual text classification, the main goal is not to classify by language but rather to classify documents in different languages by topic or similar criteria.

There has been research on reaching this goal in the case of enough training documents being available for every language, as discussed in Section 4.2.2. In such cases, Bel et al. (2003) use one single classifier for all languages. Their results prove that whether this can be done without a very negative impact on the level of accuracy depends on how the learning algorithm operates, e.g. the Rocchio algorithm, which attempts to form centroids, does not handle this situation very well. In the cross-lingual case, they propose translating selected terms from the documents. García Adeva et al. (2005) test additional setups for dealing with this scenario of enough training documents being available in each language, and their results suggest that using separate classifiers for each language leads to a higher accuracy than using a single classifier for all languages. Such scenarios do not fall under what we will later define as genuinely multilingual text classification problems, for they can be resolved using separate monolingual solutions.

In our work, we focus on more interesting cases when this condition of enough training documents being provided for each language is not given. For such cases, a simple ad hoc solution is to translate documents such that the problem is reduced to a monolingual one (Jalam, 2003; Olsson et al., 2005), and then e.g. simply use conventional bag-of-words TF-IDF feature vectors as described earlier in Chapter 2. Rigutini et al. (2005) present a similar approach that additionally applies the EM algorithm in order

to exploit unlabelled examples already provided in the destination language in addition to the translated training documents. Relying on machine translation is also the dominant approach in cross-lingual information retrieval (CLIR) (see Oard and Dorr, 1996), although in this case mostly only user queries are translated rather than the documents to be indexed. CLIR has a long history going back to at least Salton (1969) and has recently developed into a rather active area of research with workshops such as the Cross-Language Evaluation Forum (CLEF, see Peters et al., 2005). Translations in CLIR have been performed with professional machine translation tools, dictionaries, parallel corpora, and comparable corpora, and most research focuses on optimizing these translations, for instance by developing sophisticated word sense disambiguation algorithms and exploiting the corpora better. A similar approach has also been applied to multilingual text clustering, where Mathieu et al. (2004) use a bilingual dictionary to establish a cosine similarity-like distance measure. Our work, however, shows that simple translations alone lead to suboptimal results when performing multilingual text classification.

An alternative approach related to the path pursued in our work is to use multilingual thesauri. Loukachevitch (1997) uses a semi-automatically created bilingual Russian-English thesaurus. She does not use a machine learning approach but rather requires rules to be specified manually for the classes involved. Her method involves looking for occurrences of terms from the thesaurus in the document, leading to a small number of topics ('*thematic nodes*') that are considered relevant, and then identifying which of these topics are the most important, assuming that co-occurrence with other important topics in the document is what characterizes them. If these most important topics match the demands of the manually specified rules, the document is assigned to the respective class.

Lauser and Hotho (2003) use the AGROVOC thesaurus, a controlled vocabulary for the agricultural domain, for multilingual subject indexing, meaning that their objective is to label documents with keywords from the vocabulary rather than supporting arbitrary text classification tasks. Steinberger and Pouliquen (2003) use the EUROVOC thesaurus for this purpose.

In CLIR, some systems use a controlled vocabulary for indexing documents, so only a very limited number of terms can be searched for (Soergel, 1997). There has been some research on using larger thesauri to create concept-based document and query representations, e.g. Verdejo et al. (2000) use EuroWordNet. Many implementations, however, just use thesauri for creating pseudo-translations. None of these approaches

use a region weighting model like the one described in Chapter 5.

There has been further related work on multilingual solutions based on latent semantic analysis (LSA). Gliozzo and Strapparava (2005) adapt LSA for multilingual text classification by exploiting comparable corpora and requiring that the languages share certain terms (which is not necessarily the case for language pairs such as Chinese and French), and Dumais et al. (1997) use latent semantic indexing for CLIR where the initial training is based on documents present in or translated into multiple languages. Latent semantic analysis does not use formal background knowledge like our approach but rather identifies concepts implicitly present in a set of documents, computed statistically by identifying terms with similar occurrence patterns.

Chapter 4

Analysis of Problem and of Existing Approaches

The central goal in this chapter will be a thorough analysis of multilingual text classification and of strategies to tackle such problems. First, a clearer definition of multilingual text classification will be presented. This will be followed by a study of the different kinds of scenarios one might encounter. Subsequently, some possible approaches will be judged mostly in terms of their appropriateness from a linguistic perspective.

4.1 Multilingual Problems

Conventional text classification methods as described in Chapter 2 are based on the tacit assumption that the text classification problem is monolingual. Our goal is to devise methods for dealing with multilingual text classification problems while maintaining a high level of effectiveness. Bearing in mind how text classification problems were defined in Section 2.1, a formal distinction between monolingual and multilingual text classification problems ought to be established.

Unfortunately, our conventional notion of languages is rather vague, and the distinction between a language and a dialect, for instance, is often purely based on historical and political circumstances. We thus use the more precise notion of a *language form*, which we define as a maximal set of variants of written languages (or languages that are symbolic in some other way), such that all variants are sufficiently similar in phenotype to be processed by the same methods. In many cases, such language forms correspond directly to our conventional understanding of languages. However, we may observe the following.

1. In certain cases, two arguably distinct languages correspond to the same language form. For instance, the Romanian and Moldovan languages are virtually identical in their standard written variants. Similarly, the standard form of Simplified Written Chinese should be treated as a single language form, although it does not correspond to one particular spoken language, considering that several different mutually unintelligible languages or dialects exist.

2. In other cases, two variants of what is considered one single language may correspond to distinct language forms. For instance, Norwegian can be written using either Bokmål or Nynorsk, and thus the two language forms are regarded as distinct. Similar distinctions are made when languages have undergone significant reforms, as in the case of modern Turkish.

In what follows, we are going to assume that each document is provided in a single language form. When a document is available in multiple languages, each version may be treated as a separate document. Documents where multiple language forms are mixed within the text can be accommodated by regarding them as being provided in a pseudo language form L_{mixed}, and then processing each part of the text language-specifically.

At this point, the desired definition of multilingual text classification problems can finally be made as follows:

Definition 4.1.1. A text classification problem $\mathcal{T}$ is a **monolingual** text classification problem if and only if all documents $D \in \mathcal{D}$ are expected to be given in a single language form L ($L \neq L_{\text{mixed}}$). A text classification problem $\mathcal{T}$ is a **multilingual** text classification problem if and only if $\mathcal{T}$ is not a monolingual text classification problem.

4.2 Problem Analysis and General Framework

We will now propose a framework for analysing a given multilingual text classification problem $\mathcal{T}$, for, as it turns out, not all such problems demand similar solutions. The first step involved is an identification of the language forms that need to be supported. We assume that a machine learning approach is going to be used and that the following information can be determined, based on the specific requirements of the task:

1. the set of class labels $\mathcal{C}$ that need to be supported

2. the learning algorithm that is going to be used as well as its requirements (see Chapter 2 for algorithms that have proven to work particularly well in text classification)

3. $\mathcal{L}_{\text{tr}}$: the set of language forms that will need to be supported when parsing training documents

4. for every language form $L \in \mathcal{L}_{\text{tr}}$ the kinds of training documents that are going to be available to the system in terms of the classes $\mathcal{C}$ they are expected to belong to, including knowledge e.g. about whether all classes are going to be covered by them

5. $\mathcal{L}_{\text{tst}}$: the set of language forms that are to be supported when classifying documents in a test or operational setting

Note that in this chapter, when referring to *test documents*, we mean any documents that might need to be assigned class labels rather than just a limited test set used for evaluating the performance of a system. Similarly, the term *training documents* shall henceforth refer to any documents used for training, whether they are pre-classified or not, for the strategies presented here can also be used with semi-supervised learning methods that attempt to benefit from additional unlabelled examples (Zhu, 2005).

Our analysis will continue with an evaluation of whether *simple language transformations* can be applied. After that, in certain cases, our problem $\mathcal{T}$ might turn out to be rather trivial, allowing for a reduction to simpler monolingual problems.

4.2.1 Simple Language Transformations

We observed earlier in Section 4.1 that language forms can be highly related yet distinct. For instance, one might expect test documents written in Serbian using the Cyrillic alphabet yet only have training documents available that are written in Croatian using the Latin script. In order to proceed with our analysis of the problem, we need to consider whether certain documents can be converted from a language form L_1 to another form L_2 by means of some simple transformations, as this might affect the scenario we are dealing with.

Such transformations need not be completely lossless, and may involve the following:

- A transliteration (or transcription) may be applied when multiple scripts exist for a single language or for extremely closely related languages, e.g. Serbian, Croatian,

and Bosnian can be transformed to use the same script. Similarly, Simplified Chinese Characters can be converted to Traditional Chinese Characters and vice versa (although the mapping may not always be trivial).

- Orthographic changes: Transformation rules can be applied when multiple language forms are very closely related, e.g. when they merely differ in terms of different spelling systems being used, or when they correspond to very closely related dialects, sociolects, or other varieties of the same language.

Note that in the case of mixed-language documents, each part needs to be processed on a language-specific basis.

4.2.2 Monolingual Reduction

We then proceed with our analysis by determining whether a sufficient number of training documents are going to be available for every language form in $\mathcal{L}_{\text{tst}}$, i.e. for every language we expect test documents to be provided in. In such cases, a multilingual text classification problem can in fact be reduced to one or more monolingual problems. While the exact requirements will depend on the classification task and on the particular learning algorithm used, in general, many algorithms require for each language and class (or class pair) a certain amount of positive and negative examples.

If it turns out that enough training documents are going to be available in every relevant language form, one can simply reduce the problem to a monolingual one. The following solutions are conceivable:

- Whenever enough training documents are available for every language form, a multilingual text classification problem $\mathcal{T}$ with $|\mathcal{L}_{\text{tst}}|$ different possible test document languages can be reduced to $|\mathcal{L}_{\text{tst}}|$ separate monolingual text classification problems $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{|\mathcal{L}_{\text{tst}}|}$ by training a separate classifier $\widehat{\phi}_i$ for each such language form. One then creates a meta-classifier $\widehat{\phi}$ which dispatches documents to the appropriate base classifier $\widehat{\phi}_i$ based on the language of the document, e.g. sending French documents to a French classifier and Portuguese documents to a Portuguese classifier. This reduction corresponds to the polylingual training approach suggested by Bel et al. (2003).

- Alternatively, if the learning algorithm chosen is able to handle classes that are very heterogeneous, one does not need to proceed very differently from conventional text classification as described earlier in Chapter 2. Every training document with

a language in $\mathcal{L}_{\text{tst}}$ is processed using language-specific tokenization, stemming, etc. (see Section 6.3), and then a single classifier $\widehat{\phi}$ is trained using all those documents (cf. the NP1C approach by García Adeva et al., 2005). Later on, one can then use $\widehat{\phi}$ to classify any test document that has been processed using such language-specific techniques because enough training documents were submitted to the learner for each language form.
Bearing in mind that each class can contain documents written in different languages, the learning algorithm used in such a case must be able to cope with heterogeneous classes comprising training examples that do not share very much in common. With some such algorithms one will have to ensure for every single class that the numbers of training documents per language are well-balanced. The K-nearest neighbour algorithm (see Mitchell, 1997, p.231ff), which we would expect to work rather well with such heterogeneous classes, has no such requirement.

- Another possibility arises when *every* training document and *every* test document is available in *all* language forms $L \in \mathcal{L}$ (where $\mathcal{L}$ is a set of language forms containing more than one form). In such a case, one optionally can additionally use techniques from multi-view learning (please refer to Muslea, 2002) to increase the accuracy of the results. Separate classifiers $\widehat{\phi}_i$ for each language form are created as explained above, and then a meta-classifier is constructed that evaluates each document using multiple (perhaps all) base classifiers $\widehat{\phi}_i$. This meta-classifier then determines the final result by combining the single results in an appropriate manner, perhaps using some voting system.

In each of these cases, one will have to rely on certain multilingual natural language processing techniques as explained later on in Section 6.3, in particular in order to identify the language of the document when necessary.

4.2.3 Genuinely Multilingual Problems

If the conditions mentioned above fail to be met in spite of the simple language transformation processing, then our text classification problem $\mathcal{T}$ is what one might call a *genuinely multilingual* one in the sense that the classifier will need to detect similarities between documents despite them being in different language forms.

Definition 4.2.1. A text classification problem $\mathcal{T}$ is a **genuinely multilingual** text classification problem if and only if it cannot be reduced to one or more monolingual

text classification problems using the simple language transformation and monolingual reduction steps.

There is an interesting subclass of genuinely multilingual problems where this challenge becomes particularly clear.

Definition 4.2.2. A text classification problem $\mathcal{T}$ is a **cross-lingual** text classification problem if and only $\mathcal{L}_{\text{tr}} \cap \mathcal{L}_{\text{tst}} = \emptyset$, i.e. test documents are always expected to be provided in language forms distinct from the language forms used for training.

In the rest of this chapter, we are going to take a closer look at several existing solutions for tackling such genuinely multilingual problems.

4.3 Approaches to Multilingual Text Classification

We have seen that only certain multilingual text classification problems can be reduced to one or more monolingual problems. Other problems, so-called genuinely multilingual problems, require special solutions. The main challenge is finding a way of representing texts such that positive examples can be distinguished from negative ones, although the texts are provided in different languages. At the surface level, an Italian article from the field of molecular biology shares a lot more in common with an Italian text about renaissance art than with a Norwegian or Korean text about molecular biology. Ideally, however, the representation chosen should lead to data allowing a linear separation of positive and negative examples despite different languages being involved.

In what follows, several approaches will be studied from a linguistic perspective. We will begin by determining whether it makes sense to simply use the bag-of-words model as in conventional monolingual text classification, however with words from different languages.

4.3.1 Multilingual Bag-Of-Words Approach

Even when a proper monolingual reduction is not possible, we might be able to use conventional text classification methods, as presented in Chapter 2, to classify texts in multiple languages. The general idea is to use the conventional bag-of-words model with words from different languages, hoping that enough of these words are shared between those languages. Terms that are shared by different languages could enable

learning algorithms to discriminate classes properly although the documents are provided in different languages, because documents in different languages that are to be assigned the same class label might tend to contain certain characteristic terms which do not occur in documents not belonging to the respective class.

The general procedure is very close to the monolingual case. One first applies preprocessing techniques such as tokenization and stemming, paying care, however, to language-specific issues as described in Section 6.3, and the resulting terms from different languages are then all represented in one single feature space. The learning algorithm, aided by feature selection, will then presumably be able to identify terms relevant for establishing the classification.

4.3.1.1 Shared Terms

The effectiveness of this approach depends mostly on the level of agreement that can be exposed between terms taken from two documents in different languages.

A significant contribution is made by *named entities*, i.e. names of people, places, things, etc. Many named entities are transliterated, transcribed, or even translated in different languages but others remain unchanged. Named entities include:

- names of people (however, Leonardo da Vinci is '*Léonard de Vinci*' in French, '*Gorbachev*' is '*Gorbatjov*' in Swedish, 'Горбачёв' in Russian, and '戈尔巴乔夫' in Chinese)
- names of groups, organizations, and companies (however, the '*UNO*' is called '*ONU*' in many languages)
- many names of locations, rather often for instance '*Berlin*' and '*Madrid*', not that often '*Cologne*' ('*Köln*' in German), '*London*' (e.g. '*Londres*' in French)

Another important contribution is made by loanwords and internationalisms that are shared between languages, for instance words such as '*radio*', '*tennis*', '*information*' and '*tsunami*' occur in many languages.

Of course, this always depends on the languages involved, as shown in Table 4.1. The word '*civilisation*' has the same spelling in French, Swedish, and usually also in British English. However, alternative spellings such as '*civilization*', common in the USA, Canada and also in certain British publications, could negatively affect the performance of text

Table 4.1: International word '*civilization*'

Language	**Term**
English	*civilization* *civilisation*
German	*Zivilisation*
Swedish	*civilisation*
French	*civilisation*
Spanish	*civilización*
Portuguese	*civilização*
Estonian	*Tsivilisatsioon*
Hungarian	*civilizáció*

classification tasks, since two variations of the same word might not be recognized as such.

The examples demonstrate that we can yield further correspondences between language forms by performing stemming, for instance although English '*revolution*' is distinct from Spanish '*revolución*', the process of stemming could conflate both words to the common stem '*revol*'. The same holds for English '*civilization*' and Spanish '*civilización*' in Table 4.1.

Such shared words often have a Latin or Greek background, though perhaps equally many, especially in science and technology, come from English. One positive aspect is that precisely these international words present in many languages sometimes are the most relevant words. For our purposes, we do not need prepositions, pronouns, and other auxiliaries to be similar.

Another advantage that we have is that written language is a lot more conservative than spoken language, leading to cognates having identical spellings, as in the case of '*civilisation*', despite great differences in pronunciation between different languages. A special situation arises with writing systems such as the Chinese one, which historically gave birth to the Japanese kanji, the Korean hanja, and the old Vietnamese Chữ nôm, the latter two not being in common use anymore. Despite certain shifts in meaning, there is a large overlap between these writing systems, which can help when classifying.

Unfortunately, terms with different meanings but similar appearance, often called *false*

friends, might also be matched. For instance, the German word '*Roman*' corresponds to the English '*novel*' rather than referring to the Roman culture. French '*pain*' means bread rather than the sensation of pain. However, we do not expect such cases to exert a significant influence on the effectiveness of the classifier.

4.3.1.2 Discussion

Despite having seen that documents written in different languages may share many important terms, it remains clear that the learnt classifier will only work well if the level of concordance between the involved languages is sufficiently high and this agreement exhibited by them is amenable to identification by means of stemming. In the case of the German and Estonian words for '*civilization*' in Table 4.1, we can observe that stemming does not suffice to identify the two words despite their obvious similarity.

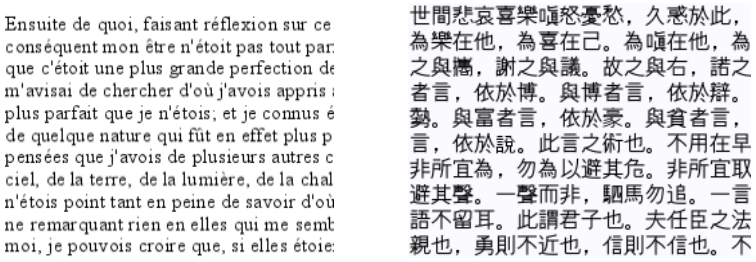

Figure 4.1: Lack of agreement between French and Chinese at the surface level.

Looking at Figure 4.1, we can get an idea of how difficult it would be to use this method in the case of two very different languages such as French and Chinese, for in Chinese normally even all non-Chinese names are transcribed, so '*Victor Hugo*' becomes '维克多·雨果' ('*Wéikèduō Yǔguǒ*') in Mandarin Chinese, and likewise, in western languages such as French and English all Chinese names are romanized using Hànyǔ Pīnyīn or other systems. If we used conventional text classification techniques on French and Chinese documents, there would usually be nearly no overlap between the features used for either language. The feature set would fall into two nearly disjoint subsets for each language, which is highly undesirable.

Another significant problem arises when the training instances for certain languages are not distributed very equally among all classes. For instance, if there are several classes $C_1, \ldots, C_{i-1}$ for which all training examples are provided in Swedish, and another class C_i for which all training documents are provided in Portuguese, then an extreme case of

underfitting for C_i is very likely to occur: Due to the overwhelming amount of agreement, the learning algorithm will likely end up associating all Portuguese test documents with class C_i regardless of what they are about thematically. Not in all cases will one be able to adapt the training set to solve such problems.

Using conventional text classification for multilingual problems thus is only viable in a limited range of cases, depending on the distribution of training documents, the level of correspondence between the languages, the required effectiveness of the system, as well as the type of documents and classes involved, among other things. It is desirable, however, to establish a more universal solution to the task of performing genuinely multilingual text classification.

4.3.2 Conventional Bag-Of-Words Model with Translations

The problem of simply using conventional text classification for multilingual documents is that documents provided in two languages may be related semantically but nevertheless might not exhibit any similarities at the surface level.

One general strategy for multilingual systems is to perform translations such that all documents are present in a single pivot language. Jalam (2003) suggests simply translating all documents and then adopting conventional monolingual text classification methods. Typically, the pivot language would correspond to one of the language forms for which a large number of training documents are present.

This amounts to an extension of the idea of simple language transformations described in Section 4.2.1 in the sense that one not only performs simple syntactic transformations but also more complex transformations that involve actual translation based on the semantics of a text, and then exclusively uses the results instead of the original text. If all required translations succeed, then the entire collection of training and test documents in the document collection $\mathcal{D}$ will be available in a single pivot language form and we can perform a reduction of our problem $\mathcal{T}$ to a single monolingual problem $\mathcal{T}'$ for which conventional monolingual text classification methods may be used.

This approach can be expected to work rather well. However, while grammatically correct translation results are not strictly necessary, producing properly disambiguated translations is a rather complex process, and high-quality machine translation software tends to be very expensive. Furthermore, some of the general drawbacks of the bag-of-words approach become particularly severe when using translations.

1. **lexical variety:** The representation scheme requires documents that belong to the same class to contain the same terms. However, this might not always be the case when using translations. The machine translation process usually involves a built-in translation dictionary that associates a particular sense of an input language word with one specific term in the destination language but not to any of the other relevant synonyms in the destination language. This means that the output documents do not contain the same variety of synonyms as a human-written document would.
   Considering the current availability of machine translation software, the pivot language is very often going to be the English language, which unfortunately makes this problem of synonymy particularly severe, as it has an extraordinarily large vocabulary, many words existing in both a Germanic form and a Romance form (e.g. '*whole*' vs. '*entire*').

   A further aggravation might occur when multiple source languages need to be translated. Translation method A might always map a certain word sense in language form L_1 to word w_1 in the destination language L_D, whereas translation method B for source language L_2 might always map to a different word w_2 of L_D, although w_1 and w_2 have the same meaning. For instance, we might have:

Spanish	'*coche*'	$\longmapsto$	'*car*'
French	'*voiture*'	$\longmapsto$	'*automobile*'

   Such differences in lexicon can present an impediment to the learning algorithm's task of detecting similarities between documents of the same class.

2. **variety of expression:** Different languages may offer similar but not identical ways of expressing things. For example, in one language it might be common to say you take the bus, while in another language you more generally speak of using public transportation, perhaps just referring to the name of the local public transportation authority. Often one term has a more general meaning than the term in the other language. Consider also the following example.

English	*I have a headache*	*I have a headache*
Spanish	*Me duele la cabeza*	**It hurts the head to me* (Babel Fish)
French	*J'ai mal à la tête*	**I have pain at the head*

   It is a well-known problem that machine translation software such as Babel Fish (AltaVista, 2006) rarely produces texts with the expressions that would most

naturally be used in a language. In the example above, the use of literal translations would not reveal the links between the term '*headache*' and the concepts corresponding to '*head*' and '*to hurt*'.

3. **lexical ambiguity in destination language:** The problem of lexical ambiguity, too, is aggravated by the use of translations. Of course, lexical ambiguity is generally an issue when dealing with texts. However, for instance when translating the Indonesian word '*timbal*' to English '*lead*', meaning the chemical element, we introduce even more lexical ambiguity because '*lead*' is a homograph that could also mean the verb '*to lead*'.

4. **lexical lacunae:** In certain cases, the destination language might not be able to express certain ideas very well unless new terms are defined or borrowed from other languages. For instance, Latin might not be the language of choice for discussing hydraulic engineering. The Portuguese term '*saudade*' refers to a very specific kind of longing for something that is gone, might return in the distant future, though very likely is gone forever.

5. **overstemming and understemming:** A general problem of conventional text classification is that stemming might identify terms which semantically have nothing or very little in common, e.g. the word '*organ*', representing a musical instrument, with the words '*organic*', '*organization*', '*organism*', a phenomenon sometimes called overstemming. Another problem is understemming, when related-words are not conflated to the same form, e.g. '*acquire*' might be stemmed to '*acquir*', but '*acquisition*' to '*acquis*' (Paice, 1996).

In order to solve such problems, we investigate alternative approaches to multilingual text classification that exploit knowledge from ontologies and thesauri.

4.3.3 Ontology-based Concept Mapping

Many of the shortcomings of the translation-based approach can be addressed by exploiting resources such as ontologies (or thesauri) and using concept-based representations as discussed earlier in Chapter 3. The basic idea is to map all words to identifiers representing language-independent concepts specified by an ontology and then counting occurrences of such concepts rather than occurrences of the original words. The main advantage of this approach is that it resolves the issue of lexical variety because all synonyms will ideally be associated with the same concept. By mapping to concepts,

one also overcomes the problem of lexical ambiguity in the destination language. Overstemming is avoided by making use of a lemmatizer instead of a stemmer (cf. Chapter 2).

In multilingual settings, the idea of mapping terms from different languages to language-neutral semantic concepts seems rather attractive from a theoretical perspective. Unfortunately, an approach that simply maps each term to the respective candidate concepts from an external resource still has several shortcomings, for certain issues mentioned earlier such as variety of expression, lexical lacunae, and understemming are not resolved. Indeed, it turns out that this approach causes some entirely new problems that might outweigh the merits of adopting it, in particular because the mappings often remain very fine-grained compared to conventional text classification using the bag-of-words model, and thus fail to comply with the demands of classification tasks. The following problems can be identified:

1. **understemming:** While lemmatization removes various types of transforming morphemes from words and thus tends to handle inflected forms much better than stemming, it does not remove as many suffixes as stemming does. For instance, the concepts corresponding to the terms '*educational*' and '*educate*' are treated as distinct, despite being closely related.

2. **family resemblance and polysemy issues:** Word sense disambiguation obviously is useful in the case of true homonyms (e.g. the word '*bank*' which can be used for a river bank as well as for a financial institute) and heteronyms (such as the word '*bass*', which can be used to describe low-frequency tones in music but with a different pronunciation can also refer to certain fish species). However, the lexical database Princeton WordNet 2.1, described earlier in Section 2.3.4, distinguishes many senses of the word '*school*' of which at least seven can be seen as a thematic cluster, listed in Table 4.2. Even if it were universally possible to capture the intended use of a word in the document context very accurately, choosing only the correct sense would mean that other highly related senses are neglected, which might compromise the performance of text classification systems. The table seems to suggest that sometimes even lexical category information ought to be ignored.

When dealing with multiple human languages, additional problems may arise.

1. **incongruent concepts:** It might not be possible to map words in different languages to the exact same concept, for different languages tend to structure

Table 4.2: Polysemy: Some of the senses of the term '*school*' distinguished by Princeton WordNet 2.1 (see Section 2.3.4). Here *(n)* is used for nouns and *(v)* indicates a verb.

Lexical category	**sense**
(n)	(an educational institution) '*the school was founded in 1900*'
(n)	school, schoolhouse (a building where young people receive education) '*the school was built in 1932*'; '*he walked to school every morning*'
(n)	school, schooling (the process of being formally educated at a school) '*what will you do when you finish school?*'
(n)	school (an educational institution's faculty and students) '*the school keeps parents informed*'; '*the whole school turned out for the game*'
(n)	school, schooltime, school day (the period of instruction in a school; the time period when schools is in session) '*stay after school*'; '*he didn't miss a single day of school*'; '*when the school day was done we would walk home together*'
(v)	school (educate in or as if in a school) '*The children are schooled at great cost to their parents in private institutions*'
(v)	educate, school, train, cultivate, civilize, civilise (train to be discriminative in taste or judgment) '*Cultivate your musical taste*'; '*Train your tastebuds*'; '*She is well schooled in poetry*'

reality differently. For instance, the concept corresponding to the English word '*woods*' is much narrower than the respective ones for Danish '*skov*' or French '*bois*' (HJELMSLEV, 1943). Similarly, Vietnamese '*xanh*' corresponds to both '*blue*' and '*green*' in English, and English '*wall*' corresponds in German to both '*Wand*' (an interior wall) and '*Mauer*' (an exterior wall) (HUTCHINS, 2003).

One might come to think that this is merely a matter of word sense disambiguation but in fact this is not always the case. Rather, the concept corresponding to the French word '*bois*' may be regarded as a super-concept of the concepts corresponding to English '*wood*' and '*woods*', and the Vietnamese '*xanh*' as a hypernym of '*blue*' and '*green*'. Even that would be a simplification, however, as all of these concepts are rather vague ones with fuzzy boundaries and connotations that vary from language to language. This means that even terms that seem to correspond, such as Spanish '*selva*' and French '*forêt*' in reality may have slightly different meanings and an ontology may or may not represent them using the same concept. Consider also the Welsh word '*glas*', which, though usually translated as '*blue*', is also used to refer to the colour of grass or of silver.

2. **lexical lacunae:** Due to the idiosyncrasies exhibited in different languages, concepts lexicalized in one language are not necessarily lexicalized in another. For example, an ontology might link the commonly used German word '*Friedensnobelpreisträgerin*' with a concept representing women awarded the Nobel Peace Prize, yet be unable to map any English term to this same concept, simply because the English language does not possess a fixed term with the same designation. Fortunately, in some cases, the words might correspond to commonly used compound terms for which we also have an appropriate ontology mapping in the second language (e.g. German '*Pech*' corresponds to English '*bad luck*'). In other cases, however, the term might be less common in one language, e.g. in Japanese and Chinese, there are separate words for older and younger sisters, but a typical English ontology mapping is not that likely to provide a mapping for the expressions '*younger sister*' ('妹妹', '*mèimei*' in Mandarin Chinese, '妹', '*imouto*' or '妹さん', '*imoutosan*' in Japanese depending on whether it is the own or an other family) or '*elder sister*' ('姐姐', '*jiějie*' in Mandarin, '姉', '*ane*' or 'お姉さん', '*oneesan*' in Japanese). In even other cases, one might encounter concepts that are lexicalized in one language but truly unlexicalized in another language, e.g. Cantonese Chinese has a word for the paternal younger great uncle's wife ('叔婆'). In certain polysynthetic languages, one single word can capture enough information to represent a complex statement, e.g. the Finnish '*juoksentelisinkohan*' meaning '*I wonder*

whether I should run around aimlessly' (WIKIPEDIA, 2006).

3. **variety of expression:** Furthermore, as mentioned before, even aside from such fundamental differences in morphology and lexicon, different languages may simply choose different words when expressing the same idea. Recall the difference between French '*J'ai mal à la tête*' vs. Spanish '*Me duele la cabeza*' mentioned earlier in Section 4.3.2. This problem is not solved by merely mapping to concepts because the French noun '*mal*' will be mapped to a concept distinct from the one corresponding to the Spanish verb '*doler*'.

This simple concept mapping approach thus turns out to be inadequate for identifying connections between terms in different languages, although a well-structured ontology would be expected to reflect the connections present between such similar concepts.

4.4 Summary

Our formal characterization of multilingual text classification problems has shown that different classes of problems need to be distinguished. While certain problems can simply be reduced to monolingual ones, others require specifically multilingual solutions. The multilingual bag-of-words approach is limited to a narrow range of cases. Translation-based solutions, in contrast, are universal; however a linguistic analysis shows that they suffer from several drawbacks. Some of these issues may be resolved by mapping terms to concepts specified by an ontology or lexical resource, however this introduces several new problems, including polysemy issues and incongruent concepts.

Chapter 5

Ontology Region Mapping

In order to overcome many of the shortcomings of the simple concept mapping approach and other approaches outlined in Chapter 4, we present an alternative solution for text classification problems, called *Ontology Region Mapping*, where ontologies are construed as semantic networks (see Figure 5.1), for all of the shortcomings mentioned earlier involve concepts being treated as distinct despite being closely related. The intuition is that we should not only map terms occurring in a document to the respective concepts representing their meaning, but rather map to entire regions of semantically related concepts in the network, for whole groups of concepts or even entire topics are relevant when classifying, not just individual concepts.

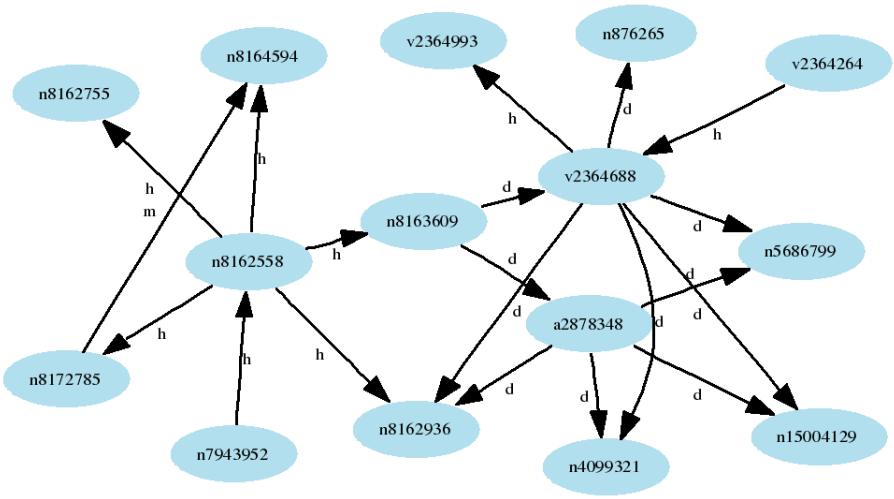

Figure 5.1: Ontologies as graphs: Concepts are linked via various types of relations.

By propagating weight from concepts to other related concepts we fully exploit the background knowledge offered by the ontology and overcome many of the problems

mentioned in Section 4.3.3. For instance, even if terms in different languages correspond to incongruent concepts, the respective concepts may nevertheless have a lot in common and are expected to be related. Similarly, when certain concepts are unlexicalized in one language but lexicalized in another, it is assumed that the ontology will relate the complex concept to some of the simple concepts it is based on, e.g. the concept associated with the German term '*Friedensnobelpreisträgerin*' is expected to be related to concepts associated with terms such as '*Nobel prize*' and '*peace*'. The same holds for issues such as understemming, variety of expression as well as problems arising from family resemblance and polysemy.

This is akin to how semantically related words presumably lead to similar regions of neurons being activated according to many theories of cognition. One of these theories, going back to Quillian (1968), Neely (1977), and many others, attempts to elucidate the phenomenon known as semantic priming (Meyer and Schvaneveldt, 1971) by assuming that an activation of neurons associated with specific concepts results in activation energy being spread to neurons associated with semantically related concepts (see McNamara, 2005).

5.1 Overview of Procedure

Since we can map words from several languages to the same concept space, this approach can be used directly for multilingual text classification. However, given that this method may also be applied to monolingual problems, an alternative setup is possible for multilingual problems, involving the use of our technique in conjunction with machine translation in order to overcome at least some of the deficiencies of using translations with conventional text classification. In this case, all documents are translated to some pivot language, and then monolingual text classification is performed using Ontology Region Mapping.

Regardless of which of these alternatives is chosen, the following tasks will need to be tackled in order to establish a representation of a document's content. After tokenizing the document and eliminating superfluous stop words, each source term will be mapped to one or more concepts specified by an ontology. This process is far from trivial as it involves word sense disambiguation, i.e. determining which candidate concepts are the most appropriate for a given occurrence of a term, and lemmatization, i.e. reducing inflected word forms such as '*children*' to their citation form ('*child*').

Whenever a direct mapping from a term to some concept has been established, additional related concepts will be evaluated. For this, a graph algorithm will be used in order to repeatedly propagate weight from one concept to another according to the relations captured by the ontology, thereby activating entire regions of concepts.

In the end, when all source terms have been mapped to concepts and all propagation steps have been completed, the sum of all weight assigned to any given concept will be used to compute its corresponding feature value. Together, all such feature values then constitute the final feature vector that represents the document.

5.2 Ontologies

In order to develop this idea of mapping terms to regions of concepts, we shall first attempt to formally characterize some of the vague notions of ontologies presented earlier in Chapter 2.3.

For our purposes, it suffices to see an ontology as a system of concepts with binary relations between them. We do not require ontologies to provide us with formal definitions for entities, because all that is needed is a set of identifiers for concepts (*concept terms*) and a function that returns information about the relations holding between such concept terms. We thus propose the following definition.

Definition 5.2.1. An *ontological resource* is a tuple

$$O = (C_O, R_O, \tau_O)$$

where C_O is a set of terms that specify concepts (*concept terms*), R_O is a set of relations that may hold between concept terms, and

$$\tau_O : C_O \longrightarrow \mathcal{P}(C_O \times R_O \times [0,1])$$

provides information about how concept terms are related by returning finite sets of entries in $C_O \times R_O \times [0,1]$ such that for each entry (c, r, w) the concept term c refers to a related concept, $r \in R_O$ indicates the type of relation between the two concept terms (hypernymy, antonymy, etc.), and $w \in [0,1]$ is the relation weight, a value that specifies to what degree the two concept terms are related, ranging from 0.0 (unrelated) to 1.0 (fully related). Here, $\mathcal{P}(S)$ denotes the power set of a set S, i.e. the set of all subsets of S.

For an extremely simple example, consider an ontological resource $O = (C_O, R_O, \tau_O)$ using a universe of terms consisting only of $C_O = \{$'*Kenya*', '*Nairobi*', '*country*', '*national capital*', '*city*'$\}$. The relation types list could be $R_O = \{r_{\text{instance}}, r_{\text{holonymy}}, r_{\text{hypernymy}}\}$, and the knowledge specified could include $\tau_O($'*Nairobi*'$) = \{($'*national capital*'$, r_{\text{instance}}, 1.0)$, $($'*Kenya*'$, r_{\text{holonymy}}, 1.0)\}$ as well as $\tau_O($'*national capital*'$) = \{($'*city*'$, r_{\text{hypernymy}}, 1.0)\}$, meaning that Nairobi is the national capital of Kenya and a part of Kenya, and that every national capital is a city.

The set C_O may be uncountable and can contain terms for any type of entity, including various types of particulars, properties, and relations, and we refer to the meanings of any such terms as *concepts*. In fact, C_O may even include terms that denote states of affairs and similar entities, considering that the human language terms we would like to map to concept terms might include cases like the Finnish word '*juoksentelisinkohan*' which means '*I wonder whether I should run around aimlessly*' or Turkish '*Nasılsınız*' which means '*How are you*'. Mapping such words to concept terms may be possible because the latter can be complex expressions constructed in some formal language L_O rather than just simple identifiers. Knowing which language L_O is used for the concept terms is not required for our purposes, although we do assume that the concept terms in C_O are somewhat normalized so that identical concepts tend to correspond to identical concept terms[1].

The ontological resource further provides us with knowledge about relations that hold between concept terms from C_O. For every ontological relation between entities there is a corresponding semantic relation between concept terms or other words. For instance, identity of concepts implies synonymy of concept terms, super-concept relationships correspond to hypernymy of terms, and part-whole relationships between concepts are captured by meronymic term relations. Although one may deny that thesauri and similar lexical resources define ontologies, it is clear that they fulfil our requirements for ontological resources. However, ontologies may, of course, also capture much more specific relationships between entities, e.g. expressing that one entity was created by another entity.

The relational knowledge is manifested in a function τ_O that returns related concepts with respect to a set R_O of relation types. We assume that for any concept term $c_0 \in C_O$

[1]For instance, in the description logics $\mathcal{AL}$ and $\mathcal{ALC}$ (Schmidt-Schauss and Smolka, 1991), the equivalent concept terms $Student \sqcap Female$, $Female \sqcap Student$ and $Female \sqcap \top \sqcap Student$ could all be normalized to $Female \sqcap Student$, where the identifiers are in lexicographical order ($\sqcap$ is reflexive) and no superfluous intersection with the top concept $\top$ is carried out, as $\top$ by definition contains all individuals.

the function $\tau_O(c_0)$ provides us with a finite set of relation tuples of the form (c, r, w) where r indicates the type of relation, c is the related concept term, and w with $0 \leq w \leq 1$ is the weight of the relation, i.e. roughly specifies to what degree the two concept terms are related in this way. For instance, for the concept term $c_{\text{schoolhouse}}$ representing the concept of a schoolhouse we might have an entry $(c_{\text{building}}, r_{\text{hypernymy}}, 1.0)$, indicating that the concept term representing the concept of a building is a full hypernym, and for c_{class} there might be an entry $(c_{\text{lesson}}, r_{\text{similarity}}, 0.8)$, indicating to what degree the concepts of classes and lessons are related.

5.3 Ontology Mapping

Ontological resources on their own cannot serve as lexical resources, for the concepts they specify are normally not linked to specific terms of a human language. Although it is advisable to use concept terms that are intuitively understandable, e.g. '*UndergraduateStudent*', ontological resources might just as well use cryptic names such as '*X3738*'.

In order to link words found in a text to concepts from an ontology O, additional auxiliary *ontology mapping functions* μ are required, which, roughly speaking, map terms from a human language L_{H} such as English, Japanese, or Esperanto to concept terms from a set C_O.

Definition 5.3.1. An ontology mapping function is a function

$$\mu : \quad C_{L_{\text{H}}} \times \Delta_{L_{\text{H}}} \quad \longrightarrow \quad \mathcal{P}(C_O \times [0,1])$$

that maps human language terms from $t \in C_{L_{\text{H}}}$ depending on their local context $\delta \in \Delta_{L_{\text{H}}}$ in a text to a set of 2-tuples (c, w) such that $c \in C_O$ is a concept term that is possibly relevant and w is the weight, viz. the degree of relevance for c. Here, C_O is a set of concept terms associated with an ontological resource O, and L_{H} is a human language with terms $C_{L_{\text{H}}}$ and $\Delta_{L_{\text{H}}}$ as the set of all texts written in L_{H}, i.e. the set of all finite sequences of symbols adhering to L_{H}.

Such an ontology mapping function μ could map the word '*bank*' in the sentence '*The nearest bank is the one at the River Thames*' to $\{(c_{\text{bankbranch}}, 0.6), (c_{\text{riverbank}}, 0.4)\}$, meaning that a weight of 0.6 is associated with the concept term representing branches of financial institutes, while an inferior weight of 0.4 is associated with the concept term representing river banks.

The degree of relevance could for example be set based on an estimation of the probability that c represents the true meaning of term t in the respective context, or more precisely, a probability estimate for the event $X = c$ given the context δ, where X is a random variable denoting the concept term that reflects the true meaning of t in the respective context. In what follows, however, we will argue against seeing the weights as probability estimates and present our own particular approach to determining them.

5.3.1 Word Sense Disambiguation

Ontology mapping functions need to look up which concepts might correspond to a human language term, and then tackle the task of determining to what degree the candidate concepts are relevant in a particular context. While certain lexical resources such as WordNet (cf. Section 2.3.4) provide a mapping from human language terms to concept terms, the process of determining which of those concepts are relevant in a particular context is not trivial. Word sense disambiguation is often considered one of the most difficult problems in natural language processing (Russell and Norvig, 1995, 687). Given a human language term, the method we propose starts out with all possible candidate concepts delivered by a resource such as WordNet for that term, except for the use of lexical category (i.e. part of speech) information determined via morphological analysis, which allows an elimination of many entries, e.g. if one knows that '*desert*' is a verb, then the concept associated with dry landscapes does not need to be considered.

In order to disambiguate among the remaining senses, the mapping functions can use the technique described earlier in Section 2.3 (Theobald et al., 2003). One regards the words surrounding a term in the text document as the term's local context, and then attempts to construct a similar context string for the candidate concept terms of the ontology, based on the assumption that the ontology provides human language descriptions for concept terms, written in the same language L_{H} as the text document. This allows for a concatenation of the description of the candidate concept term with the descriptions of its immediate holonyms and hyponyms as well as two levels of hypernyms. Feature vectors for both context strings are created using TF-IDF weighting (in conjunction with lexical analysis, stop word removal and stemming). Given two feature vectors $\mathbf{t}$, $\mathbf{c}_i$ for a human language term and a concept, respectively, the mapping function can then interpret the cosine measure as a measure of the relatedness between the word in the text document and the respective concept.

Our approach deviates from the one by Theobald et al. (2003) in that we do not

merely choose the concept with the highest score. Instead, we take all candidate concepts with their respective weights into account in our mappings. There is a theoretical as well as a practical motivation for this. Kilgarriff (1996) argues that the senses of a word should not be conceived as discrete objects. This is in accordance with the later Wittgenstein's view that what connects the different isolated uses of a word is simply *family resemblance* rather than the existence of some essential core meaning (Wittgenstein, 1953, 66ff). Since the ontological resources, however, merely offer a selection of discrete concept terms, the true meaning of a particular occurrence of a human language term can be approximated by selecting the closest available concept terms and assigning weights to them rather than forcing merely one sense to be chosen while neglecting the others. The practical motivation for this behaviour is that for text classification purposes, capturing the exact true sense of a term is usually unnecessary because many related senses might be equally relevant, as has been pointed out in Section 4.3.3.

We thus propose the following heuristic. If $S = \{c_1, \ldots, c_n\}$ is the set of context vectors for all candidate concept terms from C_O for human language term t with context vector $\mathbf{t}$, and $\mathbf{c}_i \in S$ is the context vector for concept $c_i \in C_O$, then the final weight assigned to concept c_i by the mapping function is

$$w(\mathbf{t}, \mathbf{c}_i, \delta) := \frac{cossim'_\delta(\mathbf{t}, \mathbf{c}_i)}{\sum_{j=1}^{n} cossim'_\delta(\mathbf{t}, \mathbf{c}_j)} \tag{5.1}$$

for some constant δ and a function $cossim'_\delta$ defined as

$$cossim'_\delta(\mathbf{t}, \mathbf{c}) := \begin{cases} cossim(\mathbf{t}, \mathbf{c}) + \delta & cossim(\mathbf{t}, \mathbf{c}) > -\delta \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

In other words a fixed value δ is added to every cosine similarity, and after eliminating result values smaller than zero (a negative δ could otherwise result in negative weights), each result is set in relation to the respective result values of other candidate concept terms.

Considering that

$$\lim_{\delta \to \infty} w(\mathbf{t}, \mathbf{c}_i, \delta) = \frac{1}{|S|} \tag{5.3}$$

and that the cosine similarity values lie in $[0, 1]$, we very quickly approach the uniform distribution with increasingly positive values for δ. In contrast, negative values for δ tend to produce a stricter disambiguation where concepts with small cosine similarity values are largely ignored.

5.3.2 Lemmatizing Ontology Mapping Functions

Our hope is that for a high number of document terms d_i we are able to obtain a mapping $\mu(d_i, \delta_i) \neq \emptyset$ with our ontology mapping function μ. However, typical external resources only provide mappings of lemmata, so if d_i is an inflected word form such as '*children*' rather than a lemma such as '*child*', then the mapping might often simply yield $\mu(d_i, \delta_i) = \emptyset$.

This problem cannot be solved by means of a normal stemmer because stemming algorithms often generate word stems that are not valid word forms and hence are even less likely to be recognized by the ontology mapping function. Instead, we use a lemmatizing function σ (see Section 2.3) for each human language, which delivers the most likely base form of terms in that language based on their local context. These functions are used to define lemmatizing ontology mapping functions that first look up the inflected form using the regular ontology mapping, and then, if no entry is found, resort to looking up the base form returned by σ. Compared to the alternative procedure of simply always using the lemmatizer, this policy allows for more precise results in the case of words such as '*glasses*' which might refer to eyeglasses, whereas the form '*glass*' cannot be used in that way.

Definition 5.3.2. A lemmatizing ontology mapping function is a function

$$\begin{aligned} \mu' :\ C_{L_\mathrm{H}} \times \Delta_{L_\mathrm{H}} &\longrightarrow \mathcal{P}(C_O \times [0,1]) \\ \mu'(d_i, \delta) &:= \begin{cases} \mu(d_i, \delta) & \text{if } \mu(d_i, \delta) \neq \emptyset \\ \mu(\sigma(d_i, \delta), \delta) & \text{otherwise} \end{cases} \end{aligned} \tag{5.4}$$

for a set C_O of concept terms, a human language L_H with a corresponding set of context texts Δ_{L_H}, an ontology mapping function μ that maps from L_H to terms in C_O, and a lemmatizing function σ for L_H.

A lemmatizing mapping function is able to provide mappings for terms such as '*children*' even when the original mapping function only has mappings for the citation form '*child*'. However, in cases when the original mapping function does have mappings for an inflected form, no lemmatization is performed, so the results for '*glasses*' in '*I lost my glasses*' gives us perhaps $\{(c_{\text{eyeglasses}}, 0.8), (c_{\text{drinkingglass}}, 0.2)\}$ rather than the incorrect $\{(c_{\text{eyeglasses}}, 0.0), (c_{\text{drinkingglass}}, 1.0)\}$ which one might expect if the singular form '*glass*' were used.

5.4 Weight Propagation

The ontology mapping functions as well as the corresponding lemmatizing mapping functions map human language terms to the concept terms that immediately represent their respective meanings. As outlined earlier, our approach, however, differs from existing approaches in that we do not merely map to individual concepts but to entire regions of concepts. For this, we regard the relational knowledge provided by the ontological resource as a labelled multidigraph, i.e. a directed graph where every node is labelled with a concept term, every arc is labelled with a relation between concepts, and multiple arcs are allowed between any two nodes (i.e. there is a multiset of arcs). One can then traverse this graph of concepts in order to additionally propagate a part of each concept's weight to related concepts. Consider, for instance, that a lemmatizing mapping function μ maps a human language term to some concept term c_0 with weight 1.0. For every relation type $r \in R_O$ we have an associated *relation type weight* $\beta_r \in [0, 1)$, so, for example, we could have 0.8 for hypernym concept terms and 0.2 for hyponym terms. If one decides to recursively pass on 80% of the weight to related hypernyms, then the direct parent hypernym of c_0 would receive weight 0.8, and the grandparent would receive a weight of 0.64, and so on. This propagation stops when the weights fall below some predetermined threshold. In addition to these relation type weights as parameters specific to a given term weighting process, the amount of weight passed on from node to node is also governed by the fixed *relation weights* delivered by the ontological resource, which capture the degree of relation between two specific concept terms (cf. Section 5.2).

The model chosen for this in our research avoids feedback and fan-in effects, instead giving each concept term c' that is in some way related to the starting concept term c_0 the weight associated with the path from c_0 to c' that maximizes the weight of c'. In order to determine these most optimal paths and assign the correct weights to related concepts, we thus need to traverse the multidigraph, making sure not to get caught in cycles or choosing suboptimal paths to related nodes (see Figure 5.2). To this end, an algorithm inspired by the A* search algorithm is used (see Russell and Norvig, 1995, p. 92ff).

The objective of Algorithm 5.4.1 is to determine the optimal weights for related concepts and then accordingly update global concept term counts ctc_c which represent the sum of all weights assigned to a concept while processing an entire document. In addition to the initial starting concept c_0 and its weight w_{c_0} it takes several other parameters as input.

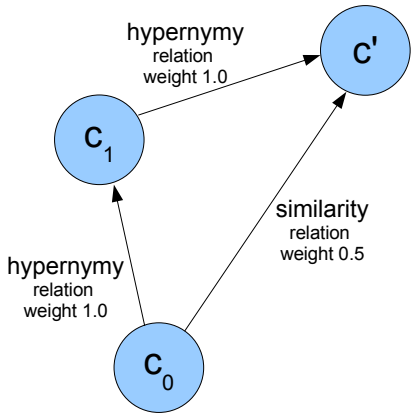

Figure 5.2: Suboptimal paths: c' is reachable from c_0 via two different paths, a long one as well as a direct one with relation weight 0.5. If c_0 has weight 1.0 and 80% is passed to hypernyms (relation type weight 0.8) and 40% for similarity (relation type weight 0.4), then the direct path from c_0 to c' would only lead to a weight of $0.5 \cdot 0.4 = 0.2$ for c' whereas via the indirect path we obtain $(1.0 \cdot 0.8)^2 = 0.64$. The indirect path is thus the optimal path to c'.

These include the ontology to be used, the concept term counts ctc_c to be updated, a function β that returns the relation type weights β_r as well as a threshold $w_{\min}$ that determines when the weight propagation stops. A more detailed discussion of the $w_{\min}$ and β_r parameters as well as a simple parameter space search heuristic that can be used to empirically determine suitable values for them will be presented in Section 5.6.

Our weight propagation algorithm maintains a list of nodes to be visited, sorted by weight, as well as a list of nodes already visited, and each node corresponds to a concept term. Initially, c_0 is added to the list of nodes to be visited in conjunction with its weight w_{c_0}, as provided by the lemmatizing mapping function. The algorithm then repeatedly removes the node c with the highest weight from this list of nodes to be visited and increments the counter ctc_c by the weight of c. It then evaluates all neighbours of c, by computing their weights and adding them to the list of nodes to be visited (provided their weight is over the pre-determined threshold $w_{\min}$).

Theorem 5.4.1. *When updating concept term counts, Algorithm 5.4.1 always chooses the weight associated with the optimal path.*

Proof. This stems mainly from the fact that the algorithm always chooses the node with the highest weight from the *open* list. For a proof by means of a reductio ad

Algorithm 5.4.1 Ontology-relation-based feature weighting

Input: initial concept c_0 with weight w_{c_0} from Ontology $O = (C_O, R_O, \tau_O)$, initial term counts ctc_c for concepts $c \in C_O$, function β which returns the relation type weight $\beta_r \in [0, 1)$ for any relation type $r \in R_O$, weight propagation threshold $w_{\min} > 0$

Objective: update term counts ctc_c for all relevant concepts $c \in C_O$ by following relational paths

1: $weight_{c_0} \leftarrow w_{c_0}$, $weight_c \leftarrow -\infty$ for all $c \neq c_0$
2: $open \leftarrow \{c_0\}$, $closed \leftarrow \emptyset$
3: **while** $open \neq \emptyset$ **do**
4: choose concept c with greatest $weight_c$ from $open$
5: $open \leftarrow open \backslash \{c\}$, $closed \leftarrow closed \cup \{c\}$ ▷ Move c to *closed*
6: $ctc_c \leftarrow ctc_c + weight_c$ ▷ increase term count
7: **for each** relation entry $(c_i, r_i, w_i) \in \tau_O(c)$ **do** ▷ visit neighbours c_i of c
8: **if** $c_i \notin closed$ **then**
9: $w \leftarrow weight_c \cdot \beta(r_i) \cdot w_i$
10: **if** $w \geq w_{\min}$ **then** ▷ proceed only if over threshold
11: $open \leftarrow open \cup \{c_i\}$
12: $weight_{c_i} \leftarrow max\{weight_{c_i}, w\}$

absurdum, assume that the algorithm has arrived at concept term c with computed weight w but an optimal path to c exists which leads to a computed weight of $w^* > w$ for c. For this optimal path $c_0, c_1, \ldots, c_n, c$ with computed weights $w_0^*, w_1^*, \ldots, w_n^*, w^*$ we have $w_0^* > w_1^* > \cdots > w_n^* > w^* > w$ because all relation type weights $\beta_r < 1$ and, according to Definition 5.2.1, all relation weights $w_i \leq 1$. Since c_0 is our starting node and initially is the only node available in the *open* list, the node c_1 must have been seen and added to the *open* list with weight w_1^* in the first iteration. Furthermore, from $w_1^* > w$ it follows that c_1 must have been visited before we reached c. This, in turn, inductively implies that c_2 was seen with weight w_2^* and visited before c, and so on. Finally, c must have been seen with the optimal weight w^* and visited with that optimal weight before it could have been visited with the suboptimal weight $w < w^*$. This contradicts our original assumption. □

Theorem 5.4.2. *Algorithm 5.4.1 terminates.*

Proof. For the relation type weights β_r we have $0 \leq \beta_r < 1$ by definition. Also, according to Definition 5.2.1, the relation weights w_i returned by the ontology relation function τ_O may not exceed 1. Any concept term added to the *open* list thus is added with a weight strictly smaller than that of its predecessor. Since Definition 5.2.1 also specifies that the number of related concepts returned for any $\tau_O(c)$ must be finite, at some point no nodes with weights greater or equal $w_{\min}$ will remain in the *open* list, for we have required $w_{\min} > 0$. □

Moreover, the fact that cyclic graphs do not cause infinite loops already follows trivially from the use of a *closed* list which ensures that each node is removed only once from the *open* list. Nevertheless, in order to decrease the runtime, one can optionally limit the number of iterations to some fixed value n by adding the condition $|closed| < n$ to the while-loop. When such a limit is in effect, the algorithm will always visit n highest-ranking concepts.

Theorem 5.4.3. *If Algorithm 5.4.1 is stopped after n iterations, n highest-ranking related concepts are guaranteed to have been visited.*

Proof. Imagine, for sake of argument, that the node associated with concept term c would be visited at the m-th iteration with $m > n$, but that the weight w that would be assigned to c is greater than the weight w' assigned to some concept term c' visited in one of the first n iterations. Then there exists some path $c_0, \ldots, c_k, c$ from the starting concept

term c_0 to c that causes c to receive weight w. Let the associated computed weights be $w_0, \ldots, w_k, w$, respectively. We then have $w_0 > \cdots > w_k > w$ because all $\beta_r \leq 1$ and all $w_i \leq 1$ (Definition 5.2.1). The algorithm obviously chooses the starting node c_0 first, because initially it is the only node available. When expanding the neighbours of c_0, the algorithm will add c_1 to the *open* list. In virtue of the fact that $w_1 > w > w'$, node c_1 is guaranteed to be chosen before c'. When selecting c_1 and visiting its neighbours, the algorithm will then see c_2 with weight $w_2 > w > w'$ and visit it before choosing c', and so on. Finally, the algorithm will see c and choose c before c' because $w > w'$. This contradicts our original assumption. □

5.5 Feature Vector Construction

The general procedure is thus as follows. In order to use the ontology to generate feature vectors for text classification, we import the documents and identify the languages involved, as described earlier. If language forms turn out to be used for which no appropriate ontology mapping function is available, we may perform language transformations or resort to machine translation, though in this case, the original versions are not retained in addition to the transformed or translated ones.

The documents are then tokenized and stop words are removed, resulting in a representation of the source document text as a finite sequence of terms $\mathbf{t} = (t_1, \ldots, t_k)$, as described earlier.

These terms are then mapped to ontology concepts, based on a weighting configuration $\Omega = (O, \mu, \beta, w_{\min})$ that specifies the ontology O to be used, the lemmatizing mapping function μ for the language of the document, a function $\beta(r)$ which returns the relation type weight β_r for any relation type $r \in R_O$, as well as the weight transfer threshold $w_{\min}$.

The conventional bag-of-words representation described in Section 2.1 is based on counting the number of occurrences of a particular term t in the term sequence $\mathbf{t} = \{t_1, \ldots, t_k\}$ for the document obtained after preprocessing. Here, however, for each term t_i in the term sequence, the ontology mapping function μ lemmatizes and disambiguates as described earlier, returning a list of candidate concepts with associated weights (concept terms $c_{i,j} \in C_O$ with associated weights $w_{i,j}$). Instead of counting how often a particular human language term t occurs, we count how often a concept term c occurs among

the $c_{i,j}$, incrementing the counter ctc_c by $w_{i,j}$ whenever $c_{i,j} = c$ for some i and j, i.e. whenever one of the concepts $c_{i,j}$ that a term t_i from $\mathbf{t}$ is mapped to turns out to be c.

Additionally, each concept term $c_{i,j}$ is submitted as input to Algorithm 5.4.1 with its respective weight $w_{i,j}$ such that the concept term counts ctc_c for any further relevant concept terms are updated, too. Formally, one might say that the final concept term counts are returned by a function ctc_Ω where $ctc_\Omega(\mathbf{t}, c)$ returns the sum of all weight values associated with concept term c when the terms in $\mathbf{t}$ are mapped to concept terms using the Ontology Region Mapping configuration Ω. These concept term counts normally have fractional values rather than the integer values corresponding to normal term counts for conventional text classification (as defined in Chapter 2).

Simply adding up weights, however, does not suffice to arrive at an adequate feature space for texts such that documents with similar topics are positioned close to each other. The concept term count values depend to a large extent on the length of a document because the longer a document is, the more often certain source language terms will tend to appear. A more adequate representation thus needs to capture the *relative* importance of terms. In order to accomplish this, we embrace ideas from conventional text classification and compute *concept term frequencies* that are very similar to normal term frequencies as defined earlier in Equation 2.7.

Definition 5.5.1. The Concept Term Frequency ctf_Ω is defined as

$$ctf_\Omega(\mathbf{t}, c) := \begin{cases} \dfrac{ctc_\Omega(\mathbf{t}, c)}{\sum\limits_{c'} ctc_\Omega(\mathbf{t}, c')} & ctc_\Omega(\mathbf{t}, c') \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

for a processed token sequence $\mathbf{t}$, a concept c, and an Ontology Region Mapping configuration Ω as defined earlier, and c' iterating over the set of all unique concept terms occurring in t.

Finally, one can further improve the performance by taking into account the number of documents a concept term is associated with, as in the case of the conventional TF-IDF weighting scheme (Equation 2.10) where the inverse document frequency is used. However, applying such a factor to concept term frequencies is a bit less straightforward. The rationale behind the multiplication with the inverse document frequency is the idea that terms that may be found in a vast number of documents bear little discriminatory information and thus should receive less weight. The notion of occurrence required for such document frequencies, however, does not emanate from our definition of concept

term counts as it does in the case of conventional term counts, for it is unclear whether a concept term with a minuscule weight should count as occurring in the document. We thus allow the use of different thresholds α when deciding whether a document is to be considered as containing a particular concept term.

Definition 5.5.2. For a term sequence $\mathbf{t}$ for some document (that need not be from the training set), a concept $c \in C_O$, a constant $\alpha \in \mathbb{R}$, a set of training document term sequences $D_{\text{tr}} = \{\mathbf{t}_1, \ldots, \mathbf{t}_m\}$, and an Ontology Region Mapping configuration Ω, we define

$$ctfidf_{\Omega,\alpha,D_{\text{tr}}}(\mathbf{t}, c) \;:=\; ctf_{\Omega}(\mathbf{t}, c) \cdot \log \frac{1}{cdf_{\Omega,\alpha,D_{\text{tr}}}(c)} \tag{5.6}$$

with

$$cidf_{\Omega,\alpha,D_{\text{tr}}}(c) \;:=\; \frac{|\{\mathbf{t}_i \in D_{\text{tr}} \mid ctc_{\Omega}(\mathbf{t}_i, c) > \alpha\}| + 1}{m + 1} \tag{5.7}$$

Here, adding 1 to the numerator and denominator avoids undefined results when $m = 0$.

The feature space is then constructed by associating each concept term with a separate dimension, and then the respective $ctfidf_{\Omega,\alpha,D_{\text{tr}}}$ values can be used to create the individual feature vectors which are then normalized, as described in Section 2.2.

5.6 Parameters and their Tuning

This algorithm has introduced new parameters that need to be adapted to the respective classification task. The relation type weight settings depend on the following factors:

1. The set of class labels $\mathcal{C}$: Very fine topic distinctions, between municipal bonds and mortgage-backed bonds for example, tend to require lower relation type weights than more general distinctions, as in the case of biology vs. physics.

2. Precision vs. recall trade-off: higher relation type weights typically induce an improved recall but often also a lower precision (cf. Section 7.2.4). Which values work best will thus depend on the particular application.

3. The structure of the ontology: When ascending the hierarchy of hypernyms of a concept term, for instance (i.e. moving along a path of nodes in the graph constituted by the ontology corresponding to a series of hypernym concept terms), having an initial weight of 1.0, a relation type weight of β_{h} and some propagation threshold $w_{\min}$ means that $\lfloor \log_{\beta_{\text{h}}}(w_{\min}) \rfloor$ levels of hypernyms will receive some weight, assuming that all relation weights are 1.0. Setting $\beta_{\text{h}} = 0.7$, $w_{\min} = 0.3$

causes $\lfloor \log_{\beta_h}(w_{\min}) \rfloor = \lfloor 3.38 \rfloor = 3$ levels of hypernyms to receive weight before the threshold is reached. In Princeton WordNet 2.1, for the first sense of '*academy*' the following path of hypernyms would be followed: '*secondary school*', '*school*', '*educational institution*'. However, another ontology might more directly lead from '*academy*' to '*school*', '*institution*', and then to '*entity*', which would be too general.

4. The type of documents: There is a big difference between categorizing very long articles and cases when just a short abstract or perhaps merely the title is available. Generally, the shorter the available text fragments are, the greater the problem of recall becomes (cf. Section 7.2.4) since we have fewer words we can match with. A single occurrence of a word might have to make the difference between a positive and a negative example. The relevance of this word thus is more likely to be recognized when higher relation type weights are used, leading to more related concepts being weighted.

5. Performance issues: In order to ensure the termination of the algorithm we specified earlier that the relation type weights β_r must not have the value 1. However, if one does not impose a limit on the number of concepts to visit, even values that come too close to 1 are not very practical. Defining $\beta = \max_r \beta_r$, the maximum distance from the initial concept turns out to be $\lfloor \log_\beta(w_{\min}) \rfloor$. According to Definition 5.2.1, the number of related concept terms for any concept is finite. Let k be the maximal number of related concepts a concept can be assigned to. Then for each initial concept our algorithm visits $O(k + k^2 + \cdots + k^{\lfloor \log_\beta(w_{\min}) \rfloor}) = O(k^{\log_\beta(w_{\min})})$ related concepts. This also means that the memory requirements, too, are exponential with respect to the maximum distance. Great care thus needs to be taken not to use extremely high β_r values.

   Concern about performance issues, however, is not a factor that is completely independent from the previously mentioned factors that influence the choice of parameters. Too big numbers of visited concepts normally also indicate that excessively many concepts are being visited that do not have very much in common with the original concept from a semantic perspective. Our experiments showed that configurations bringing forth useful results do not require overly large quantities of related concepts to be visited.

Since all of these issues need to be considered, determining fixed relation type weights that universally lead to optimal results is impossible. Instead, a suitable configuration will have to be found for each particular text classification problem. Indeed, this is a general challenge in machine learning, as many algorithms, from neural networks to

support vector machines, are highly configurable.

Since testing all possible configurations generally turns out to be infeasible, we propose a simple procedure for searching the parameter space based on hill-climbing algorithms (see RUSSELL and NORVIG, 1995, p. 111ff). If there are n parameters to be optimized, i.e. $n-1$ relation type weights in addition to the threshold $w_{\min}$, then an n-dimensional parameter value vector $\mathbf{w} = w_0, \ldots, w_n$ is used to capture their current values. Initially, one may begin with values randomly chosen from their respective domains.

In each iteration, one then creates new neighbouring parameter vectors $\mathbf{v}_j$ of the original vector $\mathbf{w}$, by keeping all values unchanged except for one single parameter value w_i ($v_{j,k} = w_k$ for each $k \neq i$). The value of w_i is instead increased or decreased by a reasonable amount, for example moving from 0.6 to 0.5 ($v_{j,i} = w_i - 0.1$ in this case). Since each of the n parameter values w_i can be increased as well as decreased, but not all such moves are valid (e.g. the relation type weights cannot become negative), there are up to $2n$ neighbours that can serve as candidates. As mentioned above, it is best to avoid moving to values higher than 0.8 for the relation type weights in order to avoid the problem of a combinatorial explosion of the number of concepts to be visited by the Ontology Region Mapping algorithm, unless of course a fixed limit has been imposed on this number. One should also avoid making moves to parameter values that are very close to the original because it might not be possible to detect any particular tendency if two neighbouring settings are very close.

Each candidate setting $\mathbf{v}_j$ is then evaluated by running the text classification system on a small training and validation dataset, using the parameter configuration designated by $\mathbf{v}_j$, and the performance is evaluated, perhaps in terms of averaged F_1 scores (see Section 7.2.4) obtained with different feature selection parameters.

The neighbour candidate $\mathbf{v}_j$ that has received the highest score then becomes the new $\mathbf{w}$, and another iteration can begin. The optimization stops when no more improvement is made. However, since hill climbing algorithms might end up encountering mere local optima rather than global ones, multiple runs with different random starting configurations $\mathbf{w}$ may be performed.

Optionally, the process of convergence can be accelerated whenever multiple candidates $\mathbf{v}_j$ exhibit significant performance increases compared with $\mathbf{w}$. In such cases, the next configuration $\mathbf{w}$ can be set to a combination of the respective $\mathbf{v}_j$ by modifying multiple component values of $\mathbf{w}$ simultaneously rather than just changing one value.

5.7 Combined Representations

The ontology-based feature weighting schemes still have one major downside: Document terms that cannot be mapped to an ontology concept are completely ignored. Yet, specialized technical terms as well as names of politicians, organizations, or companies, for instance, often are very helpful – occasionally even crucial – in correctly categorizing a document despite not being covered by an ontology mapping function. The same applies to product names, or simply new buzzwords that have not found their way into lexical resources yet. Some lexical resources go so far as to even exclude, as a matter of principle, terms such as '*England*' because they are names.

Most machine translation approaches simply maintain the original term in the translation whenever they do not recognize it. In a similar fashion, one may thus choose to maintain the original human language term whenever the ontology mapping is unable to map some t to a concept term from C_O, i.e. whenever $\mu(t, \delta) = \emptyset$. As explained in Section 4.3.1, such terms often are identical in different languages. Alternatively, one can also combine all concept terms with all original human language terms.

This ideally can be generalized to encompass arbitrary combined representations. Two different scenarios need to be dealt with. In many cases, two representations will be based on different feature types, e.g. features based on concept terms differ significantly from the human language term features of the bag-of-words model. In other cases, however, two representations will be able to share the same feature space, e.g. when two different translations are used as input to our Ontology Region Mapping approach.

The general approach thus is as follows. Each base representation is associated with a zone identifier. When constructing the base representations for a document, we ensure that representations sharing the same zone identifier are constructed in a shared vector space, which implies that the same terms are associated with the same feature dimensions. The first combination step then involves merging all representations that share the same zone identifier. If one document has representations $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$, then the combined representation will be the linear combination $\mathbf{w} = (u_1 + v_1, \ldots, u_n + v_n)$. This gives us one vector for each zone identifier.

In a second step, the vectors for the different zones are combined as suggested by Siersdorfer (2005, p.95) by creating a combined vector space. If we have two vectors $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_m)$, then the combined representation will be $\mathbf{w} = (u_1, \ldots, u_n, v_1, \ldots, v_m)$.

One can then use a variety of combined representations, including one that incorporates four base representations: Ontology Region Mapping using some translation method A (Zone 1), Ontology Region Mapping using some translation method B (also Zone 1), bag-of-words weighting with translation method A (Zone 2), and finally bag-of-words weighting with translation method B (also Zone 2).

In such cases, the $ctfidf_{\Omega,\alpha,D_{\mathrm{tr}}}$ values are computed globally with respect to all term counts and the feature space has dimensions for all zones, e.g. dimensions for concept terms as well as additional dimensions for stems of the original terms occurring in the text.

5.8 Summary

We have seen that Ontology Region Mapping is a document representation technique allowing us to exploit background information from ontologies and lexical resources such as thesauri, thereby, however, going beyond a direct mapping from document terms to concepts. This is achieved by adopting a graph traversal algorithm that identifies additional concepts that characterize the contents of the document. The techniques presented can be used for genuinely multilingual problems, either by employing multilingual ontological resources, or alternatively by first applying machine translation and then using monolingual ontological resources. Since such resources do not necessarily cover all terms one might encounter in the document, one can attempt to further increase the performance by combining this concept-based approach with conventional term-based approaches. A simple heuristic can be used to determine suitable parameter values for the algorithm.

Chapter 6

Implementational Issues

This chapter presents additional ideas intended to contribute to a well-designed implementation of the approach mentioned in Section 5, based on our own experience gained from developing a general multilingual framework for testing and deploying alternative text representation methods, which consists of over 18,000 lines of Java code (physical LOC) in over 80 classes, organized into five main packages, not including external libraries.

The Java programming language was chosen for our framework because it offers a high level of portability, is widely supported by third-party libraries, and, unlike many rapid prototyping languages, facilitates the development of very robust code. Ideally, much of the system architecture should already have been specified when the platform and programming language is chosen, as special care needs to be taken in ensuring that any external libraries one wishes to use are available for the setup chosen.

Figure 6.1 provides an outline of the main operations that need to be performed. After loading labelled training documents, feature vectors are created that represent their contents. A learning algorithm is used to establish a classifier based on a model of the training data. Subsequently, the feature vectors of new test documents can be passed to the classifier in order to obtain class predictions. If the test documents are labelled, one can evaluate the performance of the classifier.

This setup suggests four main packages to be implemented. A machine learning (`ml`) package is responsible for classifying as described in Section 2.1 and thus needs to implement machine learning algorithms. A second `nlp` package must implement natural language processing methods as described in Sections 2.2 and 2.3. A `util` package will normally be used for small helper functions, including for example input/output, string manipulation, configuration, and logging routines. Finally, one needs some kind of `main`

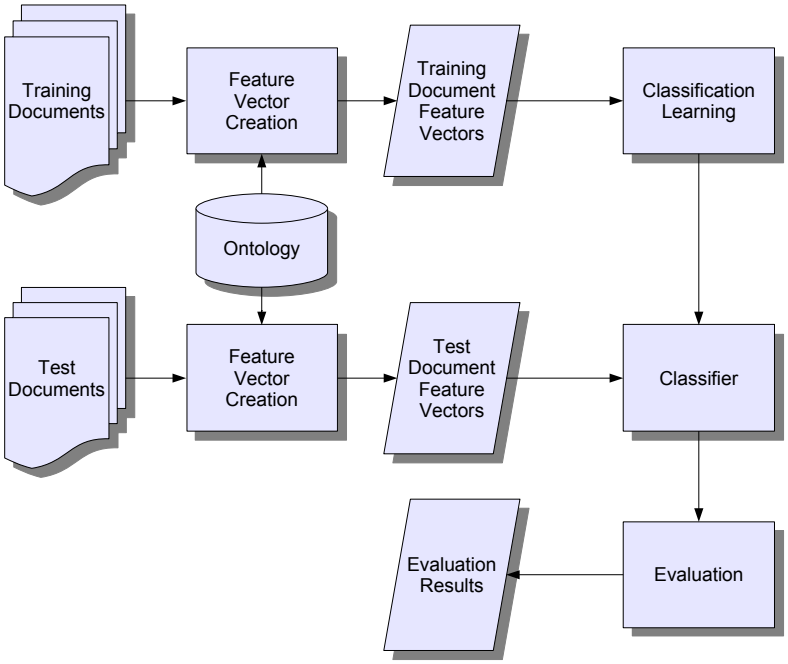

Figure 6.1: Simplified model of our implementation's general operation

package that ties everything together and is responsible for loading the configuration, using the NLP routines to create appropriate feature vector representations for documents as described in Chapter 5, using the learning algorithms in order to train a classifier, and perhaps evaluate the performance of that classifier for testing and tuning purposes, creating output in log files. An additional `tools` package may be used for various small tools that are created in order to manage resources and data.

6.1 Main Program

In our implementation, the main program is highly configurable and thus begins its operation by loading configuration files. The configuration system uses an XML format that allows for a specification of multiple values for each parameter such that each combination of parameter values is evaluated in a separate test run. Furthermore, dependencies between these parameter values can be captured. For example, one might want to test a feature selection of 500 and 1000 features when using one learning algorithm, but instead 300, 500, and 700 when using another. This approach is useful when many different configurations need to be evaluated, in particular for parameter tuning.

For each classification task, the program needs to load the respective sets of documents. One might choose to use an external database as the data source for such documents. This gives the program the advantage of being able to execute very specific data queries, e.g. in order to quickly retrieve all documents assigned to a specific class without having to open many single document files or having to specifically cache such information in a separate class association list. A disadvantage is that it might not always be possible to set up a database server on all computer systems one desires to use, so in our implementation, the documents are loaded from files stored in a proprietary XML format developed for our data, or alternatively using the Reuters newsML format. A proprietary format such as the one chosen in our implementation is preferred since it permits caching of several processed versions of the document text, each version being identified using the processing steps involved to generate it. For example, we might first need to annotate the text with lexical category tags and lemmata by performing morphological analysis, then generate a translation, and finally morphologically analyse the translation. Our system would cache three different processed versions of the document in this case.

The next step then is to create feature vectors for the training documents. In addition to the caching system mentioned above which is mainly intended to cache annotations and

translations, it is advisable to also cache additional processing steps in order to increase performance, because when dealing with multiple classes, documents will normally be used multiple times. One such caching step should occur before computing the TF-IDF values as those values depend on the entire document set rather than on single documents. In our implementation, we use a proprietary binary format for this since this reduces the overhead compared to XML or text-based formats. If no cached version of the document is available, the feature vector is created using appropriate document versions (e.g. translations, annotated versions) and then plugging various token stream providers and manipulators from the `nlp` package together, as will be described later on. Post-processing of the feature vectors then occurs, involving for example a conversion of term counts to TF-IDF and optionally also feature selection.

The program then needs to learn a classifier using classes from the machine learning package. As shown in Figure 6.1, this classifier is evaluated on feature vectors created for the test set using the error rate as well as F_1 and other performance measures in our implementation (see Section 7.2.4), and various log files are written to disk during operation in order to allow manual supervision and evaluation.

6.2 Machine Learning Classes

Our machine learning (`ml`) package contains classes for dealing with feature vectors, lists of feature vectors, as well as for performing feature selection, classification learning, and making actual classification predictions.

The `FeatureVector` and `FeatureVectorList` classes aid in managing feature vectors including such tasks as conversion to TF-IDF and determining cosine similarities.

Feature selection is performed by interfacing with version 3.4.8 of the WEKA library (see Witten and Frank, 2005) developed at the University of Waikato, New Zealand, and using its implementation of Information Gain or χ^2 rankings (see Section 2.2.4 and Sebastiani (2002)). For learning and deploying classifiers, we use version 6.01 of SVMlight (described in Joachims, 1999) as well as various algorithms provided by the WEKA library. While SVMlight is a very efficient implementation of Support Vector Machine-related algorithms, WEKA offers an enormous variety of machine learning algorithms for many different purposes. For this reason, we chose to support both libraries.

6.3 Natural Language Processing Classes

The natural language processing (`nlp`) package of our implementation offers a wide array of services, including, inter alia, token management and processing, stop word removal, stemming, morphological analysis, translation, ontological resource interfacing, and word sense disambiguation.

6.3.1 Text Import and Lexical Analysis

Many systems dealing with multilingual data need to support international characters going beyond what is offered by conventional 8-bit character sets such as ISO 8859/1. The most flexible solution is to internally use the Unicode standard or the roughly equivalent ISO 10646.

Most multilingual text classification techniques presume that the language and the character set used for the document text are known. If no additional meta-information is provided within the file format used to store it or through the database that delivers the documents, one may attempt to automatically detect the character set and language form, a task frequently performed using learning algorithms applied to feature vectors representing character n-grams, explained e.g. by Beesley (1988) and Kikui (1996).

The process of tokenizing the document or separating words from another becomes less trivial in multilingual settings because it must be implemented language-specifically. First of all, the set of characters that may occur as part of words is often distinct from the Latin alphabet as used for the English language, and may even include characters such as the colon (‘*:*’) which in Finnish sometimes is added to terms before appending suffixes. There are a few rare cases of languages that use characters other than a normal space for interword separation, e.g. the Ethiopic wordspace used in Ethiopian languages. However, a much more common case is when writing systems have not incorporated any general indication of interword separation at all, as in Japanese, Chinese and traditionally also often in Ancient Greek and Latin. In these cases, specialized word segmentation algorithms such as maximum matching (Palmer and Burger, 1997) or stochastic finite-state models (Sproat et al., 1994) need to be employed. Special attention might also need to be paid to clitics, which include e.g. the abbreviated form of ‘*to be*’ in ‘*you're*’, or ‘*me*’ and lo in Spanish ‘*dámelo*’. In certain languages, separating clitics and contractions turns out to be much more important than for English, e.g. to ensure that the term ‘*information*’ is extracted from French ‘*l'information*’. The tokenizers implemented in

our framework support languages that have interword separation by using whitespace and certain interpunctuation characters in order to detect word boundaries, while also attempting to recognize abbreviations such as '*E.U.*'. Most interpunctuation characters are discarded, as they are not relevant when classifying. Languages without interword separation such as Japanese are not supported, with the exception of the Chinese written language, for which we implemented the simple character-as-word algorithm, treating each Chinese Hanzi character as a separate token. This algorithm is known to work reasonably well for information retrieval tasks (cf. PALMER and BURGER, 1997), although better alternatives may be considered.

6.3.2 Term processing

In order to support weighted translations as well as establish an extensible program architecture we recommend storing all tokens with an associated weight and with other optional token attributes such as lexical category and lemma. Although the tokenizer simply gives all tokens a weight of 1.0 and does not set any additional attributes, this model allows an efficient integration of various operations into a single chain of token stream processors. The initial token stream is normally provided by a tokenizer, and then additional subsequent token processors may be added to create a token pipeline that can include operations such as case normalization, stop word removal, stemming or lemmatizing, mapping from terms to ontology concept terms as well as weighting related concept terms from the ontological resource. A simplified model of how this works in our implementation can be seen in Figure 6.2. In reality, additional processors can be embedded into such a pipeline, e.g. multiplexers and demultiplexers for arbitrarily combined representations or binary file input/output for caching feature vectors.

Apart from tokenization, another operation that occasionally becomes less trivial is the term normalization process. Whereas for monolingual conversion this process often merely involves a case conversion for the 26 letters of the Latin alphabet, in multilingual applications several transformations need to be performed, the first being a normalization of the character encoding. Consider, for example, that in Unicode the letter '*á*' can be encoded either with `U+00E1` (an '*a*' with an acute accent) or using a normal `U+0061` ('*a*') followed by `U+0301` (a combining acute accent). Moreover, for compatibility reasons, Unicode has the property of including code points such as `U+FB03`, which encodes the characters '*ffi*' (as in '*office*') in the form of a single ligature. It is thus recommended to use a normalization algorithm as described by the Unicode standard (see DAVIS and DÜR, 2005). This is usually followed by a conversion to lower-case characters, where

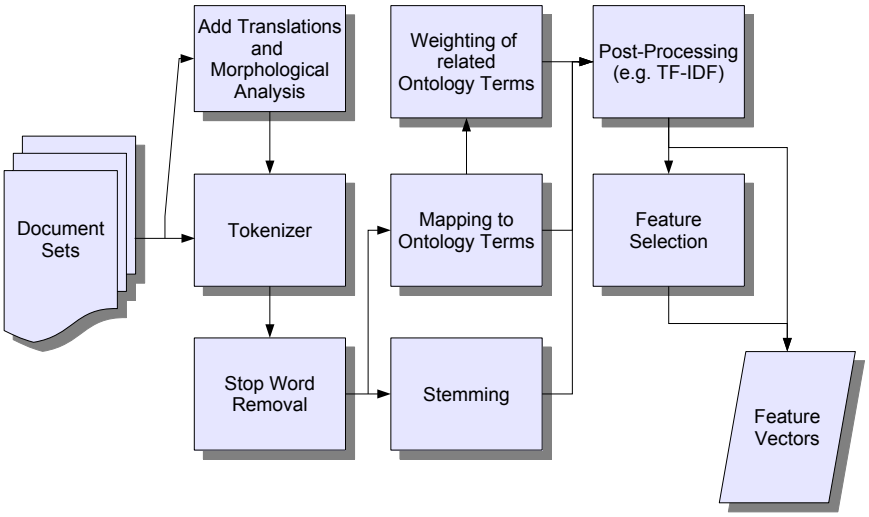

Figure 6.2: Simplified model of the feature vector construction process in our implementation.

various accented characters and other scripts need to be supported. Additionally, one also needs to pay respect to language-specific rules, e.g. in Turkish a capital letter '*I*' should be converted to a dotless ('*ı*') rather than the normal lower-case letter '*i*', the latter corresponding instead to a variant of the capital letter '*İ*' with a dot above ('*İ*'). Finally, the normalization process might also need to take care of certain spelling variations. This, of course, can be done for English (e.g. '*organize*' vs. '*organise*') but is more important in certain other languages such as Japanese, where often there is a choice between the kanji and hiragana scripts.

For *stop word removal*, separate stop word lists are required for each language involved. Our implementation supports several different languages. For the English language we use the very popular list of 571 stop words from the SMART information retrieval system (Buckley, 1985). The Spanish stop word list was also taken from the same source and contains 351 stop words. Should a stop word list for a specific language not be freely available (see e.g. Savoy, 2005), one may also resort to generating it manually from word frequency lists (Fox, 1990).

Stemming needs to be performed differently for each language, yet preferably in a consistent manner so that different language words with the same origins are reduced to the same stem, making it easier to identify related words across different languages. In our

implementation we use the *Snowball* package by Martin Porter due to its support for a wide range of different languages (see PORTER, 2001). The English-language stemmer included in the package deviates slightly from the original Porter stemmer (PORTER, 1980). Improvements include the removal of the '*-ly*' suffix and the incorporation of a small list of exception terms that cannot be processed normally. GOLDSMITH (2001) presents an algorithm that is able to learn the morphology of European and other languages in an unsupervised manner.

6.3.3 Other Techniques and Resources

Since proper morphological analysis is a rather complex process, it is usually best to use external tools. Our implementation is designed to work with the *TreeTagger*, a tool for morphological analysis, including lemmatization, developed at the University of Stuttgart (see SCHMID, 1994), which unfortunately is only available in binary form. The application has been trained on annotated corpora in several different languages and is able to estimate transition probabilities for lexical categories using binary decision trees. Since we do not need very fine-grained distinctions of lexical categories, our interfacing code converts the various tagsets used for different languages to an internal format that makes much coarser distinctions than the original ones.

Another task that usually requires external tools is machine translation, and depending on the languages that need to be supported and on the level of accuracy that is desired, one might have to resort to commercial software. In our implementation, we use AltaVista's Babel Fish online service as well as word-to-word translation based on bilingual translation dictionaries, as described later on in Section 7.2.1.

A very important component of our approach is the support of ontological resources. In our implementation, we import and store ontologies in a proprietary binary format that allows for more efficient access. This enables us for example to precompute the context vectors of concepts required for our word sense disambiguation technique, presented in Section 5.3.1. The ontological resources used in our tests are described in detail in Section 7.2.1.

6.4 Utilities and Tools Packages

A complete implementation of a text classification application will normally require various additional classes for operations such as I/O, string manipulation, etc. The

utilities (`util`) package of our implementation is a general library of such commonly used functions and contains classes for parsing configuration files and for logging, as well as simple routines for string manipulation, XML input and output, and comma-separated value (CSV) output.

Finally, our `tools` package contains various small programs that facilitate working with resources and evaluating data. It contains, for instance, code to import ontologies and lexical resources, in our case Princeton WordNet as well as the Spanish and Catalan versions of WordNet, including the application of external mapping tables (see DAUDÉ ET AL., 2003) in order to synchronize them with version 2.1 of Princeton WordNet. It also contains tools to import bilingual translation dictionaries as required by our implementation of word-to-word translation.

Various additional tools are used in order to prepare training and test sets, e.g. for generating category distribution statistics, for randomly choosing test and training files for all class pairs and for generating cached lexical category and lemma annotations as well as translations.

Finally, we created several small tools to assist in carrying out evaluation tests, e.g. by uploading configuration files to servers via the SSH file transfer protocol (SFTP), by starting and surveilling tests via SSH, as well as by downloading and evaluating the classification test log files, for instance generating statistics which can be used for plotting graphs. The next chapter will provide a detailed description of our experimental setup.

Chapter 7

Experimental Setup and Evaluation

We will now proceed to explain how the new techniques described in the previous chapters were evaluated with the aim of assessing their performance in terms of certain performance measures that will be described in a moment. The chapter begins with a description of the resources and then continues to describe the experimental procedure including the process of fine-tuning certain parameters. This is followed by a presentation and analysis of the main test results.

7.1 Material and Resources

7.1.1 WordNet

Princeton WordNet, described earlier in Section 2.3.4, is not only a resource that can be regarded as defining an ontological resource in the sense of Section 5.2, but also provides all information required for establishing mappings between English words and concepts of the ontology. We used version 2.1 of the resource, which specifies about 120,000 different synsets, the majority corresponding to nouns, and provides over 200,000 mappings from words to synsets.

For the Spanish language, we used the Spanish WordNet, which is structured in the same way as the original and contains over 100,000 synsets (Farreres et al., 1998). Although many WordNet-like resources have been created for various international languages, the majority of them are not freely available for research purposes. A Japanese version of WordNet currently does not exist, so only translation-based approaches were tested when Japanese documents were involved.

7.1.2 Reuters Corpora

Our training and test datasets were extracted from corpora provided to the scientific community by the Reuters news agency. *Reuters Corpus Volume 1: English Language, 1996-08-20 to 1997-08-19* (RCV1) contains over 800,000 English-language news stories (Reuters, 2000a) and served as a source for English-language articles in our experimental setup. It is one of the most commonly used collections for benchmarking text classification performance. Another corpus, titled *Reuters Corpus Volume 2 (RCV2): Multilingual Corpus, 1996-08-20 to 1997-08-19*, features nearly 500,000 news stories in several other languages (Reuters, 2000b), viz. Chinese, Danish, Dutch, French, German, Italian, Japanese, Norwegian bokmål, Portuguese, Russian, Spanish (Latin America), Spanish (Spain), and Swedish.

The documents are labelled with category codes, based on topic, industry, and geographical region. Reuters' policies required each document to be associated with at least one topic category and one region category – most are associated with several relevant categories. The classification process that produced the labels involved the initial use of rule-based text classification software and subsequent manual correction (Lewis et al., 2004).

Comparing the two Reuters corpora used, we noted that, although many classes are shared, the existing classification schemes differ in many respects. For instance, the class '*Biographies, Personalities, People*' contains over 5000 documents in the English corpus, but not one in either of the two Spanish-language collections. The fact that humans often disagree about classifications (Cleverdon, 1984) is obviously particularly relevant when classifications are made by people having different cultural backgrounds, but this is even further amplified when the set of available classes varies. However, although these factors might compromise the results, the Reuters corpora nevertheless are able to give us a good idea of how well our approach performs for multilingual problems.

7.1.3 Wikipedia

Wikipedia (Wikimedia Foundation, 2006) is a multilingual enyclopedia created and maintained collaboratively by Internet users all over the world in an open manner, meaning that anyone can perform changes almost instantaneously and all content is licensed under the GNU Free Documentation License (GFDL). Some have criticized Wikipedia

as failing to meet the standards of a reliable, authoritative source of knowledge. Nevertheless, the encyclopedia remains a very valuable resource providing vast amounts of information that cannot be found in traditional encyclopedias. Currently, several million articles exist, and a very extensive category system is used to organize them.

The differences in the classification schemes for different languages noted above in the case of the Reuters corpora obviously is a lot more extreme for Wikipedia, where each version has an entirely different category scheme and links between these schemes are only occasionally established on an ad hoc basis.

7.2 Setup

The resources mentioned above were then used for our evaluation setup. Our main tests were preceded by a number of preparatory steps, including the preparation of the datasets.

7.2.1 Training and Test Data

We used the Reuters corpora as well as Wikipedia as sources for our training and test data. In our tests, we decided to focus on cross-lingual text classification, as it is the most interesting case and shows how well a given method is able to establish similarities between documents in different languages. Most importantly, methods that perform well for cross-lingual problems can easily be adapted to perform well for other multilingual problems, while the converse, in contrast, is not generally true. Due to the need to perform machine translation in our tests, we extracted a subset of the corpora for use in our experimental setup. Two datasets were extracted from Reuters Corpus Volume 1 and 2 (RCV1/RCV2) using English documents from RCV1 for training and Spanish ones from the RCV2 collection as test documents. One of the datasets was based on the topic codes and another one on the geographical class codes that are available in the corpora. The industry codes were not used due to significant inconsistencies in the class labelling schemes between RCV1 and RCV2. For this extraction process, we initially started out with the corpora in their entirety, meaning over 800,000 documents from the English corpus, and c. 80,000 Latin American as well as c. 17,000 European Spanish documents from the multilingual Reuters collection (differences between the written forms of Latin American and European Spanish are very small and hence negligible). In neither case did we manually remove documents that are obviously misclassified (cf. Lewis et al.,

2004), considering that in production settings, too, such documents may very well occur. The Reuters newsML files were converted into our own proprietary format.

An additional dataset with Japanese test documents and English training documents was generated from Wikipedia using all articles assigned to a category or to any subcategory as members of the respective classes. The wikitext markup language was converted to plain text and class membership was determined recursively.

In our experimental setup, we decided to focus solely on binary classification. As Sebastiani (2002) argues, binary problems are more general than multi-label problems because the latter can be reduced to binary classification problems using class decomposition methods, as described in Section 2.1. The converse, in contrast, does not hold: An algorithm designed for multi-label classification cannot necessarily be used for binary classification because it might assign 0 or 2 class labels to a document.

In order to create such binary decision problems, we used 15 classes for each dataset, selected by a random number generator among all classes with sufficient numbers of documents in both languages involved. The 15 classes led to $\binom{15}{2} = 105$ class pairs and binary classification problems. For each pair of classes, we created training datasets consisting of 100 labelled training documents, and a test set consisting of 600 further documents (300 in the case of Wikipedia) not occurring in the respective training dataset. The documents were also selected randomly, however, in order to avoid getting biased error rates due to an unequal number of positive and negative examples, this was done in way that ensured that the ratio of positive to negative examples was 1:1 in every training and test set, and without any overlap, i.e. the respective subsets were disjoint.

The resulting datasets included class pairs where the classes involved should be rather easy to distinguish, e.g. '*Labour issues*' vs. '*Crime, Law Enforcement*', as well as classes that share a lot in common and thus require more fine-grained distinctions, e.g. '*Bond Markets*' vs. '*Equity Markets*'.

For our Reuters topic dataset, we used the contents of the `<headline>`, `<dateline>`, and `<text>` elements as the document contents, not the `<title>` element which contains a semi-automatically inserted country code (Lewis et al., 2004). Our geographical dataset is slightly different in that we excluded the contents of the `<dateline>` element, as it contains geographical references that would have oversimplified the task of classifying the data geographically. In addition, some text entries already had an embedded dateline preceding the actual article. Such occurrences, too, were removed in order to eliminate such explicit geographical information.

We then performed morphological analysis and added English translations for the documents using two different translation methods. The first was to use AltaVista's Babel Fish Translation service, which in turn is based on software by SYSTRAN, in order to translate Spanish and Japanese test documents to English (AltaVista, 2006). The second was a bilingual dictionary-based approach that uses freely available dictionaries (cf. Röder, 2002) in order to perform word-for-word translations, in our case from Spanish to English. Here we adopted a policy of maintaining all n translations when multiple translations were available for a term, assigning each token a weight of $\frac{1}{n}$. When a word is not found in the dictionary, an attempt is made to find a translation of its lemma.

7.2.2 Preparation of Ontologies

As mentioned earlier, for the English language, we used version 2.1 of the Princeton WordNet project. When looking up terms in WordNet, the setup we tested does not involve attempting to identify multiple word expressions but this feature could straightforwardly be added.

We also imported the Spanish WordNet and used mappings between WordNet 1.6 and WordNet 2.1 in order to synchronize it with our version of the Princeton WordNet data (see Daudé et al., 2003). We then merged both language versions of WordNet, creating a bilingual Spanish-English resource.

7.2.3 Implementation Settings

As mentioned earlier in Chapter 6, our algorithms were implemented into a general text classification framework that allows for experimentation with alternative feature weighting models.

Only the training documents were taken into account when defining the feature space, and the IDF was computed only with respect to the training documents. Following (Lewis et al., 2004), we perform an L_2 normalization of all vectors after feature selection (as explained earlier in Section 2.2.4).

For testing our representation with Support Vector Machines, we used SVMlight with its default settings that are known to work well for text classification tasks. In particular, this means that we used a linear kernel, as is customary in text classification, and the κ parameter for the soft margin (see Section 2.1), which characterizes the trade-off between

training error and margin, was set to the reciprocal of the average Euclidean norm of training examples (which is 1.0 due to the normalization of the feature vectors).

The Voted-Perceptron algorithm by Freund and Schapire (1998) was used in order to test a simple variant of an Artificial Neural Network. Considering previous results by Wiener et al. (1995), we chose to use 100 iterations in order to arrive at representative results.

Additional tests were performed using an implementation of Adaptive Boosting (Freund and Schapire, 1997) in conjunction with an entropy-based decision stump learner (see Section 2.1).

7.2.4 Effectiveness Measures

Having learnt a classifier $\widehat{\phi}$ using the training set, we would like to evaluate its performance by estimating how well $\widehat{\phi}$ coincides with the true classification ϕ. As mentioned before, pre-classified documents from a test set are normally used for this purpose. For each document $D \in \mathcal{D}_{\text{tst}}$ and every class $C \in \mathcal{C}$ we know whether $\phi(D, C)$ holds. We can then apply $\widehat{\phi}$ to the documents in the test set and compare the two classifications. It should be pointed out that, from a theoretical perspective, the idea of a classification being correct or incorrect should be taken with care. Since the semantic intensions of the classes are not formally specified, the choice of whether a document belongs to a specific class or not is a rather subjective and fuzzy decision (Sebastiani, 2006). For practical reasons, however, we normatively take any case in which our classifier $\widehat{\phi}$ gives us some $\widehat{\phi}(D, C) \neq \phi(D, C)$ to be error and consider all evaluation measures to be relative to ϕ.

For binary single-label classifications, we can then calculate the so-called *(sample) error rate* as the percentage of documents that were classified correctly.

As the error rate does not provide a neutral indication of effectiveness when the numbers of positive and negative examples differ, other effectiveness measures, too, have been considered in text classification research. Two other popular measures are *recall* and *precision*, originally from the field of information retrieval. The *recall* value indicates the sensitivity, and in the context of IR is defined as the ratio of all relevant material that is actually returned as a search result. The precision, in contrast, indicates the positive predictive value and is defined as the ratio of truly relevant documents in the

returned search results. In the context of binary text classification problems, class labels are regarded as search queries.

Definition 7.2.1. If TP, FP, TN, FN are the set of true positives, false positives, true negatives, and false negatives, respectively, then precision p and recall r can be defined as follows:

$$p := \begin{cases} \dfrac{TP}{TP+FP} & TP+FP \neq 0 \\ 1 & TP+FP+FN=0 \\ 0 & otherwise \end{cases} \tag{7.1}$$

$$r := \begin{cases} \dfrac{TP}{TP+TN} & TP+TN \neq 0 \\ 1 & otherwise \end{cases} \tag{7.2}$$

Neither precision nor recall is very useful on its own. The trivial classifier $\widehat{\phi}$ with $\widehat{\phi}(D,C) = \bot\ \forall D,C$ leads to an optimal precision while $\widehat{\phi}(D,C) = \top\ \forall D,C$ results in an optimal recall value.

In order to compare different evaluation results, one of several composite measures can be used, for instance the break-even point of precision and recall (criticized in Sebastiani, 2002, p.36), or the F_β-measure (van Rijsbergen, 1979, ch.7):

Definition 7.2.2. If p denotes the precision score, and r denotes the recall score, then the F_β-measure is computed as

$$F_\beta := \begin{cases} \dfrac{(\beta^2+1)pr}{\beta^2 p + r} & \beta p + r \neq 0 \\ 0 & otherwise \end{cases} \tag{7.3}$$

for a nonnegative β, where $\beta = 2$ for example would imply that precision is weighted twice as much as recall.

For example, in an e-mail filtering system, letting certain unsolicited e-mails pass through the filter is less disastrous than deleting important e-mails that in reality were not unsolicited. Since we are not evaluating our methods with respect to any particular application among the ones mentioned in Chapter 1, we set $\beta = 1$, adopting the commonly used F_1-measure which is equivalent to the harmonic mean of precision and recall.

$$F_1 = \begin{cases} \dfrac{2 \times p \times r}{p+r} & p+r \neq 0 \\ 0 & otherwise \end{cases} \tag{7.4}$$

For instance, obtaining a precision value of 0.9 in conjunction with a recall of 0.4 would lead to an F_1 score of only about 0.55. Obtaining 0.9 for both precision and recall induces an F_1 score of 0.9.

When testing multiple class pairs the results can be combined in two different manners. *Micro-averaged* scores are obtained by globally summing over all individual decisions, i.e. TP, FP, TN, and FN refer to global numbers of documents. *Macro-averaged* scores are obtained by averaging the local precision and recall scores obtained for each class or class pair (Sebastiani, 2002). Micro-averaging, used by the majority of researchers, gives classes with a large number of examples more weight compared to macro-averaging, which gives very small classes the same weight as large classes. Since we use a fixed number of documents for each class, the distinction is not very significant in our case. We focus on micro-averaged F_1 scores in what follows, as this currently is the most commonly used measure in text classification research.

7.2.5 Tuning of Term Weight Processing

A number of settings need to be made before the machine learning algorithms and our feature weightings algorithms can be used, and it is not *a priori* clear which settings lead to the best performance. Before the actual evaluation, we thus carried out several preliminary tests in order to find suitable parameter values. Since testing all possible combinations is not feasible, it is common practice to tune one parameter at a time, while leaving the others at some default value.

We took care not to use our test sets for tuning, as that would entail biased final results, instead using a separate validation dataset generated using the same procedure as for our Reuters topic dataset with English training and Spanish test documents, albeit with only 10 randomly chosen topic categories and $\binom{10}{2} = 45$ class pairs.

We started out by examining which term weighting post-processing settings best suit our algorithm. Our variant of the TF-IDF weighting scheme presented in Section 5.5 can be used with different α-values.

Figure 7.1 shows the experimental results of tests conducted with Support Vector Machines using four different configurations (see also Table 7.5 in Section 7.3 for an overview): a simple concept mapping (CM) without weight propagation to related concepts as outlined in Section 4.3.3, Ontology Region Mapping (ORM) from Chapter 5,

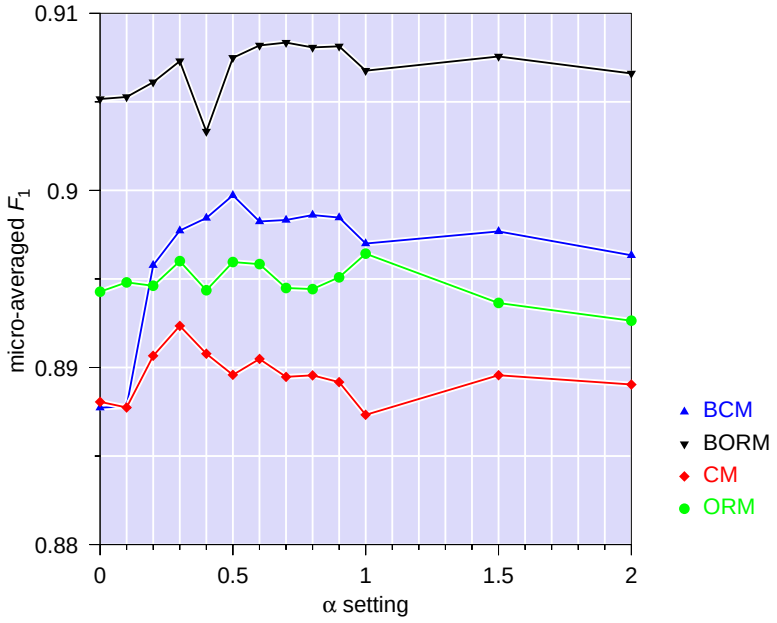

Figure 7.1: Performance when using $ctfidf$ with different α values

simple concept mapping using Babel Fish translations (BCM), as well as Ontology Region Mapping using Babel Fish translations (BORM). In all cases we used $\delta = -0.05$ for our word sense disambiguation algorithm (cf. Section 5.3.1) and an Information Gain-based feature selection of 1000 features. The relation type weights were intuitively chosen on an ad-hoc basis and are listed in Table 7.1.

While the results do not give us a clear indication of which α-value is best, they do seem to indicate that a threshold of 0.0 is suboptimal. The purpose of the multiplication with the inverse document frequency is to reduce the weight of high-frequency terms. The results demonstrate that counting concept terms as occurring in a document despite them being associated with miniscule weights is not the best approach. We therefore chose a threshold of 0.5, which had delivered optimal or near-optimal values in a number of cases.

Since our concept term counts (ctc_Ω) are in many ways different from normal term counts, another test was performed with the aim of ensuring that the conversion to concept term frequencies and concept TF-IDF values really is beneficial to the effectiveness. Table 7.2 suggests that this is indeed the case. Concept TF-IDF with $\alpha = 0.5$ consistently led to the best micro-averaged F_1 scores.

Table 7.1: Relation type weights chosen for preliminary tests

relation type	**weight** β_r
hypernymy	0.8
derivation	0.7
derived	0.7
participle	0.7
member/ part holonymy	0.7
substance holonymy	0.4
similarity	0.4
other types	0.0
$w_{\min}$ threshold	0.3

Table 7.2: Comparison of micro-averaged F_1 values (in %) for ctc_Ω, ctf_Ω and $ctfidf_{\Omega,0.5,D_{\text{tr}}}$ (our variant of TF-IDF with threshold $\alpha = 0.5$ for the document frequency)

	ctc	**ctf**	**ctfidf**
CM	88.4	88.9	89.0
ORM	83.7	86.5	89.6
BCM	88.8	89.3	90.0
BORM	86.0	87.5	90.7

7.2.6 Tuning of Word Sense Disambiguation

The next component examined was word sense disambiguation as introduced in Section 2.3.2. We tested different algorithms, using a rather large document text context window, based on the 'one sense per discourse'-heuristics according to which a word with multiple senses, when appearing multiple times in a discourse will nearly always be used in the same sense within that discourse. Tests by Gale et al. (1992) have shown this hypothesis to be correct in 97% of the cases.

Our tests were again conducted with Support Vector Machines applied to the validation dataset that is based on the topic codes of the Reuters corpora. The results in Table 7.3 show that for the simple concept mapping approach (CM and BCM) word sense disambiguation often does not manage to yield significantly better results than using

Table 7.3: Comparison of micro-averaged F_1 scores (in %) for *CM*, *ORM*, *BCM*, *BORM* setups with different WSD methods: equal distribution, CosDscMax, CosDscDist with different δ values

	CM	**ORM**	**BCM**	**BORM**
uniform distribution	88.2	88.8	88.2	89.2
CosDscMax	78.4	85.5	85.9	88.1
CosDscDist ($\delta = -0.5$)	88.2	88.8	88.2	89.2
CosDscDist ($\delta = -0.1$)	87.7	88.7	88.0	89.2
CosDscDist ($\delta = -0.05$)	87.8	88.5	88.2	89.6
CosDscDist ($\delta = -0.04$)	87.8	89.0	88.4	90.0
CosDscDist ($\delta = 0$)	87.9	89.9	87.9	91.0
CosDscDist ($\delta = 0.05$)	88.6	88.7	88.3	89.1
CosDscDist ($\delta = 0.1$)	88.5	88.8	88.4	89.3
CosDscDist ($\delta = 0.5$)	88.1	88.8	88.2	89.4

a uniform distribution, i.e. simply giving each of the n candidate concept terms for a given human language word the same weight $\frac{1}{n}$. One possible reason for this might be that an overly fine-grained disambiguation makes it more difficult to detect similarities between feature vectors for documents of the same class. As explained in Section 4.3.3, often many of the candidate concept terms have very similar meanings.

Our distribution based word sense disambiguation described in Section 5.3.1 (called CosDscDist in this context) is able to compensate for this to a certain extent by allowing multiple relevant concepts to be maintained. When an appropriate value for δ is used (see Section 5.3.1), it is shown to lead to better results than merely using the candidate term with the highest cosine similarity value as proposed by Theobald et al. (2003) (the latter approach is labelled CosDscMax in the table). The use of Ontology Region Mapping (ORM and BORM) acts as another counterforce against a too fine-grained weighting and thus enables the word sense disambiguation to perform better than uniform distribution.

It must be pointed out, however, that the fact that our method (CosDscDist) performed better than the original CosDscMax does not mean that the disambiguation in itself is more exact, but simply that our method is better suited for text classification purposes. For our main tests, we chose to use the CosDscDist method with $\delta = 0$.

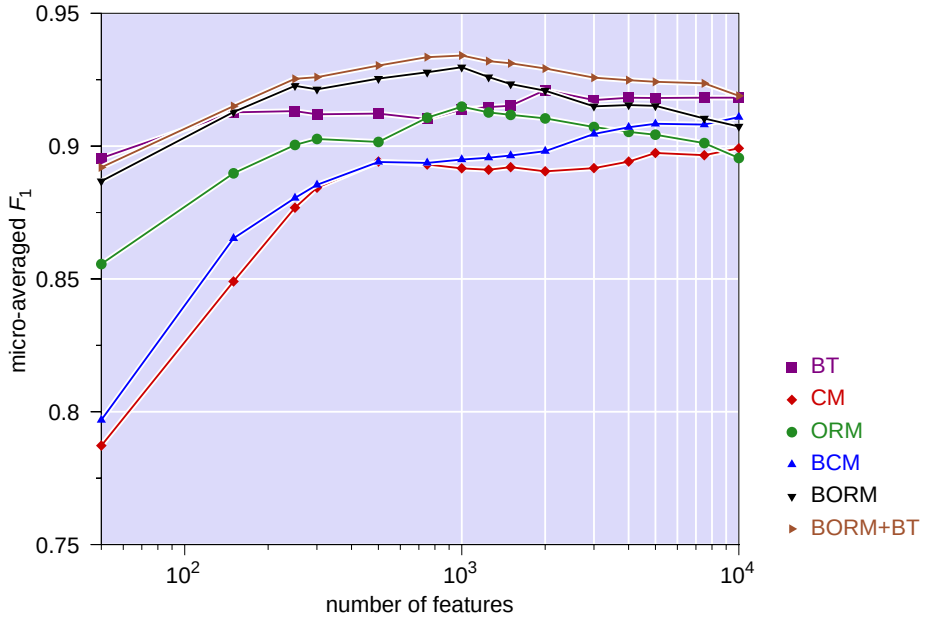

Figure 7.2: Feature selection analysis for Support Vector Machines (using a logarithmic scale from 50 to 10,000 for the number of features))

7.2.7 Tuning of Feature Selection

Different learning algorithms exhibit different levels of resistance towards irrelevant features. We thus tested the effects of feature selection on different classifiers using a wide range of different numbers of features.

Previous results by Yang and Pedersen (1997) caused us to focus on the Information Gain ranking as explained in Section 2.2.4. Initial tests showed that this method even consistently outperformed the χ^2 statistic in our setups (for a description of the χ^2 measure see Sebastiani, 2002).

Figure 7.2 shows our test results for the SVM algorithm, tested with five different setups: bag-of-words with Babel Fish based translations (BT), simple concept mapping (CM), Ontology Region Mapping (ORM), simple concept mapping with Babel Fish translations (BCM), Ontology Region Mapping with Babel Fish translations (BORM), and finally Ontology Region Mapping combined with bag-of-words weighting using Babel Fish translations (BORM+BT). All tests were performed with our concept TF-IDF weighting ($\alpha = 0.5$). Where applicable, we used our word sense disambiguation algorithm (CosDscDist) with $\delta = 0$.

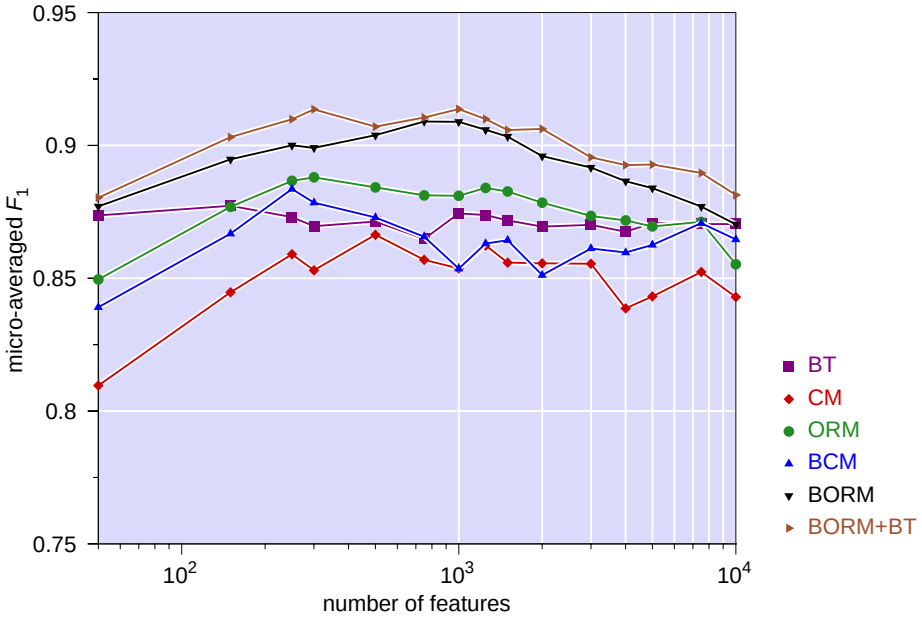

Figure 7.3: Feature selection analysis for the Voted-Perceptron algorithm (using a logarithmic scale from 50 to 10,000 for the number of features)

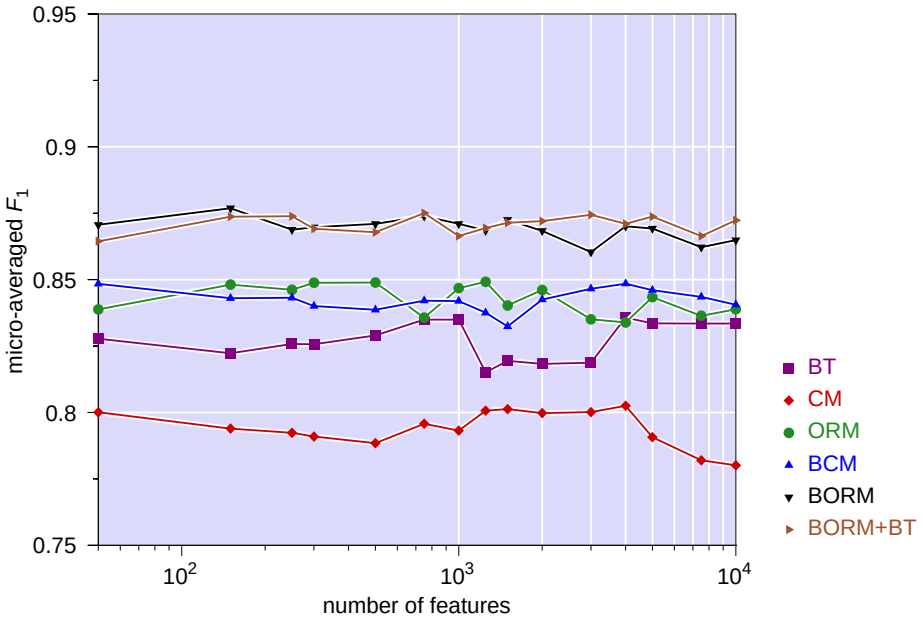

Figure 7.4: Feature selection analysis for the Adaptive Boosting algorithm (using a logarithmic scale from 50 to 10,000 for the number of features)

Based on previous results by Joachims (1998) and Brank et al. (2002), among others, many authors have considered Support Vector Machines robust enough to obviate the need for any additional feature selection process (e.g. Lewis et al., 2004). In accordance with Gabrilovich and Markovitch (2005), our results indicate that this assumption is not universally true when alternative text representation methods are used. With Ontology Region Mapping, the optimum number of features highly depends on the relation type weights chosen. For the weights used in this tuning test, the performance of ORM, BORM, and BORM+BT peaked at around 1000 to 1100 features, depending on the configuration. Even with the bag-of-words model (BT), using the Information Gain ranking led to slightly improved results in one case. For CM and BCM, ceteris paribus, the best performance is indeed achieved when no feature selection is used.

The test results obtained with the Voted-Perceptron algorithm, shown in Figure 7.3, demonstrate that the benefit gained from feature selection depends greatly on the algorithm. For the Voted-Perceptron, eliminating too few features or dispensing with feature selection altogether has a detrimental effect on performance. The AdaBoost algorithm, in contrast, when used with decision stump functions, in many cases does not seem to be influenced significantly from feature selection, as evidenced in Figure 7.4. The reason for this is that the decision stump functions chosen are unlikely to be based on rather uninformative features such as the ones typically eliminated by feature selection.

For each learning algorithm and each text representation method, we chose the feature selection threshold that produced the highest performance.

7.2.8 Tuning of Ontology Relation Type Weights

Finally, we fine-tuned the relation type weights β_r for the Ontology Region Mapping algorithm (BORM setup with SVM classifier), starting out with some intuitively chosen values very close to the ones in Table 7.1 and then using the procedure described in Section 5.6 to find more preferable values. The final relation type weights are depicted in Table 7.4.

The importance of hypernymy is rather evident. By going from specific to more general concepts, we abstract from irrelevant details. Following hyponymy relations means moving in the opposite direction, explaining why hyponymy merely received a weight of 0.3. Derived word forms (e.g. nominalizations) are linked in WordNet by derivation relations. Such derived forms are obviously semantically very close. Meronymic relations are most useful when moving from part to whole (holonymy). Instance relations connect

individuals such as the Amazon River with general concepts such as '*river*', in this case, and thus are obviously very useful.

Table 7.4: Relation type weights chosen for final tests

relation type	**weight** β_r
hypernymy	0.9
hyponymy	0.3
derivation	0.5
derived	0.7
participle	0.7
member holonymy	0.45
part holonymy	0.45
substance holonymy	0.5
meronymy	0.0
similarity	0.4
pertainymy	0.0
instance	0.6
other types	0.0
$w_{\min}$ threshold	0.3

7.3 Test Results

Having fine-tuned the relation type weights, feature selection as well as other parameters using the validation set, we then carried out the main tests. Table 7.6 displays the results obtained for all three datasets using Support Vector Machines, while Table 7.7 shows the corresponding results for the Voted-Perceptron algorithm. Results obtained with Adaptive Boosting are shown in Table 7.8.

First of all, we noticed that using the conventional bag-of-words method without any translation whatsoever (T method) worked remarkably well for the English/Spanish setup, probably to a large part due to named entities, which occur particularly often in news articles, and because of the relatedness of the two languages. Nonetheless, the error rates delivered are likely to be unsatisfactory for production use and similar results cannot be expected for arbitrary language pairs. For Japanese, in fact, this method

Table 7.5: Overview of approaches tested

abbreviation	description
T	conventional bag-of-words term approach without translations
CM	simple concept mapping approach without weight propagation
ORM	Ontology Region Mapping
ORM+BT	Ontology Region Mapping combined with bag-of-words representation of Babel Fish translations
DT	dictionary translations used with bag-of-words method
DCM/DORM	same as CM/ORM but using dictionary translations as input to ontology mapping
BT	Babel Fish translations used with bag-of-words method
BCM/BORM	same as CM/ORM but using Babel Fish translations as input to ontology mapping
BCM+BT	same as CM but using Babel Fish translations as input and combined with terms from BT
BORM+BT	same as ORM but using Babel Fish translations as input and combined with BT
BDORM+BDT	both Babel Fish and dictionary translations used for Ontology Region Mapping as well as for bag-of-words representation

could not be used directly because it would require the use of advanced tokenization heuristics. As expected, the translation approach (DT and BT) leads to significant improvements compared to T, however, using simple dictionary translations without word sense disambiguation (DT, DCM, DORM) generally yields inferior results compared to approaches using higher quality translations (BT, BCM, BORM, etc.) based on word sense disambiguation.

As predicted in our analysis in Chapter 4, a pure concept mapping approach without propagation (CM, DCM, BCM) suffers from serious drawbacks, so it is not surprising that such solutions are unable to generally outperform translation-based approaches. Ontology Region Mapping (ORM, DORM, BORM), in nearly all cases, has a very positive impact on the performance compared to a simple concept mapping approach without propagation. Several reasons for this were given in Section 4.3.3. The results, however, depend to a large extent on the ontology employed. Combining the English and Spanish WordNet versions (CM and ORM) did not always produce convincing results. Instead, we found that we can obtain high levels of accuracy by using Ontology Region Mapping with translations (BORM) to improve on the results obtained for BT, even more so by including both the concepts and the terms in the final representation (BORM+BT). This is a positive result, as it implies that it suffices to use the freely available English WordNet combined with translation software that also tends to be more available than multilingual lexical resources. For instance, despite the non-existence of a Japanese version of WordNet, we were able to perform multilingual text classification for Japanese documents. When necessary, one can yield even further improvements by using multiple translations as input and thus effectively combining four different representations as explained in Section 5.7 (BDORM+BDT).

When thematically classifying news articles and encyclopedic entries, outperforming the translation approach BT is a rather difficult task because introduction paragraphs exist with terms that have a lot of discriminatory power. In these cases, our methods delivered superior results that are significant from a statistical perspective. Geographical references, in contrast, are not always explicit and the classes are thus rather heterogeneous, so the use of our methods pays off even more. Given that the relation type weights were tuned with respect to the Reuters topic-based validation set, we can presume that sets of weights exist that lead to even better results than the ones indicated.

A similar situation holds with respect to the learning algorithms tested. Although we used the same general parameters for all algorithms (with the exception of the feature selection setup) rather than tuning in accordance with the demands of particular ones,

very encouraging results were also achieved with the Voted-Perceptron and Adaptive Boosting algorithms.

Further performance boosts could be made by improving one of the following components.

1. more accurate part-of-speech tagging
2. more accurate lemmatization
3. more accurate word sense disambiguation
4. greater coverage of concepts by the ontology
5. greater coverage of terms by the ontology mapping
6. better use of ontology mapping (e.g. by recognizing multiple-word expressions)
7. more adequate relations in ontology (better background knowledge)
8. better choice of relation weighting parameters
9. optimal learning algorithm setup

In summary, our experimental evaluation confirmed our intuitions about the nature of multilingual text classification made in our linguistic analysis in Chapter 4 as well as the decisions resulting from them that are described in Chapter 4 and led to our novel solution. Ontology Region Mapping not only improves on the simple concept mapping approach but also manages to deliver superior results compared to the approach of using translations with the bag-of-words model.

Table 7.6: Main test results for the Support Vector Machine algorithm tested on the Reuters English-Spanish (Topics and Geography classes) and Wikipedia English-Japanese datasets using the approaches explained in Table 7.5 (micro-averaged F_1 scores in %, average error rates in % with 95% confidence intervals).

	Reuters Spanish			
	Topics		**Geography**	
	F_1	error rate	F_1	error rate
T	80.97	18.61 ±0.30	81.86	18.12 ±0.30
CM	89.23	10.49 ±0.24	85.74	14.58 ±0.28
ORM	89.53	10.36 ±0.24	87.33	12.97 ±0.26
ORM+BT	91.88	8.04 ±0.21	91.92	8.22 ±0.21
DT	86.45	13.17 ±0.26	86.14	14.24 ±0.27
DCM	86.74	13.07 ±0.26	89.38	10.98 ±0.24
DORM	88.01	11.75 ±0.25	91.82	8.46 ±0.22
BT	90.96	8.80 ±0.22	88.76	11.43 ±0.25
BCM	90.75	9.06 ±0.22	91.12	9.16 ±0.23
BORM	91.12	8.74 ±0.22	93.89	6.28 ±0.19
BCM+BT	91.50	8.30 ±0.22	90.12	10.08 ±0.24
BORM+BT	92.46	7.43 ±0.20	94.44	5.68 ±0.18
BDORM+BDT	92.47	7.40 ±0.20	94.83	5.29 ±0.17

	Wikipedia Japanese	
	F_1	error rate
BT	86.26	14.00 ±0.38
BCM	85.38	15.10 ±0.40
BORM	86.67	13.52 ±0.38
BCM+BT	86.41	13.91 ±0.38
BORM+BT	87.29	12.86 ±0.37

Table 7.7: Main test results for the Voted-Perceptron algorithm using the same three datasets and the same setups as in the case of our tests with SVM classifiers in Table 7.6.

	Reuters Spanish			
	Topics		**Geography**	
	F_1	error rate	F_1	error rate
T	73.96	25.30 ±0.34	75.97	23.89 ±0.33
CM	84.20	15.60 ±0.28	83.78	16.57 ±0.29
ORM	85.81	13.84 ±0.27	86.27	13.87 ±0.27
ORM+BT	89.17	10.71 ±0.24	89.74	10.47 ±0.24
DT	83.01	16.67 ±0.29	89.03	11.02 ±0.24
DCM	81.98	17.88 ±0.30	90.34	9.91 ±0.23
DORM	85.13	14.65 ±0.28	90.33	10.01 ±0.23
BT	87.80	12.15 ±0.26	91.70	8.35 ±0.22
BCM	85.93	13.81 ±0.27	92.85	7.40 ±0.20
BORM	88.03	11.78 ±0.25	92.58	7.62 ±0.21
BCM+BT	88.31	11.53 ±0.25	92.02	8.10 ±0.21
BORM+BT	89.37	10.49 ±0.24	92.83	7.34 ±0.20
BDORM+BDT	90.02	9.77 ±0.23	93.63	6.54 ±0.19

	Wikipedia Japanese	
	F_1	error rate
BT	80.28	20.46 ±0.45
BCM	78.29	22.83 ±0.46
BORM	84.06	16.22 ±0.41
BCM+BT	81.22	19.39 ±0.44
BORM+BT	83.86	16.23 ±0.41

Table 7.8: Main test results for the AdaBoost algorithm using the same three datasets and the same setups as in the case of our tests with SVM classifiers in Table 7.6.

	Reuters Spanish			
	Topics		**Geography**	
	F_1	error rate	F_1	error rate
T	65.10	39.58 ±0.38	61.22	32.27 ±0.37
CM	74.41	24.94 ±0.34	79.71	18.54 ±0.30
ORM	78.23	21.71 ±0.32	78.35	20.28 ±0.31
ORM+BT	80.72	19.15 ±0.31	83.24	15.73 ±0.28
DT	76.77	24.22 ±0.33	93.66	6.24 ±0.19
DCM	75.37	25.13 ±0.34	93.39	6.47 ±0.19
DORM	81.37	18.46 ±0.30	94.34	5.67 ±0.18
BT	82.02	18.18 ±0.30	95.99	3.97 ±0.15
BCM	79.52	20.15 ±0.31	95.47	4.45 ±0.16
BORM	84.35	15.47 ±0.28	96.53	3.47 ±0.14
BCM+BT	81.00	18.77 ±0.30	96.21	3.76 ±0.15
BORM+BT	83.44	16.47 ±0.29	96.91	3.08 ±0.13
BDORM+BDT	85.08	14.86 ±0.28	96.93	3.07 ±0.13

	Wikipedia Japanese	
	F_1	error rate
BT	76.70	22.58 ±0.46
BCM	72.54	26.59 ±0.49
BORM	79.90	20.12 ±0.44
BCM+BT	77.69	21.79 ±0.46
BORM+BT	82.97	17.00 ±0.41

Chapter 8

Conclusions and Future Work

In the past, studies have been conducted with the goal of using natural language processing techniques in order to improve the performance of monolingual text classification. Many of these attempts have failed to deliver convincingly superior results (Sebastiani, 2002). An analysis of some of the drawbacks of simply mapping terms to concepts led us to a novel approach called Ontology Region Mapping where related concepts, too, are taken into consideration when mapping from terms to concepts, thus allowing us to exploit additional background knowledge. We have explained why this approach is particularly useful in multilingual settings, and our experimental evaluation confirmed our intuitions.

Our new algorithm requires additional resources, however, most of them are freely available, most notably Princeton WordNet for the English language. Multilingual ontology mappings, unfortunately, are not that readily available but this issue is alleviated by the fact that one can instead use translations in conjunction with Princeton WordNet.

Nevertheless, especially considering that multilingual ontologies are hard to come by, in the future we would also like to devise strategies for arriving at an ontology with mappings for multiple languages that better reflects the semantic relatedness between concepts than WordNet. Kilgarriff and Yallop (2000), for instance, point out that senses that are very closely related from a semantic perspective, often are not closely linked in Princeton WordNet, where the main focus lies on ontological relations.

In conjunction with such an endeavour, we would also like to evaluate the effects of integrating and using background knowledge more rigorously. The underlying idea is that in order to classify certain documents very specific background knowledge might be required, e.g. knowing that a person used to be a famous sports star might be the only way of knowing that an article needs to be categorized as being sports-related. Similarly,

knowing that Álamos is a town in Mexico and that Almendralejo is a small town in Spain might be the only way of knowing that one document is Mexico-related and another is Spain-related rather than vice versa. For our research, this might also mean using a more powerful concept description language and ontology interface, perhaps one that has inference capabilities. It would make sense to attempt to identify more complex concepts, for instance using language-specific noun phrase parsing.

Finally, it could be explored how well the feature weighting schemes perform when used for multilingual information retrieval and multilingual text clustering. The latter requires an unsupervised machine learning technique that usually involves some kind of similarity measure. Since such similarity measures are unlikely to ignore irrelevant feature dimensions in the same way as support vector machines and other classifying algorithms are able to, we might be required to adapt our weighting schemes such that the number of features is restricted to a further extent.

Indeed, we believe our feature weighting approach or extensions of it to have a wide range of interesting applications, in multilingual as well as monolingual settings, because it captures the general meaning of a text more adequately than mainstream weighting schemes.

Bibliography

AltaVista (2006). Babel Fish Translation. URL `http://babelfish.altavista.com/`.

ANSI/NIZO (2005). *ANSI/NISO Z39.19-2005. Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies*. NISO Press, Bethesda, MD, USA.

Baeza-Yates, Ricardo A. and Ribeiro-Neto, Berthier A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

Beesley, K. R. (1988). Language identifier: A computer program for automatic natural-language identification on on-line text. In: *Proceedings of the 29th Annual Conference of the American Translators Association*, pp. 47–54.

Bel, Nuria, Koster, Cornelis H., and Villegas, Marta (2003). Cross-lingual text categorization. In: Traugott Koch and Ingeborg Torvik Sølvberg (Eds.), *Proceedings of ECDL-03, 7th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 126–139. Springer Verlag, Heidelberg, Germany, Trondheim, Norway. Published in the "Lecture Notes in Computer Science" series, number 2769.

Bloehdorn, Stephan and Hotho, Andreas (2004). Boosting for Text Classification with Semantic Features. In: *Proceedings of the Workshop on Mining for and from the Semantic Web at the 10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2004)*, pp. 70–87.

Brank, Janez, Grobelnik, Marko, Milic-Frayling, Natasa, and Mladenic., Dunja (2002). Interaction of Feature Selection Methods and Linear Classification Models.

Brause, Rüdiger (1999). *Neuronale Netze. Eine Einführung in die Neuroinformatik.* Teubner Verlag, Stuttgart, Germany.

Buckley, Chris (1985). Implementation of the SMART Information Retrieval System. Technical report, Cornell University, Ithaca, NY, USA.

de Buenaga Rodríguez, Manuel, Gómez-Hidalgo, José María, and Díaz-Agudo, Belén (1997). Using WordNet to Complement Training Information in Text Categorization. In: Ruslan Milkov, Nicolas Nicolov, and Nilokai Nikolov (Eds.), *Proceedings of RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing*. Tzigov Chark, Bulgaria.

Cleverdon, Cyril, Optimizing convenient online access to bibliographic databases. *Information Services and Use*, volume 4(1-2): (1984) pp. 37–47.

Cortes, Corinna and Vapnik, Vladimir, Support-Vector Networks. *Machine Learning*, volume 20(3): (1995) pp. 273–297.

Daudé, Jordi, Padró, Lluís, and Rigau, German (2003). Making Wordnet Mappings Robust. In: *Proc. Congreso de la Sociedad Española para el Procesamiento del Lenguage Natural (SEPLN'03)*. Universidad Universidad de Alcala de Henares, Madrid, Spain.

Davis, Mark and Dür, Martin (2005). Unicode Standard Annex #15. Unicode Normalization Forms. Technical report, Unicode 4.1.0. URL `http://www.unicode.org/reports/tr15/tr15-25.html`.

Dumais, Susan T., Letsche, Todd A., Littman, Michael L., and Landauer, Thomas K. (1997). Automatic cross-language retrieval using latent semantic indexing. In: *AAAI Symposium on CrossLanguage Text and Speech Retrieval*. American Association for Artificial Intelligence.

Farreres, Xavier, Rigau, German, and Rodríguez, Horacio (1998). Using WordNet for Building WordNets. In: Sanda Harabagiu (Ed.), *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pp. 65–72. Association for Computational Linguistics, Somerset, NJ, USA.

Fellbaum, Christiane (Ed.) (1998). *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Fox, Christopher, A stop list for general text. *SIGIR Forum*, volume 24(1-2): (1990) pp. 19–21.

Freund, Y. and Schapire, Robert E., A decision–theoretic generalization of on–line learning and an application to boosting. *Journal of Computer and System Sciences*, volume 55(1): (1997) pp. 119–139.

Freund, Yoav and Schapire, Robert E. (1998). Large margin classification using the perceptron algorithm. In: *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pp. 209–217. ACM Press, New York, NY, USA.

Freund, Yoav and Schapire, Robert E., A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, volume 14(5): (1999) pp. 771–780.

Gabrilovich, Evgeniy and Markovitch, Shaul (2005). Feature Generation for Text Categorization Using World Knowledge. In: *Proceedings of The Nineteenth International Joint Conference for Artificial Intelligence*, pp. 1048–1053. Edinburgh, Scotland, UK.

Gale, William A., Church, Kenneth W., and Yarowsky, David (1992). One sense per discourse. In: *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pp. 233–237. Association for Computational Linguistics, Morristown, NJ, USA.

García Adeva, Juan José, Calvo, Rafael A., and de Ipiña, Diego López, Multilingual Approaches to Text Categorisation. *The European Journal for the Informatics Professional*, volume VI(3): (2005) pp. 43 – 51.

Genesereth, Michael R. and Nilsson, Nils J. (1987). *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Gliozzo, Alfio Massimiliano and Strapparava, Carlo (2005). Cross Language Text Categorization by acquiring Multilingual Domain Models from Comparable Corpora. In: *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pp. 9–16. Ann Arbor, MI, USA.

Goldsmith, John, Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, volume 27(2): (2001) pp. 153–198.

Grefenstette, Gregory and Segond, Frédérique (2003). Multilingual On-Line Natural Language Processing. In: Ruslan Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics*. Oxford University Press, New York, NY, USA.

Grefenstette, Gregory and Tapanainen, Pasi (1994). What is a word, what is a sentence? Problems of tokenization. In: *The 3rd International Conference on Computational Lexicography*, pp. 79–87. Budapest, Hungary.

Gruber, Thomas R., A translation approach to portable ontology specifications. *Knowledge Acquisition*, volume 5(2): (1993) pp. 199–220.

Hjelmslev, Louis (1943). *Omkring sprogteoriens grundlaeggelse*. Akademisk Forlag, København, Denmark.

Hutchins, John (2003). Machine Translation: General Overview. In: Ruslan Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics*. Oxford University Press, New York, NY, USA.

Ifrim, Georgiana, Theobald, Martin, and Weikum, Gerhard (2005). Learning Word-to-Concept Mappings for Automatic Text Classification. In: Luc De Raedt and Stefan Wrobel (Eds.), *Proceedings of the 22nd International Conference on Machine Learning - Learning in Web Search (LWS 2005)*, pp. 18–26. ICMLW4-LWS2005, Bonn, Germany.

Jalam, Radwan (2003). *Apprentissage automatique et catégorisation de textes multilingues*. Ph.D. thesis, Université Lumière Lyon 2, Lyon, France.

Joachims, Thorsten (1998). Text categorization with support vector machines: learning with many relevant features. In: Claire Nédellec and Céline Rouveirol (Eds.), *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398 in Lecture Notes in Computer Science, pp. 137–142. Springer Verlag, Heidelberg, Germany.

Joachims, Thorsten (1999). Making large-scale support vector machine learning practical. In: A. Smola B. Schölkopf, C. Burges (Ed.), *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, USA.

Kikui, G. (1996). Identifying the Coding System and Language of On-line Documents on the Internet. In: *Proceedings of the Sixteenth International Conference on Computational Linguistics: COLING'96*. Copenhagen, Denmark.

Kilgarriff, Adam (1996). Word senses are not bona fide objects: implications for cognitive science, formal semantics, NLP. In: *Proceedings of the 5th International Conference on the Cognitive Science of Natural Language Processing*, pp. 193–200. Dublin, Ireland.

Kilgarriff, Adam and Yallop, Colin (2000). What's in a thesaurus. Technical Report ITRI-00-28, Information Technology Research Institute, University of Brighton. Also published in Proc. of the Second Conference on Language Resources and Evaluation, pp. 1371-1379.

Kipfer, Barbara Ann (2006). *Roget's New Millennium Thesaurus, First Edition (v 1.3.1)*. Lexico Publishing Group.

Lauser, Boris and Hotho, Andreas (2003). Automatic multi-label subject indexing in a multilingual environment. In: *Proc. of the 7th European Conference in Research and Advanced Technology for Digital Libraries, ECDL 2003*, volume 2769, pp. 140–151. Springer.

Lewis, David D., Yang, Yiming, Rose, Tony G., and Li, Fan, RCV1: A New Benchmark Collection for Text Categorization Research. *The Journal of Machine Learning Research*, volume 5: (2004) pp. 361–397.

Loukachevitch, Natalia V. (1997). Knowledge Representation for Multilingual Text Categorization. In: *AAAI Symposium on CrossLanguage Text and Speech Retrieval, AAAI Technical Report.*

Luhn, Hans Peter, The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, volume 2(2): (1958) pp. 159–165.

Mathieu, Benoit, Besançon, Romaric, and Fluhr, Christian (2004). Multilingual Document Clusters Discovery. In: *RIAO 2004, Avignon, France.*

McNamara, Timothy P. (2005). *Semantic Priming. Perspectives from Memory and Word Recognition.* Essays in Cognitive Psychology. Psychology Press, New York, USA.

Meyer, D.E. and Schvaneveldt, R.W., Facilitation in recognition pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology*, volume 90: (1971) pp. 227–234.

Mitchell, Tom M. (1997). *Machine Learning.* McGraw-Hill, New York, NY, USA.

Moschitti, Alessandro and Basili, Roberto (2004). Complex Linguistic Features for Text Classification: A Comprehensive Study. In: Sharon McDonald and John Tait (Eds.), *Proceedings of ECIR-04, 26th European Conference on Information Retrieval Research*, volume 2997 of *Lecture Notes in Computer Science*, pp. 181–196. Springer Verlag, Heidelberg, Germany, Sunderland, UK.

Muslea, Ion (2002). *Active Learning with Multiple Views.* Ph.D. thesis, Department of Computer Science, University of Southern California.

Neely, J. H., Semantic priming and retrieval from lexical memory: Roles of inhibitionless spreading activation and limited-capacity attention. *Journal of Experimental Psychology: General*, volume 106: (1977) pp. 22–254.

Oard, Douglas W. and Dorr, Bonnie J. (1996). A survey of multilingual text retrieval. Technical report, University of Maryland at College Park, College Park, MD, USA.

Olsson, J. Scott, Oard, Douglas W., and Hajič, Jan (2005). Cross-language text classification. In: *Proc. ACM SIGIR Conference on Research and Development in Information Retrieval 2005*, pp. 645–646. ACM Press, New York, NY, USA.

Paice, Chris D., Method for evaluation of stemming algorithms based on error counting. *Journal of the American Society for Information Science*, volume 47(8): (1996) pp. 632–649.

Palmer, David D. and Burger, John D. (1997). Chinese Word Segmentation and Information Retrieval.

Peters, Carol, Clough, Paul, Gonzalo, Julio, Jones, Gareth J. F., Kluck, Michael, and Magnini, Bernardo (Eds.) (2005). *Multilingual Information Access for Text, Speech and Images, 5th Workshop of the Cross-Language Evaluation Forum, CLEF 2004, Bath, UK, September 15-17, 2004, Revised Selected Papers*, volume 3491 of *Lecture Notes in Computer Science*. Springer.

Porter, Martin F., An algorithm for suffix stripping. *Program*, volume 14(3): (1980) pp. 130–137.

Porter, Martin F. (2001). Snowball: A language for stemming algorithms. URL `http://snowball.tartarus.org/texts/introduction.html`. (accessed June 2006).

Quillian, R. (1968). Semantic Memory. In: Marvin Minsky (Ed.), *Semantic Information Processing*, pp. 216–270. MIT Press.

Ramakrishnanan, G. and Bhattacharyya, P., Text Representation with WordNet Synsets Using Soft Sense Disambiguation. *Ingénierie des systèmes d'information*, volume 8(3): (2003) pp. 55–70.

Reuters (2000a). Reuters Corpus, Volume 1: English Language, 1996-08-20 to 1997-08-19. URL `http://trec.nist.gov/data/reuters/reuters.html`.

Reuters (2000b). Reuters Corpus, Volume 2, Multilingual Corpus, 1996-08-20 to 1997-08-19. URL `http://trec.nist.gov/data/reuters/reuters.html`.

Rigutini, Leonardo, Maggini, Marco, and Liu, Bing (2005). An EM Based Training Algorithm for Cross-Language Text Categorization. In: *WI '05: Proceedings of the*

The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), pp. 529–535. IEEE Computer Society, Washington, DC, USA.

van Rijsbergen, Cornelis Joost (1979). *Information Retrieval (2nd Ed.)*. Butterworth, London, UK.

Rosenblatt, Frank, The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, volume 65: (1958) p. 386–408.

Rosso, Paolo, Ferretti, Edgardo, Jiménez, Daniel, and Vidal, Vicente (2004). Text Categorization and Information Retrieval Using WordNet Senses. In: Petr Sojka, Karel Pala, Pavel Smrž, Christiane Fellbaum, and Piek Vossen (Eds.), *Proceedings of the Second International WordNet Conference (GWC 2004)*, pp. 299–304. Masaryk University, Brno, Czech Republic.

Russell, Stuart and Norvig, Peter (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, USA.

Röder, Jens (2002). The Magic-Dictionary Magic-Dic. URL `http://magic-dic.homeunix.net/`.

Salton, Gerard (1969). Automatic processing of foreign language documents. In: *Proceedings of the 1969 Conference on Computational Linguistics*, pp. 1–28. Association for Computational Linguistics, Morristown, NJ, USA.

Salton, Gerard and Buckley, Christopher, Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, volume 24(5): (1988) pp. 513–523.

Savoy, Jacques (2005). CLEF and Multilingual Information Retrieval. URL `http://www.unine.ch/info/clef/`.

Schmid, Helmut (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In: *International Conference on New Methods in Language Processing*. unknown, Manchester, UK.

Schmidt-Schauss, M. and Smolka, G., Attributive concept descriptions with complements. *Artificial Intelligence*, volume 48: (1991) pp. 1–26.

Scott, Sam and Matwin, Stan (1998). Text Classification Using WordNet Hypernyms. In: Sanda Harabagiu (Ed.), *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pp. 38–44. Association for Computational Linguistics, Somerset, NJ, USA.

Sebastiani, Fabrizio, Machine learning in automated text categorization. *ACM Computing Surveys*, volume 34(1): (2002) pp. 1–47.

Sebastiani, Fabrizio (2006). Classification of text, automatic. In: Keith Brown (Ed.), *The Encyclopedia of Language and Linguistics*, volume 2, pp. 457–463. Elsevier Science Publishers, Amsterdam, Netherlands, second edition.

Siersdorfer, Stefan (2005). *Combination Methods for Automatic Document Organization*. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany.

Singhal, Amit, Buckley, Chris, and Mitra, Mandar (1996). Pivoted document length normalization. In: *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 21–29. ACM Press, New York, NY, USA.

Soergel, Dagobert (1997). Multilingual thesauri in cross-language text and speech retrieval. In: D. Hull and D. Oard (Eds.), *AAAI Symposium on Cross-Language Text and Speech Retrieval*. Menlo Park, CA, USA.

Sproat, Richard, Shih, Chilin, Gale, William, and Chang, Nancy (1994). A Stochastic Finite-State Word-Segmentation Algorithm for Chinese. In: *Meeting of the Association for Computational Linguistics*, pp. 66–73.

Spärck Jones, Karen, A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation*, volume 28(1): (1972) pp. 11–21.

Steinberger, Ralf and Pouliquen, Bruno (2003). Cross-lingual Indexing. Final Report for the IPSC Exploratory Research Project. Technical report, JRC Internal Note.

Theobald, Martin, Schenkel, Ralf, and Weikum, Gerhard (2003). Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data. In: Vassilis Christophides and Juliana Freire (Eds.), *6th International Workshop on the Web and Databases (WebDB-03)*, pp. 1–6. OGI School of Science and Engineering / CSE, San Diego, CA, USA.

Vapnik, Vladimir N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Verdejo, Felisa, Gonzalo, Julio, Peñas, Anselmo, López-Ostenero, Fernando, and Fernández-Amorós, David (2000). Evaluating Wordnets in Cross-language Text Retrieval: the ITEM multilingual search engine. In: *Proceedings LREC 2000*.

Wiener, Erik D., Pedersen, Jan O., and Weigend, Andreas S. (1995). A neural network approach to topic spotting. In: *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pp. 317–332. Las Vegas, NV, USA.

Wikimedia Foundation (2006). Wikipedia. URL `http://www.wikipedia.org/`.

Wikipedia (2006). Synthetic language. URL `http://en.wikipedia.org/w/index.php?title=Synthetic_language&oldid=93062137`. (accessed 2006-12-16).

Wittgenstein, Ludwig (1953). *Philosophical Investigations*. Blackwell, Oxford.

Witten, Ian H. and Frank, Eibe (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann, San Francisco, CA, USA.

Wolff, Ekkehard (2000). Language and Society. In: Bernd Heine and Derek Nurse (Eds.), *African Languages - An Introduction*, pp. 298–347. Cambridge University Press, Cambridge, MA, USA.

Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In: *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 42–49. Berkeley, CA, USA.

Yang, Yiming and Pedersen, Jan O. (1997). A comparative study on feature selection in text categorization. In: Douglas H. Fisher (Ed.), *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pp. 412–420. Morgan Kaufmann Publishers, Nashville, TN, USA.

Zhu, Xiaojin (2005). Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison. Http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.

Index