EU-IST Project IST-2003-506826 SEKT

SEKT: Semantically Enabled Knowledge Technologies

D1.9.1 Simultaneous ontologies

Blaž Fortuna , Marko Grobelnik, Dunja Mladenić
(Jožef Stefan Institute)

Abstract

In this deliverable we describe a solution for incorporating background knowledge into the OntoGen system for semi-automatic topic ontology construction which was developed as a part of deliverable *D1.7.1 Ontology generation from scratch*. This makes it easier for different users to construct different topic ontologies on the top of same document collection. To achieve this, a word weighting is learned based on the user's background knowledge and than used bz OntoGen's machine learning and text mining algorithms.

Keyword list: simultaneous ontologies, multi-view, word weighting schemas, learning word weighting

WP1

Prototype/Report PU

Contractual date of delivery: 31.12.2005

Actual date of delivery: 22. 12. 2005

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.
Orion 5/12, Adastral Park
Ipswich IP5 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Empolis GmbH
Europaallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540
Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

Jozef Stefan Institute
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Karlsruhe, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592
Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891
Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

University of Innsbruck
Institute of Computer Science
Techikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475
Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Intelligent Software Components S.A.
Pedro de Valdivia, 10
28006
Madrid
Spain
Tel: +34 913 349 797
Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Kea-pro GmbH
Tal
6464 Springen
Switzerland
Tel: +41 41 879 00
Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Ontoprise GmbH
Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912
Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Sirma Group Corp., Ontotext Lab
135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Vrije Universiteit Amsterdam (VUA)
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Universitat Autonoma de Barcelona
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vall` es)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

Siemens Business Services GmbH & Co. OHG
Otto-Hahn-Ring 6
81739 Munich
Germany
Contact person: Dirk Ramhorst
Tel: +49 (89)63640225; Fax: +49 89 63640233
Email: Dirk.Ramhorst@siemens.com

Executive Summary

The aim of this document is to present a way for incorporating background knowledge into the semi-automatic topic ontology learning process. This enables the machine learning algorithms to discover concepts and relations from a document collection which better reflect the user's view of the collection. More importantly, it enables users with different background knowledge (or different interests, views, etc.) to construct different topic ontologies from the same document collection using the same machine learning algorithms. This is achieved trough the use of word weighting schemas which form the basis of bag-of-words text document representation most commonly used in machine learning. Different schemas are learned for different users, reflecting their background knowledge.

This deliverable starts by making an overview of different approaches to learning word weights. It is followed by a description of the chosen approaches, the experiments and details about the software implementation. Detailed description of the requirements, availability and use of the developed software tools is given in the Appendix.

The methods developed as part of this deliverable are integrated into the tool for semi-automatic topic ontology construction called OntoGen which was developed within SEKT project (delivereable *D1.7.1 Ontology generation from scratch*) and other tools from Text Garden.

Contents

1 Introduction

When using ontology-based techniques for knowledge management it is important for the ontology to capture the knowledge in a proper way. Very often different tasks require the knowledge to be encoded in ontology in different ways, depending on the task. For instance, the same document-database in a company may be viewed differently by marketing, management, and technical staff. Therefore it is crucial to develop techniques for building multiple ontologies from the same data where a specific ontology captures one of the possible views. In other words, we build *simultaneous ontologies* on the same data.

As part of deliverable D1.7.1 [Fortuna05] we developed a system called OntoGen for semi-automatic construction of topic ontologies. Topic ontology consists of a set of topics (or concepts) and a set of relations between the topics which best describe the data. The OntoGen system helps the user by discovering possible concepts and relations between them within the data.

As a part of deliverable D1.9.1 we improve this system so that the user can supervise the methods for concept discovery by providing background knowledge on the specific user's view on the data used by the system. This allows the user to build an ontology which captures the user view on the data.

To encode the user's background knowledge we require from the user to group documents into categories. These categories do not need to describe the data in details, the important thing is that they show to the system the user's view of the data – which documents are similar and which are different from the user's perspective. The process of manually marking the documents with categories is time consuming but can be significantly speeded up by the use of active learning, for instance, using the work developed inside deliverable D1.2.1 [Novak04].

In this report we present how background knowledge provided in this way can be used for guiding the methods for concept discovery. The main focus will be on finding the similarity measure which captures the user's background knowledge. This measure can then be used to guide the concept discovery and other text mining methods used in OntoGen.

Please note that terms "category" and "topic" are not the same in this report. The input needed for incorporating user's background knowledge into the similarity measure is given by providing groups of documents which are similar to in user's view. We refer to these groups as "categories". The term "topic" refers to concepts in the topic ontology.

In Chapter 2 we review the work that has been done so far in the field of learning similarity measures. In Chapter 3 we present our approach to this task and provide an idea about its performance while in Chapter 4 we present the architecture of our system. In Chapters 5 and 6 we lay down the plans for future development and we finish with conclusions. Detailed information about the developed software components is available in Appendixes.

2 Related Work

An important part of OntoGen [Fortuna05] are methods for discovering concepts from a collection of documents. For the representation of documents we use the well established bag-of-words representation, where each document is encoded as a vector of term frequencies and the similarity of a pair of documents is calculated by the number and the weights of the words that these two documents share. This method heavily relies on the weights associated with the words – the higher the weight of a specific word the more probable that the two documents are similar if they share this word. The weights of the words are commonly calculated by so called TFIDF weighting. We argue that this provides just one of the possible views on the data and propose an alternative word weighting that enables taking into account the background knowledge providing the user's view on the documents.

OntoGen discovers concepts using Latent Semantic Indexing (LSI) [Deerwester90] and k-means clustering [Jain99]. The LSI is a method for linear dimensionality reduction by learning and optimal sub-basis for approximating documents' bag-of-words vectors. The sub-basis vectors are treated as concepts. The k-means method discovers concepts by clustering the documents' bag-of-words vectors into k clusters where each cluster is treated as a concept.

Both methods heavily rely on the representation of the documents. Namely, the document representation provides the vectors of the documents which LSI tries to approximate and, the basis for clustering algorithm is the similarity of document which depends on the document representation.

By incorporating background knowledge directly into document representation via word weighting reflecting similarity between the documents we enable our methods to discover concepts which resemble the view that the user has on the data.

For the purpose of building simultaneous ontologies from the same data, we calculate word weights from the background knowledge. The background knowledge is included by documents' category information provided by the user.

In the rest of this chapter we first review some basic definitions of the known bag-of-words document representation and the most commonly used heuristic approaches for calculating word weights. In the second part of the chapter we review the existing work on automatically learning similarity measures in the field of machine learning. The notation used throughout this report is established in Section 2.1.

2.1 Basic definitions

Most commonly used representation of the documents in text mining is *bag-of-words* representation. Let $V=\{w_1,\ldots,w_n\}$ be vocabulary of words. Let TF_k be the number of occurrences of the word w_k in the document. In the bag-of-words representation a single document is encoded as a vector $\boldsymbol{x}$ with elements corresponding to the words from a vocabulary providing some word weight, eg. $x^k = TF_k$. These vectors are in general very sparse since the number of different words that appear in the whole collection is usually much larger than the number of different words that appear inside one specific document.

In a similar way we can store category information for each document. Let $C=\{c_1,\ldots,c_m\}$ be set of all categories. For each document we have a bag-of-categories vector $\boldsymbol{y}$ where i-th element y^i of vector $\boldsymbol{y}$ is one if the document belongs to category c_i and zero otherwise. Please note that storing category information in this way does not include any relations between categories and all categories are regarded equal.

A labeled document collection is represented by pairs of bag-of-words vectors and by bag-of-categories vectors. More formally, labeled document collection is a set of pairs

$$D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}.$$

Measure usually used to compare text documents is *cosine similarity* and is defined to be the cosine of the angle between two documents' bag-of-words vectors,

$$sim(x_i, x_j) = \frac{\sum_{k=1}^{n} x_i^k \cdot x_j^k}{\sqrt{\sum_{k=1}^{n} x_i^k \cdot x_i^k}\sqrt{\sum_{k=1}^{n} x_j^k \cdot x_j^k}}.$$

Performance of both bag-of-words representation and cosine similarity can be significantly improved by introducing word weights. Each word from vocabulary V is assigned a weight and elements of vectors $\boldsymbol{x}_i$ are multiplied by the corresponding weights. Elements of vectors $\boldsymbol{x}_i$ can be also extended from simple word frequency TF_k to the output of a function providing the word statistics $\boldsymbol{\theta}_{i,k}$ of the k-th word inside the i-th document.

Let μ_k be the weight of word w_k, let $\boldsymbol{\theta}_{i,k}$ be statistics of the word w_k inside the i-th document and $g: \Re^h \rightarrow \Re$ function on word statistics. The elements of *weighted bag-of-words vectors* are

$$x_i^k = \mu_k \cdot g(\boldsymbol{\theta}_{i,k}).$$

We can see that the basic, commonly used bag-of-words model is a specific case of that having: $\mu_k = 1.0$, $\boldsymbol{\theta}_{i,k} = [TF_k]$, $g(x) = x$. Note that there is no need to treat word weights μ_k separately, as we can have the weights as a part of function g. As we will see in the next sections, some methods are based on finding a good function g and ignoring weights μ_k, while some other are based on finding better weights μ_k and keeping function g and word statistics from the basic bag-of-words model.

Note that the word statistics $\boldsymbol{\theta}_{i,k}$ does not only depend on the k-th word but also on the i-th document since in some cases information such as document length and document category are needed. Sometimes (eg., in meta-learning) word statistics also depend on categories. In that case we mark it as $\boldsymbol{\theta}_{i,l,k}$ meaning the statistics of the k-th word inside the l-th category and inside the i-th document.

As we already mentioned, our approach is based on the word weights being the key to viewing the same data from different angels. We can use the weights to store the background knowledge since the weights define which words are important. We will now focus on different ways for calculating weights which were proposed and used till now. Note that not all of them are able to incorporate extra knowledge about categories.

2.2 Traditional approach – TFIDF

Most of the research on word weighting schemas was traditionally done in the information retrieval community. The most common goal in information retrieval is to find the most relevant documents from the document collection for a given query. Many popular methods from information retrieval are based on measuring cosine similarity between the documents and a query and their performance can be significantly improved by appropriate weighting of the words.

Most of the popular methods for this task developed in last decades do not involve learning. Word weights are calculated by predefined formulas from some basic statistics of the word frequencies inside the document and inside the whole document collection [Salton91]. These methods are base on intuition and experimental validation and not so much on the theoretical foundation of the proposed formulas.

2.2.1 TFIDF based weighting methods

Most term weighting schemas [Salton91, Robertson96, Singhal96] within this family are a combination of the following simple statistics of the words, documents and collection:

- *Term Frequency* (TF_k) – number of times *i*-th word occurs inside the document,
- *Inverse Document Frequency* (IDF_k) – inverse of the number of documents from the collection in which *i*-th word occurs,
- *Inverse Category Frequency* (ICF_k) – inverse of the number of categories in which *i*-th word occurs,
- Number of documents in the collection (N),
- Number of categories in the collection (M),
- Length of the document (DL_i).
- Average document length (ADL)

These methods only try to define function g and relevant word statistics and do not affect the word weights μ_k.

The most widely used is the *TFIDF weighting schema* [Salton91] which defines elements of bag-of-words vectors with the following formula:

$$x_i^k = TF_k \cdot \log(N \cdot IDF_k).$$

We can place this formula inside our framework, the word statistics $\boldsymbol{\theta}_{i,k}$ and function g for the TFIDF weighting schema are:

$$\boldsymbol{\theta}_{i,k} = [TF_k, N, IDF_k], \qquad g([x,y,z]) = x \cdot \log(y \cdot z).$$

The intuition behind this weighting schema is that the words which occur very often are not so important for determining if a pair of documents is similar while a not so frequent words occurring in the both documents is a strong sign of similarity. The TFIDF weighting can be easily modified to include category information by replacing *IDF* and number of documents with *ICF* and number of categories.

There are many extensions of this schema most famous being Okapi weighting schema [Roberston96] which we will skip here since it does not incorporate category information. Some of the learning methods that we will mention in next section try to learn functions of word statistics similar to the TFIDF or Okapi formulas. This can be seen as a more systematic and optimized way of choosing the function g.

2.3 Learning approaches

In recent years many authors tried to more systematically search for the most appropriate functions of word statistics and word weights by means of machine learning. In the next subsections we will review the major work done in this area. Names of the forthcoming subsections are selected according to how the authors referred to their proposed methods.

2.3.1 Meta-learning

In [Do05] authors propose a method for learning function of word statistics for the task of multi-class text classification with disjoint categories. The word statistics they use depend both on the category and on the documents. The category is needed when calculating weighted bag-of-words vector for a specific document even if the correct category is not know.

In the learning process they searched the function space for a function *g* which maximizes the elements of training documents' vectors when weights are calculated for the correct category (provided with the training data) and minimizes the elements when weights are calculated for all the other categories.

The resulting function *g* can be used for classification. When a new document arrives its vector is calculated for all possible categories. Category for which the document's vector has the largest elements is chosen for prediction.

The authors first limit function *g* to be linear function of the word statistics. It turns out that word statistics only appears inside inner product which allows them to use the kernel trick and to search for *g* in much larger space of possible functions.

2.3.2 Inferring document similarity

Learning a good function of word statistics is also the goal for authors in [Grangier05] but this time the task is information retrieval. At learning the functions *g* authors use hyperlinks between the documents instead of category information for guiding the search. Their hypothesis is that if two documents are connected by a hyperlink then they are similar.

For word statistics and function *g* they choose:

$$\boldsymbol{\theta}_{i,k} = [TF_k,\ IDF_k,\ DL_k/ADL], \qquad g([x,y,z]) = MPL(x) \cdot MPL(y) \cdot MPL(z),$$

where *MPL* stands for *Multi-Layer Perceptron*. In the learning process they optimize each MPL so that the documents are more similar to their direct neighbours in the hyperlinks graph than to the other documents.

2.3.3 Distance learning

The paper [Xing03] is different from the others presented in this chapter because the authors try to learn the optimal *Euclidian distance* and not the optimal *cosine similarity*. However, their ideas can be easily translated to learning optimal weights for cosine similarity.

In order to find the optimal Euclidian distance the authors search over the space of *Mahalanobis* distances in $\Re^n$ which are parameterized with positive symmetric definite matrices *A*:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = ||\ \boldsymbol{x}_i - \boldsymbol{x}_j||_A = \text{sqrt}[(\boldsymbol{x}_i - \boldsymbol{x}_j)^T A(\boldsymbol{x}_i - \boldsymbol{x}_j)].$$

The inputs for the method are sets

$$S = \{(\boldsymbol{x}_i, \boldsymbol{x}_j);\ \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are similar}\}$$

$$D = \{(\boldsymbol{x}_i, \boldsymbol{x}_j);\ \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are not similar}\}$$

and the goal of the learning process is to find the matrix *A* which solves the following optimization problem:

$$\begin{array}{ll} \min\limits_{A} & \sum\limits_{(x_i,x_j)\in S} || x_i - x_j ||_A^2 \\ s.t. & \sum\limits_{(x_i,x_j)\in D} || x_i - x_j ||_A^2 \geq 1 \\ & A \succ 0 \end{array}$$

The intuition is that we want distance to be small between documents known to be similar and we want to keep some "margin" between the documents know for not being similar. The problem can be significantly simplified by restricting matrix A to be diagonal.

2.3.4 Weights learning

A different approach was taken in [Rong05] where category information is used for learning word weights μ_k while the function g and the word statistics are taken from the basic bag-of-words model. The authors' hypothesis is that two documents are more similar if they belong to similar categories. For this purpose they use cosine similarity also on the bag-of-categories vectors. the authors argue that in the same way as not all words are of equal importance not all categories are of the equal importance. For determining the similarity between documents based on categories they introduce category weights η_k.

The learning process finds word weights μ_k and category weights η_k so that the similarities based on bag-of-categories vectors and similarities based on bag-of-words vectors match as much as possible. The authors formulate that into the following optimization problem:

$$\begin{aligned} \min_{\mu,\eta} \quad & \sum_{i,j=1}^{N} \left(\sum_{k=1}^{m} \eta_k^2 y_i^k y_j^k - \sum_{k=1}^{n} \mu_k^2 x_i^k x_j^k \right)^2 \\ s.t. \quad & \sum_{k=1}^{m} \eta_k^2 + \sum_{k=1}^{n} \mu_k^2 \geq 1 \\ & \eta_k \geq 0, \quad k = 1,\dots,m \\ & \mu_k \geq 0, \quad k = 1,\dots,n \end{aligned}$$

The optimization problem can be rewritten in the form of standard quadratic programming formulation and solved using Matlab's Optimization Toolbox. However, the criteria function is not convex and that can get us stuck in one of the local minima.

2.4 SVM feature selection methods

As we will see in the next chapter a different approach can also be taken for generating word weights based on feature selection methods. Feature selection methods based on Support Vector Machine (SVM) [Cristianini00] has been found to increase the performance of classification by discovering which words are important for determining the correct category of a document [Brank02].

The method proceeds as follow. First linear SVM classifier is trained using all the features. Classification of a document is done by multiplying the document's bag-of-words vector with the normal vector computed by SVM,

$$\mathbf{x}^T\mathbf{w} = x^1w^1 + x^2w^2 + \dots + x^nw^n,$$

and if the result is above some threshold b then the document is considered positive. This process can also be seen as voting where each word is assigned a vote weight w^i and when document is being classified each word from the document issues x^iw^i as its vote. All the votes are summed together to obtain the classification. A vote can be positive (document should belong to the category) or negative (the document should not belong to the category).

A simple and naïve way of selecting the most important words for the given category would be to select the words with the highest vote values w^i for the category. It turns

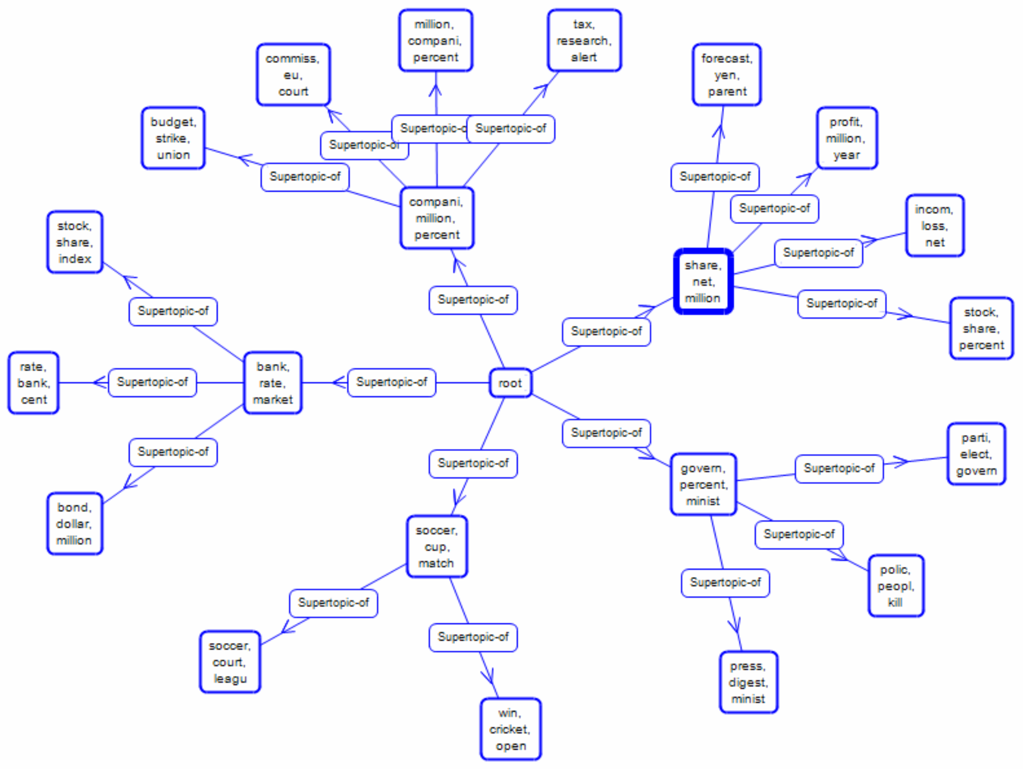

Figure 1 An example of a topic ontology that nicely follows the view provided by the topic labels originally associated with the news articles.

out that it is more stable to select the words with the highest vote $x^i w^i$ averaged over all the positive documents.

The votes w^i could also be interpreted as word weights since they are higher for the words which better separate the documents according to the given categories.

3 Description of approach

For the purpose of building simultaneous topic ontologies on the same data we need methods which can learn optimal weights for words based on the provided category information. We have based our approach on *Weights learning* and *SVM feature selection approach* (see Section 2.34, 2.4), as they seemed the most appropriate for our task. Here is a brief description of our reasoning when selecting the two methods. The first criterion which we used was to choose methods which involve learning, because they can offer better insurance that the final results are “optimal” – optimal in the sense of the criteria they optimize. Also, the *Meta-learning* and *Inferring document similarity* approaches try to learn single optimal function *g* which is then used for calculating all the word weights. But, since we are searching for a method that treats each word separately we could not use them. The other methods described in Section 2 are appropriate for our task and we focus on the last two since the *Weights learning* was already successfully applied to cosine similarity [Rong05] and *SVM feature selection* was shown to discover important category specific words [Brank02].

In the rest of this chapter we first show how SVM feature selection can be used for calculating word weights. Then we give a description of the dataset we used for testing the methods and we finish with the results we obtained using the two methods.

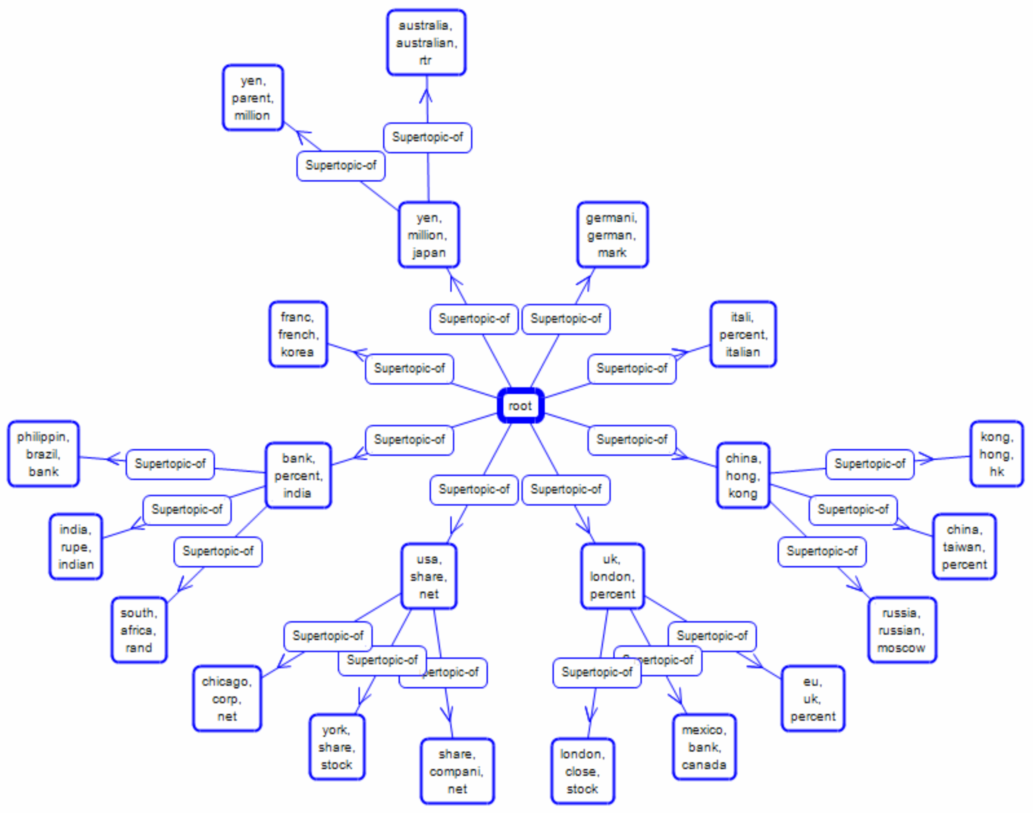

Figure 2 An example of a topic ontology that groups the news articles based on countries and not on the topics or pure content of the news.

3.1 Word weighting with SVM

The algorithm we developed for assigning weights using SVM feature selection method is the following:

1. Calculate a classifier for each category from the document collection (one-vs-all method for multi-class classification). TFIDF weighting schema can be used at this stage. Result is a set of SVM normal vectors $W = \{\mathbf{w}_j \,;\, j=1,\ldots,m\}$, one for each category.
2. Calculate weighting for each of the categories from its classifier weight vector. Weights are calculated by averaging votes $x^i w^i$ across all the documents from the category. Only weights with positive average are kept while the negative ones are set to zero. This results in a separate set of word weights for each category. By μ^j_k we denote weight for the *k*-th word and *j*-th category.
3. Weighted bag-of-words vectors are calculated for each document. Let $C(d_i)$ be a set of categories of a document d_i. Element of vector $\mathbf{x}_i$ are calculated in the following way:

$$x_i^k = \left(\sum_{j \in C(d_i)} \mu_k^j \right) \cdot TF_k \,.$$

This approach has another strong point. Weights are not only selected so that similarities correspond to the categories given by the user but they also depend on the context. Let us illustrate this on a sample document which contains words “machine learning”. If the document would belong to category “learning” then the word "learning" would have high weight and the word "machine" low weight. However, if the same document would belong to category "machine learning", then most probably both words would be found important by SVM.

This is not the case in *Weights learning* method where weight for a specific word is always the same and does not dependent on the context (content of the document).

3.2 Reuters RCV1 dataset

As a document collection for testing the above methods we chose Reuters RCV1 [Lewis04] dataset. The reason for which we chose it is that each news article from the dataset has two different types of labels (categories). Each news article is assigned labels according to (1) the topics covered and (2) the countries involved in it. We used a subset of 5000 randomly chosen documents for the experiments.

Bellow is a table with the 20 most frequent categories from the used subset of RCV1 dataset. The statistics are for the subset used in the experiments. Topics are also arranged into a simple taxonomy, first two levels of it are visualized in Figure 4.

TOPICS VIEW			COUNTRIES VIEW	
CCAT	*corporate/industrial*	*46%*	USA	*33%*
GCAT	*government/social*	*30%*	UK	*11%*
MCAT	*markets*	*24%*	Japan	*6%*
C15	*performance*	*19%*	Germany	*4%*
ECAT	*economics*	*14%*	France	*4%*
C151	*accounts/earnings*	*10%*	Australia	*3%*
M14	*commodity markets*	*10%*	India	*3%*
C152	*comment/forecasts*	*9%*	China	*3%*
GPOL	*domestic politics*	*7%*	EEC	*3%*
M13	*money markets*	*7%*	Hong Kong	*2%*
C18	*ownership changes*	*7%*	Russia	*2%*
M11	*equity markets*	*6%*	South Africa	*2%*
M141	*soft commodities*	*6%*	Canada	*2%*
C181	*mergers/acquisitions*	*6%*	Italy	*2%*
C31	*markets/marketing*	*5%*	Poland	*2%*
E21	*government finance*	*5%*	Netherlands	*2%*
GDIP	*international relations*	*5%*	South Korea	*2%*
C17	*funding/capital*	*5%*	Indonesia	*2%*
GCRIM	*crime/police*	*5%*	Mexico	*1%*
C13	*regulation/policy*	*5%*	Belgium	*1%*

3.3 Examples of discovered concepts from OntoGen

Both methods for calculating word weights were used on the same dataset. With each method two separate sets of weights were constructed, one based on topic labels and second based on countries labels. It is expected that OntoGen will discover concepts which group news articles together by (1) the topics or (2) by countries involved.

The topic labels covered fields of business, macro economy, government, politics and sports and we are hoping that the discovered topic ontologies would follow this view. Note that majority of news articles is about business and economy.

The country labels cover most major world countries with emphasis on the most economically strong ones like USA, Japan, China, European countries, Russia. For

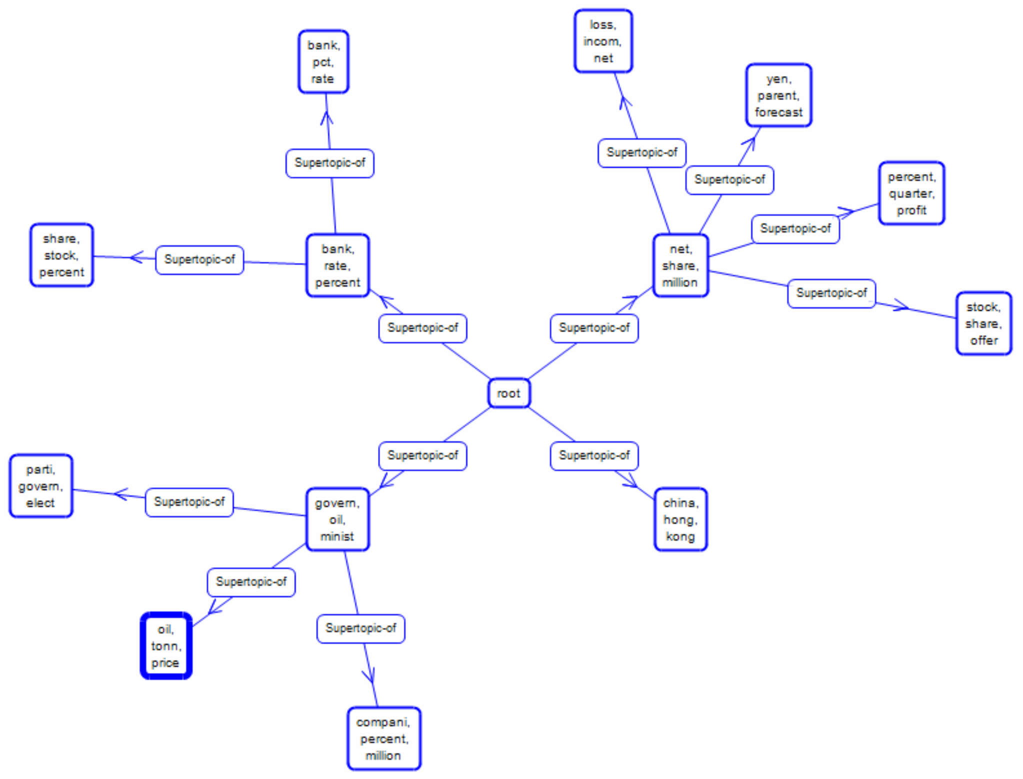

Figure 3 Topic ontology generated using TFIDF weighting schema. Results are more similar to results with weighting based on topic labels. However, the concepts are not so clear for TFIDF weighting schema as they are for learned weightings.

this view we are hopping that the countries will be grouped based on similarities of their economies and on geo-political similarities since this are the areas mostly cover with the news articles in the dataset.

Ontology construction in OntoGen was done with k-means clustering used as a method for concept discovery. Because OntoGen is a semi-automatic tool for ontology construction the user has to be involved during the construction. The user selects the number of concepts for suggestion, (parameter *k* for k-means algorithm), decides which concepts to add to the ontology, which concepts should be further broken into smaller concepts, etc. This makes automatic and fully objective comparison of different word weighting schema hard.

For constructing ontologies that are used in the next paragraph we spent same amount of time and work for all word weighting schemas with the goal of obtaining as good topic ontology of the news articles as possible. Only suggested concepts were used and no concepts were generated from scratch or by manually adding/removing news articles.

The results for SVM feature selection method are presented in a form of visualization of the discovered topic ontologies. For each view the top two levels of topic ontologies were constructed. For illustration of the approach we show the results obtained using SVM future selection method. Figure 1 shows the result for topic view while Figure 2 shows the country view. In Figure 3 we provide results obtained when no category information and only the basic TFIDF weighting schema is used for word weighting.

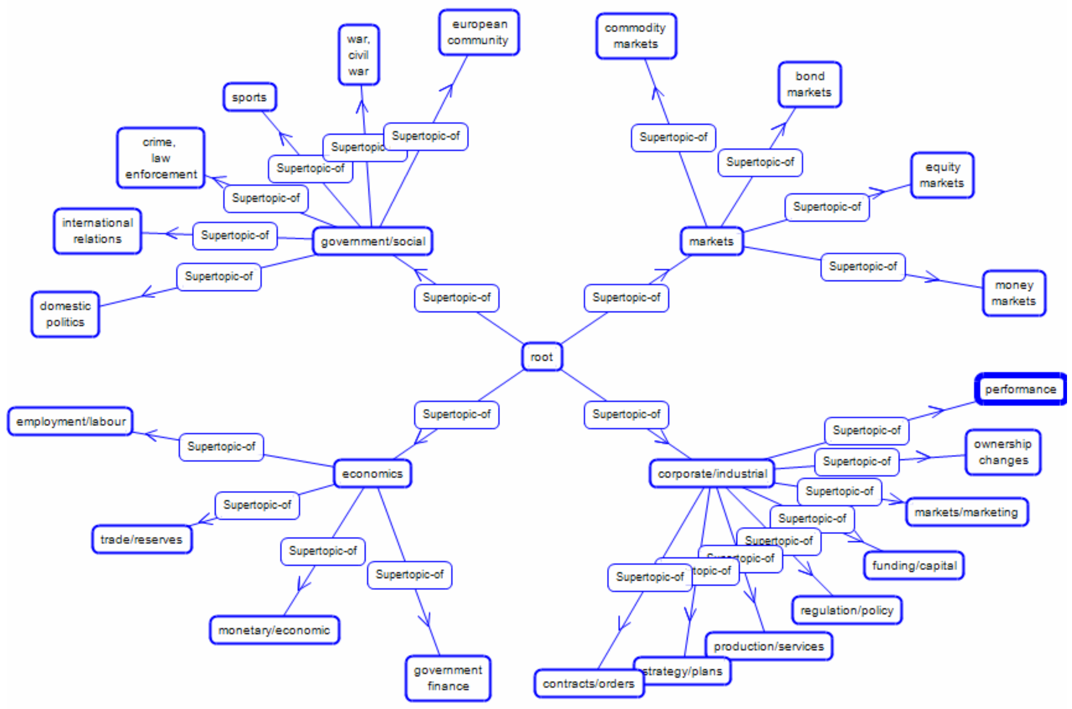

Figure 4 First two levels of topic taxonomy used in Reuters RCV1 (topics with less than 100 documents in the subset selected for testing are omited).

From the results in Figures 1 and 2 we can see that SVM feature selection method succeeds with discovering weights which can reflect the category information, when compared to Figure 3. Also, the concepts discovered with OntoGen look clearer when SVM feature selection is used for word weighting than the concepts discovered with OntoGen when TFIDF schema is used. This is expected since by providing category information the similarity measure is supervised to reflect human expectations.

It is more difficult to compare Figures 1 and 3 to the original taxonomy of topics in Reuters dataset in Figure 4. The concepts in Figure 4 are assigned names that were carefully chosen by domain experts while the concepts in the other Figures are assigned names based on most important keywords. As feature work we plan to make the Figures more comparable but already we can see closer resemblance between Figures 1 and 4 than between Figures 3 and 4.

Topic ontologies generated from *Weight learning* method are omitted here since the results failed to show any visual correlation with the topic or country label. Reasons for that still need to be investigated because the original paper [Rong05] reports very positive results. One possible explanation for our poor results is that different dataset was used and the method was applied to different task.

4 Architecture

The technology and methods developed as part of deliverable D1.9.1 and described in this report is tightly integrated into Text Garden [Grobelnik06] tools. Since the traditional word weighting methods are already part of the functionality we only extended it with the new word weighting methods.

The computational complexity of the traditional methods is very low so until now the word weights could be calculated on the fly when needed. However, the new methods which involve learning are computationally much more complex and time consuming so we created an extra tool which pre-computes the weights and stores them on hard-disk for later use. The utility is called *Bag-Of-Words to Bag-Of-Words-Weights* (Bow2Boww.exe). The files which store weighted bag-of-words vectors have the prefix `.boww`.

A new option was also added to OntoGen which enables it to load weighted bag-of-words vectors. The option is available under the menu "Load→Bag-of-words with weights". The user is required to provide the location of standard bag-of-words file (`.bow`) and of the weighted bag-of-words file (`.boww`).

Because *Weights learning* method relies on Matlab Optimization Toolbox functionality it currently only runs inside Matlab. Tools are provided which simplify the process of connecting Text Garden with Matlab:

- Bag-of-words file are transformed into Matlab sparse matrices using Bow2Txt.exe utility. To sparse matrices are generated, one with bag-of-words vectors and one with bag-of-categories vectors.
- Code that learns the weights is nicely packaged inside two Matlab functions:
    - Function load_data reads two sparse matrices from hard-drive and prepares them for the learning
    - Function optimize_weights performs the learning algorithm and outputs vector of word weights. The result can be saved to disk using Matlab's save function
- Utility Bow2Boww.exe gets bag-of-words file (`.bow`) and Matlab vector containing word weights on the input and generates weighted bag-of-words file (`.boww`) as output.

The *SVM feature selection* approach is already integrated into Text Garden and can be accessed trough the Bow2Boww.exe utility. It uses Text Garden's SVM implementation. Detailed description on the availability, system requirements and a user guide for the utilities developed as part of this deliverable can be found in the appendixes.

5 Future development

For the future we plan to fully integrate the *Weights learning* approach into Text Garden so that whole pipeline will be implemented in Text Garden. This will eliminate the dependency on Matlab. We also plan to extensively test Weights learning approach and try it on the same data as in the paper [Rong05].

Another important step for the future is to test the system on more real world data and to apply the word weights learning methods to some practical scenarios. SEKT Case studies offer great opportunity for this. For example, in the Legal case study the data is already partly labeled and this could be used to improve the quality of the Legal Ontology.

As feature work we also plan to add automatic discovery of complex relations between concepts to the OntoGen system. Word weights learning methods will allow us to include the user's background knowledge into this process.

Conclusions

In this report we have shown that different topic ontologies can be constructed with OntoGen based on the same data by using category information for determining word weights. This allows users to incorporate their background knowledge into the similarity measure used for discovering concepts in the data which in turn allows different users to provide different view on the data and enables construction of simultaneous ontologies.

On the test data used in this report we have also show that SVM feature selection method provides better means of incorporating category information for the concept discovery methods.

We can therefore conclude that SVM feature selection method together with Active Learning [Novak04] (for labeling the documents with categories) provide good and efficient approach to construction of simultaneous ontologies from a given set of data.

Appendix A – Availability and System Requirements

All the tools presented here are available trough the website http://www.textmining.net/ and run in the Windows operating system. OntoGen also needs *.NET framework 2.0* which can be freely downloaded from internet.

Appendix B – User Guide

B.1 Bow2Txt

Bow2Txt is a utility which can take Text Garden bag-of-words file on the input ("-i") and can save it as Lined-Documents ("-olndoc") or generate sparse matrices which can be loaded in the Matlab. Two matrices are generated, the one with bag-of-words is stored into the file provided with parameter "-oml" and the bag-of-categories is stored in the file provided with parameter "-omlcat". Statistics of the bag-of-words file are also stored if parameter "-ostat" is provided. When generating Matlab sparse matrix the bag-of-words vectors can be weighted. The type of weighting is provided with parameter "-w". Note that in this stage of processing only traditional weighting methods are available.

Usage: Bow2Txt.exe

`-i:`	Input-BagOfWords-FileName
`-olndoc:`	Output-LineDocuments-FileName
`-w:`	Weighting (none, norm, bin, tfidf) (default:'tfidf')
`-oml:`	Output-Matlab-FileName
`-omlcat:`	Output-Matlab-Category-FileName
`-ostat:`	Output-Statistics-FileName

Example:

```
Bow2Txt.exe –i:reuters.bow –oml:reutres_docs.dat
            -omlcat:reuters_cats.dat –w:none
```

In this example we take bag-of-words file *reuters.bow* and generate two Matlab sparse matrices. One with bag-of-words vectors is stored into file *reuters_docs.dat* and the one with bag-of-categories vectors is stored into file *reuters_cats.dat* file. No weighting schema is used which means the basic bag-of-words model.

B.2 Matlab functions

B.2.1 Load_data

Input parameters:

`DocFNm`	Name of the file with bag-of-words sparse matrix
`CatFNm`	Name of the file with bag-of-categories sparse matrix
`min_word_fq`	In how many documents must word appear so it is used
`min_cat_fq`	In how many documents must category appear so it is used

The last two parameters can help at reducing the learning time by cutting out less frequent words and categories which can be considered noise.

Output parameters:

`X`	Bag-of-words matrix
`Y`	Bag-of-categories matrix
`Words`	Number of all words

`Cats`	Number of all categories
`selected_word_id`	Vector with Ids of words with high enough frequency
`selected_cat_id`	Vector with Ids of categories with high enough frequency

Example:
```
[W C words cats selected_word_ids selected_cat_ids] =
        load_data(topic_docs.dat', 'topic_cats.dat', 7, 10);
```

The upper example loads two sparse matrices and ignores all the words with frequency less than 7 and categories with frequency less than 10.

B.2.2 Optimize_weights

Input parameters:

`max_iter`	Number of iterations
`W`	Bag-of-words matrix
`C`	Bag-of-categories matrix
`word_sub_size`	Size of sub-problems which are solved in each iteration

Output parameters:

`Mi`	Calculated word weights
`Ni`	Calculated category weights

Example:
```
[word_wgt, cat_wgt] = optimize_weights(300, W, C, 500);
```

The upper example runs 300 iteration of the optimization procedure with sub-problem size of 500 words.

B.2.3 Save

This is Matlab's command for storing the vectors on the hard-drive. Results of `optimize_weights` function should be stored in this way so *Bow2Boww* utility can get it on the input.

Example:
```
full_word_wgt = zeros(words,1);
full_word_wgt(selected_word_ids) = word_wgt;
save topic_wgt.dat full_word_wgt -ascii -double
```

B.3 Bow2Boww

Bow2Boww is a utility which can take bag-of-words file on the input ("-i:") and generate weighted bag-of-words file as output ("-o:"). This can be done on two ways ("-type"). First way ("load") means that word weights are loaded from a file which was generated with Matlab ("-iwgt"). Second way ("svm") is to apply SVM feature selection approach for calculating weights. Weighted bag-of-words vectors can be normalised ("-unitnorm") and words with small weights ("-cutww") or low frequency ("-mnwfq") can be deleted from the vectors.

Usage: Bow2Boww.exe

`-i:`	Input-BagOfWords-FileName
`-o:`	Output-BagOfWordWeights-FileName
`-type:`	Method-Type (load, svm)
`-iwgt:`	Input-Matlab-WordWeights-FileName
`-unitnorm:`	Normalize-Document-Vectors

`-cutww:`	Cut-Word-Weight-Sum-Percentage
`-mnwfq:`	Minimal-Word-Frequency

Examples:

```
Bow2Boww.exe -i:country.bow -o:country.boww
      -type:load -iwgt:country_wgt.dat –unitnorm:T

Bow2Boww.exe -i:country.bow -o:country.boww -type:svm
```

The first examples generates weighted bag-of-words file using weights calculated inside Matlab and the second example generates weighted bag-of-words file using SVM feature selection method.

B.4 OntoGen

Use of OntoGen as a system was already described in the deliverable D1.7.1. This deliverable provides an extension trough a possibility of including background knowledge while usage of the system as a whole remains the same.

Bibliography and references

[Brank02] Brank, J., Grobelnik, M., Milic-Frayling, N., Mladenic, D.: *Feature selection using support vector machines*. Proceedings of the Third International Conference on Data Mining Methods and Databases for Engineering, Finance, and Other Fields, Bologna, Italy, 25--27 September 2002.

[Cristianini00] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines*, Cambridge University Press, 2000

[Deerwester90] S. Deerwester, S. Dumais, G. Furnas, T. Landuer and R. Harshman, *Indexing by Latent Semantic Analysis*, Journal of the American Society of Information Science, vol. 41, no. 6, 391-407, 1990

[Do05] Chuong B. Do, Andrew Y. Ng. Meta-learning for text classification. Advances in NIPS, vol. 17, 2005.

[Fortuna05] Fortuna, B., Mladenic, D., Grobelnik, M., (2005a). Semi-automatic construction of topic ontology. Proceedings of the ECML/PKDD Workshop on Knowledge Discovery for Ontologies.

[Grangier05] D. Grangier and S. Bengio. *Inferring Document Similarity from Hyperlinks*. Presented at ACM Conference on Information and Knowledge Management, CIKM, 2005

[Grobelnik06] M. Grobelnik, D. Mladenic. *Text Mining Recipes*, Springer-Verlag, Berlin; Heidelberg; New York (to appear), 2006, (accompanying software available at http://www.textmining.net).

[Jain99] Jain, Murty and Flynn: *Data Clustering: A Review*, ACM Comp. Surv., 1999

[Lewis04] Lewis, D. D.; Yang, Y.; Rose, T.; and Li, F. *RCV1: A New Benchmark Collection for Text Categorization Research*. Journal of Machine Learning Research, 5:361-397, 2004.

[Novak04] Novak, B., (2004a). Use of unlabeled data in supervised machine learning. Proceedings of the 7th International multi-conference Information Society IS-2004, Ljubljana: Institut "Jožef Stefan", 2004.

[Robertson96] Robertson, S. E., S. Walker, M. M. Hancock-Beaulieu, M. Gatford and A. Payne. *Okapi at TREC-4*. The Fourth Text REtrieval Conference (TREC-4). 1996

[Rong05] Rong Jin, Joyce Y. Chai and Luo Si. *Learn to weight terms in information retrieval using category information*. Proceedings of the 22nd international conference on Machine learning, pages 353-360, 2005

[Salton91] G.Salton. *Developments in Automatic Text Retrieval*. Science, Vol 253, pages 974-979. 1991

[Singhal96] Singhal, A., C. Buckley and M. Mitra. *Pivoted Document Length Normalization*. Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval. 1996

[Xing03] E. Xing and A. Ng and M. Jordan and S. Russell. *Distance metric learning, with application to clustering with side-information*. Advances in NIPS, vol. 15, 2003