NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 – "Semantic-based knowledge and content systems"

D3.2.4 Context-sensitive Search of Ontologies

Deliverable Co-ordinator: **Boštjan Pajntar**

Deliverable Co-ordinating Institution: **J. Stefan Institute (JSI)**

Other Authors: **Dunja Mladenić (JSI), Marko Grobelnik (JSI)**

This deliverable provides an extension of the web application SearchPoint, which enhances ontology web search scenarios with the use of other ontologies that provide context for the search. Finding a good ranking is one of the main problems of ontology search engines. In SearchPoint we propose a solution that automatically generates topics related to a query and its results. These topics are then visualized in a panel which the user can interact with. Every point on this panel represents a weighted ranking. Every such ranking is aligned with the selected point. I.e. results semantically closer to the topics near the selected point get ranked higher. Effectively, the user can get a specifically tailored ranking of the results with a single click.

The context-sensitive search of ontologies as proposed in this deliverable provides one of the three core technologies for contextualization developed in NeOn WP3. Topics are generated via classification of results into a given ontology. Concepts that are the most represented get selected as topics. Since SearchPoint easily upgrades any textual search engine, it effectively contextualizes any textual general knowledge by re-ranking it in accordance with user selected importance of the concepts in the ontology.

Document Identifier:	NEON/2009/D3.2.4/v1.0	Date due:	August 31, 2009
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	August 31, 2009
Project start date:	March 1, 2006	Version:	V1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is a part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta} @open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.upm.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SI-1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 655 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Koblenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Martino della Battaglia, 44 - 00185 Roma-Lazio, Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 1 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts: JSI, OU.

Change Log

Version	Date	Amended by	Changes
0.1	20-07-2009	Boštjan Pajntar	Overall structure of the report
0.2	01-08-2009	Boštjan Pajntar	Executive Summary, introduction
0.3	03-08-2009	Boštjan Pajntar	Began approach description chapter
0.4	15-08-2009	Dunja Mladenić	Chapter 2, Overall revision
0.5	28-08-2009	Boštjan Pajntar	Chapter3
0.6	12-09-2009	Marko Grobelnik	Overall revision
0.7	28-09-2009	Boštjan Pajntar	Figures, Overall revision
0.8	15-10-2009	Christopher Buttenshaw	Final QA

Executive Summary

This report is describing a software deliverable developed as an extension of the web application SearchPoint [Pajntar and Grobelnik, 2008]. The main goal of SearchPoint is to enhance search engines by allowing the users to get multiple rankings of the results for each query. We achieve this by generating topics for the given query and its result set and visualizing these topics on a panel named “Ranking Space”. Each point in this ranking space maps to specific ranking. For example, if a point is selected near one topic, results that are on that topic are ranked higher.

The topics can be generated by clustering of the results and we have implemented this method as a baseline to compare with. The more advanced method for generating topics is the classification of hits and query into a selected ontology and then selecting the concepts with most results to serve as topics. This allows for visualization of a small enough number of topics that can be understood by the user on the one hand whilst retaining as much of the domain covered by the current result set.

Topics must be visualized in an intelligent way. Since the selection of a point in between two topics promotes hits that cover either of them, it makes sense to visualize similar topics close together. In the baseline scenario of taking centroids of clusters for the topics, they are visualized by drawing a complete weighted graph, each node representing a topic and edges being weighted by the similarity of the two nodes. In the scenario of classifying and selecting most prominent concepts from the ontology, we visualize in accordance to the underlying structure of the ontology.

This is the last of the three core contextualizing technologies stemming from WP3. It provides means for study of any non-structured, general knowledge, in a context of a selected ontology. The other two core contextualizing technologies are complementary: the context can be provided by one networked ontology for the other. This was implemented in OntoConto (D3.2.2, D4.5.2), consuming Alignment Server (D3.3.1, D3.3.2) and second, a general background knowledge can provide means for the contextualization of an ontology, which was implemented in OntoAtlas (D3.7.1, D4.3.1, D3.7.2)

Table of Contents

List of Figures

1. Introduction

Big search engines (i.e. www.google.com, www.yahoo.com) use a simple, effective and user-friendly method. A user must enter a query in the form of typed words or sentences which constitutes the engine's input, at which point, the engine returns a ranked result-set. Calculating the correct result-set is a well known problem with a well known solution. On the other hand, there is no apparent best way of calculating ranking. A lot of resources are spent on calculating optimal ranking and a lot of features are used for this process. There is continuous debate on how much the ranking should be personalized. Usually results are very efficient in the general web search setting. The rankings are at least initially calculated from the underlying graph of linked web pages. However, the ranking quickly becomes worse if there is no underlying structure. For example, corporate or specialized (image search, ontology search) search scenarios do not have a good solution for rankings.

Some web applications build on top of the usual search method. It is possible to identify subcategories of the result-set. These categories are in one way or another presented beside the hits and a user can reformulate his query by clicking on them. Examples of such site's are: www.vivisimo.com, www.clusty.com, www.kartoo.com, www.ujiko.com.

On mindset.research.yahoo.com another approach is presented. After the user receives the results of his query, he can tune the ranking by defining if results should be more in the sense of "shopping" or of "researching". This web application gave us the initial idea for our approach.

SearchPoint web application builds on the idea of re-ranking hits in accordance to the selected topics. We have extended the idea in a sense that topics are not predefined but dependant on the query and result set. There are also an arbitrary number of topics per query, providing a challenge on how the user can select his preferences.

The main work of this deliverable lies in extending SearchPoint with the possibility of adding any pre-trained OntoLight Classifier [Grobelnik et al., 2008] for the use of classification of hits into an ontology, which in turn provides best concepts for the visualized topics (Section 2.3). Apart from the classification into ontologies, we also use clustering of results for topic generation (Section 2.2).

Since there are more than two topics, it is not trivial to position them on the ranking space. With two topics it was easy since a simple slider bar sufficed for determining how much of each topic the user selects. Here, we must use the similarity of topics to draw a graph of topics. In this way, similar topics are visualized closer than non-similar. This enables the user to select points between topics if he is interested in results featuring two similar topics with a single click (Section 2.4).

The remainder of this deliverable initially describes the approach (Section 2), first giving an overview of the architecture, before describing topic generation and visualization techniques. Then we offer some discussion about the possible usage (Section 3.2) and showcase the system with screenshots (Section 3.3). In the end we discuss on future work that will incorporate current effort in the NeOn toolkit (Section 4).

2. Approach Description

SearchPoint in essence is a search engine add-on. This means it needs a search engine it can consume to provide additional functionality. The basic search functionality, input query, output ranked result-set, is unhindered. In the beginning a user has to provide a query and a ranked result-set is returned. Besides this, topics of interest are calculated. How this is done will be explained later, for now, all we have to know is, these topics correspond to the search query and top portion of the result-set.

Topics are placed on a plane as part of a graphical user interface (GUI) in such a manner that similar topics lay close. A focus point is placed at the origin. This focus can be moved by either dragging it to a desired location or alternatively a point on the plain can be clicked in order for the focus to move there. Each position of this focus corresponds with one ranking.

For example a user who clicks near one topic will get hits ordered mostly by the sense of that topic, if the focus is moved to a position between two topics, results that share similarity with both these topics will tend to be higher ranked. In truth, at any moment all the topics influence the ranking; however influence decreases with the distance between topic and focus point. For an illustration of usability see Figure 1 and Figure 2.

This approach can be used with any search engine that provides textual results. However, the problem of a general web search has been mostly solved, as underlying graph of linked web sites offers a lot of information for the importance of a single node – web page. On the other hand, our approach is very useful in the search scenarios without an underlying graph structure. For example, corporate search engines must work on a relatively small site with a mostly tree like structure, yet important content could be anywhere on this graph. Another example is that of prolific search scenarios, for example image search or ontology search.

In this deliverable, we have concentrated on the ontology search scenarios. There are several ontology search engines, so we tested our approach on swoogle.umbc.edu and google.com search with the defined result type as .owl or .rdf. This work will be integrated with Watson search [d'Aquin 2008] and NeOn toolkit platform as part of effort in WP4.

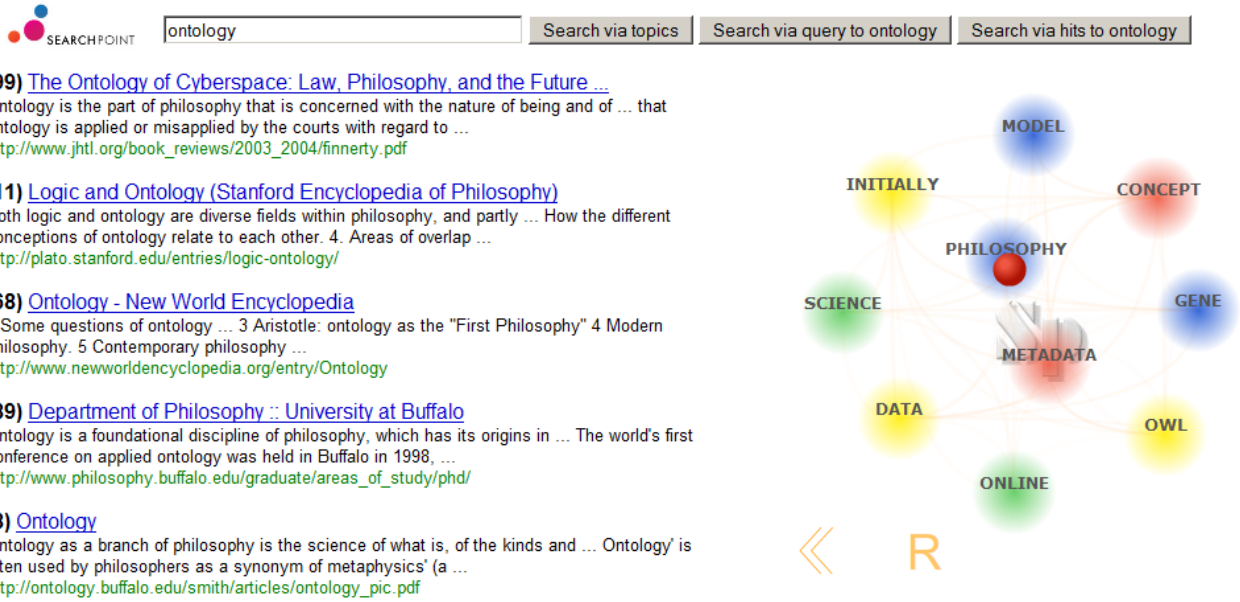

Figure 1: illustration of basic SearchPoint in action. For a query “ontology” ambiguous hits get returned. To see only the hits in the context of philosophy the red focus point is moved near the automatically generated topic “philosophy”. Hits returned are initially low ranked (99, 11, 68...) but all talk about philosophy, logic, Aristotle...

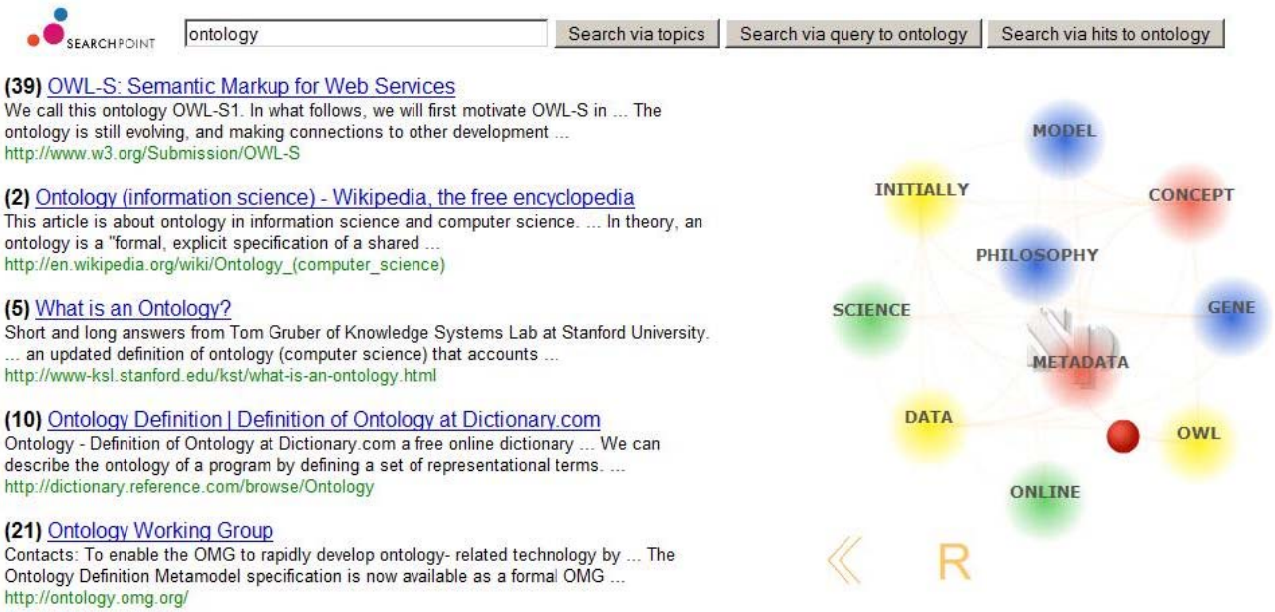

Figure 2: The same result page, only the focus point is moved between topics “metadata”, “owl”, “online”. The user immediately gets returned a list of hits about i.e. semantic mark-up, information and computer science.

2.1 System architecture

The SearchPoint is a tool for searching. However, this is not the usual search we are accustomed to for example on the web. In fact the usual search engine is merely one module in a longer chain of processes, which we will call modules.

The implementation of SearchPoint is modular as this makes it very easy to change a single module, for example, to change a search engine, adopt a new classifier, and provide a new distribution channel. The architecture of SearchPoint (Fig 1) consists of the following modules chained into a pipeline (Figure 3):

1. Agent (usually user) provides a query (usually in a GUI)
2. Search engine processes the query and returns a result-set of short textual documents – snippets
3. Automatic topic generation module (implemented as a web service)
4. Graph drawing and sub-graph extraction for visualization of the topics (implemented as a web service)
5. Graphical User Interface (GUI) for the visualization of the topics, focus point selection and rearranging the results dynamically.

Some comments on the modules:

In the current application, the modules 3 and 4 are enveloped inside a single web service. This is in order to minimize the number of calls made; however, if needed, they could be easily separated. Next, modules 1 and 5 are a component of the same GUI provided in the form of a web application. It seems natural to provide the user with a single place for posing queries, receiving and manipulating results. It is, however, very easy to change the distribution channel (i.e. to NeOn Toolkit) or to separate them (for use in an even bigger solution).

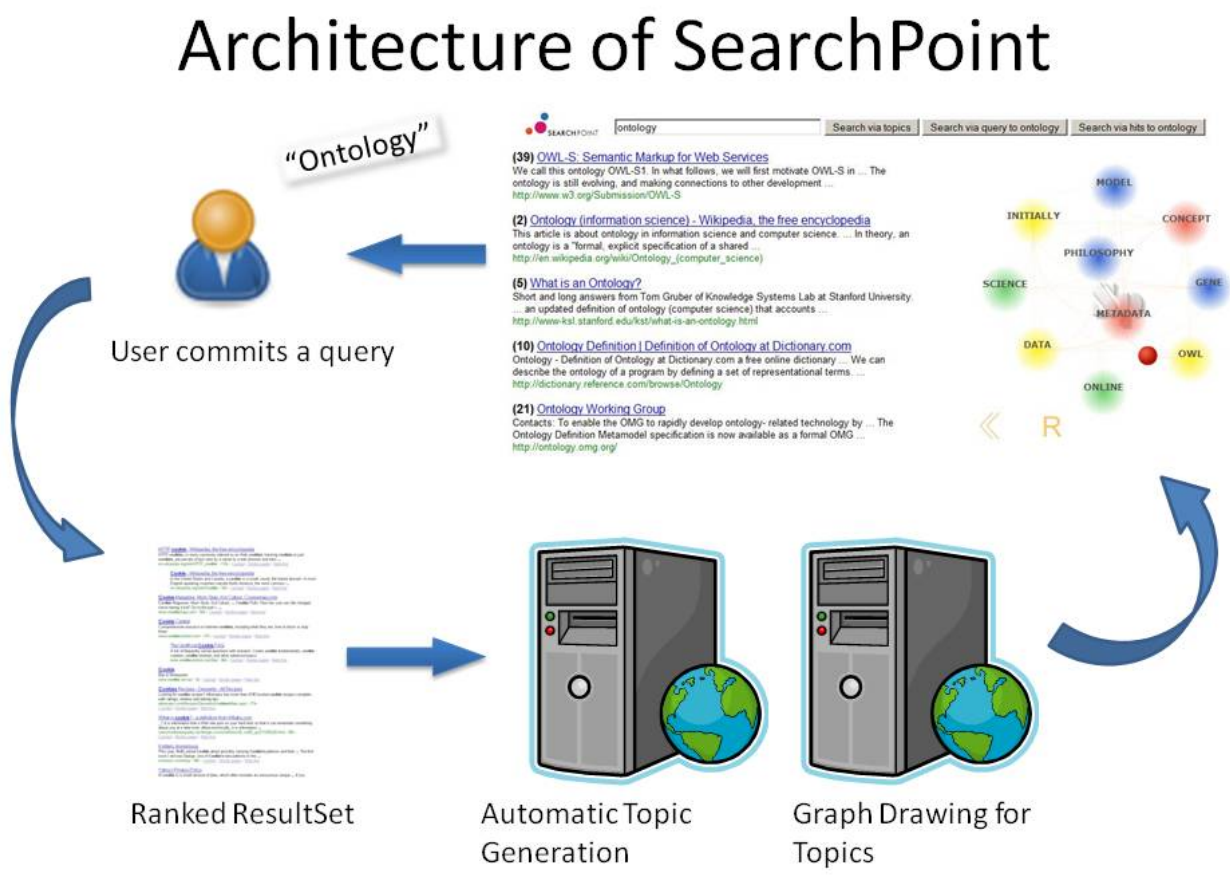

Figure 3: Architecture of SearchPoint

2.2 Topic generation – Clustering

The most obvious way to automatically generate topics out of a corpus of documents is to cluster the available results into a predefined number of clusters and use a centroid or medoid of each cluster as an individual topic. The technical details of the implementation follow below.

Document clustering (Steinbach et al., 2000) is based on a general data clustering algorithm adopted for textual data by representing each document as a word-vector, which for each word contains some weight proportional to the number of occurrences of the word (usually TFIDF weight as given in equation (2.1)).

$$d^{(i)} = TF(W_i, d)IDF(W_i), where IDF(W_i) = \log \frac{D}{DF(W_i)} \qquad (2.1)$$

Where D is the number of documents; document frequency DF(W) is the number of documents the word W occurred in at least once; and TF(W,d) is the number of times word W occurred in document d. The exact formula used in different approaches may vary somewhat but the basic idea remains the same – namely, that the weighting is a measure of how frequently the given word occurs in the document at hand and of how common (or otherwise) the word is in an entire document collection.

The similarity of two documents is commonly measured by the cosine-similarity between the word-vector representations of the documents (see equation (2.2)). The clustering algorithm groups documents based on their similarity, putting similar documents in the same group.

Several clustering algorithms can be used on TFIDF represented documents. For our approach we have selected k-means because of its high speed, since the clustering must be done at run time when returning results and topics to the users.

The basic k-means clustering algorithm [Kanungo et al 2002] is one of the oldest and simplest clustering algorithms to be applied to text, which still produces good results because of its fast execution and apparent suitability to textual data distributions. It involves randomly choosing k points to be the centroids of clusters, and grouping documents around centroids based on proximity with regards to a certain distance/similarity measure. We have chosen cosine similarity in our approach, because they are invariant to the length of each document. Then, centroids are iteratively recomputed for each cluster, and documents regrouped until there is sufficiently little change in centroid positions. This algorithm depends heavily on the choice of k, which we specify as a parameter in our web-service. We chose random initial positioning of centroids. The whole k-means algorithm is run ten times with different random positions of initial centroids. Final cluster assignments are determined on the basis of best intra – inter ratio for cluster similarity of the independent clustering results.

The calculated centroids are actually vectors in TFIDF space, so each centroid has a ranked list of words which are particularly prominent for the whole cluster. From this we derive the best word to represent the cluster that is visualized in the GUI and we also list several top words the user can read with some interaction.

The webservices for this method and other topic generation methods can be found at:

http://searchpoint.ijs.si/Classifier/WS_Classify.asmx

More information is given in the Appendix.

2.3 Topic generation – Classification

The clustering approach is very efficient in separating the documents into a selected number of topics. The main disadvantage however, is the presentation of these topics to the user. The most important words found in the centroid that define and separate the topic the most are not necessarily also the most informative to the user.

Therefore, we choose a different approach. Instead of automatically generating the topics through clustering, we classify each document into a preset list of topics that have been generated by a human and can therefore also be understood by a human.

Instead of classifying into an arbitrary list of topics we can do much better with classification into a given ontology. Such a classification provides a deeper understanding of available topics and can, besides being used as a query refinement tool, also be used to investigate the actual subtopics of a given interest of the user.

Any knowledge represented in a textual corpus and therefore being searchable, can in this way be contextualized with the use of different ontologies. On the same data set that is being searched with the same query, different ontologies return different topics the user can navigate through, providing different rankings of the hits. Each ranking can be considered as a different view into the whole corpus.

2.3.1 Text Classification

Text classification can be applied when a set of predefined categories (classes), such as “arts, education, science”, are provided as well as a set of documents labelled with those categories. The task is to classify new (previously unseen) documents by assigning each document one or more content categories. This is usually performed by representing documents as word-vectors (usually referred to as the ‘bag-of-words’ representation) and using documents that have already been assigned the categories, to generate a model for assigning content categories to new documents. In the word-vector representation of a document, a vector of word frequencies is formed taking all the words occurring in all the documents (usually several thousands of words). The representation of a particular document contains many zeros, as most of the words from the collection do not occur in a particular document. The categories can be organized into an ontology, for example, the MeSH ontology for medical subject headings or the Yahoo! hierarchy of Web documents that can be seen as a topic ontology. Other applications of document categorization into hierarchies/taxonomies are of US patents, Web documents (McCallum et al., 1998; Mladenić, 1998; Mladenić and Grobelnik, 2003), and Reuters news articles (Kholer and Sahami, 1997).

Cosine-similarity that is commonly used in document clustering can be also used for document classification as follows. Given a new document, cosine-similarity is used to find the most similar documents (e.g., using *k*-Nearest Neighbour algorithm (Mitchell, 1997)). Cosine-similarity between all the documents and the new document is used to find the *k* most similar documents whose categories (topics) are then used to assign categories to a new document. For documents d_i and dj, the similarity is calculated as given in equation (2.2). Note that the cosine similarity between two identical documents is 1 and between two documents that share no words is zero.

$$\cos(d_i, d_j) = \frac{\sum_k d_{ik} d_{jk}}{\sqrt{\sum_l d_{il}^2 \sum_m d_{jm}^2}} \qquad (2.2)$$

2.3.2 OntoLight

OntoLight [Grobelnik et al., 2008] is a software suite which implements basic reasoning functionalities for contextualized ontologies. It is limited to light-weight ontologies which are grounded with appropriate text corpora. The representation and reasoning scales to the largest currently available ontologies, comprising up to one million concepts. In particular, OntoLight currently incorporates the following five ontologies: AgroVoc and ASFA (relevant for the Food and Agricultural Organization of the UN), EuroVoc (EU legislation), Cyc (common-sense knowledge) and DMoz (a WWW directory).

There are two basic reasoning mechanisms implemented in OntoLight. First, new textual instances without a known class can be classified into the selected ontology. Second, soft (probabilistic) mappings between a pair of selected ontologies can be computed, thus providing a contextual relationship between the ontologies.

OntoLight was used as a basic building block for extensions to OntoGen [Fortuna et al., 2006], where contextual mappings are used to improve semi-automatic construction of light-weight ontologies from text corpora. The same mechanism of contextual reasoning will be used to extend OntoGen to support simultaneous, collaborative development of an ontology. Soft mappings between grounded ontologies also complement methods for ontology alignment, where mappings are computed on the basis of common, background ontologies (as provided by Swoogle, for example) [Sabo et al. 2008]. The main functionality we cover is the contextualization of ontologies through generation of soft mappings between ontologies, thus enabling us to view concepts of one ontology through the perspective of another one. OntoLight also supports the scalability needed for large case studies – i.e. being able to deal with large ontologies such as AgroVoc and ASFA. To

achieve this, the representation is constrained to a light-weight ontology model which covers targeted functionality needed in the case studies.

For the use of topic generation in SearchPoint we will be able to use any of the classifiers that are trained with OntoLight. OntoLight is a tool that easily transforms any grounded ontology into a classifier. We can use this classifier to find the concepts that are most relevant for the current query and use them to provide topics.

Since ontologies can be huge and we can actually visualize only so many distinct topics, we only choose those concepts that have the most documents classified into. Instead of just counting how many documents are classified into each concept we rather follow a different approach. When classifying a document into an ontology, OntoLight can actually provide document-concept similarities. In order not to miss any concept that actually covers many documents in the corpus but is not really the most important one for any, we rather use these similarities and for each concept we sum all the similarities of similar documents. Similar documents mean that we only take into account those similarities that are above a particular threshold. In this way we can get a ranked list of concepts which get to be used in the visualization and GUI.

2.4 Visualization and ranking space

Our main goal is to visualize automatically generated topics and position them on the screen in such a way that similar or related topics lay close together. Apart from this the user can select a *focus point* by dragging a red point that is also part of the GUI.

Once we achieve this, it is possible to calculate new rankings of the documents for each position of the focus point. Since each point on the panel maps to a specific ranking, we call it the *ranking space.*

2.4.1 Graph drawing

The input for the visualization is always a list of topics, which are presented as vectors. When using the clustering method we can select the desired numbers of topics-centroid, by selecting k in the k-means algorithm. In SearchPoint we leave this number as a parameter which is by default ten. When using OntoLight classifiers, we get a long list of ranked classes-concepts and we also select the desired top n (default ten).

While the visualization described below could be used for both clustering and classifier methods, we use it only for clustering. The changes for the classifier will be described in the next subsection.

To visualize these topics we need to position them. Our application of being able to select a point between topics also requires that similar topics lay close together. In both cases the topics are actually represented as vectors so we can calculate their cosine similarities.

In order to position or draw the topics, we model them as a full weighted graph. Each topic represents a node and similarity between two topics gives a weighted edge. In the classifier model we can also weight the nodes by the classifier score of the topics.

To draw a graph we use Fruchterman-Reingold (FR) algorithm [Fruchterman and Reingold 1991], which is very robust and for such a small graph of around ten nodes works in a fraction of a second. We restate some of the observations in [Pajntar 2006].

In this algorithm only two criteria are demanded for a good graph drawing:

- Connected nodes should be close.
- No two different vertices should be too close.

There is no penalization for edge crossings in this algorithm. Edge crossings are very important for clear drawing of a graph in general; however, we mostly need the final position of the nodes and graph structure is not very interesting since we have a full graph. There would also be a heavy computational penalty in calculating all the edge crossings (n^4) in every iteration.

In every iteration of the algorithm we calculate all the attracting forces from connected vertices, and the repulsing forces from all the nodes. Since there is no scalar penalty for edge crossings, the result is a vector, which points out not just how much out of place the vertex is (vector size) but also into which way it should move. The size of the actual move is confined to current temperature, which is linearly decreased in every iteration.

This very simple algorithm provides excellent results. Its main advantage is its speed and robustness. Even today, years after its creation, it is still one of the most popular algorithms for graph drawing.

After we get positions we can visualize the topics on the screen. We also visualize the most important word or bigram (a common pair of words) on top of each topic. Since for the clustering method there is a possibility of a poor word on the top place, we also visualize some additional top ranking words in a tooltip on a mouse over (Figure 4).

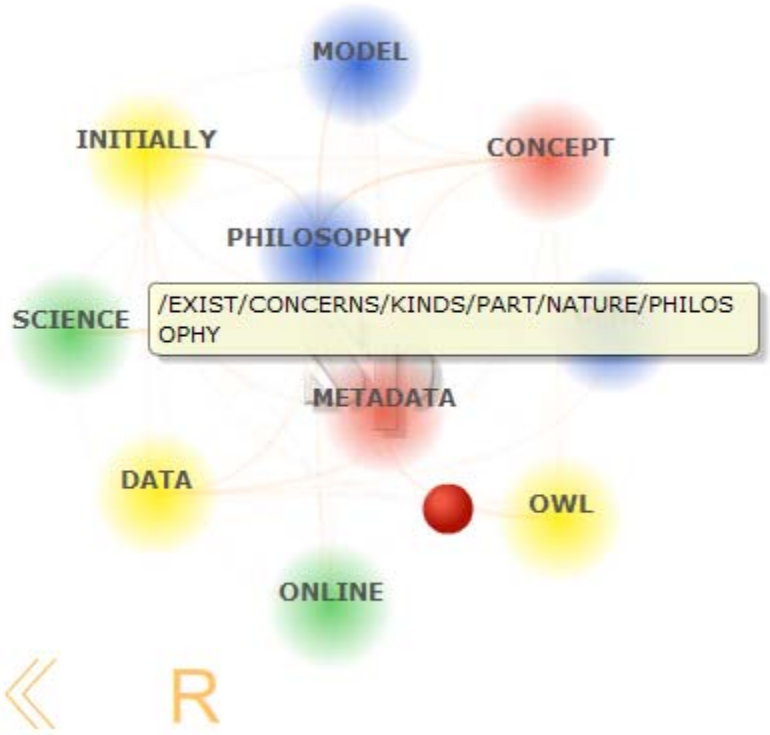

Figure 4: The hand cursor is above "Philosophy" topic. The additional words are: exist, concerns, kinds, part, nature.

2.4.2 Sub Graph Extraction

In the classifier model we could easily adopt graph drawing to visualize the best topics-classes. However, we would like to retain some of the structure that is inherent in the ontology model. The problem is that we can only visualize a small number of topics for one query. We would like our solution to work on any ontology even on simple taxonomies. This is why we first extract a connected subgraph that contains all the relevant concepts and then modify it so it can be visualized.

From the given set of relevant concepts we must first extract a subgraph. Because we want everything to work even on taxonomies we must use a very basic relation from which we will extract this subgraph. Since the only relation in treelike taxonomy is the subsumtion relation, in a case where the graph provided by the SubConceptOf relation is not connected, we connect every top concept in every connected part to a virtual node we name “Top”. The end result of this process is a connected tree, reaching all the relevant concepts that provide topics.

To visualize this we first need some sort of abstraction. After qualitative analysis we have decided to visualize three levels of this tree. We want to visualize as much as possible, however, more than three levels are too much information for the user and also negatively affect the reranking done with focus point selection.

For the first level we choose the root of the tree. This basically shows the user which part of the ontology models the current query and its results. For the third level we chose to visualize the computed relevant topics. In the ontology these concepts are not generally on the same level. However, we found it important to unify the visualization of the results of the classifying phase. For the second level we chose to visualize the actual second level below the root concept of the ontology. This informs the user of which part of the ontology the third level topics belong to, which is especially useful when the root topic is also the root of the ontology, or the virtual “Top” node.

The actual visualization is straightforward. We position the root at the origin and draw each consecutive level on concentric circles. Each third level node has equal space, while second level nodes get space proportional to the third level nodes they parent. The links between levels are also visualized and represent the ancestry relation between nodes.

Some visual aides have also been implemented in GUI. The actual score of third level topics is visualized as the size of nodes and for the second level nodes we sum the scores of all the descendant concepts in the ontology. Actual score from the classifier could be used for the visualization; however, we found it provides more information to visualize how much the whole subpart of the ontology models the current results in contrast to how much a single concept and its instances do. On mouse over a single topic the ancestor list of the concept is written in the tooltip. All of this can be seen on Figure 5, 6.

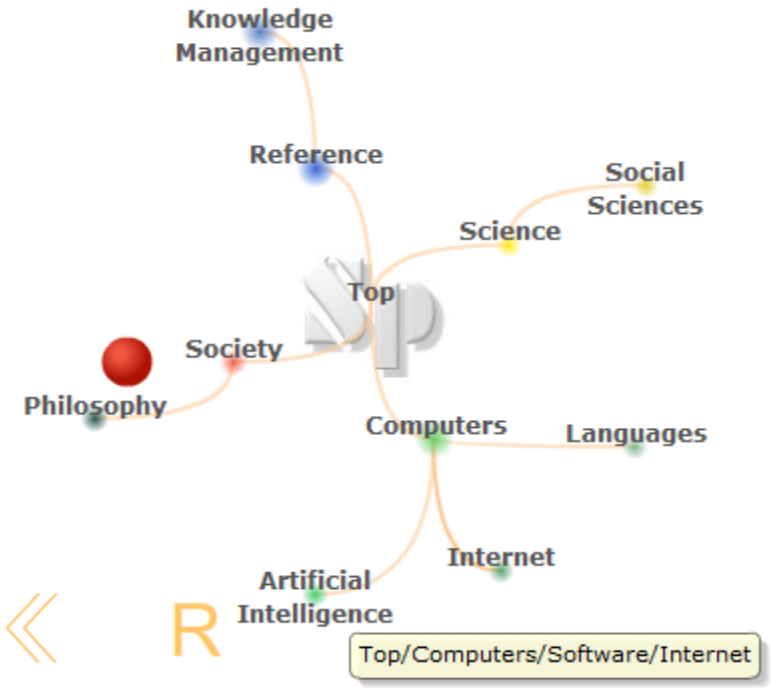

Figure 5: Classifier method for the query “ontology”. The ontology (taxonomy) used is DMOZ. The relevant concepts classified are: Philosophy, Knowledge Management, Social Sciences, Languages, Internet and Artificial Intelligence. The four main categories are Society, Reference, Science and Computers. Mouse over the concept “Internet” shows Top/Computers/Software/ Internet. The concept Software was left out of the visualization in order to make it clearer.

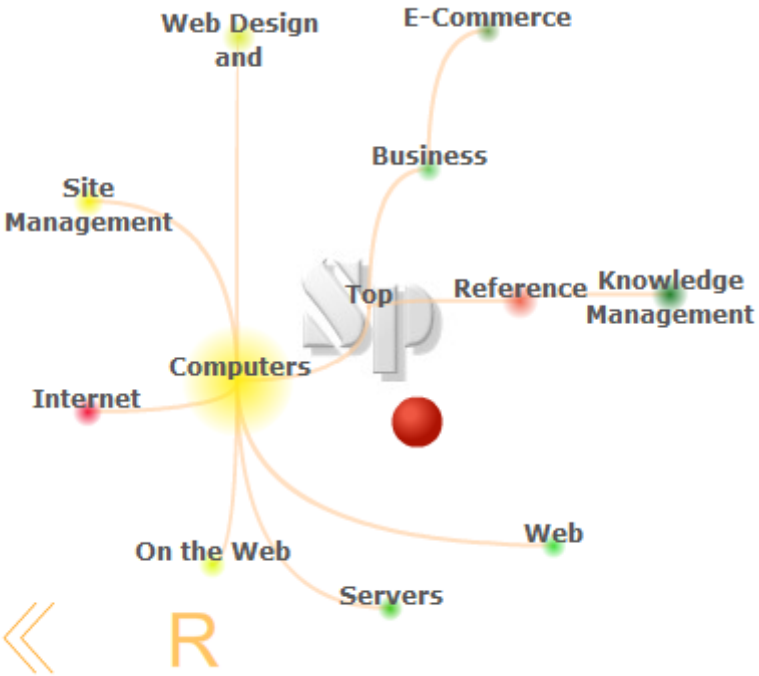

Figure 6: Visualization for the query “semantic web”. Immediately it is seen that this topic is mostly modelled with Computers and its descendants. Additionally there is some business and Knowledge management linked to it.

2.5 Calculation of the ranking space

Once the topics are positioned and visualized and all the documents are equipped with similarities to each topic, we must calculate the ranking space. Each selected focus point gets calculated in runtime when the user actually moves it. When dragging, this happens tens of times per second so this must be done efficiently.

When the SearchPoint starts, we also want to maintain the original ordering that is provided by the search engine, as we do not want to hinder the basic functionality. Because of this, we add a virtual invisible node and position it in the origin – the same as the starting position of the focus point. We also add document – virtual node similarities to documents in such a way that they observe the original ranking.

When the focus point moves, we must calculate a score for each document. We first define the position of the focus f(x, y), position of topics $t_i(x,y)$, documents d_i, and Documents – Topics similarity matrix S(i, j). The score for each document is proportional to the similarity of each topic and inversely proportional to the distance to that topic.

$$score\,(d_i) = \sum_{t_j \in T} \frac{S(i,\,j)}{dist\,(f\,, t_j) + \varepsilon}$$

The set T of all topics t_j also contains the invisible node that provides the original ranking. We take Euclidian distance for the dist(f, t_j) and we add a small ε in order to prevent division by zero. The hits get ordered and visualized by this scoring function.

3. Example usage of the system

In this section we will briefly describe the basic functionalities of the prototype (Section 3.1), then delve a bit deeper in the scenarios where the SearchPoint benefits the users (Section 3.2) and in the end we will showcase one use case accompanied with the screenshots (Section 3.3)

3.1 Basic description of the functionalities

GUI consists of an input field where the user can provide the query. The user can submit a query in several ways. Each method for topic extraction is connected with one submit button. Apart from this, there is a result list and the ranking space, where the user will be able to choose the focus point.

The search engine is selected by the URL of the web application. For the testing in this deliverable the following search engines have been used:

- Yahoo: http://searchpoint.ijs.si/
- Swoogle: http://searchpoint.ijs.si/swoogle/
- Google ontology search: http://searchpoint.ijs.si/googleowl/

After the user inputs the query and selects the method for topic extraction the result list is returned by the current search engine. At the beginning the results are ranked the same way as in the search engine. Each result item is visualized together with the original ranking. At the start this is therefore (1, 2, 3, 4…). The user can now inspect the topics for the current search and select any point on the ranking space or drag the focus point around the ranking space. The hits get reordered in real time, so the user is able to pinpoint a good selection for the focus easily, by just observing the quality of the displayed results.

There is also a history of the positions of the focus point. In case the user has found a good position and then moved the focus to see other rankings it is easily possible to navigate to a previous position by clicking on one of the three buttons of the history bar.

In the history bar there are three positions:

- Back: for returning the focus to one step back
- Forward: to move it to the next position
- Origin: to clear the history and return to default ranking provided by the search engine.

The buttons are only visible when they are available. For example the Forward button is not visible until Back has been clicked. No button is visible in the start, or upon clicking origin button, since there is no history yet at that time.

3.2 Discussion on Usability of SearchPoint

SearchPoint provides several benefits for the user. Here we will discuss and give examples for many of them. However, first we would like to point out that the basic functionality of the underlying search engine is not hindered in any way. For example, if the search engine provides adequate top ranked results for the query, the user can immediately follow that information without interacting with SearchPoint. SearchPoint provides additional functionalities at no extra cost to the user.

3.2.1 Query disambiguation

The most obvious use of SearchPoint is query disambiguation. Users are accustomed to writing very short queries, usually consisting only of a single word. In the case this word is ambiguous as, for example: owl, ontology, jaguar, a4 or kiwi, the results of both or all the meanings get returned.

This can be a problem when results of one meaning dominate the search space, since in such a case it is very hard to find the few results of the other meaning(s). Big web search engines deal with this problem by ranking the results of the less presented meanings higher in order to present a variety of meanings in the first ten results. Another possibility for the user is to refine a query, providing some additional words that further separate the meanings. This approach is better than the former in the sense that after the refinement, the user actually has ten results he can choose from. However, as simple as it is to refine a query it is still tasking the user.

SearchPoint actually serves as semi automatic query refinement. The user is presented with topics, which usually contain the one meaning the user wants and the one click re-ranking of the hits to the selected topic, can be seen by the user as a redefinition of the query. Another advantage is that the user does not have to come up with the correct best word for the separation of the meanings, which can cause problems to a user new to the topic or to a non-native speaker. The user merely has to recognize the topic or even only recognize the results, by randomly moving the focus point.

3.2.2 Sub-Topic exploration

Even when the actual meaning of the query is not ambiguous there can be several subtopics related to the results. For an uninformed user the very summarization of the main subtopics provided by the topics in the ranking space can be beneficial. What is more, the user can select a subtopic and immediately get documents about it. When two subtopics are related, they are visualized close and the user can select the focus point in between to get documents touching both of the subtopics.

For an example of such a sub-topic exploration, we give query “password” with the clustering method returning the following sub-topics:

- “Password Manager”... for the documents about software for managing passwords
- “Administrator”... for tools to moderate the passwords
- “Encryption”... for the various algorithms connected with passwords
- “Recovery”... for tools when losing a password
- “Forgotten Password”... also dealing with lost passwords, but more in the context of procedure of what to do than tools.
- “Change Password”... for best practices when dealing with passwords

Even for the user who is familiar with the topic it is sometimes hard to find the best query to get just some subtopics in the results. SearchPoint is extremely useful when dealing with subtopic profiling as it is designed in order to enable the user to select several subtopics with a single click.

3.2.3 Ranking in a general corpus

The biggest advantage of SearchPoint is creation of the ranking space. In a general corpus, which is usually a bag of unrelated documents, the usual search engines fail to produce any ranking. When the corpus is extensive, returning a large number of results for a single query, this becomes

a real problem. The visualized ranking is usually nothing more than a randomized order of all the relevant documents.

Most of the special searches have this problem. This is the reason we have chosen ontology search engines to showcase our solution. Another example would be a repository of images. We have an example at: http://searchpoint.ijs.si/photo12. This is a repository of textually annotated images provided by a company Photos12 that sells them. SearchPoint provides a useful service by providing ranking, sub-topic profiling and also by quickly finding images with two motifs which are presented as two topics in the ranking space.

3.3 Showcase

In the showcase we will demonstrate the contextualization power of SearchPoint. The main goal is to contextualize search over a given background knowledge with the use of automatic topic generation techniques and re-ranking possibilities of the ranking space.

The scenario will be that of a user tasked with finding the most suitable ontology to model the domain of fisheries. The user can first get a general overview of the domain by profiling the sub topics of the general web search engine results.

On searchpoint.ijs.si yahoo search engine is used. By querying for “fisheries” and selecting the classification method into Dmoz, the user can get a basic model of what concepts are connected with fisheries in an everyday context of general public (Figure 7).

Next, the user can assess how well represented topics are in truth on the web. This can be achieved using a clustering method. As can be seen on Figure 8, several topics (Alaska, Lake) seem to be over represented. This can probably be explained by the great importance the fishing industry has for special regions. Science, the most represented topic from before, is much less prominent. This is mostly on account of the industry (products, processes, fisheries management) that was missing before.

For the same query, “fisheries”, there is an obvious context switch from the more scientifically oriented Dmoz to the more economical World Wide Web.

The last context the user can interpret the fisheries results with is that of a legal vocabulary of European Union (EuroVoc). As can be seen in Figure 9, there is a broad range of topics. There is a cluster of more environmental topics (Aquaculture, Fishing regulations and Fisheries policy), however, more in a governing context than a scientific one. Industry is also well represented (Fishing industry, Fishery Product, Fishery Produce).

The user now has a firm understanding of the domain and several ways of modelling. On searchpoint.ijs.si/swoogle he can access the Swoogle ontology search engine. The third method is chosen for contextualization. When changing the search engine, topics also change (Figure 10). This is because ontology search space is different than general web search. Topics are smaller, probably due to somewhat lacking descriptions of the result ontologies, also, the industrial concepts seem to dominate over environmental concepts in the ontology space.

The user can nevertheless search for the ontology more easily and with respect to the chosen context.

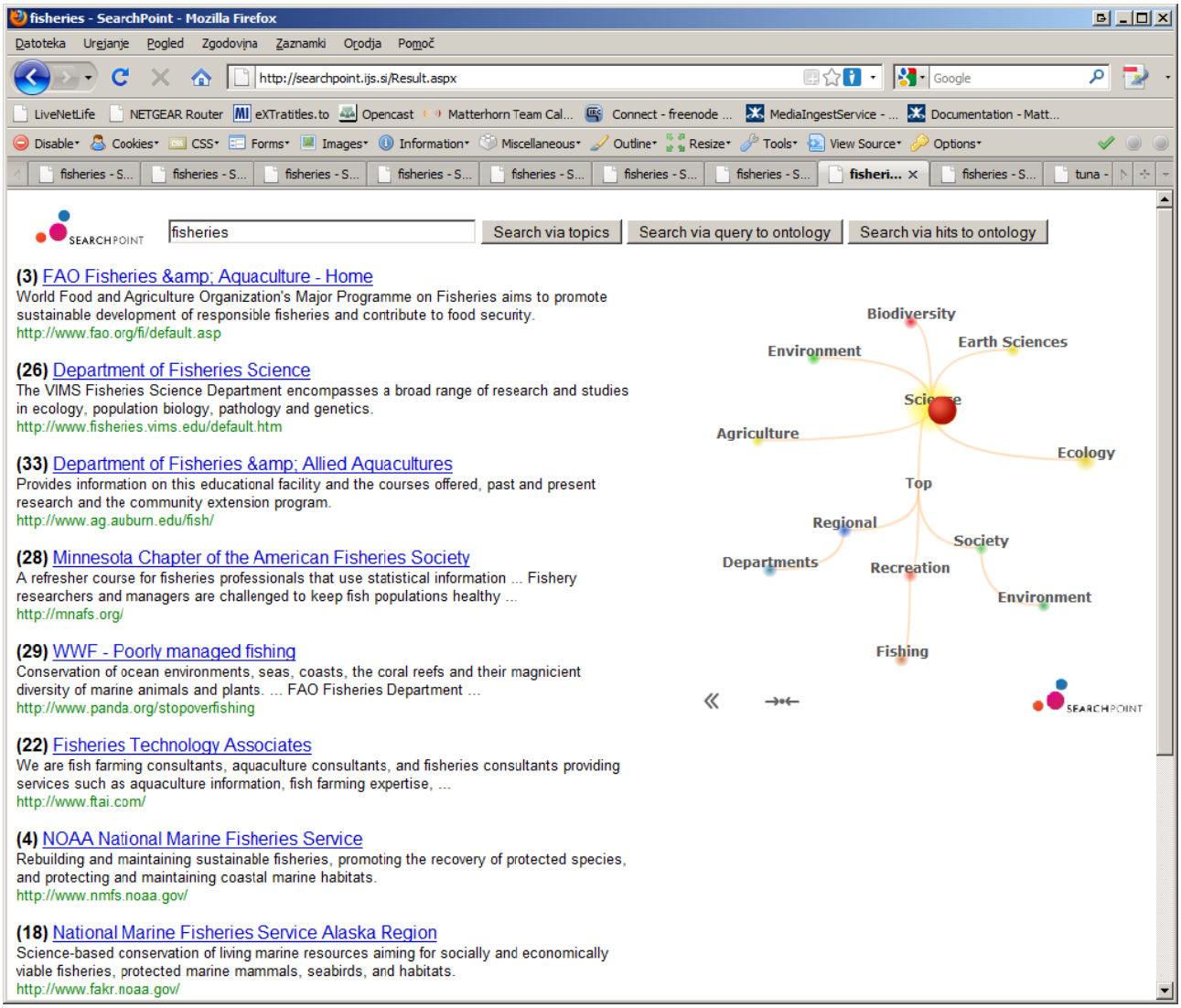

Figure 7: Yahoo web search in the context of Dmoz.

General topic of interest as provided in the Dmoz open directory. The topics most related to fishing mostly come from science (Agriculture, Environment, Biodiversity, Earth Sciences). There is also some notion of regional fisheries. Fisheries also seem connected to recreational fishing and there is an environmental concern in the society.

The focus point is moved to the centre of the science group and in the first eight results mostly institutions and science departments are ranked on top.

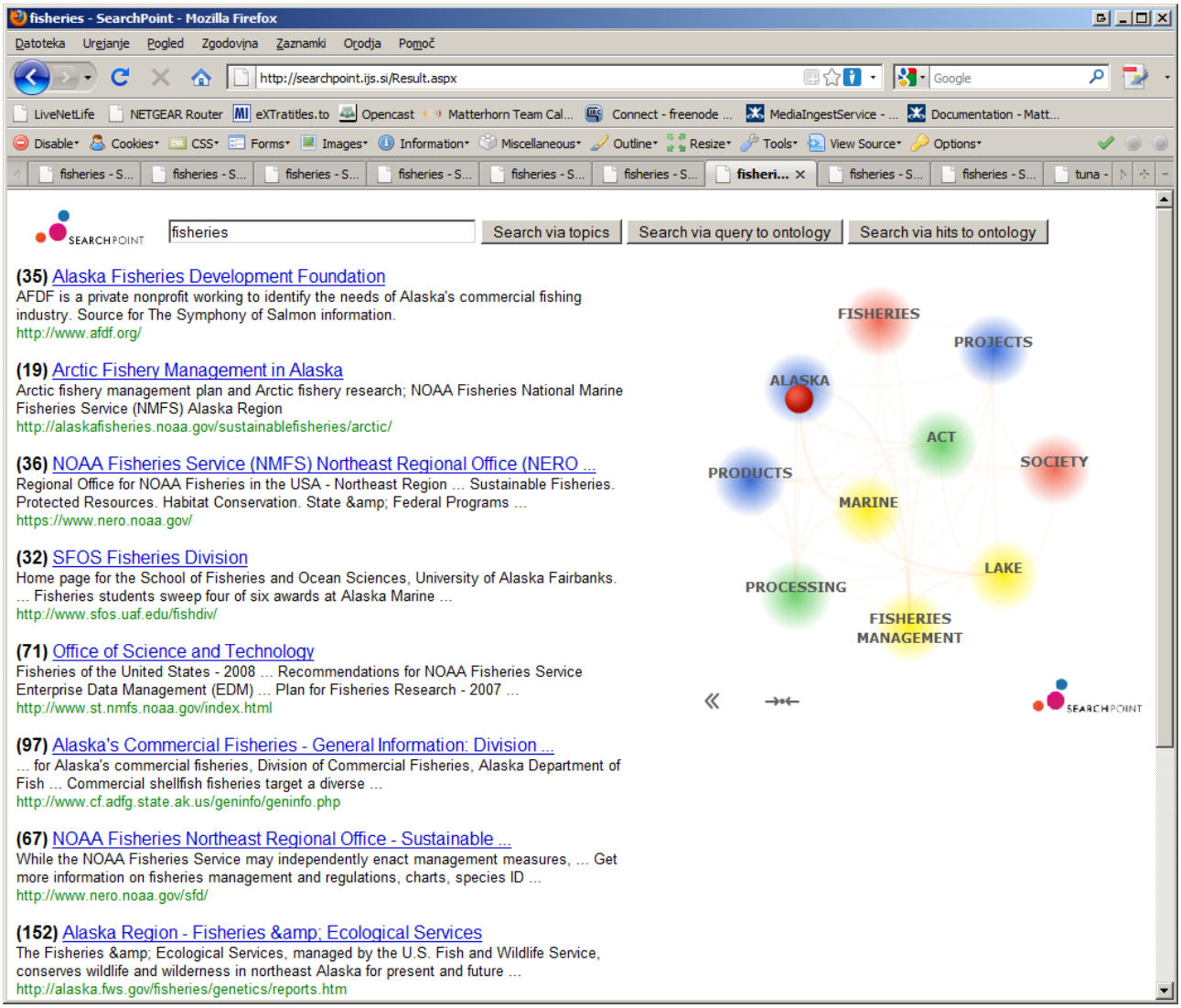

Figure 8: Yahoo web search with clustering.

Several prominent topics are extracted. There is a clear presence of economical topic (fisheries management, processing, products) and some more specific (Alaska, Lake).

The focus is moved towards Alaska topic, and there truly are many websites talking about Alaskan fisheries.

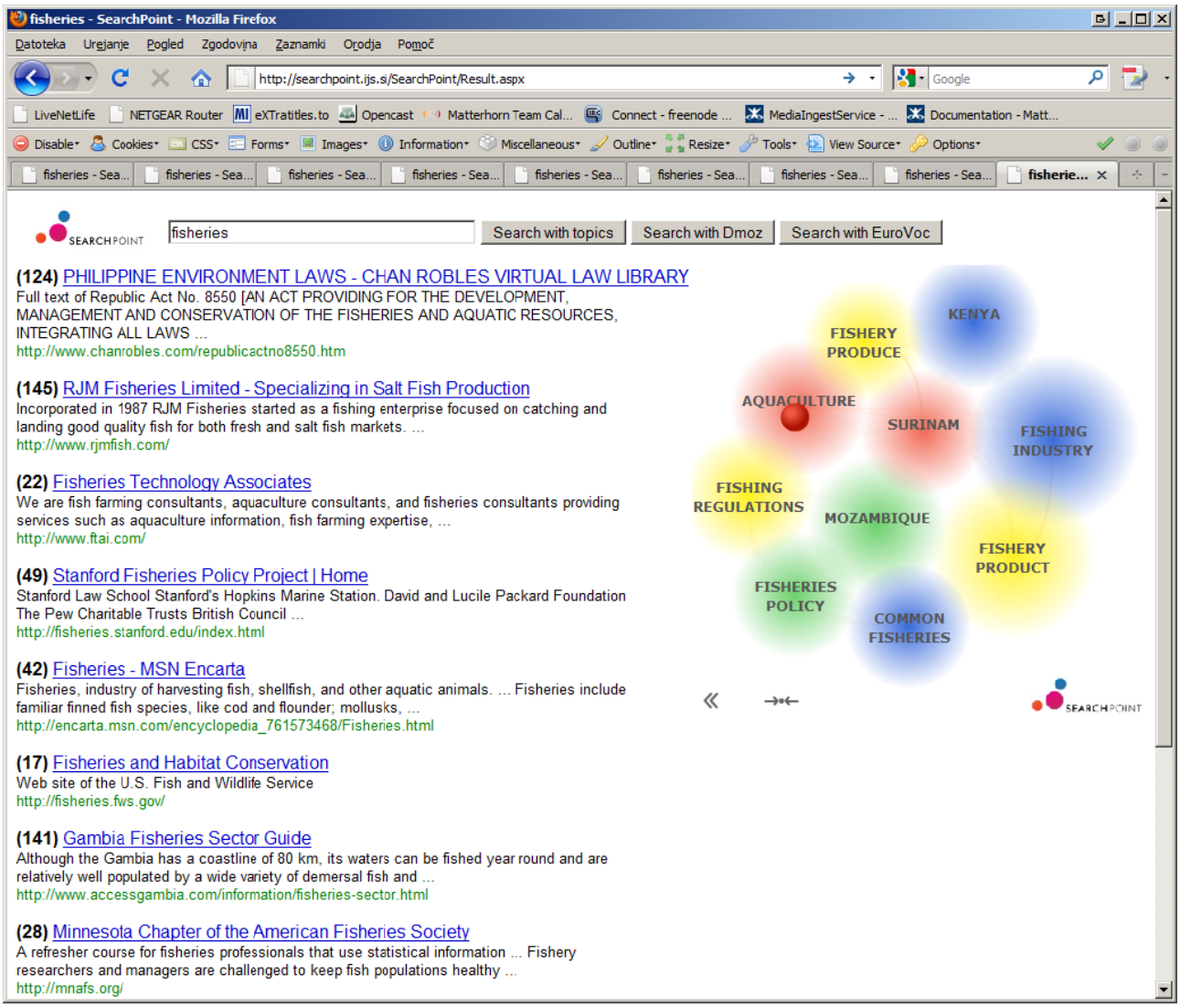

Figure 9: Yahoo web search in the context of EuroVoc.

Both industrial (Fishing Industry, Fishery Produce, Fishery Produce) and environmental (Aquaculture, Fishing regulations, Fisheries policy) aspects are well presented. However, this time more in the context of a government.

The focus is moved to the environmental part of the panel near the Aquaculture topic. Websites ranging from environmental (124 Philippine environment laws, 17), environmentally aware business (145, 22) and fishing societies (49, 28) share the Aquaculture context.

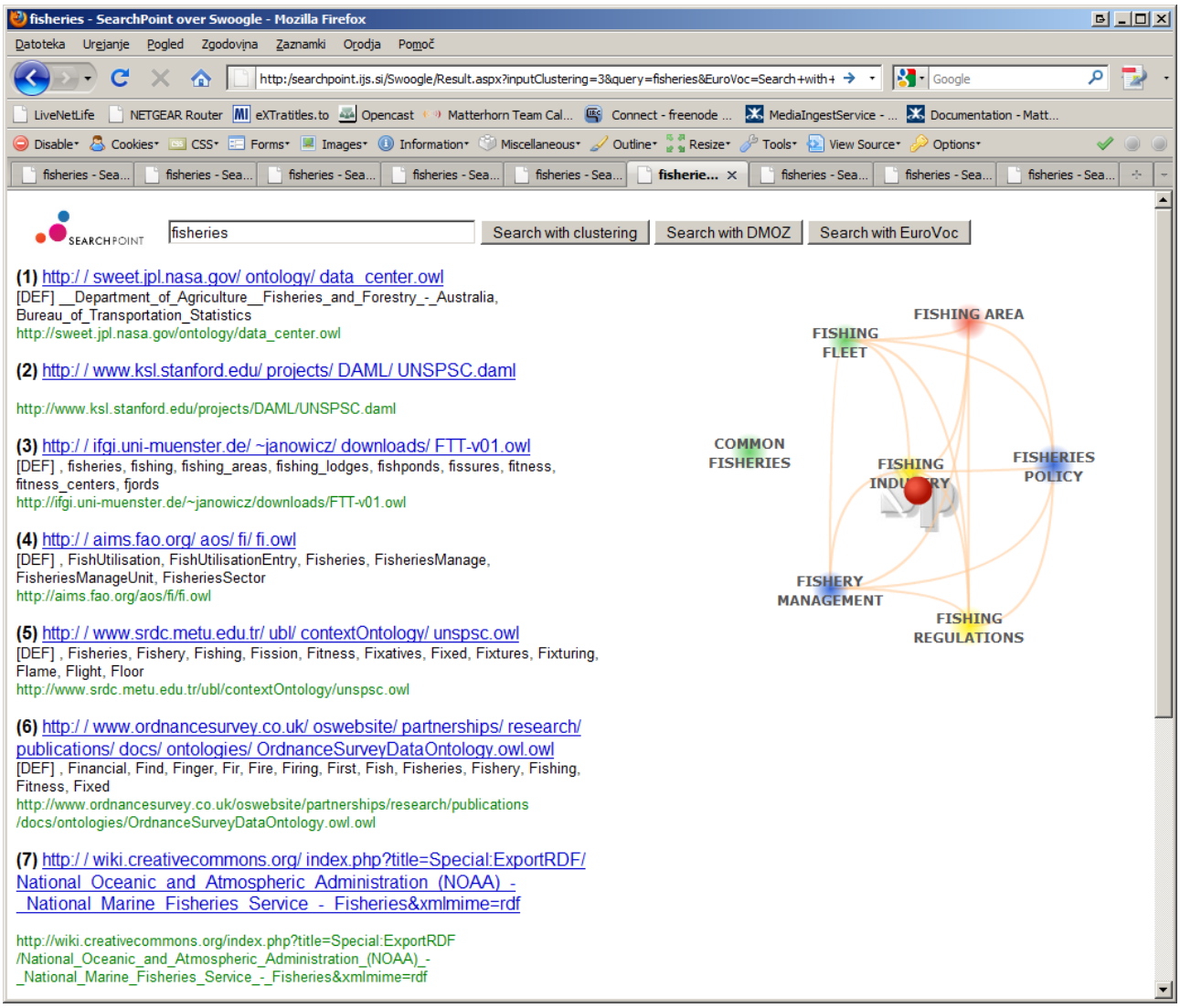

Figure 10: Swoogle Ontology Search in the context of EuroVoc.

The concepts in the ontology search space show most ontologies deal with the business oriented modelling. Also, the concepts are much smaller than before, because of lacking descriptions provided by the swoogle search engine.

4. Conclusion and future work

In this work we have extended a search engine add-on, SearchPoint, in order to enable searching within a context. Any general knowledge represented in a textual form can now be easily reranked by the user, selecting a focus point in the ranking space. Ranking space is created by visualizing topics stemming from clustering or classification into an ontology. Since the different choice of topic extraction provides different results for the user, this provides contextualization of general knowledge with the selection of the ranking space ontology.

Any textual background knowledge can be indexed and documents containing query words can easily be retrieved. However, when the corpus being searched is extensive, the problem of good ranking occurs. This problem has been solved for web-search mainly due to the underlying graph structure of the linking pages and due also to the redundancy of information on the web. This approach does not work on an arbitrary corpus of documents or other more specific search tasks where there is no underlying graph structure, and indeed different rankings are needed for different users and even for the same user when searching in a different context.

SearchPoint enables the user to get a continuum of rankings for one query which helps to disambiguate the query, provide a one-click query reformulation, or can even be used for subtopic profiling of the current result space. All this is provided by the standard SearchPoint. On top of this, it is possible for the user to exchange the ranking space ontology and, through this, change the context in which the ranking space is calculated. Only topics that are relevant to both the search results and the current context are visualized and create a special contextualized ranking space in which the user can rerank the results.

This is a finished working prototype available as a web application. This work will continue in WP4 where it will be considered how to best integrate it inside the NeOn Toolkit platform, to be most easily and intuitively used by the users. Thought will also be given as to how it will be integrated with Watson Ontology search engine, probably with key concepts (developed in WP4) and ontology similarities (developed in WP3) to help with the automatic topic extraction and actual ranking process.

Appendix A

Here we give some technical information about the SOAP/http webservice which does the automatic topic extraction and also provides the positioning of the topics with graph drawing. The WSDL specification of this web service can be found at http://searchpoint.ijs.si/Classifier/WS_Classify.asmx?WSDL and is also given in A.1.

The main methods are:

- ClassifyKMeans which does the clustering method for topic extraction and graph drawing for the positioning of the topics
- ClassifyDMoz which does the classification to DMOZ open directory for topic generation and positions the topics by extracting the relevant sub-tree.
- ClassifyEuroVoc which does the classification into EuroVoc Terminology extended with Acquis communautaire legislation documents that provide grounding.

The data is returned as a BowPartStruct structure, which contains three sub structures:

- Information about the topics, with relevant keywords or place in the ontology for each cluster and concept respectively is stored in Node structure.
- It also contains the similarities between each topic. This is stored in Links structure
- Next, all the results are given a vector of similarities to each topic. This is stored in Documents structure.

Several search engines can be called for the actual results:

- Big web search engines: Google, Yahoo, Bing
- Newspapers: NYTimes, About.com, Mladina (Slovenian)
- Ontology search: Watson, Swoogle, GoogleOWL
- Some specific search engines: EBay, Enron, CCA, Photo12
- It is also possible to provide results of any search engine:
    - As a serialized xml: String
    - As a url to such an xml: File

A.1 The current WSDL file to the web service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://searchpoint.ijs.si/Classifier"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://searchpoint.ijs.si/Classifier" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
 <wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://searchpoint.ijs.si/Classifier">
   <s:import namespace="http://searchpoint.ijs.si/Classifier/BowPart.xsd" />
```

```
    <s:import schemaLocation="http://localhost:60107/Classifier/WS_Classify.asmx?schema=BowPart"
namespace="http://searchpoint.ijs.si/Classifier/BowPart.xsd" />
    <s:element name="ClassifyKMeans">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="DS" type="tns:DataSource" />
       <s:element minOccurs="0" maxOccurs="1" name="Query" type="s:string" />
       <s:element minOccurs="1" maxOccurs="1" name="NumHits" type="s:int" />
       <s:element minOccurs="1" maxOccurs="1" name="NumMinHits" type="s:int" />
       <s:element minOccurs="1" maxOccurs="1" name="NumCategories" type="s:int" />
      </s:sequence>
     </s:complexType>
    </s:element>
    <s:simpleType name="DataSource">
     <s:restriction base="s:string">
      <s:enumeration value="Google" />
      <s:enumeration value="Yahoo" />
      <s:enumeration value="Live" />
      <s:enumeration value="About" />
      <s:enumeration value="AboutViaGoogle" />
      <s:enumeration value="AboutViaYahoo" />
      <s:enumeration value="EBay" />
      <s:enumeration value="NYTimes" />
      <s:enumeration value="Mladina" />
      <s:enumeration value="Watson" />
      <s:enumeration value="Enron" />
      <s:enumeration value="CCA" />
      <s:enumeration value="Photo12" />
      <s:enumeration value="String" />
      <s:enumeration value="File" />
      <s:enumeration value="GoogleOnto" />
      <s:enumeration value="Swoogle" />
     </s:restriction>
    </s:simpleType>
    <s:element name="ClassifyKMeansResponse">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="ClassifyKMeansResult">
        <s:complexType>
         <s:sequence>
          <s:any namespace="http://searchpoint.ijs.si/Classifier/BowPart.xsd" />
         </s:sequence>
        </s:complexType>
       </s:element>
```

```
      </s:sequence>
     </s:complexType>
    </s:element>
    <s:element name="ClassifyKMeansIn">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="DS" type="tns:DataSource" />
       <s:element minOccurs="0" maxOccurs="1" name="Query" type="s:string" />
       <s:element minOccurs="1" maxOccurs="1" name="NumHits" type="s:int" />
       <s:element minOccurs="1" maxOccurs="1" name="NumMinHits" type="s:int" />
       <s:element minOccurs="1" maxOccurs="1" name="NumCategories" type="s:int" />
      </s:sequence>
     </s:complexType>
    </s:element>
    <s:element name="ClassifyKMeansInResponse">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="ClassifyKMeansInResult"
type="tns:BowPartStruc" />
      </s:sequence>
     </s:complexType>
    </s:element>
    <s:complexType name="BowPartStruc">
     <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="Clusters" type="tns:ArrayOfCluster" />
      <s:element minOccurs="0" maxOccurs="1" name="Links" type="tns:ArrayOfLink" />
      <s:element minOccurs="0" maxOccurs="1" name="Documents" type="tns:ArrayOfDocument" />
     </s:sequence>
    </s:complexType>
    <s:complexType name="ArrayOfCluster">
     <s:sequence>
      <s:element minOccurs="0" maxOccurs="unbounded" name="Cluster" type="tns:Cluster" />
     </s:sequence>
    </s:complexType>
    <s:complexType name="Cluster">
     <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="Clusters_Id" type="s:int" />
      <s:element minOccurs="0" maxOccurs="1" name="Title" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="Color" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="Quality" type="s:double" />
      <s:element minOccurs="1" maxOccurs="1" name="X" type="s:double" />
      <s:element minOccurs="1" maxOccurs="1" name="Y" type="s:double" />
     </s:sequence>
    </s:complexType>
```

```
<s:complexType name="ArrayOfLink">
 <s:sequence>
  <s:element minOccurs="0" maxOccurs="unbounded" name="Link" type="tns:Link" />
 </s:sequence>
</s:complexType>
<s:complexType name="Link">
 <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="id" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="id1" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="id2" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="Quality" type="s:double" />
 </s:sequence>
</s:complexType>
<s:complexType name="ArrayOfDocument">
 <s:sequence>
  <s:element minOccurs="0" maxOccurs="unbounded" name="Document" type="tns:Document" />
 </s:sequence>
</s:complexType>
<s:complexType name="Document">
 <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="id" type="s:double" />
  <s:element minOccurs="1" maxOccurs="1" name="relevance" type="s:double" />
  <s:element minOccurs="0" maxOccurs="1" name="dcSim" type="s:string" />
 </s:sequence>
</s:complexType>
<s:element name="ClassifyDMoz">
 <s:complexType>
  <s:sequence>
   <s:element minOccurs="1" maxOccurs="1" name="DS" type="tns:DataSource" />
   <s:element minOccurs="0" maxOccurs="1" name="Query" type="s:string" />
   <s:element minOccurs="1" maxOccurs="1" name="NumHits" type="s:int" />
   <s:element minOccurs="1" maxOccurs="1" name="NumMinHits" type="s:int" />
   <s:element minOccurs="1" maxOccurs="1" name="NumCategories" type="s:int" />
  </s:sequence>
 </s:complexType>
</s:element>
<s:element name="ClassifyDMozResponse">
 <s:complexType>
  <s:sequence>
   <s:element minOccurs="0" maxOccurs="1" name="ClassifyDMozResult">
    <s:complexType>
     <s:sequence>
      <s:any namespace="http://searchpoint.ijs.si/Classifier/BowPart.xsd" />
```

```
        </s:sequence>
       </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="ClassifyDMozContext">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="DS" type="tns:DataSource" />
      <s:element minOccurs="0" maxOccurs="1" name="Query" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="NumHits" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="NumMinHits" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="NumCategories" type="s:int" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="ClassifyDMozContextResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="ClassifyDMozContextResult">
        <s:complexType>
          <s:sequence>
            <s:any namespace="http://searchpoint.ijs.si/Classifier/BowPart.xsd" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="ClassifyGY">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="DS" type="tns:DataSource" />
      <s:element minOccurs="0" maxOccurs="1" name="Query" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="NumHits" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="NumCategories" type="s:int" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="ClassifyGYResponse">
  <s:complexType>
    <s:sequence>
```

```
      <s:element minOccurs="0" maxOccurs="1" name="ClassifyGYResult">
       <s:complexType>
        <s:sequence>
         <s:any namespace="http://searchpoint.ijs.si/Classifier/BowPart.xsd" />
        </s:sequence>
       </s:complexType>
      </s:element>
     </s:sequence>
    </s:complexType>
   </s:element>
   <s:element name="ClassifyEuroVoc">
    <s:complexType>
     <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="DS" type="tns:DataSource" />
      <s:element minOccurs="0" maxOccurs="1" name="Query" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="NumHits" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="NumMinHits" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="NumCategories" type="s:int" />
     </s:sequence>
    </s:complexType>
   </s:element>
   <s:element name="ClassifyEuroVocResponse">
    <s:complexType>
     <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="ClassifyEuroVocResult">
       <s:complexType>
        <s:sequence>
         <s:any namespace="http://searchpoint.ijs.si/Classifier/BowPart.xsd" />
        </s:sequence>
       </s:complexType>
      </s:element>
     </s:sequence>
    </s:complexType>
   </s:element>
   <s:element name="BowPart" nillable="true">
    <s:complexType>
     <s:sequence>
      <s:any namespace="http://searchpoint.ijs.si/Classifier/BowPart.xsd" />
     </s:sequence>
    </s:complexType>
   </s:element>
   <s:element name="BowPartStruc" type="tns:BowPartStruc" />
  </s:schema>
```

```
</wsdl:types>
<wsdl:message name="ClassifyKMeansSoapIn">
  <wsdl:part name="parameters" element="tns:ClassifyKMeans" />
</wsdl:message>
<wsdl:message name="ClassifyKMeansSoapOut">
  <wsdl:part name="parameters" element="tns:ClassifyKMeansResponse" />
</wsdl:message>
<wsdl:message name="ClassifyKMeansInSoapIn">
  <wsdl:part name="parameters" element="tns:ClassifyKMeansIn" />
</wsdl:message>
<wsdl:message name="ClassifyKMeansInSoapOut">
  <wsdl:part name="parameters" element="tns:ClassifyKMeansInResponse" />
</wsdl:message>
<wsdl:message name="ClassifyDMozSoapIn">
  <wsdl:part name="parameters" element="tns:ClassifyDMoz" />
</wsdl:message>
<wsdl:message name="ClassifyDMozSoapOut">
  <wsdl:part name="parameters" element="tns:ClassifyDMozResponse" />
</wsdl:message>
<wsdl:message name="ClassifyDMozContextSoapIn">
  <wsdl:part name="parameters" element="tns:ClassifyDMozContext" />
</wsdl:message>
<wsdl:message name="ClassifyDMozContextSoapOut">
  <wsdl:part name="parameters" element="tns:ClassifyDMozContextResponse" />
</wsdl:message>
<wsdl:message name="ClassifyGYSoapIn">
  <wsdl:part name="parameters" element="tns:ClassifyGY" />
</wsdl:message>
<wsdl:message name="ClassifyGYSoapOut">
  <wsdl:part name="parameters" element="tns:ClassifyGYResponse" />
</wsdl:message>
<wsdl:message name="ClassifyEuroVocSoapIn">
  <wsdl:part name="parameters" element="tns:ClassifyEuroVoc" />
</wsdl:message>
<wsdl:message name="ClassifyEuroVocSoapOut">
  <wsdl:part name="parameters" element="tns:ClassifyEuroVocResponse" />
</wsdl:message>
<wsdl:message name="ClassifyKMeansHttpGetIn">
  <wsdl:part name="DS" type="s:string" />
  <wsdl:part name="Query" type="s:string" />
  <wsdl:part name="NumHits" type="s:string" />
  <wsdl:part name="NumMinHits" type="s:string" />
  <wsdl:part name="NumCategories" type="s:string" />
```

```
</wsdl:message>
<wsdl:message name="ClassifyKMeansHttpGetOut">
  <wsdl:part name="Body" element="tns:BowPart" />
</wsdl:message>
<wsdl:message name="ClassifyKMeansInHttpGetIn">
  <wsdl:part name="DS" type="s:string" />
  <wsdl:part name="Query" type="s:string" />
  <wsdl:part name="NumHits" type="s:string" />
  <wsdl:part name="NumMinHits" type="s:string" />
  <wsdl:part name="NumCategories" type="s:string" />
</wsdl:message>
<wsdl:message name="ClassifyKMeansInHttpGetOut">
  <wsdl:part name="Body" element="tns:BowPartStruc" />
</wsdl:message>
<wsdl:message name="ClassifyDMozHttpGetIn">
  <wsdl:part name="DS" type="s:string" />
  <wsdl:part name="Query" type="s:string" />
  <wsdl:part name="NumHits" type="s:string" />
  <wsdl:part name="NumMinHits" type="s:string" />
  <wsdl:part name="NumCategories" type="s:string" />
</wsdl:message>
<wsdl:message name="ClassifyDMozHttpGetOut">
  <wsdl:part name="Body" element="tns:BowPart" />
</wsdl:message>
<wsdl:message name="ClassifyDMozContextHttpGetIn">
  <wsdl:part name="DS" type="s:string" />
  <wsdl:part name="Query" type="s:string" />
  <wsdl:part name="NumHits" type="s:string" />
  <wsdl:part name="NumMinHits" type="s:string" />
  <wsdl:part name="NumCategories" type="s:string" />
</wsdl:message>
<wsdl:message name="ClassifyDMozContextHttpGetOut">
  <wsdl:part name="Body" element="tns:BowPart" />
</wsdl:message>
<wsdl:message name="ClassifyGYHttpGetIn">
  <wsdl:part name="DS" type="s:string" />
  <wsdl:part name="Query" type="s:string" />
  <wsdl:part name="NumHits" type="s:string" />
  <wsdl:part name="NumCategories" type="s:string" />
</wsdl:message>
<wsdl:message name="ClassifyGYHttpGetOut">
  <wsdl:part name="Body" element="tns:BowPart" />
</wsdl:message>
```

```
<wsdl:message name="ClassifyEuroVocHttpGetIn">
  <wsdl:part name="DS" type="s:string" />
  <wsdl:part name="Query" type="s:string" />
  <wsdl:part name="NumHits" type="s:string" />
  <wsdl:part name="NumMinHits" type="s:string" />
  <wsdl:part name="NumCategories" type="s:string" />
</wsdl:message>
<wsdl:message name="ClassifyEuroVocHttpGetOut">
  <wsdl:part name="Body" element="tns:BowPart" />
</wsdl:message>
<wsdl:portType name="WS_ClassifySoap">
  <wsdl:operation name="ClassifyKMeans">
    <wsdl:input message="tns:ClassifyKMeansSoapIn" />
    <wsdl:output message="tns:ClassifyKMeansSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ClassifyKMeansIn">
    <wsdl:input message="tns:ClassifyKMeansInSoapIn" />
    <wsdl:output message="tns:ClassifyKMeansInSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ClassifyDMoz">
    <wsdl:input message="tns:ClassifyDMozSoapIn" />
    <wsdl:output message="tns:ClassifyDMozSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ClassifyDMozContext">
    <wsdl:input message="tns:ClassifyDMozContextSoapIn" />
    <wsdl:output message="tns:ClassifyDMozContextSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ClassifyGY">
    <wsdl:input message="tns:ClassifyGYSoapIn" />
    <wsdl:output message="tns:ClassifyGYSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ClassifyEuroVoc">
    <wsdl:input message="tns:ClassifyEuroVocSoapIn" />
    <wsdl:output message="tns:ClassifyEuroVocSoapOut" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="WS_ClassifyHttpGet">
  <wsdl:operation name="ClassifyKMeans">
    <wsdl:input message="tns:ClassifyKMeansHttpGetIn" />
    <wsdl:output message="tns:ClassifyKMeansHttpGetOut" />
  </wsdl:operation>
  <wsdl:operation name="ClassifyKMeansIn">
    <wsdl:input message="tns:ClassifyKMeansInHttpGetIn" />
```

```
    <wsdl:output message="tns:ClassifyKMeansInHttpGetOut" />
   </wsdl:operation>
   <wsdl:operation name="ClassifyDMoz">
    <wsdl:input message="tns:ClassifyDMozHttpGetIn" />
    <wsdl:output message="tns:ClassifyDMozHttpGetOut" />
   </wsdl:operation>
   <wsdl:operation name="ClassifyDMozContext">
    <wsdl:input message="tns:ClassifyDMozContextHttpGetIn" />
    <wsdl:output message="tns:ClassifyDMozContextHttpGetOut" />
   </wsdl:operation>
   <wsdl:operation name="ClassifyGY">
    <wsdl:input message="tns:ClassifyGYHttpGetIn" />
    <wsdl:output message="tns:ClassifyGYHttpGetOut" />
   </wsdl:operation>
   <wsdl:operation name="ClassifyEuroVoc">
    <wsdl:input message="tns:ClassifyEuroVocHttpGetIn" />
    <wsdl:output message="tns:ClassifyEuroVocHttpGetOut" />
   </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="WS_ClassifySoap" type="tns:WS_ClassifySoap">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="ClassifyKMeans">
    <soap:operation soapAction="http://searchpoint.ijs.si/Classifier/ClassifyKMeans" style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ClassifyKMeansIn">
    <soap:operation   soapAction="http://searchpoint.ijs.si/Classifier/ClassifyKMeansIn"   style="document"
/>
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ClassifyDMoz">
    <soap:operation soapAction="http://searchpoint.ijs.si/Classifier/ClassifyDMoz" style="document" />
    <wsdl:input>
     <soap:body use="literal" />
```

```
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ClassifyDMozContext">
    <soap:operation soapAction="http://searchpoint.ijs.si/Classifier/ClassifyDMozContext"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ClassifyGY">
    <soap:operation soapAction="http://searchpoint.ijs.si/Classifier/ClassifyGY" style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ClassifyEuroVoc">
    <soap:operation soapAction="http://searchpoint.ijs.si/Classifier/ClassifyEuroVoc" style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="WS_ClassifySoap12" type="tns:WS_ClassifySoap">
   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="ClassifyKMeans">
    <soap12:operation soapAction="http://searchpoint.ijs.si/Classifier/ClassifyKMeans" style="document"
/>
    <wsdl:input>
     <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap12:body use="literal" />
```

```
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClassifyKMeansIn">
    <soap12:operation soapAction="http://searchpoint.ijs.si/Classifier/ClassifyKMeansIn" style="document"
/>
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClassifyDMoz">
    <soap12:operation soapAction="http://searchpoint.ijs.si/Classifier/ClassifyDMoz" style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClassifyDMozContext">
    <soap12:operation                soapAction="http://searchpoint.ijs.si/Classifier/ClassifyDMozContext"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClassifyGY">
    <soap12:operation soapAction="http://searchpoint.ijs.si/Classifier/ClassifyGY" style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClassifyEuroVoc">
    <soap12:operation  soapAction="http://searchpoint.ijs.si/Classifier/ClassifyEuroVoc"  style="document"
/>
    <wsdl:input>
```

```
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="WS_ClassifyHttpGet" type="tns:WS_ClassifyHttpGet">
  <http:binding verb="GET" />
  <wsdl:operation name="ClassifyKMeans">
    <http:operation location="/ClassifyKMeans" />
    <wsdl:input>
      <http:urlEncoded />
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClassifyKMeansIn">
    <http:operation location="/ClassifyKMeansIn" />
    <wsdl:input>
      <http:urlEncoded />
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClassifyDMoz">
    <http:operation location="/ClassifyDMoz" />
    <wsdl:input>
      <http:urlEncoded />
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ClassifyDMozContext">
    <http:operation location="/ClassifyDMozContext" />
    <wsdl:input>
      <http:urlEncoded />
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body" />
```

```
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ClassifyGY">
    <http:operation location="/ClassifyGY" />
    <wsdl:input>
     <http:urlEncoded />
    </wsdl:input>
    <wsdl:output>
     <mime:mimeXml part="Body" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ClassifyEuroVoc">
    <http:operation location="/ClassifyEuroVoc" />
    <wsdl:input>
     <http:urlEncoded />
    </wsdl:input>
    <wsdl:output>
     <mime:mimeXml part="Body" />
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="WS_Classify">
   <wsdl:port name="WS_ClassifySoap" binding="tns:WS_ClassifySoap">
    <soap:address location="http://localhost:60107/Classifier/WS_Classify.asmx" />
   </wsdl:port>
   <wsdl:port name="WS_ClassifySoap12" binding="tns:WS_ClassifySoap12">
    <soap12:address location="http://localhost:60107/Classifier/WS_Classify.asmx" />
   </wsdl:port>
   <wsdl:port name="WS_ClassifyHttpGet" binding="tns:WS_ClassifyHttpGet">
    <http:address location="http://localhost:60107/Classifier/WS_Classify.asmx" />
   </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

References

[d'Aquin 2008] d'Aquin, M. Building Semantic Web Based Applications with Watson. 17th International World Wide Web Conference (WWW2008) Developers Track.

[Fortuna et all, 2005] B. Fortuna, M. Grobelnik, D. Mladenic: Semi-automatic Construction of Topic Ontology. *Semantics, Web and Mining, Joint International Workshop*, EWMF 2005 and KDO 2005, Porto, Portugal, October 3–7, 2005.

[Fruchterman and Reingold, 1991] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force directed placement. Softw. Pract. Exper., 1991.

[Grobelnik and Mladenić, 2006] Grobelnik, M., Mladenić, D., (2006). Automated Knowledge Discovery in Advanced Knowledge Management, journal of Knowledge Management.

[Grobelnik et al., 2008] Marko Grobelnik, Janez Brank, Blaž Fortuna, Igor Mozetič. Contextualizing Ontologies with OntoLight: A Pragmatic Approach, Informatica. 2008.

[Kanungo et all, 2002] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu (2002). An efficient k-means clustering algorithm: Analysis and implementation, IEEE Trans. Pattern Analysis and Machine Intelligence, 24 (2002), 881-892.

[Koller and Sahami, 1997] D., Sahami, M., (1997). Hierarchically classifying documents using very few words, Proceedings of the 14th International Conference on Machine Learning ICML-97, pp. 170-178, Morgan Kaufmann, San Francisco, CA.

[McCallum et all, 1998] McCallum A., Rosenfeld R., Mitchell T., Ng A., (1998). Improving Text Classification by Shrinkage in a Hierarchy of Classes, Proceedings of the 15th International Conference on Machine Learning ICML-98, Morgan Kaufmann, San Francisco, CA.

[Mitchell, 1997] Mitchell, T.M. (1997). Machine Learning. The McGraw-Hill Companies, Inc.

[Mladenić, 1998] Mladenić, D. (1998). Turning Yahoo into an Automatic Web-Page Classifier. Proc. 13th European Conference on Artificial Intelligence (ECAI'98, John Wiley & Sons), 473–474.

[Mladenić and Grobelnik, 2003] Mladenić, D., Grobelnik, M. (2003). Feature selection on hierarchy of web documents. Journal of Decision support systems, 35, 45-87.

[Pajntar, 2006] Pajntar B. (2006), Overview of algorithms for graph drawing, In proc. of Slovenian KDD Conference 2006. Oct. 2006

[Pajntar and Grobelnik, 2008] Boštjan Pajntar, Marko Grobelnik. SearchPoint – a New Paradigm of Web Search. 17th International World Wide Web Conference (WWW2008) Developers Track.

[Sabou et all, 2006] M. Sabou, M. d'Aquin, and E. Motta: Using the Semantic Web as Background Knowledge for Ontology Mapping, In *Proceedings of the International Workshop on Ontology Matching (OM-2006)*, collocated with ISWC-06.

[Steingbach et all, 2000] Steinbach, M., Karypis, G. and Kumar, V. (2000). A comparison of document clustering techniques. Proc. KDD Workshop on Text Mining. (eds. Grobelnik, M., Mladenić, D. and Milic-Frayling, N.), Boston, MA, USA, 109–110.