AALTO UNIVERSITY
SCHOOL OF SCIENCE
Degree Programme of Information Networks

Katariina Nyberg

Document Classification Using Machine Learning and Ontologies

Master's Thesis
Espoo, January 31, 2011

Supervisor:	Professor Eero Hyvönen, Aalto University
Instructor:	Professor Eero Hyvönen, Aalto University

AALTO UNIVERSITY
SCHOOL OF SCIENCE
Degree Programme of Information Networks

ABSTRACT OF
MASTER'S THESIS

Author:	Katariina Nyberg		
Title of thesis:			
Document Classification Using Machine Learning and Ontologies			
Date:	January 31, 2011	**Pages:**	9 + 71
Professorship:	Mediatechnology	**Code:**	T-75
Supervisor:	Professor Eero Hyvönen		
Instructor:	Professor Eero Hyvönen		

This master's thesis explores a way in which documents can be automatically classified based on their contents. Automatic classification of data is one of the main applications of machine learning. With the help of already classified data a model for the most likely class can be learned.

Whether adding background knowledge from ontologies can be added to the model in order to improve the classification accuracy, is also explored in this master's thesis. A new machine learning model is introduced that incorporates ontology information.

The proposed method for learning a classification model and enhancing it with ontology information is used in a case study for the Finnish National Archives and a set of digital documents that have been manually classified.

An RDF schema for representing documents, sentences and words is created in order to prepare tha data for the machine learning analysis. The words are put into base form and matched semi-automatically with concepts of the General Finnish Ontology YSO. Then the ontology enhanced model is applied on the data and the most likely classes for documents are learned.

The master's thesis shows that the classification accuracy of the model increases when ontology information is added to it.

Keywords: document classification, ontologies, syntactical analysis, machine learning, logistic discriminant, bag of words, YSO
Language: English

AALTO YLIOPISTO
PERUSTIETEIDEN KORKEAKOULU
Informaatioverkostojen koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Katariina Nyberg		
Työn nimi:			
Asiakirjojen luokittelu koneoppimista ja ontologioita käyttäen			
Päiväys:	31. tammikuuta 2011	**Sivumäärä:**	9 + 71
Professuuri:	Mediateknologia	**Koodi:**	T-75
Työn valvoja:	Professori Eero Hyvönen		
Työn ohjaaja:	Professori Eero Hyvönen		

Tässä diplomityössä tutkitaan asiakirjojen automaattista luokittelua niiden sisällön pohjalta. Tiedon automaattinen luokittelu on yksi koneoppimisen keskeisiä aihepiirejä. Oppivasta luokittimesta luodaan malli jo valmiiksi luokitetulla esimerkkidatalla.

Tehtävänä on kokeilla ontologisen taustatiedon hyödyntämistä oppivassa luokittimessa ja selvittää parantaako taustatiedon lisääminen mallin luokittelutarkkuutta. Diplomityö esittelee uuden oppivan luokittimen, joka sisällyttää ontologiatiedon analyysiinsä. Luokitinta testataan Suomen Kansallisarkiston sähköisillä asiakirjoilla, jotka ovat käsin luokiteltuja.

Asiakirjojen ja niiden sisältämien lauseiden sekä sanojen esittämistä varten diplomityössä on kehitetty RDF skeema, jota käyttäen sanat voidaan muuttaa perusmuotoon ja yhdistää puoliautomaattisesti Yleisen suomalaisen ontologian käsitteisiin. Skeemaa hyödynnetään datan valmisteluun oppivan luokittimen analyysia varten.

Diplomityössä on osoitettu, että luokittelutarkkuus paranee, kun oppivaan luokittimeen lisätään ontologiatietoa.

Avainsanat:	asiakirjojen luokittelu, ontologiat, kieliopillinen analyysi, koneoppiminen, logistinen diskriminantti, YSO
Kieli:	englanti

Acknowledgements

This work was done as a collaboration between the Semantic Computing Research Group and the research group Bayesian Algorithms for Latent Variable Models. I thank Dr. Tapani Raiko and Teemu Tiinanen from the latter research group for their collaboration and hope that in the future our research teams will work closely together on numerous other projects, where the challenges of the semantic web and machine learning meet.

The project was financed by the FinnONTO 2.0 project funded mostly by Tekes and the Academy of Finland project 'Bayesian Machine Learning Algorithms for Discovering Relations with Latent Variable Models'. My thanks extend to Päivi Happonen and Jaana Pohjonen from the Finnish National Archives, who provided the case study problem with datasets and help.

I would like to thank Professor Eero Hyvönen for guiding me on this master's thesis. I also thank my colleagues and especially the researcher I share my work room with, Matias Frosterus, for answering my numerous questions on the English language and letting me think out loud to him. Special thanks extend to Jessica Priestley who agreed to read through my thesis and make helpful suggestions on the English language.

My thanks also extend to my mother who has with her example given me faith to myself as a student of Information Technology.

Espoo January 31, 2011

Katariina Nyberg

Abbreviations and Acronyms

BT	Broader Term
CSV	Comma Separated Values
MIME	Multipurpose Internet Mail Extensions
NT	Narrower Term
OCR	Optical Character Recognition
PCA	Principal Component Analysis
PDF	Portable Document Format
RDF	Resource Description Framework
RT	Related Term
RTF	Rich Text Format
TIFF	Tagged Image File Format
TTL	Terse RDF Triple Language
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Universal Resource Locator
WSD	Word Sense Disambiguation
XML	Extensible Mark-Up Language
YSA	Yleinen Suomalainen Asiasanasto (in English: General Finnish Thesaurus)
YSO	Yleinen Suomalainen Ontologia (in English: General Finnish Ontology)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The digital handling of documents represents the present for many organisations: documents and letters are written, sent and stored in digital form. From an ecological and practical point of view it makes sense to store digital documents digitally and not to print them out and put them into a folder. The cost of digital storage space decreases as more bits fit to a smaller space on a hard disk, as noted by Mark Kryder [Wal05]. Thus digital storing makes also economically more sense than physical storing.

A convenient way of storing also creates the need for a convenient way of retrieval: what is the use of storing documents if they cannot be found? Naturally categorisation of documents has been used to make it easier to find relevant information.

This master's thesis is part of the FinnONTO 2.0 project[1] of the Semantic Computing Research Group (SeCo) of the Aalto University School of Science and Technology. The Finnish National Archives provided the project with a set of documents and wanted to know how the documents could be classified automatically. The work was divided between the University of Tampere in the following way: the researchers in Tampere would try to find a way to categorise the documents by the context and the metadata of the documents and SeCo would concentrate on exploring the ways in which documents could automatically be classified based on their contents, that is to say the words of the document.

The benefit of digital documents is that they can be computationally anal-

[1]FinnONTO 2.0: `http://www.seco.tkk.fi/projects/finnonto/`

ysed, because a computer programme can extract the document text and process it for further analysis. Text is for the computer programme only a string representation without any meaning, therefore it may be of use to add knowledge on the meaning of words, that is to say ontology information, to the text.

It was proposed that the benefits of machine learning in automatic classification be explored to see, whether adding ontology information could improve those classification results. In order to combine the best of both worlds, machine learning and ontology knowledge, a collaboration with the research group Bayesian Algorithms for Latent Variable Models at the Aalto University was proposed. The result of this collaboration was a machine learning model based on logistic discrimination that extended the bag of words representation with ontology information using binary matrices.

The text from the digital documents was read and parsed into RDF form. An RDF schema was developed to map the relations of all terms, sentences and documents. The individual words were matched with concepts from YSO, the General Finnish Ontology, and a programme was written, that turned the resulting RDF file into a two dimensional table, that could be used for the purpose of training, validating and testing the proposed machine learning model. Dr. Tapani Raiko and Teemu Tiinanen from the research group Bayesian Algorithms for Latent Variable Models developed the model to be used for the learning of the documents' classes, applied, trained, validated and tested it.

Chapter 2

Machine Learning

This chapter provides a basis for understanding machine learning (ML) and how it can be used to learn the classification of documents. When example data exists, supervised learning can be applied for creating a classifier model for documents. This chapter discusses two different ways for learning the model and provides an outlook on different ways of dimensionality reduction and visualisation.

2.1 Supervised Learning

In his book, Introduction to Machine Learning, Ethem Alpaydin describes Machine Learning as:

> programming computers to optimise a performance criterion using example data or past experience. [Alp04, p. xxv]

ML is applied in cases where a programmer cannot explicitly tell the computer programme what to do and what steps to take. In supervised ML the idea is to show results and variables that might have lead to the results to a programme and hope that the examples are good enough and that there are enough of them to teach the programme how the variables lead to the wanted result. In ML the aim is to obtain a general solution for yet unseen data. Of course the discipline is not based on faith alone: there are many ways to optimise the programme's learning process and ways of validating the performance. ML works like a past master who shows his apprentice how things are done, because he cannot explicitly tell the apprentice how they are done. He just knows implicitly the ways of his craft.

In supervised ML one way for the programme to learn how the results are obtained is to model, for example, the statistical dependencies of the variables and the results from the past data. When statistical dependencies are measured association rules are formed [Alp04, p. 3]. The confidence in an association from A to B is measured with the conditional probability $P(B \mid A)$:

$$P(B \mid A) = \frac{P(A,B)}{P(A)} \ . \tag{2.1}$$

For the association rule to hold the conditional probability should be quite high that is to say close to one and higher than the probability of B $P(B)$ alone [Alp04, p. 56].

One ML application is to model a regression curve from data plots. For example a polynomial curve of different orders can be used:

$$f(x) = \sum_{j=0}^{n} a_j x^j, \tag{2.2}$$

where n is the order of the polynomial. A programmer can try out models of different orders and make her programme learn the different parameters $(a_0,a_1,\ldots,a_n)$ of the polynomial curve. The parameters are learned by using a training set of data points that should represent the whole data well. The training set should be divided into two parts: the actual training set, with which the parameters are obtained and a validation set, that tests the accuracy of the learned model [Alp04, p. 35].

The most accurate model for the validation set should be chosen. Validation prevents at its best the common pitfall of choosing the model: a model is chosen that most accurately replicates the example data, but is not the best model to predict future data [Alp04, p. 33]. This phenomenon is called over-fitting and the problem with over-fitting is that a too complex or precise model might falsely create a general rule for an accidental feature or an out-lier. Manning et al. [MRS08, p. 251] give as an example a data set of documents with different topic classes, where a rare term “arachnocentric” happens to be featured only in documents of the topic class “China”. A too precise model would learn an association with the term “arachnocentric” and the topic class “China” and incorrectly classify everything with that word to the topic class “China”. Manning et al. also warn not to use too complex models, because they tend to make mistakes on noisy data – mistakes especially occur if there is not enough data [MRS08, p. 267]. While over-fitting might be a problem, under-fitting is one too. A too simple model just will not be of use, because it cannot really trace any possible patterns in the data and provide the solution [Alp04, p. 77].

2.2 Classification

In addition to regression, ML can be applied to classification problems. In supervised ML training sets are used where the example data is been classified by hand. The task of the ML programme is to model the connection of different variables to the classes they belong to.

For example ML could be used to learn whether an Email message belongs to the class spam or ham (not-spam). The SpamAssasin[1] tool is designed to recognise certain features of Email messages (such as that Viagra is mentioned and a fast delivery is promised). These features are all some sort of indications whether an Email is spam or not. With the help of an example data set of Emails that are classified by hand, the correlation between the features and the two classes, spam or ham, can be calculated exactly. This creates a general model to be used for predicting not yet classified Email messages.

In regression the idea is to model and solve a problem, where the answer is a continuous value [Alp04, p. 8] [Bis07, p. 38]. In classification the idea is to predict the group to which a to-be-classified item belongs [MRS08, p. 234]. The target is not a continuous value but a class label [Bis07, p. 38]. The classification is learned by modelling the conditional probability of the class C, $P(C \mid x_1,x_2,\ldots,x_T)$, given the variables $(x_1,x_2,..,x_T)$, which can be represented by a vector $\bar{x}$. The Bayes rule can be used to solve for $P(C \mid \bar{x})$:

$$P(A,B) = P(A \mid B) \cdot P(B) \tag{2.3}$$

[1]`http://spamassassin.apache.org/`

and

$$P(A,B) = P(B \mid A) \cdot P(A), \tag{2.4}$$

which can be written as:

$$P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A). \tag{2.5}$$

By dividing with $P(B)$ on both sides, the equation is as follows:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}. \tag{2.6}$$

Using this $P(C \mid \bar{x})$ can be written as:

$$P(C \mid \bar{x}) = \frac{p(\bar{x} \mid C)P(C)}{p(\bar{x})}, \tag{2.7}$$

where $P(C)$, called the prior probability, is the probability mass function of the class C. The probability density function of the variables $\bar{x}$ given the class C, $p(\bar{x} \mid C)$, is called the class likelihood. The denominator is the probability density function of the variables' occurrence, $p(\bar{x})$, and it is called the evidence. It is the sum of each class' $C_1,\ldots,C_K$ prior probability multiplied by the appropriate class likelihood: $\sum_{i=1}^{K} p(\bar{x} \mid C_i)P(C_i)$.

In order to choose the best class for a new unclassified $\bar{x}$, the following decision model applies:

$$\arg\max_k P(C_k \mid \bar{x}) = \arg\max_k \frac{p(\bar{x} \mid C_k)P(C_k)}{\sum_{i=1}^{K} p(\bar{x} \mid C_i)P(C_i)}, \tag{2.8}$$

and the class C for which the posterior probability is at maximum should be chosen. [Alp04, pp. 42][Bis07, p. 43]

2.2.1 Evaluation

Different classification results can be represented in confusion matrices [Alp04, p. 333] such as the general one represented in Table 2.1. The function of the matrix is to show the number of items that were classified correctly (TP and TN) and falsely (FP and FN). The matrix in Table 2.1 is a confusion matrix for a two-class classification task. The sum of the number of classified items

		Predicted class	
		Yes	No
True Class	Yes	True Positive (TP)	False Negative (FN)
	No	False Positive (FP)	True Negative (TN)

Table 2.1: Confusion matrix [Alp04, p. 333]

in every cell is the total number of the classified items. Useful measurement rates can be calculated from the numbers in the cells of the confusion matrix:

- error rate: $\frac{FN+FP}{N}$, where N is the total of the classified items
- accuracy rate: $\frac{TN+TP}{N}$, the error rate and accuracy rate sum up to 1
- precision rate: $\frac{TP}{TP+FP}$, the ratio of correctly classified items to all items classified to that class
- recall rate: $\frac{TP}{TP+FN}$, the ratio of correctly classified items to all items of that class

The error rate or the accuracy rate measures how well a model correctly classifies items. When training the model, one tries to get the accuracy rate as high as possible and chooses the model with the best accuracy on the validation set.

Precision measures the correctness of classification and recall measures its usefulness. For example in spam messages it is important that the precision rate is quite high, because a user would be annoyed if an important email were to be classified incorrectly as spam and thus left unread. It is less annoying to get some spam into one's Inbox from time to time. It is up to each classification task to decide whether it is more harmful to have False Negatives than False Positives. The decision model mentioned in Section 2.8 would have to be altered appropriately so, that if False Negatives are not as harmful as False Positives, then the decision model would prefer high precision rate over a high recall rate. This applies also the other way around: if False Positives are not as harmful as False Negatives, a high recall rate is of importance rather than a high precision rate.

2.2.2 Discriminant Function

The choice of the class depends on the variables denoted by the vector $\bar{x}$. A discriminant function can be derived using the earlier mentioned decision model (see Equation 2.8) for the maximum posterior probability. The denominator of the equation, $\sum_{i=1}^{K} p(\bar{x} \mid C_i)P(C_i)$, is the same for each class. It is only a normalising factor on which the maximum does not depend on. Thus the discriminant function can be written in the following way:

$$g_i(\bar{x}) = p(\bar{x} \mid C_i)P(C_i). \tag{2.9}$$

The product of probabilities can cause underflow in computations. To prevent this the logarithm of $g_i(\bar{x})$, which breaks up the product to a sum, can be used in search for the maximum. The discriminative function can be written accordingly [Alp04, p. 62]:

$$g_i(\bar{x}) = \log p(\bar{x} \mid C_i) + \log P(C_i), \tag{2.10}$$

and a new unseen variable $\bar{x}_n$ should be assigned to the class C_k for which $g_i(\bar{x}_n) = \arg\max_k g_k(\bar{x}_n)$ stands. [Alp04, p. 45][Bis07, p. 180]

This modification is valid, because the logarithm is a monotonic function and thus the $\bar{x}$ for which $g_i(\bar{x})$ is at maximum is the same for which the logarithm of $g_i(\bar{x})$ is at maximum.

In a classification problem the discriminant function splits the variable space into regions appropriate for each class, which are called decision regions [Alp04, p. 45]. The decision regions are separated by decision boundaries and each region might not consist of only one continuous region [Bis07, p. 39].

In this thesis two different approaches, the generative and the discriminative model for solving for the discriminant function, will be discussed in Section 2.3 and Section 2.4.

2.3 Likelihood-based Classification

In likelihood-based classification a generative model is created when solving for the Equation 2.10 by estimating the prior class probabilities $P(C_i)$ and the class likelihoods $p(\bar{x} \mid C_i)$ [Bis07, p. 43]. The advantage of the generative model is that it models missing values, data that does not belong to the training set, well [RVOK03], because it assumes a distribution for all possible variables.

The probability densities can be estimated using different kinds of distributions. Alpaydin [Alp04, p. 62] discusses three different distributions: the Bernoulli distribution for a two-class problem, the multinomial distribution for a multiple-class problem and the popular Gaussian normal distribution for continuous-valued variables.

A model is trained using the estimates of the probability distributions. Therefore it is important that the distribution of the set should be checked so that the right kind of distribution gets chosen for the estimation of the prior class probabilities $P(C_i)$ and the class likelihoods $p(\bar{x} \mid C_i)$ [Alp04, p. 72].

2.3.1 Maximum Likelihood Estimate

When choosing a distribution type it also needs to be trained using a sample data $\mathbf{X} = \{\bar{x}_d\}_{d=1}^{N}$ of documents. The parameters of the distribution could be estimated using Maximum Likelihood Estimation (MLE). This searches for the parameters θ of the distribution for which the likelihood of the samples being featured is at maximum, in other words the likelihood $p(\mathbf{X} \mid \theta)$ is maximised. The $\bar{x}_d$ in the sample are assumed independent and therefore the density function of the probability of $\mathbf{X}$ given θ can be written in the following way:

$$p(\mathbf{X} \mid \theta) = \prod_{d=1}^{N} p(\bar{x}_d \mid \theta). \tag{2.11}$$

In this particular case $\mathbf{X}$ is known. It is the sample data at hand and the parameters θ are to be estimated. The likelihood function for θ given $\mathbf{X}$ is therefore $l(\theta \mid \mathbf{X}) \equiv p(\mathbf{X} \mid \theta)$. As argued before in Section 2.2.2, the θ that maximises the likelihood function l is the same θ that maximises the logarithm of the function called the log likelihood $L(\theta \mid \mathbf{X})$ and it is defined as follows:

$$L(\theta \mid \mathbf{X}) = \sum_{d=1}^{N} \log p(\bar{x}_d \mid \theta). \text{ [Alp04, p. 62]} \tag{2.12}$$

To find the parameters θ that maximises log likelihood function, the function is derived with respect to the parameters and finds the θ for which the derivative is at zero:

$$\frac{dL(\theta \mid \mathbf{X})}{d\theta} = 0. \tag{2.13}$$

2.3.2 Maximum a Posteriori Estimate

The maximum a posteriori estimate (MAP) focuses on the posterior density of the parameters when looking at the sample

$$p(\theta \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \theta)p(\theta)}{p(\mathbf{X})}, \tag{2.14}$$

and is solved by choosing the posterior density that maximises the estimate:

$$\theta_{MAP} = \arg\max_{\theta} p(\theta \mid \mathbf{X}) \text{ [Alp04, pp. 67].} \tag{2.15}$$

2.3.3 Naive Bayes Classifier

If $p(\bar{x} \mid C_i)$ is estimated using normal Gaussian density $N(\bar{\mu}_i, \mathbf{S}_i)$ with a multivariate data set of dimension $t = 1 \ldots T$, where the variables can take real values, $\bar{x} \in \mathcal{R}^T$, the discriminant function is a quadratic function. The quadratic term comes from the different covariance matrices $\mathbf{S}_i$ for each distribution $p(\bar{x} \mid C_i)$. If the covariance matrices are presumed to be the same for all class likelihoods $\mathbf{S}_i = \mathbf{S} \; \forall \; i$, the discriminant function g_i becomes linear and is defined as follows:

$$g_i(\bar{x}) = \bar{w}_i^{\mathrm{T}} \bar{x} + w_{i0} \text{ [Alp04, p. 95],} \tag{2.16}$$

where w_i and w_{i0} are dependent on the covariance matrix $\mathbf{S}$ and the class means μ_i of the data.

The advantage of a linear discriminant function is that the decision regions are convex: if two known points of the region are connected by a line, then all points on that line belong to that region. [Alp04, pp. 92]

In the Naive Bayes (NB) Classifier the class likelihood $p(\bar{x} \mid C_i)$ is normally distributed and the covariance matrices are all presumed to be the same. Thus it is a linear discriminant model. In addition to this, the variables are presumed to be independent of each other and thus the common covariance matrix is diagonal. Computationally this is of significance because the complexity of determining the covariance matrix drops from $O(d^2)$ to $O(d)$ [Alp04, p. 96].

The NB classifier is quite popular and often mentioned in text books concerning ML. Manning et al. [MRS08] suggest using a NB classifier for a text classification problem. The class likelihoods of each term of a document d are assumed to be independent of each other and therefore they multiply up to the class likelihood of the document $p(\bar{x}_d \mid C_i)$, where $\bar{x}_d$ is the term vector of the document d. The terms of the document, $\bar{x}_d = (x_1, x_2, \ldots, x_T)$ are assumed to be independent of each other [MS00, p. 237]. This assumption is called bag-of-words and is discussed later in detail in Section 2.5.

A multinomial distribution is assumed instead of a normal Gaussian distribution for the document's terms and the class likelihood can be written in the following way:

$$P(\bar{x}_d \mid C_i) = \prod_{t=1}^{T} P(x_t \mid C_i) \tag{2.17}$$

The discriminant function (Equation 2.10) is then:

$$g_i(\bar{x}_d) = \log p(\bar{x}_d \mid C_i) + \log P(C_i) \tag{2.18}$$

and further using Equation 2.17

$$g_i(\bar{x}) = \sum_{t=1}^{T} \log P(x_t \mid C_i) + \log P(C_i) \text{ [MRS08, pp. 238].} \tag{2.19}$$

The class likelihood is approximated with a multinomial distribution of its terms and thus the MLE for the likelihood of each term belonging to a document of a certain class is the term frequency (see Section 2.5) of a term in a document $tf_{t,d}$ [MRS08, p. 107].

The estimate for the prior probability is:

$$p(C_i) = \frac{N_{C_i}}{N}, \tag{2.20}$$

where N_{C_i} is the number of documents belonging to the class C_i and N is the number of all documents. [MRS08, p. 240]

2.4 Discriminant-based Classification

ML solution models that assign a new $\bar{x}$ with a class label by solving for the maximum posterior probability $\arg\max_k P(C_k \mid \bar{x})$ directly are called discriminative models [Bis07, p. 43]. In discriminant-based classification the focus is not on the distribution for each class but on the decision boundaries that separate the decision regions from each other (see Section 2.2.2). Thus the assumptions on the form of the distribution and the prior probability are not made but the form of the discriminant is assumed to be of a certain kind. [Alp04, pp. 197]

2.4.1 Linear Discriminant

In ML textbooks a linear discriminant is usually discussed. The discriminant function is assumed linear and is defined the same way as in Equation 2.16. For simplicity Equation 2.16 can also be written in the following way:

$$g_i(\bar{x}) = \bar{w}^{\mathrm{T}} \begin{bmatrix} \bar{x} \\ 1 \end{bmatrix}, \tag{2.21}$$

where w_{T+1} replaces w_0.

Alpaydin [Alp04, p. 198] praises the linear discriminant for its computational simplicity and also for its understandability as the coefficient $\bar{w}$ reveals the weighted values for each factor x_1,x_2,...,x_T of $\bar{x}$.

The coefficient $\bar{w}$ is not solved by estimating the prior and class likelihood's distribution like it was done in Section 2.3.3, but they are optimised by minimising the classification error E on the data set. Alpaydin discusses the option of using Gradient Descent for finding the optimal $\bar{w}$. [Alp04, pp. 206]

In Gradient Descent the training set is used to find the coefficients w and w_0 by determining the $E(\bar{w} \mid \bar{x})$ that is the smallest. An arbitrary $\bar{w}$ is chosen and it is updated on each step by adding to it $\Delta\bar{w}$, which is written as:

$$\Delta\bar{w}_i = -\eta\frac{\partial E}{\partial \bar{w}_i}, \forall i, \tag{2.22}$$

where the step size η should be chosen with care: too small a step size makes for a slow computation and too large a step size might lead to oscillation. The purpose of Gradient Descent is to find a local minimum that might be the absolute minimum.

2.4.2 Logistic Discrimination

The decision between classes is made by deciding whether a data point is on the positive side of a decision boundary or not. Therefore the following should apply:

$$g_i(\bar{x} \mid \bar{w}_i,w_{i0}) \begin{cases} > 0 & \text{if } \bar{x} \in C_i \\ \leq 0 & \text{otherwise.} \end{cases} \tag{2.23}$$

This might not apply in all cases, though, and thus the class with the highest discriminant should be chosen. A decision between two classes can be made by taking the difference of their discriminant functions. Thus a new decision discriminant function $g_{ij}(\bar{x})$ is defined:

$$g_{ij}(\bar{x}) = g_i(\bar{x}) - g_j(\bar{x}) \begin{cases} > 0 & \text{if } \bar{x} \in C_i \\ \leq 0 & \text{if } \bar{x} \in C_j, \end{cases} \tag{2.24}$$

which is a linear function, as well. [Alp04, p. 204]

In the generative approach the discriminant function was the logarithm of the class likelihoods and the prior (see Equation 2.10). The $g_{ij}(\bar{x})$ would then be:

$$g_{ij}(\bar{x}) = \log p(\bar{x} \mid C_i) + \log P(C_i) - \log p(\bar{x} \mid C_j) - \log P(C_j), \tag{2.25}$$

which can be represented as:

$$g_{ij}(\bar{x}) = \log \frac{p(\bar{x} \mid C_i)}{p(\bar{x} \mid C_j)} + \log \frac{P(C_i)}{P(C_j)}. \tag{2.26}$$

In a two class problem this is the same as the logit function of $P(C_1 \mid \bar{x})$, which has the following estimate:

$$\hat{P}(C_1 \mid \bar{x}) = \frac{1}{1 + \exp\left[-(\bar{w}_i^{\mathrm{T}} \bar{x})\right]}. \text{ [Alp04, pp. 208]} \tag{2.27}$$

When generalised into a multiple class problem, Alpaydin shows that the estimate of $P(C_i \mid \bar{x})$ is:

$$\hat{P}(C_i \mid \bar{x}) = \frac{\exp\left[\bar{w}_i^{\mathrm{T}} \bar{x}\right]}{\sum_{j=1}^{K} \exp\left[\bar{w}_i^{\mathrm{T}} \bar{x}\right]}. \tag{2.28}$$

Depending on whether the estimate is solved by maximising it or by minimising the error of $\bar{w}$, gradient ascent or gradient descent are used respectively. [Alp04, pp. 211]

2.5 Bag of Words Model

A bag of words representation of a document assigns a weight value for each term occurring in the document. It is a simplified representation of a document, because it assumes that the document's terms are independent of each other [MS00, p. 237], that they are all of equal importance and that the term's ordering is of no importance [MRS08, pp. 105]. The representation is based on the assumption that two documents with similar bag of words representations are similar in content [MRS08, p. 107].

Term frequency is often used when determining the weight value for each term t in a document d. The number of times that a term occurs in a document is counted and normalised by the total number of words in the document in the following way:

$$tf_{t,d} = \frac{\text{number of occurrences of the term } t}{\text{number of terms in document } d} \text{ [Alp04, p. 64].} \tag{2.29}$$

The number of times a term occurs in a document does not necessarily measure a term's relevance to the contents of the document. A term that occurs frequently in a document may also occur frequently in other documents and therefore make no difference in an analysis done on the contents of the document. A common way to scale the term frequency is to use the document frequency of a term noted by df_t. The document frequency is the number of documents in which a term t occurs. The inverse of the document frequency, idf_t, with which the term frequency is scaled, is defined as:

$$idf_t = \log \frac{N}{df_t}, \tag{2.30}$$

where N is the total number of documents. [MRS08, p. 108]

By multiplying the term frequency with the inverse document frequency the tf-idf weight is defined for a term t in a document d in the following way:

$$tf\text{-}idf_{t,d} = tf_{t,d} \cdot idf_t, \tag{2.31}$$

and a document can be represented by a vector of the tf-idf weights for all possible terms. The terms that do not occur in the document have the weight 0.[MRS08, p. 109]

2.6 Dimensionality Reduction

Much like in a detective novel all the facts and all the information are not necessarily crucial clues that will lead to the culprit. This is also the case in machine learning: not all variables are of importance when trying to find the right class label. Sometimes variables can always have the same value for each data point and hold no information about the right class label.

Alpaydin argues between a trade-off of classifiers and feature selection: On the one hand a classifier should by itself be able to distinguish between important and less important variables and thus reduce its dimension [Alp04, p. 129], but at the same time the complexity of the classifier can be reduced by reducing its dimensions beforehand [Alp04, p. 105]. Also when unnecessary variables are extracted the necessary variables can create a picture of the underlying problem and create new knowledge, which is called knowledge extraction [Alp04, p.106].

Unnecessary variables can be extracted or important variables can be selected in dimensionality reduction. It is called respectively Feature extraction or selection [Alp04, p. 106].

2.6.1 PCA

Extracting some variables is not the only way to lower dimensionality. The principle component analysis (PCA) does not take a look on variables and decide which ones are the best to represent the data, but instead it projects the variables on to a lower dimension in such a way that the variance of the data points is maximised [Alp04, pp. 108]. This comes from the assumption that big variance gives more information on the data behaviour than low variance [TM03].

PCA starts by choosing the first principal component, which is an eigenvector of the covariance matrix of the data. The eigenvector with the largest eigenvalue has the largest variance, thus the first principal component is the eigenvector of the covariance matrix with the largest eigenvalue [Bis07, p. 562]. Each next principal component is an eigenvector of the covariance matrix with the next largest eigenvalue and thus all components are orthogonal to each other [Alp04, p. 110].

The data $\mathbf{X}$ can be represented with its spectral decomposition

$$\mathbf{X} = \mathbf{CDC}^{\mathrm{T}}, \tag{2.32}$$

where $\mathbf{D}$ is a diagonal matrix containing the eigenvalues and $\mathbf{C}$ contains the appropriate eigenvectors of $\mathbf{X}$. For the dimension reduction of $\bar{x}$ of size $N \times 1$ to $\bar{z}$ of size $M \times 1$, where $M < N$, the following must apply:

$$\bar{z} = \mathbf{U}^{\mathrm{T}}\bar{x}, \tag{2.33}$$

where $\mathbf{U}$ contains the M greatest eigenvectors of $\mathbf{C}$ [Alp04, pp. 108]. If $\bar{x}$ would be a bag of words representation of a document, then $\bar{z}$ would be a reduced representation of that, where $\bar{z}$ would contain not weight values of terms but those of the terms eigenvector representations.

In PCA dimensionality is not reduced by picking out the variables of importance, thus knowledge extraction as such cannot be done. PCA is still very useful for visualising clusters and groups of the data. The first two or three principal components contain already a lot of the useful information in them, so that it is useful to plot a 2 or 3-dimensional graph from the components and mark the appropriate class labels to the data points. Because PCA chooses its components by maximising the variance of the data points, it is sensitive to outliers. There are several ways for isolating outliers from the analysis such as robust estimation. [Alp04, pp. 113]

PCA is commonly used for dimensionality reduction and often discussed. Ella Bingham and Heikki Mannila [BM01] discuss the computational complexity of PCA and argue, that a random projection is computationally efficient and proves to be a sufficient enough method of dimensionality reduction.

2.6.2 Decision Tree

Like PCA a decision tree is very useful for visualisation of the data. It is also useful knowledge extraction, because it splits the data according to the data's variables. Instead of looking at the variance of eigenvectors the decision tree implements a "divide-and-conquer strategy" [Alp04, p. 173] and makes splits according to low entropy.

The main principle of a decision classification tree is to find the best set of step-by-step rules so that by each step the data is separated to a given set of classes. The construction of a decision tree can be done by the ID3 algorithm [Alp04, Figure 9.3]. The algorithm checks each node's entropy to be less than a given threshold, and if it holds, a new leaf of a tree is created based on the majority class of the node. This is the so called "stop criterium" for the algorithm. If the entropy is still bigger than a given threshold, the algorithm continues and makes a split, after which the algorithm continues its recursion. The algorithm tries to choose "the best split", which causes the largest decrease in impurity. [Alp04, pp. 176]

Each leaf of the decision tree is a rule. In a multivariate problem a rule tests the value of a certain variable. A decision tree can either be build by setting a minimum value for the number of items that have to reach a node after a split. If the minimum value cannot be reached, a split will not be made and a rule will not be created. This method of building a decision tree is called pre-pruning and Alpaydin points out that it is not as good as post-pruning. In post-pruning the decision tree is build to its full size and subtrees that make no difference to the accuracy of the tree are removed. [Alp04, pp. 182]

A pruned tree for a multivariate classification problem contains in each leaf a variable and the test that should be done for the variable in order to decide on a class label. The leafs of the tree thus show the variables to be considered and thus a decision tree model is also a way to reduce dimensionality. Decision trees tend not to be very accurate for classification but they work well for visualising creating a general look-and-feel to an underlying problem. In a problem, where there are multiple possible class labels, it will be quite hard to visualise a decision tree for it. The tree as a visual aid works best for a two-class problem.

Chapter 3

Ontologies and the Semantic Web

This chapter describes the semantic web and explores its definition. The building blocks of the semantic web, ontologies, are explored. This chapter illustrates how ontologies are built and explains the syntax of expressing ontologies.

3.1 Ontologies

Ontology in ancient Greek philosophy means the theory of being, existence and reality. The field of Artificial Intelligence (AI) has taken the word ontology to mean something a little different but in relation to the word's original meaning. Gruber [Gru93] defined an ontology as

> an explicit specification of a conceptualisation. [Gru93, p. 1]

In his dissertation Borst [Bor97] quotes Gruber's definition, but changes the word "explicit" to "formal" and points out that the conceptualisation must be based on an agreement and adds the adjective "shared" to the definition. Studer et al. [SBF98] see both definitions as right and define an ontology as

> a formal, explicit specification of a shared conceptualisation. [SBF98, p. 25]

Borst describes a conceptualisation as a “structured interpretation” of the world that people communicate about [Bor97, p. 12]. When people communicate about the world, the words they use hold a meaning for them. This is not the same for machines. For them these words are just meaningless strings of symbols. Therefore in AI there is a need for putting the words into a structure of concepts, an ontology, where the connections between concepts corresponding to the words are described in a formal and explicit way. Formal description of the concepts allow for an ontology to be machine readable [SBF98, p. 25]. Even though an ontology is defined to be explicit, the definition for the accuracy of the explicit description of the concepts is problematic [Alm07, p. 4].

3.1.1 Semantic Relations

The concepts in an ontology are related to human readable words (literals) and to each other through semantic relations. To formally map different relations between concepts a set of semantic relations are used to describe them.

The backbone of an ontology is usually a set of abstract concepts that are in a hierarchical subclass-of relation to each other [Bor97, p. 18]. Depending on the ontology and ontology language used for representing it, concepts can be enriched with attributes and other relations to each other. The constraints on them make up a set of rules called axioms [SBF98, p. 28] that can be used for reasoning [Bor97, p. 66]. There is a semantic difference in the type of hierarchical relation that a human can easily distinguish and takes for granted but that needs to be explicated to the machine [Hyv05].

Hyponymy

The apple is a subconcept of the fruit, where the fruit is the hypernym of apple and the apple is a hyponym of the fruit. Hyponymy is a hierarchical is-a or subclass-of relation. It exists when all the instances of a concept X, that is a subclass of a concept Y, are also an instance of Y [Gru93, p. 28]. For example, the Titanic is an instance of the watercraft concept, which is a subclass of the vehicle concept. Thus all instances of the watercraft are also instances of the vehicle.

Meronymy

A hierarchical relation different from the hyponymy is the part-of relation. A branch could be associated to being a subclass of a tree, but to the machine it has to be distinguished as part of a tree [Hyv05]. In a hyponymy the concepts inherit the characteristics of their hypernyms [GW04] and in a meronymy they are only seen as physically being part-of a whole. The difference has to be made, so that the machine can reason appropriately.

Associative

The ontology also contains other relations that are not hierarchical. This can be expressed with an associative relation between them. There are numerous ways of associating concepts and it has proven to be quite challenging to find a compact way to describe them explicitly to the machine, because there are so many different reasons for associating words with another [Hyv05]. For example "rain" is associated with an "umbrella", because rain is the reason for using an umbrella. "Water" can be associated with a "well", because that is the place where water could be carried from.

3.1.2 Meaning of Concepts

Words cannot necessarily be disambiguated to unique meanings from their text representation alone. For example the word light can either mean the adjective that is the opposite of heavy or the noun for expressing the radiation of the sun. Words whose string representation is the same but, which have different meanings are called homonyms. Words with equal string representations and related meanings are polysemous words. An example of that is the word crane, which can mean the bird or the construction rig. Concepts with identical labels can be set apart by a computer, if they are given different unique identifiers. [Hyv05]

3.2 Building the Semantic Web

Tim Berners-Lee et al. [BLHL01] paint the picture of the semantic web,

> where software agents roaming from page to page can readily carry out sophisticated tasks for users. [BLHL01, pp. 34]

The software agents use ontologies to deduce the needed information for their users. Ontologies are, as said, formally described structures of concepts and by using ontology information software agents can recognise a piece of information to be a certain concept and use the relations described for that concept in the ontology to reason on. Berners-Lee et al. note that a computer programme does not understand the concepts, but can process the information for the advantage of its users. [BLHL01, pp. 34]

Attaching information to a document is called annotation and it is ontology-based if the information is machine-readable and formally connected to an ontology [Alm07]. When the annotation is ontology-based a computer programme can process the information for deduction and reasoning [Bor97, p. 66]. Annotation also helps search functions. For example a document annotated with the concept apple can be recognised by a search engine reasoner to be a document concerning fruit, because the apple is a subclass of the fruit.

Oscar Corcho [Cor06] introduces document annotation by illustrating the many ways in which information about a document can be expressed and added to. Information about information is called metadata and it can be expressed, for example, with a filled form about the document. The form would have fields such as the author, date of publishing, owner of the document, place of origin, title and so on. For the information to be of use for the purposes of the semantic web, plain text metadata will not suffice, but the metadata has to be machine-readable and linked to an ontology [HSM01].

Ciravegna et al. [CDPW02] note that document annotation requires a lot of manual work and argue for the need of information extraction to make the process automatic or at least semi-automatic.

The Semantic Computing Research Group (SeCo) of the Aalto University School for Science and Technology has developed tools for semi-automatic and automatic annotation. The Opas system, that is used by librarians for annotation in the Ask-the-Librarian service, suggests automatically ontology concepts for annotation. The users of Opas still have to pick the right concepts and thus the system allows for semi-automatic annotation. [Veh06]

Another semi-automatic annotation tool is the browser-based annotation tool SAHA [Val06] that allows for multiple users to annotate a shared material. It has been developed by the research team and the newest version of the SAHA-tool allows even for social interaction between the collaborating annotators.

Annotation tools are usually semi-automatic, because some words are ambiguous and can match the string representations of multiple concepts. Stephen Dill et al. [DEG$^+$03] have developed an automatic web-page annotation tool, SemTag, which tries to solve this problem automatically. They tested it on a large corpora of web pages using the TAP ontology [GMM03]. To find the right concept that represents a word best, they created an algorithm. It explores both the context of the word and the ancestory and decendants of all possible concepts that could fit the word. The algorithm calculates the similarity between the word's context and each possibly matching concept's relational environment and chooses the most similar concept.

The POKA-tool developed by Olli Alm is an automatic annotation tool for text documents. With the POKA-tool among other things words of a document can be turned into their base form and documents can be annotated based on the document's words by connecting the words with the concepts of the ontology POKA uses. [Alm07]

3.2.1 Standards for Representing Ontologies

Ontologies are widely used on the semantic web, and the following W3C recommendations[1] are often used in practise for representing them: the Resource Description Framework (RDF), the RDF Schema [BG04], and the Web Ontology Language (OWL) [BvHH$^+$04]. Here concepts of ontologies are resources that are identified with a unique Uniform Resource Identifier (URI) and are described with properties, which themselves are resources with URIs, too. Everything is described using triples of the form

```
<subject, predicate, object>
```

where the subject is the resource to be described, the predicate is its property, and the object the value of the property. The object can be either a resource or a literal piece of data [MM04]. RDF triples constitute labelled directed

[1] http://www.w3.org/standards/semanticweb/

graphs, and this data model will be used in the case study described in Chapter 4.

There are different serialisations for the data model of RDF, such as RDF/XML, full or abbreviated.

In this master's thesis the resources are described using the compact textual syntax Turtle Terse Triple Language (TTL) [BBL08]. In TTL a triple is represented so that all triples corresponding to a subject and a subject-predicate pair are grouped together. If there is only one triple for a subject, the triple is:

```
subject1 predicate1 object1 .
```

The grouping for multiple triples for one subject, looks as follows:

```
subject1
    predicate1 object1 ;
    predicate2 object2 ;
    predicate3 object3 ;
    predicate4 object4 .
```

Let's say that there are multiple triples for `subject1` and `predicate1`, then the grouping would look like:

```
subject1
    predicate1
        object1 ,
        object2 ,
        object3 ;
    predicate2 object2 ;
    predicate3 object3 ;
    predicate4 object4 .
```

URIs look very much the same like Uniform Resource Locators (URL) but URIs do not necessarily point to a network location like URLs do. URI's consist of a namspace and a local name, for example:

```
http://www.narc.fi/onto#document1234,
```

where the first part of the URI until the hash sign is the namespace and the following part the local name. The namespace is common for many resources and represents the affiliation of the instance the URI identifies. the local name may not be globally unique, but needs to be different for each resource under the same namespace.

Because URIs tend to be long and hard to write, the namespace can be abbreviated to a short namespace prefix. The prefix is followed by a colon sign and the local name of the URI. The principle of combining the namespace prefix and the local name is used in QNames. The prefix has to be declared in the beginning of a document. For example in TTL the declaration would be:

```
@prefix narc:    <http://www.narc.fi/onto#> .
```

and later in the document the URI would be:

```
narc:document1234
```

URIs can be abbreviated also using CURIEs, which are recommended for language designers. In CURIEs the format for the part after the colon is not as strict as in QNames, because in QNames for example the local name cannot begin with a number. [BM09]

The human readable name of a resource is usually described with the property `rdfs:label`[2], its literal property value and a language tag [Alv01], that can be added to the property value to mark multilingual labels in the following way:

```
narc:document111
    rdfs:label
        "asiakirja"@fi ,
        "document"@en ;
    ...
```

where “fi” stands for the Finnish language and “en” for the English language.

[2] http://www.w3.org/TR/rdf-schema/#ch_label

3.2.2 General Finnish Ontology YSO

The General Finnish Ontology YSO has been developed from the commonly used Finnish General Thesaurus YSA [HVK$^+$05]. The thesaurus YSA uses relations such as narrower term (NT), broader term (BT), and related term (RT) between the thesaurus' terms. Even though the relations BT and NT point to a hierarchical relation between terms, the distinction between hyponymy and meronymy is not made. In order to turn the YSA into a machine-readable ontology, it was transformed using the OWL-language and the distinction between different kinds of BT and NT relations was made. First all BT/NT relations were turned into subclass-of relations and the concepts that were actually in a part-of relation to each other were corrected. In addition to this a group of abstract concepts not from YSA were added to the ontology for arrangement purposes. [HSVF07]

The RT relation expresses an associative relation between terms, for example an umbrella is associated with rain. The number of different associative relations can become unreasonably large. In YSO the type of the associative relations is left undecided. If needed, further refinements need to be made in applications. [Hyv05]

YSO's concepts are labelled not only with their Finnish names from YSA but also with their equivalent Swedish names from the General Swedish Thesaurus Allärs [HSVF07]. As of the year 2010 YSO contains also English labels for concepts, when appropriate.

The purpose of YSO is to serve as a national top-ontology for other ontologies in Finland [HSVF07]. The research group SeCo has developed domain specific thesauri into ontologies and used YSO as a basis and reference for them. For example the Finnish Thesaurus for Music MUSA contains around 900 terms. The thesaurus was automatically transformed into the OWL-language and the terms were first automatically matched with concepts from YSO. Therefore, after the automatic transformation, the syntactical ontology was manually checked and corrected into a true ontology. Approximately half of the MUSO concepts were equivalent concepts of YSO. The rest of MUSO now enriches YSO with concepts from the musical field. The BT and NT relations were like in the YSA to YSO transformation turned into subClassOf relations. Thus not all of the resulting relations were correct and had to be corrected. [Nyb08]

Chapter 4

Preprocessing of Data for the Case Study

This chapter describes the way in which the text of the Archive's documents was extracted, analysed and turned into a form that could be used by the Machine Learning analysis.

4.1 Documents of the Finnish National Archive

This case study on document classification is based on a data set from the Finnish National Archive (in short the Archive), which is a government body under the Ministry of Education. A test set of the Archive's own case management system was provided for this master's research. The test data contains documents and a XML file with all the metadata concerning the information of the archive holder, archive, groups, cases, actions and documents according to the SÄHKE metadata scheme.

The Archive provided the research also with a listing of a classification hierarchy. The listing contains a 2-levelled classification with 70 classes and 45 subclasses. Only 13 classes have subclasses and a class with subclasses has on average 3.23 subclasses.

Of the provided classification the test data uses only 67 classes of which 31 are subclasses. In the metadata these are represented by the group entities. The unused classes of the classification are not included in the metadata file.

The data set contains 7,252 documents that are linked to inquiries directed to the Archive. The inquiries are part of the Archive's service. Normal citizens or researchers ask for example for access to certain kind of information that the Archive holds or may hold.

The Archive's data set can be viewed with the HAKO-tool[1]. The HAKO-tool makes it possible to search for a certain data through multiple facets. A search is made easier by filtering the content through its multiple properties. For example when using the HAKO-tool with this particular data set it is possible to narrow down the number of documents to documents that belong only to a certain group. It is also possible to narrow down the search further and view only the documents of a certain group that were archived by a particular person. This narrows down the search quite effectively and helps to achieve a reasonably quick access for a large data set.

4.1.1 Metadata Model

The Archive dictates that any national or municipal organisation that wishes to store documents permanently and only in digital form needs to ask for the Archive's permission and has to follow the SÄHKE metadata model. The model is concerned with the digital handling, managing and finally storing of information on official documents concerning national and municipal governments. It dictates the way in which metadata, the information about the official documents, such as the author and the title, is stored. In the SÄHKE metadata model each document is part of a procedure of processing actions and cases and metadata of the procedure is also stored. The SÄHKE metadata model forms a standard under which the digital case management system of national or municipal organisations can be formed. [Ano05]

The abstract specifications of the SÄHKE metadata model [SM05] introduce an archive hierarchy under which the actual documents are placed in the metadata model (see Figure 4.1). The archive hierarchy contains the parts of the procedure in which the documents are stored. Each document is associated to one or more actions through an XML reference. Each action is then linked to a case. Inside a case a number of actions can contain the same document, if the document is of significance to that action. Each case belongs to one group and each group represents one class of a given classification. Any

[1]`http://www.seco.tkk.fi/tools/hako/`

organisation using the SÄHKE metadata model is an archive creator and holder and holds one or more archives. The archive holder is the agent that produces all the information inside the archive [SM05]. Each archive contains one or more groups under which the cases to be stored are grouped. Groups can also contain sub-groups. Because of this the groups form a hierarchical structure that can also be seen as the classification for the cases, actions and documents.

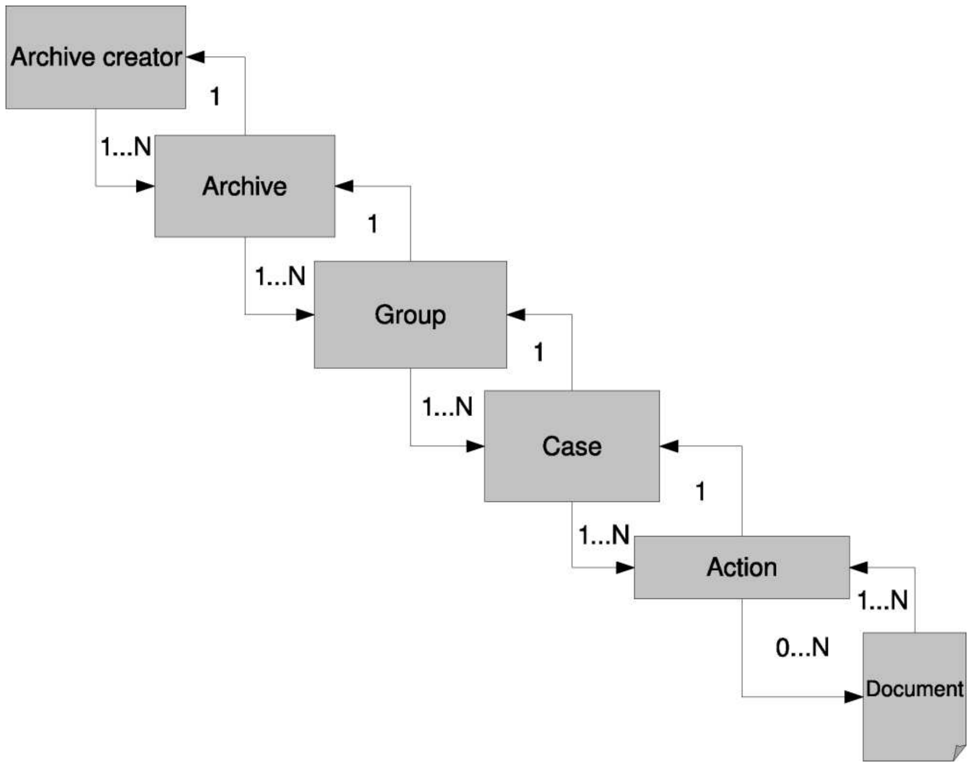


Figure 4.1: Archive hierarchy of the SÄHKE metadata model as specified in the article on the abstract modelling for the SÄHKE project [SM05]

The archive hierarchy is a direct representation of the hierarchy of the XML file that holds all the metadata information of one archive holder and its archives. In that file the archive holder is represented as an XML entity so that the entity contains one or multiple archive entities, among other things. Each archive entity contains one or several group entities. Each group entity contains one or more cases and may contain also references to its supergroup or subgroups. A case entity contains one or more action entities, and action entities contain one or more document entities.

The Archive's own data set follows the metadata model and its 7,252 documents are linked to 32,325 actions in total. They describe the actions taken during a process, where an inquiry is received and dealt with, i.e. an email request for information from the Archive is received, saved as an document and put into the data system as an action. That email is answered with an email as well, and saved as a document and action. This is done for all the steps when a request is processed. All such actions involving one matter are linked to a case. The data contains 3,469 cases. The cases are categorised under 67 groups. This particular data set is subject to only one archive.

4.1.2 Transformation of the Metadata to RDF

The metadata of the case management system was read and turned into RDF-form using the Turtle syntax. Each XML element was turned into a resource using the namespace `http://www.narc.fi/onto\#` and a local name that consisted of the element's tag name and an arbitrary unique number identifying the element. A resource class was created from the element's tag name and the class was set as the type of that resource. Every attribute of a element was turned into a triplet with the element resource as the subject, a property created from the namespace and the attribute's local name as the predicate, and the attribute's value as a literal object.

For example, the XML file describing the SÄHKE metadata of the case management system of the Archive contains an element called case and that element has the attribute ID="210". In the transformation of the metadata this resulted in the following triplets:

```
narc:case11fb2f943b914
      rdf:type narc:case ;
      narc:ID "210" ;
      ...
```

The information of each child element of an element was also saved as a triplet. If the child element had no children elements or attributes of its own, but only contained text, it was treated just like an attribute of its parent element and the information was turned into a triplet with a literal object. Thus the child element's local name served as the local name of the property. In case the child element had child elements of its own, a triplet was created with the parent element's resource as the subject and a property with a local name that started with "has_" and was followed by the child element's local name as the predicate. The object of the triplet was the child element turned into a resource with a local name that started with the child element's local name and ended with an arbitrary unique identication number. The resource of the child element was typed with a class created from the element's local name and the namespace.

In the above mentioned example, the case-element contains five action child elements. These were turned into resources of their own, as they contain more than just text and resulted in triplets describing the information the actions contain. In the above mentioned case the following five triplets were created expressing the actions that are contained in the case:

```
narc:case11fb2f943b914
    narc:has_action
        narc:action11fb2f943c88c ,
        narc:action11fb2f943c780 ,
        narc:action11fb2f943c46b ,
        narc:action11fb2f943be2a ,
        narc:action11fb2f943d2d2 ;
    ...
```

4.2 Processing the Documents for Further Analysis

A Java programme was written that handled the documents and turned them into files that could be utilized by the Machine Learning analysis. Figures 4.2, 4.5, 4.6, 4.7, 4.13, and 4.17 show the different steps of that programme.

Two Java classes, DocumentHandler and Analyser, were written in order to handle the whole process of extracting the text from documents, sending it for analysis, receiving it and transforming it so that it could be used by the Machine Learning analysis. Below are the descriptions for each class.

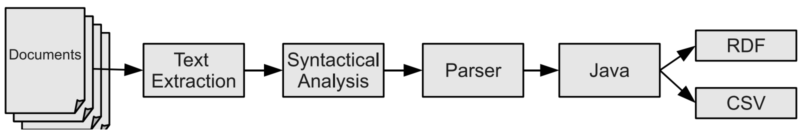
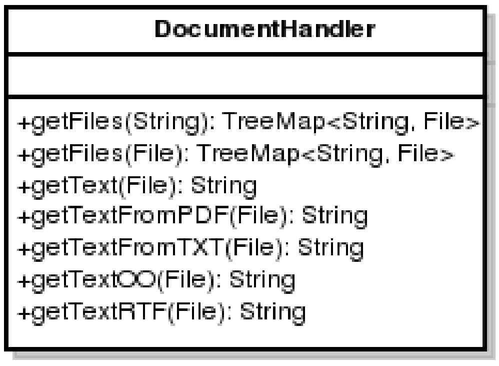


Figure 4.2: The parts of the process with which document text was read and made available for the Machine Learning analysis.

DocumentHandler Class

The DocumentHandler class (see Figure 4.3) was designed to read files and extract the text they contained. It comprises methods, that read files from a given directory path. The directory can be represented by a String or File object. The getText(File file) method analyses the file name of the input file and searches for certain kinds of file endings. This was a sensible way of distinguishing between file types as all files, from which the text could be extracted from, had file endings. For finding out the file type, determining the MIME type was tested as well. It did not prove to be sufficient enough as it was unable to distinguish between Microsoft Office documents, PDFs and TXT documents. Some RTF files were recognised to be of MIME type "application/rtf" but most documents were recognised as of MIME type "application/octet-stream".



```
DocumentHandler

+getFiles(String): TreeMap<String, File>
+getFiles(File): TreeMap<String, File>
+getText(File): String
+getTextFromPDF(File): String
+getTextFromTXT(File): String
+getTextOO(File): String
+getTextRTF(File): String
```

Figure 4.3: UML-graph for the DocumentHandler class, that acts according to its name

Analyser Class

Reading files, making them ready for the syntactical analysis and the transformation of the results of the analysis was done by the the Analyser class object (see Figure 4.4). It stored the DocumentHandler object used in this programme and all the objects of the documents, that could be analysed for this case study. It also stored a Jena model, because later on all information of the documents was stored in RDF form. The namespace that was used in the model for all elements was held in the "ns" parameter. In addition to these the class had three fields of type Resource. They were all used in connection with creating the RDF data from the object transformed analysis. One of them, the term-field, was made public, as it was used by the class' main method. The analyse method took as its input either a String representation of a directory, where all the to-be-analysed files were stored, or a map of files. The method created a document object for each file and stored the document by its ID number into a Document object map.

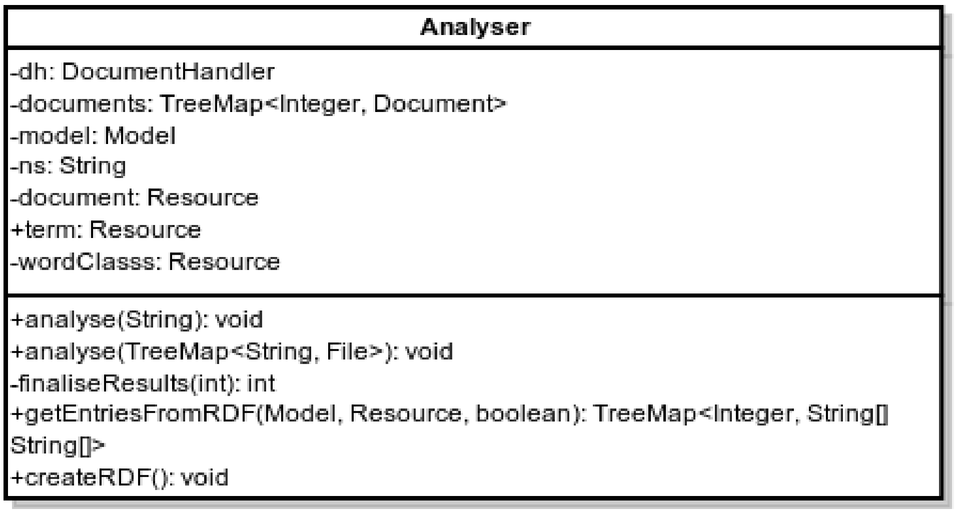

Analyser
-dh: DocumentHandler
-documents: TreeMap<Integer, Document>
-model: Model
-ns: String
-document: Resource
+term: Resource
-wordClasss: Resource
+analyse(String): void
+analyse(TreeMap<String, File>): void
-finaliseResults(int): int
+getEntriesFromRDF(Model, Resource, boolean): TreeMap<Integer, String[] String[]>
+createRDF(): void

Figure 4.4: UML-graph for the Analyser class, which is the core of the programme created for analysing the text from documents

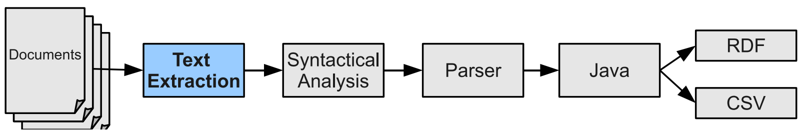


Figure 4.5: Text extraction

4.2.1 Document Text Extraction

First the text of each document in various machine readable file formats was extracted and read into String variables. Out of the 7,252 documents, 2,324 were scanned documents, mostly TIFF images. Some of these documents were written by hand and contained even hand-written Russian text. Others were written by a machine and therefore an OCR-scan of them would have helped in getting the document text in digital form. Due to lack of resources a sufficient OCR-scan was not performed and these documents were left out of the analysis. In the end 4,919 documents could be used for the analysis.

Most of the machine readable documents, 4,363, were in the form of Microsoft Office documents (.doc, .dot, .xls, .xlt) and they were turned into OpenOffice.org form using a Wizard contained in the OpenOffice.org programme. OpenOffice stores the information of its files in XML form, so that the text they contain can be easily read with Java using for example the Java org.jdom package[2] for the handling of XML files. The code for handling contents of OpenOffice.org documents was provided in a Sun Microsystems employee blog under the category of Useful codes [3].

Other documents that were stored in formats such as RTF, PDF and TXT where it was possible for a computer programme to extract the text they contained. RTF was read with Java using the javax.swing.text.rtf package[4]. PDF files were handled using the org.pdfbox package[5]. Files of file extension "txt" were read simply by using the java.io package. The text of each document was read into one String variable, sanitised from malicious characters

[2]`http://www.jdom.org/`

[3]http://blogs.sun.com/prasanna/entry/openoffice_parser_extracting_text_from

[4]Using especially the RTFEditorKit class: `http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/text/rtf/RTFEditorKit.html`

[5]`http://www.pdfbox.org/javadoc/org/pdfbox/package-summary.html`

and then passed on to a syntax analyser component.

4.2.2 Syntactical Analysis of the Text Extracted

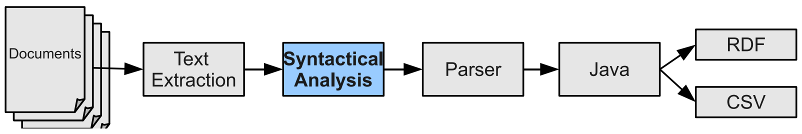


Figure 4.6: Syntactical analysis

For this case study of the documents of the Archive the Machinese Syntax[6] component created by Connexor Oy proved to be very sufficient. The component works for all of the Archive's documents' languages, which are Finnish, Swedish and English language. The Machinese Syntax was used on the text extracted from the documents. The component takes as its input text and returns that text in XML form. It recognises each sentence and numbers them, numbers each word inside a sentence, and turns them into a base form. Machinese Syntax also analyses the syntactical relations between words. [Ano06]

For example, one of the analysed documents represents an e-mail sent to the Archive. Its seventh sentence ("She is collecting information on Boris Amarantov, circus artist, mime, who has emigrated to USA from URSS in 1977.") was analysed by the Machinese Syntax. For the result of the analysis see Table 4.1 below. The first column contains the running number of the words of one sentence. The first word of each sentence is numbered with 1. The second column contains the word of the sentence in its original form and the third column the word's base form. The fourth column notes the functional dependency relations the words have with each other. For example, the syntactical relation of the first word "She" is marked as "subj:> 2", which means that word is the subject for the second word "is". The second word's functional dependency "v-ch:>3" notes that the second word is part of a

[6]Earlier known as The Functional Dependency Grammar (FDG). The Machinese Syntax Demo is available on Connexor's website: `http://www.connexor.eu/technology/machinese/demo/syntax/`

verb chain that is the predicate of this sentence. The functional dependency of the third word “main:> 0” means that this is the main verb of the main clause. The predicate is “is collecting”, its subject is “She” and by reading on the fourth row we find that the object is “information”. The fifth column contains the Language Model Tags the Machinese Syntax component provides each word with. Functional tags are marked with @, surface syntactic tags with % and morphological tags with no special character. For example the first word of the example sentence is tagged as the subject (@SUBJ) and the nominal head (%NH) of the sentence. Its morphological tags show that it is a personal (PERS) pro-nomial (PRON) in nominative case (NOM) and third person singular (SG3).[7]

1	She	she	subj:> 2	@SUBJ %NH PRON PERS NOM SG3
2	is	be	v-ch:>3	@+FAUXV %AUX V PRES SG3
3	collecting	collect	main:>0	@-FMAINV %VA ING
4	information	information	obj:>3	@OBJ %NH N NOM SG
5	on	on	mod:>4	@<NOM %N< PREP
6	Boris	boris	attr:>7	@A> %>N N NOM SG
7	Amarantov	amarantov	pcomp:>5	@<P %NH N NOM SG
8	,	,		
9	circus	circus	attr:>10	@A> %>N N NOM SG
10	artist	artist	mod:>7	@APP %NH N NOM SG
11	,	,		
12	mime	mime	mod:>10	@APP %NH N NOM SG
...	...	...	...	...

Table 4.1: Machinese Syntax output for the beginning of the sentence: “She is collecting information on Boris Amarantov, circus artist, mime, ...”

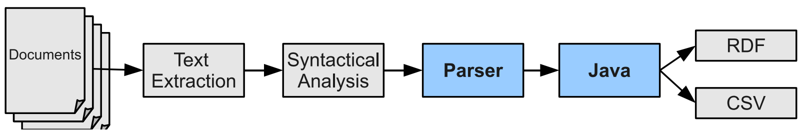


Figure 4.7: Document Information parsed and saved to Java objects

4.2.3 Parsing the Syntactical Analysis

The analysis from the Connexor Machinese Syntax component in XML form was parsed using the POKA-tool[8]. Its class FDGParser was originally designed to transform the XML output of the component into a more efficient XML form, that the POKA-tool used [Alm07]. For the purpose of this case study, the FDGParser was modified so that it stored the information from the Machinese Syntax analysis into sentence and word objects. This was done in order to store and further process the information from the analysis.

4.2.4 Document Text Transformation

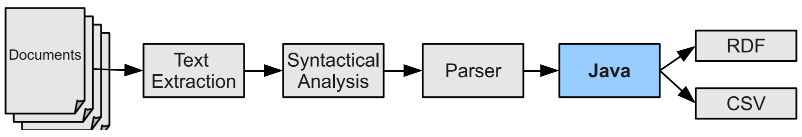


Figure 4.8: Document Information in Java objects

In order to process the analysis from the Machinese Syntax, its information was stored into Java objects. The syntactical information from the analysis was stored into Java objects of the classes Document, Sentence and Term. The functional dependency between two Term objects was stored in a object of the class Functional Dependency. Below is a description of each class.

[7]For the description of each tag see: `http://193.185.105.50/demo/machinese/doc/enfdg3-tags.html`

[8]`http://www.seco.tkk.fi/tools/poka`

Term Class

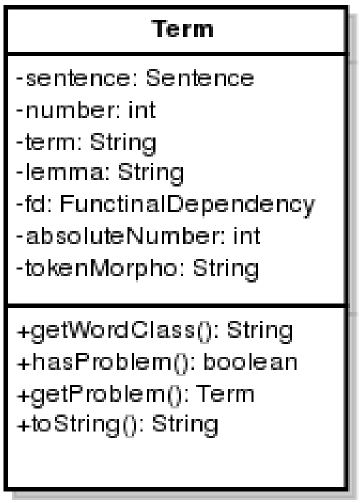


Figure 4.9: UML-graph for the Term class, whose incarnations store all the information for each word of all documents.

Figure 4.9 shows the information from the Machinese Syntax analysis that was stored for each word of all documents. In the Term class represented by the UML graph the word is called a term and its base form is called a lemma. The objects for each term stored the number of the term inside one sentence. Later, when all documents had been analysed and stored into objects, a unique absolute number was determined for each term object. It numbered all analysed words across sentences. The term object also stored the object of the sentence from which the word originated from. The tokenMorpho parameter of the term class stored the morphological tags from the Machinese Syntax analysis. The getWordClass() method of the class was designed to parse the term's word class information from the morphological tags. It searched for tags, such as N, A, DET, NUM, ADV, PREP, PRON, V, CC and CS, and set the word class to noun, adjective, determiner, numeral, adverb, preposition, pronoun, verb and, conjunction, respectively.

FunctionalDependency Class

The syntactical relation of two terms was stored as a FunctionalDependency class object (see Figure 4.10). The object stored the term object from which

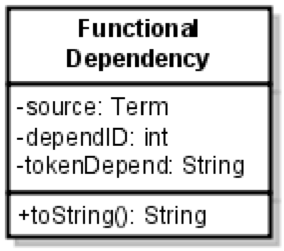


Figure 4.10: UML-graph for the FunctionalDependency class, whose incarnations store the information of the syntactical relation from one word to another.

it originated (and in which it was stored as well) in the source parameter, the String representation of the dependency, such as “subj”, “v-ch” or “obj”, in the token parameter, and the number of the word to which the syntactical relation was directed to in the dependID parameter. Later on, when all documents had been analysed and each word had its unique sentence-independent id, the dependID parameter was reset to contain the unique ID of the syntactical relation term.

Sentence Class

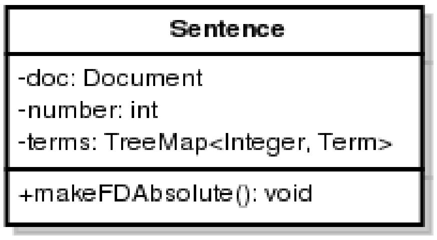


Figure 4.11: UML-graph for the Sentence class, whose incarnations store the information of each sentence.

The UML-graph for the Sentence class is shown in Figure 4.11. The class creates objects that store the object of the document, from which the sentence originated from, in the doc parameter. It also stores the order of the sentence in the document in the number parameter. All term objects inside the sentence were stored in the terms parameter with their number as their key. The Sentence class comprises a method, makeFDAbsolute(), that changes the FunctionalDependency object of each term object of the sentence. This method was called after all documents had been analysed and all term numbers made absolute. It resets the dependID to the unique ID of the term to which the dependID parameter of the Functional Dependency instance points to.

Document Class

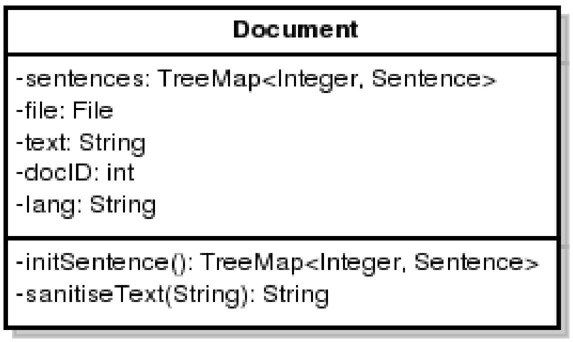

Document
-sentences: TreeMap<Integer, Sentence>
-file: File
-text: String
-docID: int
-lang: String
-initSentence(): TreeMap<Integer, Sentence>
-sanitiseText(String): String

Figure 4.12: UML-graph for the Document class, whose incarnations store the information of each document.

The Document class shown in Figure 4.12 is designed to hold the information of one document. Its method initSentence() checks the document text for bad characters and sends and receives the text to and from the Machinese Syntax analysis. Its incarnation stores all sentences of a document in the sentences parameter. The text of the document is also kept in the same named parameter of type String. The document object holds the object representation of the document file, the document id and the language code of the document.

First when the code and the output from the Machinese Syntax analysis was tested, the text taken from the documents contained malicious characters, that were placed between two words. It is sometimes convenient to write two alternative words together and divide them by the "/" character, e.g. "Attached you find the bill/invoice of my postage costs." The Machinese Syntax cannot distinguish between the two words and handles the bundle as one word (see Table 4.2, fifth row). The santiseText(String) method takes

1	Attached	attach		@-FMAINV %VP EN
2	you	you	subj:>3	@SUBJ %NH PRON PERS NOM
3	find	find	main:>0	@+FMAINV %VA V PRES
4	the	the	det:>5	@DN> %>N DET
5	bill/invoice	bill/invoice	obj:>3	@OBJ %NH N NOM SG
6	of	of	mod:>5	@<NOM-OF %N< PREP
7	my	i	attr:>8	@A> %>N PRON PERS GEN SG1
8	postage	postage	attr:>9	@A> %>N N NOM SG
9	costs	cost	pcomp:>6	@<P %NH N NOM PL
10	.	.		

Table 4.2: Machinese Syntax output of the sentence: "Attached you find the bill/invoice of my postage costs."

this problem into account and removes the "/" characters from word bundles such as these. The method ignores URLs, though, as their components did not need to be handled separately for this particular case study.

4.2.5 RDF Transformation

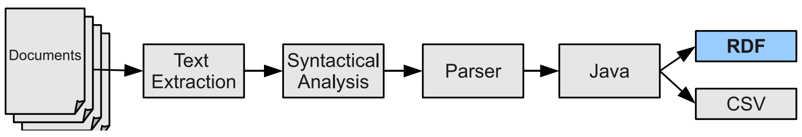


Figure 4.13: Document information in TTL file.

When all document files had been read, sent for analysis, received from analysis, and transformed into objects, the objects and the information they held were read into a Jena Model[9]. The RDF was created based on the RDF Schema represented in Figure 4.14. In this RDF schema the term is in the

[9]Jena Model Javadoc: `http://jena.sourceforge.net/javadoc/com/hp/hpl/jena/rdf/model/Model.html`

middle of the focus, because the Machinese Syntax component's analysis provides information for each term rather than for a sentence and because the Machine Learning analysis uses the bag-of-words representation of the documents.

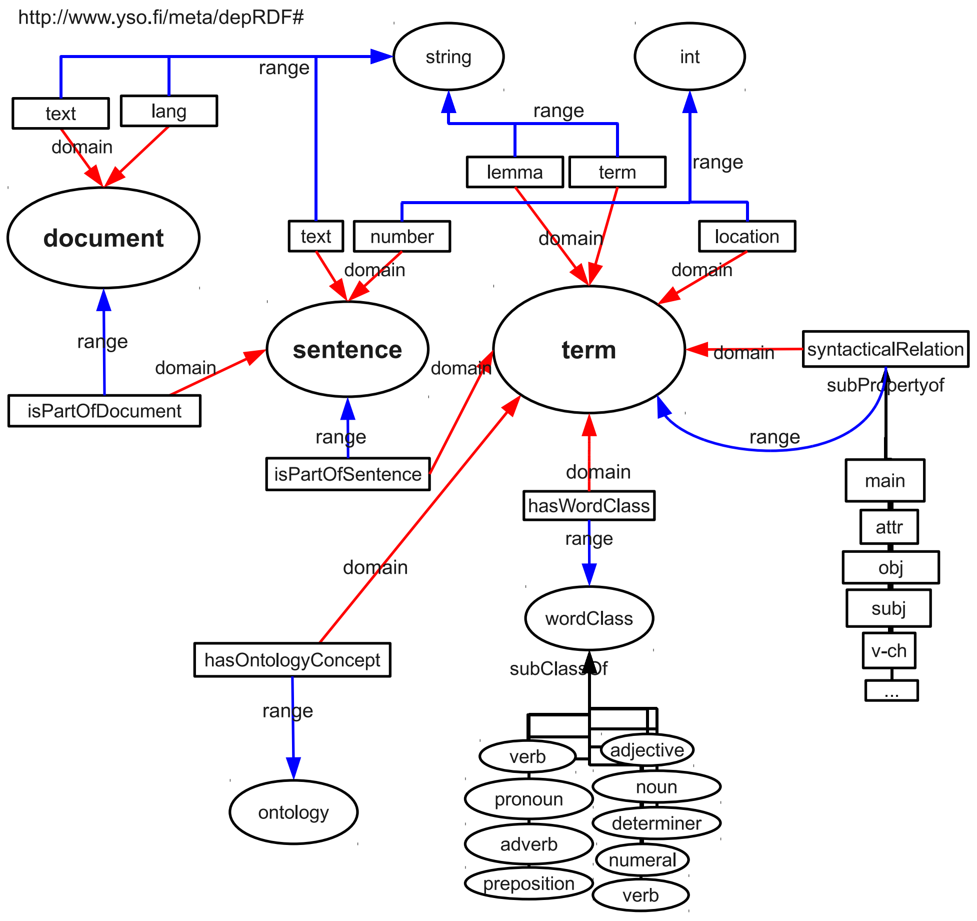


Figure 4.14: RDF Schema on how information of the documents and their Machinese Syntax Analysis was stored.

The namespace of all nodes in the schema is `http://www.yso.fi/meta/depRDF#`. The start of the namespace, `http://www.yso.fi/meta/`, is used by the research group SeCo for different kinds of metadata schemas and the last part `depRDF#` is the proposed name for ontologies adhering to the proposed schema showing syntactical dependencies between terms in sentences and documents.

The transformation from Java objects to resources of the Jena model is done in the following way: A resource of type `http://www.yso.fi/aski/term` is created for each term, so that the term's URI consists of a localname beginning with ``` ``term_'' ``` and followed by the term's unique ID number. The term resource has four literal properties that store the term's sentence number (sentenceNr), the sequence number of the term in the sentence (location), the term's base form (lemma), and its original form.

The syntactical relation from one term to another is held by various sub-properties of the property called "syntacticalRelation". The sub-properties get their localname from the Machinese Syntax analysis. In this particular case study, 42 of these kind of different syntactical relations were created[10]. For example the words of the beginning of the sentence "She is collecting information..." have the following syntactical relations with each other: "She" is the subject (subj) of the verb chain (v-ch) "is collecting", which is the main predicate (main) of the sentence. "Information" is the object (obj) of the sentence and has a syntactical relation with the main predicate.

Each document is represented by a resource of type `http://www.yso.fi/aski/document`. A document has two literal properties one of which stores the documents language code (lang) and the other the extracted text of the document. Each term resource was linked to the resource of the document from which the term originated with the "isPartOfDocument" property. The document resources have local names that start with "document_" followed by the documents unique ID number.

The word class of each term is recognised as one of nine choices: verb, pronoun, adverb, preposition, adjective, noun, determiner, numeral and conjunction. These were represented in the RDF model by nine resources of type `http://www.yso.fi/meta/depRDF#wordClass`. The "hasWordClass" property points from a term to one of the nine word class resources.

A term can also be associated with a URI reference to an ontology. In this particular case study the Finnish General Ontology (YSO) was used. It contains terms that have labels in Finnish, Swedish, and English. The

[10]Syntactical relation tags are listed here: `http://193.185.105.50/demo/machinese/doc/enfdg3-tags.html`.

POKA-tool comprises a method that matches a query with the label in one of the above mentioned languages. In Finnish some words look completely the same but have different meanings depending on the syntax of a sentence. For example, "alusta" can mean "from the beginning" or "base". Trust was put in the heuristics of the Machinese Connexor tool to recognise the right lemmas for terms, and the lemmas of each term were given as an input for this method. The result of the method was a list of URIs from the YSO ontology. Most terms were matched with only one URI. The POKA-tool was trusted and these matches were not checked. 759 terms, though, were matched with two or more URIs. The multiple alternatives came mostly from polysemous concepts. For example, the word "child" has three different meanings in YSO:

- a role of a person based on her age,
- a role of a person belonging to a certain social-economic group and
- the concept for a family member (its subclasses are daughter and son).

The multiple URIs were checked by hand, the correct YSO reference was selected, and the other references removed.

Figure 4.15 shows the resulting syntactical dependencies and term information from the Machinese Syntax analysis (see Table 4.1 for the analysis) according to the RDF schema in Figure 4.14.

A snippet of the resulting RDF in TTL form is featured in Figure 4.16. It shows the 911,753th term of this particular analysis and all the triplets, where the term is the subject. The term is from the document number 2612 and it is in Finnish. Below the term the triplets for the document number 26136 as a subject are featured. One of the document's sentences is the same sentence, whose Machinese Syntax analysis was shown as an example in Table 4.1. The letters in Cyrillic alphabet were ignored by the Machinese Syntax component and only the words Yahoo, Yahoo and the URL in the footer were taken into account in the analysis.

4.2.6 Creating the Data Set

In Machine Learning analysis models take training sets as an input. These training sets contain individual points in rows with a set of attributes in

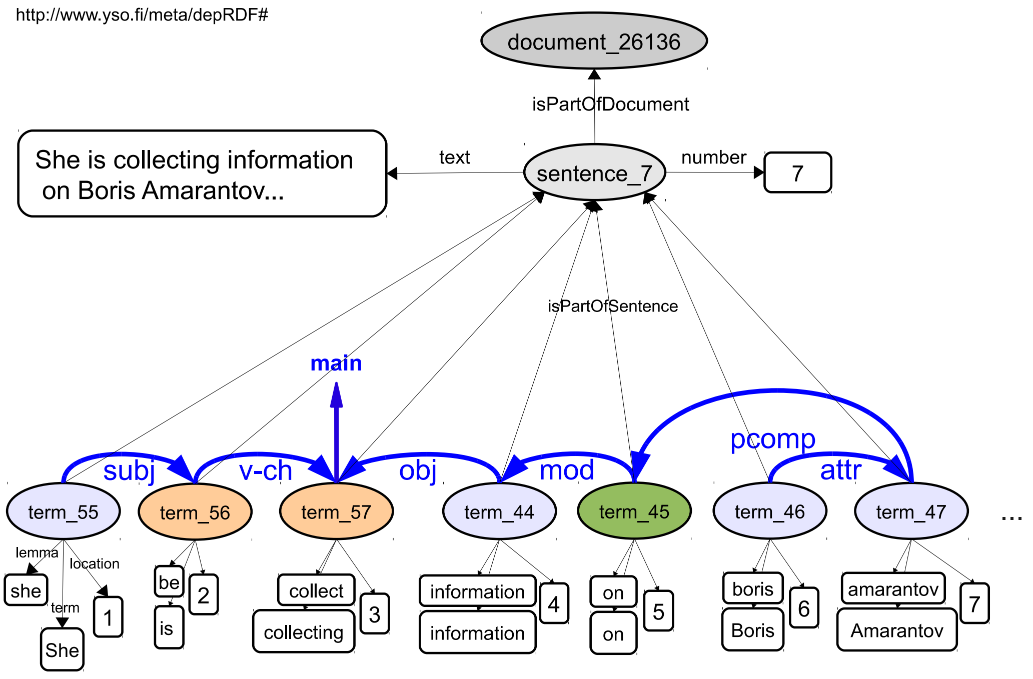


Figure 4.15: The resulting RDF incorporating the Machinese Syntax analysis.

columns. In a classification problem the class of each individual in the training set is known. From the values of the properties of each individual the model learns a set of rules that classify the individual points correctly. [Alp04]

In this case study each term was represented as one data point and the class of each term was the category to which the term's document belonged to. The category of each document was read from the SÄHKE metadata model and was represented by the group node of the metadata schema (see Figure 4.1). The properties for each individual term was all the information from the Machinese Syntax analysis that was represented using the RDF Schema of Figure 4.14.

The parsed result of the Machinese Syntax analysis, the RDF model, was turned into a two-dimensional CSV table. Each row represented an individual term and each column contained the property values for each term. For transforming an RDF model into a two-dimensional table, a programme was written that created a row for each resource of a certain given class type,

```
aski:term_911753
      <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
              aski:term ;
      aski:hasWordClass aski:verb ;
      aski:isPartOfDocument
              aski:document_2612 ;
      aski:lemma "olla" ;
      aski:location "9" ;
      aski:sentenceNr "349" ;
      aski:term "olevan" .

aski:document_26136
      <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
              aski:document ;
      <http://www.w3.org/2002/07/owl#sameAs>
              <http://www.narc.fi/onto#document11fb2fa110c177441> ;
      aski:lang "en" .
      aski:text "Date: Mon, 5 Nov 2007 22:37:24 +0300 (MSK)\nFrom: Ala
Buceatchi <buceatchi@yahoo.com>\nSubject: Youth Festival in Helsinki 1962
Russian Artist Boris Amarantov\nTo: kansallisarkisto@narc.fi\n\n\n\n\n\nDear
Sir, Madame,\n\nI am writing you on behalf of Mrs. Marianna Sorvina
\nUniversity professor, researcher, Moscow, Russia . She\nis collecting
information on Boris Amarantov, circus\nartist, mime, who has emigrated to USA
from URSS in\n1977.\nBefore emigration he performed in Europe, Japan, in\n1962
in Helsinki, at the Youth Festival;\nunfortunatelly, there is almost nothing
on his\nactivity and life after 1977.\nAfter returning to Moscow in 1987 Boris
Amarantov\ntragically died.\n\nI would be very grateful if you help us find any
\ninformation on Boris Amarantov activities in USA.\n\n \nLooking forward to
hearing from you soon,\n \nFaithfully yours, Ala Buceatchi, Belgium\n\n\n
________________________________________________________\nВы уже с Yahoo!?
\nИспытайте обновленную и улучшенную. Yahoo! Почту! http://ru.mail.yahoo.com" .
```

Figure 4.16: Example of the resulting TTL after the RDF transformation

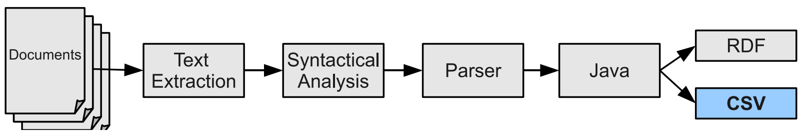


Figure 4.17: Document text in CSV file that can be utilized by the Machine Learning analysis.

`http://www.yso.fi/aski/term`. Each triplet containing the resource of this type as its subject was taken into account and the information it contained was put into the resulting table in the following way: all triplets were iterated and each of their predicates was turned into a column. Then each triplet was gone through again and its object was set as the value of the subjects row and the predicates column. If the object was a resource, its local name was set as the value. If the object was of type Literal[11] then its text was set as the value.

The programme also has two ways of setting the identity number information for each row. It either parses the ID value from a resource's URI or looks out for a given ID property. The ID information is always set as the first column.

The columns were as follows:

- ID, the unique running number for each term,
- ysoUri, the reference to a possible URI of the YSO ontology,
- term, the word in its original form,
- lemma, the term in its base form,
- isPartOfDocument, the reference to the document, from which the term originated from,
- sentenceNr, the number of the sentence in the document, where this word can be found,
- location, the order of this word in the sentence, and
- hasWordClass, the word class of this word.

In addition to these columns, each possible type of the 42 syntactical relations had a column of their own. The Machinese Syntax analysis gives only one syntactical relation for every word. Therefore on each row there was always only one syntactical relation column that had a value. This value contained the local name of the resource of the term with which this word had a dependency with.

Another way in which the syntactical relations could have been saved into the CSV table, was to create a column for the type of the relation and another for its target. This would have reduced the number of columns by 40. This option was however not chosen, because it was more interesting to

[11]`http://www.w3.org/TR/rdf-schema/#ch_literal`

create a general solution for creating two-dimensional tables from RDF files. This general solution creates a column for every property leading from the given type resource and thus generates a column for each type of syntactical relation.

The data was split into six files. The resulting data set contained 1,432,905 rows and 50 columns.

Chapter 5

Learning Document Categories

This chapter defines and describes the model that is used on the example data to learn the classification of the Archive's documents. The part of the project described in this chapter of the master's thesis was done in collaboration with the research group Bayesian Algorithms for Latent Variable Models of the Aalto University.

5.1 A Model for Logistic Discrimination

For learning the categories to which each document of the case study data set belongs, machine learning was used. The idea is to extend traditional logistic discrimination learning [Alp04] by combining it with relational background knowledge based on ontologies [SBF98, dBRHDA97]. Relational models for documents have been considered before, see for instance [PULP03], but with a very different methodology.

Each document is represented by a vector $\bar{x}_d \in \{\mathcal{R}^T\}$, which contains the tf-idf weights $tfidf_t \in \{\mathcal{R}\}$ for each term in the document and can be written as $\bar{x}_d = \{tfidf_1, tfidf_2, \ldots, tfidf_T\}$. As noted in Section 2.5 some of the weights might be 0, because a weight's respective term may not occur in the document. The matrix $\mathbf{X}$ contains all $\bar{x}_d$, $\mathbf{X} = (\bar{x}_1, \ldots, \bar{x}_N)$, where $\mathbf{X} \in \{\mathcal{R}^N \times \mathcal{R}^T\}$.

Dr. Tapani Raiko proposed to assume that the categories of the documents are separated by a linear discriminant (see Equation 2.21) and suggested using logistic discrimination for learning the vector of the weights of each term for the linear discriminant of a class C_i. The vector is $\bar{w}_i = (w_1,w_2,\ldots,w_T,w_{T+1})$, $w_t \in \mathcal{R}$, where T is the number of terms and w_{T+1} the constant of the vector. For the description of all indices used in this chapter, see Table 5.1.

symbol	range	stands for
d	$1\ldots N$	documents
t	$1\ldots T$	terms
i	$1\ldots K$	classes

Table 5.1: Meaning of indices used

A matrix $\mathbf{W} \in \{\mathcal{R}^K \times \mathcal{R}^{T+1}\}$ is defined, which contains all $\bar{w}_i$ for each class, $\mathbf{W} = (\bar{w}_1,\bar{w}_2,\ldots,\bar{w}_K)$.

The MAP-estimate for $\mathbf{W}$ is:

$$\mathbf{W}_{\text{MAP}} = \arg\max_{\mathbf{W}} p(\mathbf{W} \mid \bar{C},\mathbf{X}) = \arg\max_{\mathbf{W}} \frac{P(\bar{C} \mid \mathbf{X},\mathbf{W})p(\mathbf{W})}{P(\bar{C} \mid \mathbf{X})} \tag{5.1}$$

Because the denominator is the same for each $\mathbf{W}$ and the logarithm of the function will also find the MAP-estimate for $\mathbf{W}$, the estimate can be written in the following way:

$$\mathbf{W}_{\text{MAP}} = \arg\max_{\mathbf{W}} \left[\log P(\bar{C} \mid \mathbf{X},\mathbf{W}) + \log p(\mathbf{W})\right] \tag{5.2}$$

The priors for each w_{it} in $\mathbf{W}$ are considered as random variables and can be represented as a Gaussian distribution:

$$p(w_{it}) = N(0,\sigma_w^2) \tag{5.3}$$

where w_{it} is the weight of the i^{th} class and the t^{th} term. The weight is expected to be around zero and the σ_w^2 is learned from the data. [Alp04, p. 262]

5.2 The Model

Logistic discrimination for a multiple class problem is used as a model for the likelihood density $P(C_i \mid \bar{x}_d, \mathbf{W})$

$$P(C_i \mid \bar{x}_d, \mathbf{W}) = \frac{\exp \bar{w}_i^{\mathrm{T}} \begin{bmatrix} \bar{x}_d \\ 1 \end{bmatrix}}{\sum_j \exp \bar{w}_j^{\mathrm{T}} \begin{bmatrix} \bar{x}_d \\ 1 \end{bmatrix}}, \forall i, d. \tag{5.4}$$

The dimensionality of $\bar{x}$ was reduced by applying PCA and replaced with $\mathbf{U}^{\mathrm{T}}\bar{x}$ (see Equation 2.33), which results in the following model:

$$P(C_i \mid \bar{x}_d, \mathbf{W}) = \frac{\exp \bar{w}_i^{\mathrm{T}} \begin{bmatrix} \mathbf{U}^{\mathrm{T}}\bar{x}_d \\ 1 \end{bmatrix}}{\sum_j \exp \bar{w}_j^{\mathrm{T}} \begin{bmatrix} \mathbf{U}^{\mathrm{T}}\bar{x}_d \\ 1 \end{bmatrix}}, \forall i, d. \tag{5.5}$$

5.3 Enhancing the Analysis with Knowledge from Ontologies

The model was enhanced with matrices that represent ontology information from YSO [NRTH10]. The relations between terms can be represented by a binary matrix $\mathbf{A}$ that has as many columns and rows as there are terms, and is thus of size T×T:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \dots & \dots & \dots & a_{1,T} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \vdots & \dots & a_{i,j} & \dots & \vdots \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{T,1} & \dots & \dots & \dots & a_{T,T} \end{pmatrix}, \; a_{i,j} \in \{0,1\}$$

where $a_{i,j}$ is 1, if a relation between the i^{th} and j^{th} term exists and 0 otherwise. Different kinds of binary matrices can represent different kinds of relations. A group of matrices $\mathbf{A}_r$, $r \in \{$"hyponyms", "hypernyms", "hyponyms of hyponyms", ..., "associative relations"} of size T×T is defined to represent all relations (see Table 5.2 for full description).

$\mathbf{A}_r$	*relation*	*note*
$\mathbf{A}_1$	hyponyms	Apple is the subclass of fruit, thus the term $a_{\text{apple,fruit}} = 1$
$\mathbf{A}_2$	hypernyms	transpose of $\mathbf{A}_1$
$\mathbf{A}_3$	associative relations	rain is associated with an umbrella thus the term $a_{\text{rain,umbrella}} = 1$ and $a_{\text{umbrella,rain}} = 1$
$\mathbf{A}_4$	hyponyms of hyponyms	$\mathbf{A}_1\mathbf{A}_1$
$\mathbf{A}_5$	hypernyms of hypernyms	$\mathbf{A}_2\mathbf{A}_2$
...	...	...

Table 5.2: The matrices for ontology information. If for example an ontology expansion of two levels up and one level down is to be made and associative terms are to be accounted for, then the matrices $\mathbf{A}_1$, $\mathbf{A}_2$,$\mathbf{A}_3$ and $\mathbf{A}_5$ are used.

The attractiveness of this approach is that all possible extensions of the hyponymy relations can be created by simple multiplications or the transpose of the matrix $\mathbf{A}_1$. Note also that the matrix representing the associative relations is symmetric. A weight α_r is assigned for each binary matrix $\mathbf{A}_r$ and each weight can be learned from the training set.

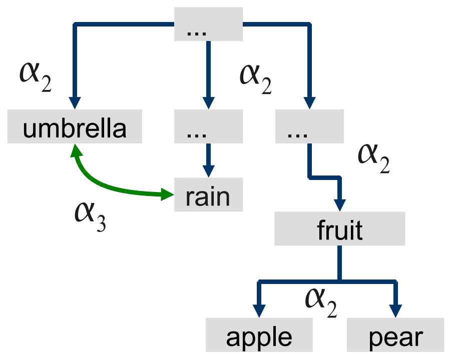


Figure 5.1: The model learns all specified α_r weights for the given ontology and the matrices representing it.

The weight α_r, once learned, can be used as a measure for the relevance of the relation r. In Figure 5.1 α_2 represents the weight that is given to the hypernym relations and α_3 that for associative relations.

The term vector $\bar{x}_d$ can be multiplied with all matrices $\mathbf{A}_r$ and their respective scalars α_r. The sum of them results to a new $\bar{y}_d$:

$$\bar{y}_d = \sum_r^R \alpha_r \mathbf{A}_r \bar{x}_d, \tag{5.6}$$

which can be used to replace $\bar{x}_d$ in the proposed model. Now dimensionality reduction can be applied and the matrix $\mathbf{U}$ multiplied with $\bar{y}_d$:

$$\begin{aligned} \mathbf{U}^{\mathrm{T}}\bar{y}_d =& \mathbf{U}^{\mathrm{T}} \sum_r^R \alpha_r \mathbf{A}_r \bar{x}_d & (5.7) \\ =& \sum_r^R \alpha_r \mathbf{U}^{\mathrm{T}} \mathbf{A}_r \bar{x}_d. & (5.8) \end{aligned}$$

Here $\sum_r^R \mathbf{U}^{\mathrm{T}} \mathbf{A}_r \bar{x}_d$ can be computed in advance, because it consists of known factors. Thus only $\bar{w}_i^{\mathrm{T}}$ and α_r remain to be trained using Gradient Ascent. The weight α_r is assumed positive and thus the initial weights were all set to 1.

By replacing $\bar{x}_d$ with $\bar{y}_d$ the model is defined in the following way:

$$P(C_i \mid \bar{x}_d,\mathbf{W}) = \frac{\exp \bar{w}_i^{\mathrm{T}} \begin{bmatrix} \sum_r^R \alpha_r \mathbf{U}^{\mathrm{T}} \mathbf{A}_r \bar{x}_d \\ 1 \end{bmatrix}}{\sum_j \exp \bar{w}_j^{\mathrm{T}} \begin{bmatrix} \sum_r^R \alpha_r \mathbf{U}^{\mathrm{T}} \mathbf{A}_r \bar{x}_d \\ 1 \end{bmatrix}}, \forall i,d. \tag{5.9}$$

Chapter 6

Results

This chapter presents the classification results of 24 different trained models and discusses the meaning of the results.

6.1 Experimental Results

6.1.1 Training the Model

The model with and without ontology information was trained with a set of 500, 1000 and 1500 documents. This was done in order to see how increasing the set size affects the results. One hypothesis was, that the added ontology information would have a bigger effect on a smaller set, because a smaller set would mean less information and ontology information would patch that missing information. Thus three different set sizes were used for training the model. A set size bigger than 1500 could not be used, because some of the documents had to be used for testing the model.

A dimensionality reduction using PCA was performed on all sets before training. This was done in order to minimise the complexity of the model. Not all information is important and should be reduced, but on the other hand dimensionality should not be reduced too much, because that would lead to too much information loss and an unfunctional model. The dimensionality was reduced to 20 and 50.

Ontology information was added to the model by trying out 0, 1, 2 and 3

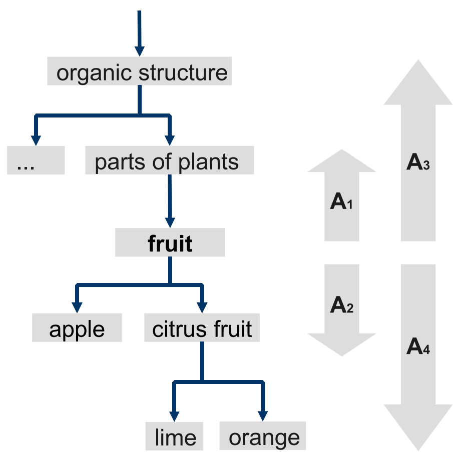


Figure 6.1: Hyponym and hypernym information extended to one, A_1 and A_2, and two levels, A_3 and A_4.

levels of ontology expansions. First the model was trained with no ontology expansions and only one binary matrix, the identity matrix, was used, so that $A_0 = I$. Then hyponyms and hypernyms extending to one level, that is to say ontology information about a term's parent(s) and children, was added to the set of A_r, r = 0,1,2 used by the model. Hyponyms and hypernyms of two levels added to that model ontology information about the term's grandparent(s) and grandchildren, and the ontology expansion of three levels added ontology information about great-grandparent(s) and great-grandchildren. An expansion up to 2 levels is illustrated in Figure 6.1, where the term fruit is extended first to comprise parts of plants (A_1), apple and citrus fruit (A_2) and then further to organic structure (A_3), and lime and orange (A_4).

6.1.2 Test Results of Trained Models

Overall 24 models were trained. Table 6.1.2 shows the accuracy rates of those models when tested with a test set of documents that were not in the training set.

The models with the best accuracy rate per dimension and data set size are shown in bold. The columns in the table titled MI1, MI2 and MI3 are

PCA	set size	accuracy rate with ontology expansions				MI1	MI2	MI3
		none	1	2	3			
20	500	60.87 %	63.39 %	**63.56 %**	63.46 %	2.69 %	0.17%	0.07%
	1000	62.99 %	68.68 %	**68.73 %**	68.68 %	5.74 %	0.05 %	0.16%
	1500	63.11 %	69.06 %	69.05 %	**69.07 %**	5.96 %	0.02 %	0.16%
50	500	64.61 %	68.60 %	**68.81 %**	68.79 %	4.20 %	0.21 %	0.12%
	1000	67.88 %	71.91 %	**71.96 %**	71.83 %	4.08 %	0.13 %	0.13%
	1500	70.26 %	74.08 %	**74.18 %**	74.12 %	3.92 %	0.10 %	0.13%

Table 6.1: Accuracy rates for models trained with a set of 500, 1000 and 1500 documents, PCA reduced dimensions to 20 and 50, and with ontology expansions of levels 0, 1, 2 and 3. MI1 and MI2 mark the maximum improvements on the accuracy rates and MI3 the maximum improvement on the error rate.

the numbers measuring the maximum improvement on accuracy and error rate. MI1 is the maximum improvement of the accuracy rate when ontology information is added, MI2 is the maximum improvement when using different kinds of ontology expansions and MI3 is the relative maximum improvement on the error rate when ontology information is added. Equation 6.1 shows how these numbers are obtained. In the equation ar_l is short for the accuracy rate for the level of ontology expansion l = 0,1,2,3.

$$\begin{aligned} \mathrm{MI1} &= \max(ar_1,ar_2,ar_3) - ar_0 \\ \mathrm{MI2} &= \max(ar_1,ar_2,ar_3) - \min(ar_1,ar_2,ar_3) \\ \mathrm{MI3} &= \frac{\mathrm{MI1}}{100\% - ar_0} \end{aligned} \tag{6.1}$$

The best accuracy rate of 74.18 % was reached with the model that was trained using a set of 1500 documents, was PCA reduced to 50 dimensions and used an ontology expansion of 2. In five out of six cases the maximum accuracy rate was reached by using an ontology expansion of 2 levels and the accuracy rate did not vary that much between different kinds of ontology expansions. Especially as the training set grew, the maximum improvement on the accuracy rate using different kinds of ontology expansions (see MI2 column) became smaller.

At its best the accuracy rate improved by 5.96 % when adding ontology

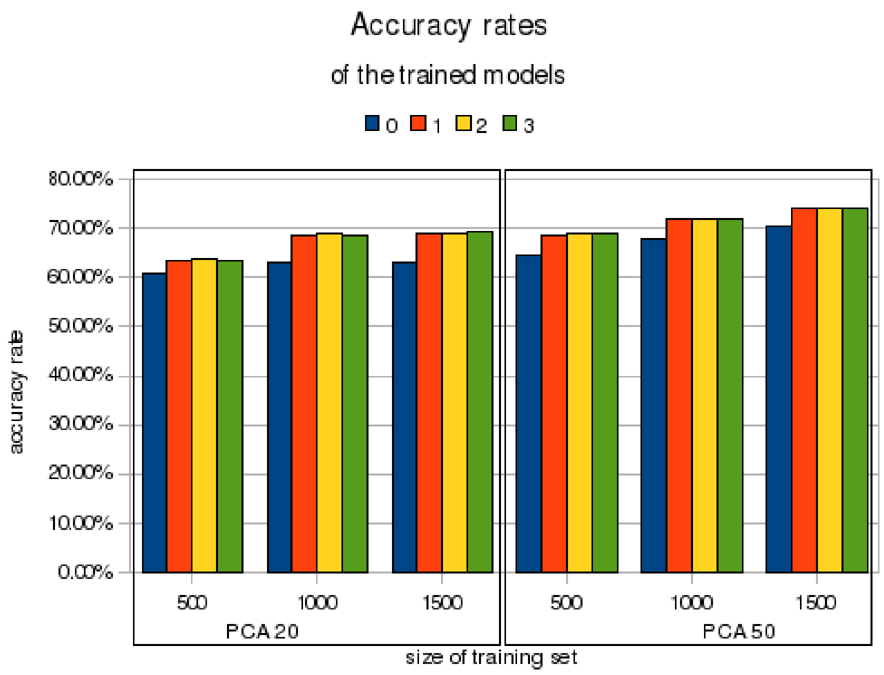


Figure 6.2: Accuracy rates for all 24 trained models

information to the model. This means that the overall error was reduced by almost a sixth of its original size (see MI3 column). The overall accuracy also always improved as the size of the training set grew. The results does not confirm the hypothesis that the added ontology information would have a bigger effect on a smaller set.

One thing that is very interesting to note here is that increasing the data set from 500 to 1500 improves the accuracy rate as much as adding ontology information to the model. The results show that with a PCA reduction of 20 an ontology expansion of one level improved the accuracy rate more than tripling the data set from 500 to 1500 documents. With a PCA reduction of 50, not even an ontology expansion of two levels improved the accuracy rate as much as tripling the data set.

Adding a binary matrix A_r adds a new weight α_r for the model to learn and increases its computational complexity by as much as doubling the set size. Therefore an ontology expansion of one level is more efficient and an ontology expansion of two levels is less efficient than tripling the data set. These points on computational efficiency should be considered when choosing

A_r matrices used in training the model.

6.2 Interpretation of the Results

The accuracy rate of the automatic classification was, at its best, almost 75 %. This might not sound like much but if one considers the alternative, no automatic classification at all, then three out of four correctly guessed classes would be of great assistance to an archiver working at the Finnish National Archives.

The trained model could be used by the Finnish National Archives to make automatic suggestions on classes on a not yet classified but already written document. An application could be built that would use all the parts that were used in this master's thesis as well. First the language, Finnish, Swedish or English, would be recognised. Then the text would be broken down to its syntactical structure and each term would be matched with an ontology concept from YSO. If the applications detects multiple matches for one term, the user of the application would be asked to assist and decide which concept is the correct match. Then the trained model would be applied on the new material and the user would be presented with a list of the 115 classes, which are ordered by the likelihood the model has assigned to them.

Instead of picking a class from a set of 115 classes the user would only need to help out a little bit in concept matching and pick a class from the list, where in three cases out of four the most correct class is at the top of the list. In addition to this the documents themselves would be more easily searchable, because they would have been annotated by the above described application. The manual help of the application user done on the application's automatic concept matching would make sure that the annotation is done in a correct way.

A model's classification success in ML is always dependent on the data set at hand. The more data available, the more accurately the model can perform [Alp04, p. 34]. The improvement of the accuracy rate when ontology information was added also suggests that this approach fits very well for documents that are written by humans, such as an email or a report. Perhaps larger data sets with documents written by humans could lead to even better results.

6.3 Evaluation of the Ontology Used

It is challenging to compare this result to other similar results as it depends so much on the classification task at hand and the material used. The purpose of this master's thesis was to answer whether adding ontology information improves the classification and the answer to that is yes. To what extent the classification is improved is trickier to answer and not explored further in this master's thesis. Whether the classification could be improved by using another ontology is a valid question and something to leave for future work.

YSO has its challenges. It is an ontology that has been created from a thesaurus that was originally created for humans to index material such as books at a library. It is a general and domain-unspecific ontology that, for example, contains detailed information on dog species but lacks weekdays except Sunday as a concept. A lot of work has been done to transform its logical structure from one for human comprehension on things to one a machine can correctly reason on. But it still requires work on its actual content so that it will accumulate even more general knowledge.

YSO contains almost 25,000 concepts[1] but WordNet contains around 200,000 of which half are nouns[2]. WordNet translated to the Finnish and Swedish language could, because of its size, contain more knowledge and lead to a greater improvement on the accuracy rate of the model. WordNet is a little bit different though from YSO because it contains synset, a set of synonymous words, for its concepts. The binary matrices A_r that provide the knowledge expansion could be modified to describe the relations of WordNet and the information about synsets. By training and testing the model, the usefulness of the background knowledge provided by the WordNet's synsets could be evaluated.

The research group SeCo has developed an ontology that is based on YSO but is specific to the domain of State Administration and is called JUHO. Perhaps JUHO contains concepts that are also present as words in the cases of the Finnish National Archives, because it is a government body under the Ministry of Education.

[1]YSO Statistics: `http://www.yso.fi/onki/yso/?p=ontology-info&l=en`

[2]WordNet statistics: `http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html`

Chapter 7

Related and Future Work

This chapter looks at other similar research work and compares it to the work done in this master's thesis. Future research work is proposed and discussed.

7.1 Related Work

Enhancing traditional machine learning techniques, such as the bag of words approach, has been researched for around 15 years. A lot of the research on enhancing ML is done on hierarchical information, because the use of it enables powerful generalisations [GM05]. For example, two different documents, where one mentions only "pork" and the other only "beef", can easily be linked together, because from an ontology one can see that the terms' hypernym is "meat" [HSS03].

Taskar et al. [TAK02] introduced Relational Markov Networks and applied it to collaborative classification of related documents. Popescul et al. [PULP03] used a relational model using both citation and coauthorship data. In collective classification, information about the predicted classes propagates over a network defined by the relations, see [SNB$^+$08] for an overview. The method in this master's thesis differs from these in two ways: Firstly, it does not use the class information of related documents, only their observed term counts. Secondly, it also uses the relations between terms.

Wittbrock et al. [WCKR09] from Cycorp, Inc. use geographical subsumption information from the company's own ontology to enhance location information for terrorist attack predictions. The probabilistic model they use benefits from the additional information. If an attack happened in Madrid, the probabilistic model can also comprehend that it happened in Spain and update the probabilities appropriately.

A hierarchical set of concepts that is repeatedly used to enhance the performance of traditional ML models is WordNet, the English lexical database. It contains words presented as pairs of a word's lexical form, the string representation, and its meaning. The concepts are linked with pointers that mark a semantic relation between them. The semantic relations can be synonymy, antonymy (the opposite of synonymy), hyponymy, meronymy, troponymy, which is the equivalent of hyponymy for verbs, and entailment, which marks an associative relation between verbs. The target of the semantic relations of a concept are packed in a set called the synset of the concept. [Mil95]

Scott and Matwin [SM98], Rodríguez et al. [dBRHDA97], Ureña-López et al. [ULBG01], and Hotho et al. [HSS03] augment automatic document classification by using the synsets of WordNet in different ways to calculate the weights for various ML models.

[ULBG01] (and its earlier version [dBRHDA97]) use the Vector Space Model (VSM) to represent the text of documents and the categories, to which the documents need to be classified. They use the cosine formula to measure similarity between documents and categories. The weights for the documents are calculated with tf-idf and the weights of the categories are calculated using the Rocchio and the Widrow-Hoff algorithm. The categories are also expanded with the synonyms of the synsets from WordNet. Both research studies show, that the automatic categorisation of documents improves as information from WordNet is added to expand the categories. In the latter research the Word Sense Disambiguation (WSD) architecture approach is elaborated more thoroughly and the precision rates have improved from circa 50% to 65% on average.

[SM98] use WordNet synsets for expanding the text representation of the documents. They compare their approach to that of [dBRHDA97] and find that Rodríguez et al. approach is not sufficient enough, as synonyms are picked manually. [SM98] add the synonyms and hypernyms of all verbs and nouns to the set of terms. As in this research they, too, tried out different levels of generalisation with hypernyms and found best results with generalisation level of 2.

[HSS03] expand the text representation with synsets from WordNet, as well. They test three different ways of augmenting the term weights by firstly adding up all terms weights of a word's synset's concepts to the word's term weight, secondly using only the first concept from the word's ordered synset, and by thirdly picking the concepts from the word's synset that at their best represent the document's context. The third approach for WSD together with an expansion of a terms hypernyms and hyponyms up to five levels seems in their research to create best results.

Gabrilovich and Markovitch [GM05] enhance their bag of words model with the hierarchical information of the Open Directory Project, which in their words *embeds a colossal amount of human knowledge*. Appropriate concepts are deduced based on the document's text only and are added to the bag of words model.

Shehata et al. [SKK08] suggest a different approach from the standard bag of words model to measure the importance of words in a document. Instead of assuming that every term is of equal importance, which is one of the assumptions made in the bag of words model, they suggest a concept-based model, that looks at the word's syntactical relation inside a sentence and determines the importance of each term to the meaning of the sentence. A sentence is broken down to labelled verb-argument structures using ProBank [KP03] and the importance of a term is measured by the number of times it occurs in each verb-argument. This measure, called the concept frequency, adds to the term's term frequency normally calculated in the bag of words approach.

7.2 Future Work

7.2.1 Relations Between Documents

The model used in this master's thesis could be extended to incorporate relations between documents. In the same way as the model was expanded with T×T binary matrices that portrayed ontology information, the model could incorporate N×N binary matrices that contain information about the relations between documents. In the same as a matrix $\mathbf{A}_1$ provides hyponym relations between terms, a matrix $\mathbf{B}_1$ could provide shared-author relations between documents. With the help of both matrices, a document describing apples and written by an author that writes a lot about fruit could be classified correctly to the category fruit. [NRTH10]

7.2.2 Measuring Ontology Applicability

The machine learning model created for the classification task in this master's thesis could also be used as a way to measure an ontology's applicability to a collection of text material. Each type of relation in an ontology would be represented by a binary matrix A_r and the corresponding weights α_r for each relation would be learned from training the model. These weights are initially set to 1, so if the trained model has a weight α_r that is above 1, then it would mean that the kind of relation represented by A_r is of relevance when classifying the specified documents to a specified set of classes.

Multiple ontologies could be tested on the material and the weights that the model learns for each ontology could be compared to each other. Reason suggests that the ontology with the highest weights for each relation would be the ontology with the highest applicability for the material at hand. Perhaps that ontology could then be used for annotating new material from the same source or similar kinds of material such as material from the same domain.

Chapter 8

Summary

This master's thesis tested a way of combining machine learning and ontology information to learn the classification of a set of documents provided by the Finnish National Archives. First it provided a basis for understanding how machine learning is used to learn the classification of documents. Then the semantic web and its definition were explored and the idea of ontologies were explained.

A way to model a document, the words it contains, and the syntactical relations between the words using RDF was suggested and described. It was used to model the documents from the Finnish National Archive. The process of extracting the information from those documents and making them ready for machine learning analysis was illustrated step-by-step. A machine learning model was proposed that could learn the classification of documents using ontology information as background knowledge. This model proved to be quite successful, because it showed that adding ontology information up to two levels improves automatic document classification.

Computers are good for completing specifically programmed tasks. Tasks that require a lot of repetition and can be explicitly explained, should be left to machines, because humans can get tired of repetition and can experience fatigue and make mistakes. Some tasks such as classification of documents cannot be explained explicitly to a machine, but a statistical model with good enough accuracy can be learned from background data. The model can then be used for the purpose of completing the task.

Machine learning and the idea of the semantic web are based on the idea of artificial intelligence, where autonomic computer agents complete tasks automatically according to the needs of humans. Perfecting such systems can take a lot of time and effort. A solution, where not all, but the majority of the work is done correctly, is easier to develop and does not need a lot of perfecting before it can be implemented. A good enough solution like that would be automatic to a point, but would then need the intervention of a human to complete its task.

The Galaxy Zoo project [RBG$^+$10] use volunteers with internet connections to classify satellite pictures of galaxies based on their morphology. The classification is hard for the machine to complete, because the galaxies are photographed from different angles and their colour and brightness against a dark and uneven background vary a lot. But for humans, recognising the shapes of the galaxies despite all variations is child's play, and that capability is harnessed to make the classification of the galaxies possible.

This master's thesis showed how machine learning and ontology information can be used to classify the majority of documents, provided an RDF model for document representation, and suggested a way in which that kind of automation can be combined with reasonable manual work to obtain best results.

Bibliography

[Alm07] Olli Alm. Tekstidokumenttien automaattinen ontologiaperustainen annotointi. Master's thesis, University of Helsinki, Department of Computer Science, September 2007.

[Alp04] Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press, 2004.

[Alv01] H. Alvestrand. RFC 3066 - Tags for the Identification of Languages. IETF, 2001. `http://www.isi.edu/in-notes/rfc3066.txt`.

[Ano05] Anonymous. Asiankäsittelyjärjestelmiin sisältyvien pysyvästi säilytettävien asiakirjallisten tietojen säilyttäminen yksinomaan sähköisessä muodossa (specifications for the permanent storage of information on digital documents to be contained in case treatment systems). Finnish National Archive, 2005. `http://www.narc.fi/Arkistolaitos/pdf-ohjeet/akj_maarays.pdf`.

[Ano06] Anonymous. Machinese linguistic analysers. Connexor Oy, 2006.

[BBL08] Dave Beckett and Tim Berners-Lee. Turtle - terse RDF triple language. W3C Recommendation, 2008. `http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/`.

[BG04] Dan Brickley and R.V. Guha. RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, 2004. `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`.

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer Verlag, 2007.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. 284(5):34–43, 2001.

[BM01] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. pages 245–250, 2001.

[BM09] Mark Birbeck and Shane McCarron. Curie syntax 1.0. W3C Recommendation, 2009. `http://www.w3.org/TR/curie/`.

[Bor97] Willem N. Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, Dutch research school for Information and Knowledge Systems (SIKS), 1997.

[BvHH$^+$04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language reference. W3C Recommendation, 2004. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`.

[CDPW02] Fabio Ciravegna, Alexiei Dingli, Daniela Petrelli, and Yorick Wilks. User-system cooperation in document annotation based on information extraction. Springer-Verlag Berlin Heidelberg, 2002.

[Cor06] Oscar Corcho. Ontology based document annotation: Trends and open research problems. *International Journal of Metadata, Semantics and Ontologies*, 1(1):47–57, 2006.

[dBRHDA97] Manuel de Buenaga Rodríguez, José María Gómez Hidalgo, and Belén Díaz-Agudo. Using wordnet to complement training information in text categorization. In *Recent Advances in Natural Language Processing II*, volume 189, pages 353–364, 1997.

[DEG$^+$03] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. SemTag and seeker: bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 178–186, New York, NY, USA, 2003. ACM.

[GM05] Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *IJCAI'05*, pages 1048–1053. Morgan Kaufmann Publishers Inc., 2005.

[GMM03] R. Guha, Rob McCool, and Eric Miller. Semantic search. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 700–709, New York, NY, USA, 2003. ACM.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[GW04] Nicola Guerino and Christopher A. Welty. An overview of OntoClean. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 151–172, 2004.

[HSM01] Siegfried Handschuh, Steffen Staab, and Alexander Maedche. Cream — creating relational metadata with a component-based, ontology-driven annotation framework. In *Proceedings of K-Cap 2001,Victoria, BC, Canada*, 2001.

[HSS03] Andreas Hotho, Steffen Staab, and Gerd Stumme. Ontologies improve text document clustering. In *3rd IEEE International Conference on Data Mining*, pages 541–544, 2003.

[HSVF07] Eero Hyvönen, Katri Seppälä, Kim Viljanen, and Matias Frosterus. Yleinen suomalainen ontologia YSO — kohti suomalaista semanttista webiä (general finnish ontology YSO — towards the finnish semantic web). *Tietolinja*, (1), May 2007. `http://www.seco.hut.fi/publications/2007/hyvonen-et-al-yso-2007.pdf`.

[HVK$^+$05] Eero Hyvönen, Arttu Valo, Ville Komulainen, Katri Seppälä, Tomi Kauppinen, Tuukka Ruotsalo, Mirva Salminen, and Anu Ylisalmi. Finnish national ontologies for the semantic web - towards a content and service infrastructure. In *Proceedings of International Conference on Dublin Core and Metadata Applications (DC 2005)*, Nov 2005.

[Hyv05] Eero Hyvönen. Miksi asiasanastot eivät riitä vaan tarvitaan ontologioita? *Tietolinja*, (2), Oct 2005. `http://www.cs.helsinki.fi/group/seco/publications/2005/hyvonen-miksi-asiasanastot-eivat-riita-2005.pdf`.

[KP03] Paul Kingsbury and Martha Palmer. PropBank: the next level of TreeBank. In *Proceedings of Treebanks and Lexical Theories*, 2003.

[Mil95] George A. Miller. Wordnet: a lexical database for english. In *Communications of the ACM*, volume 38, pages 39–41, 1995.

[MM04] Frank Manola and Eric Miller. RDF primer. W3C Recommendation, 2004. `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`.

[MRS08] Christian D. Manning, Prabhakar Raghavan, and Heinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[MS00] Christopher D. Manning and Heinrich Schütze. *Foundation of Statistical Natural Language Processing.* The MIT Press, 2000.

[NRTH10] Katariina Nyberg, Tapani Raiko, Teemu Tiinanen, and Eero Hyvönen. Document classification utilising ontologies and relations between documents. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, MLG '10, pages 86–93, New York, NY, USA, 2010. ACM.

[Nyb08] Katariina Nyberg. Ontologian arviointi OntoClean-menetelmällä, Bachelor's Thesis, Faculty of Information and Natural Sciences, Helsinki University of Technology, December 2008.

[PULP03] Alexandrin Popescul, Lyle H. Ungar, Steve Lawrence, and David M. Pennock. Statistical relational learning for document mining. In *Proceedings of IEEE International Conference on Data Mining (ICDM-2003)*, pages 275–282, 2003.

[RBG$^+$10] M. Jordan Raddick, Georgia Bracey, Pamela L. Gay, Chris J. Lintott, Phil Murray, Kevin Schawinski, Alexander S. Szalay, and Jan Vandenberg. Galaxy zoo: Exploring the motivations of citizen science volunteers. *Astronomy Education Review*, 9(1), 2010.

[RVOK03] Tapani Raiko, Harri Valpola, Tomas Östman, and Juha Karhunen. Missing values in hierarchical nonlinear factor analysis. In *Proc. of the Int. Conf. on Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP 2003)*, pages 185–189, Istanbul, Turkey, 2003.

[SBF98] Rudi Studer, V. Richard Benjamins, and Dieter Fendel. Knowledge engineering: Principles and methods. *IEEE Transactions on Data and Knowledge Engineering*, 25(1–2):161–197, 1998.

[SKK08] Shady Shehata, Fakhri Karray, and Mohamed Kamel. A concept-based model for enhancing text categorization. In C. Tang et al., editor, *ADMA*, pages 87–98, 2008.

[SM98] Sam Scott and Stan Matwin. Text classification using wordnet hypernyms. In *Proceedings of the Joint 17th International Conference on Computational Linguistics 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL'98) workshop on "Usage of WordNet in Natural Language Processing Systems"*, pages 45–51, 1998.

[SM05] Eija Sorakivi and Markus Merenmies. SÄHKE-hanke, abstrakti mallintaminen (abstract modelling of the SÄHKE project). Finnish National Archive, 2005. `http://www.narc.fi/sahke/Aineisto/SAHKE-abstrakti-V2-koko.pdf`.

[SNB$^+$08] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3), 2008.

[TAK02] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI02)*, pages 485–492, 2002.

[TM03] David M.J. Tax and Klaus-R. Müller. Feature extraction for one-class classification. In Okyay Kaynak, Ethem Alpaydin, Erkki Oja, and Lei Xu, editors, *Artificial Neural Networks and Neural Information Processing*, pages 342–349. ICANN/ICONIP, Springer-Verlag, Heidelberg, 2003.

[ULBG01] L. Alfonso Ureña-López, Manuel Buenaga, and José M. Gómez. Integrating linguistic resources in TC through WSD. *Computers and the Humanities*, 35(2):215–230, 2001.

[Val06] Onni Valkeapää. Verkkoresurssien ontologiaperustainen annotointi. Master's thesis, Helsinki University of Technology, Department of Mediatechnology, September 2006.

[Veh06] Antti Vehviläinen. Ontologiapohjainen kysymys-vastauspalvelu (ontology-based question-answer service). Master's thesis, Helsinki University of Technology, Department of Mediatechnology, October 2006.

[Wal05] Chip Walter. Kryder's Law. *Scientific American*, August 2005. `http://www.scientificamerican.com/article.cfm?id=kryders-law`.

[WCKR09] Michael Witbrock, Elizabeth Coppock, Robert Kahlert, and Benjamin Rode. Cyc-enhanced machine classification. Technical report, Cycorp Inc., 2009. `http://www.cyc.com/cyc/technology/whitepapers_dir/Cyc-Enhanced_Classification.pdf`.