1 On Knowledgeable Unsupervised Text Mining

A. Hotho and A. Maedche and S. Staab and V. Zacharias

Forschungszentrum Informatik at the Univ. Karlsruhe,
D-76131 Karlsruhe, Germany
{maedche, zach}@fzi.de, http://www.fzi.de/WIM

Institute AIFB, Univ. Karlsruhe,
D-76128 Karlsruhe, Germany
{hotho, staab}@aifb.uni-karlsruhe.de,
http://www.aifb.uni-karlsruhe.de/WBS

Ontoprise GmbH
D-76131 Karlsruhe, Germany
staab@ontoprise.de, http://www.ontoprise.de

Abstract Text Mining is about discovering novel, interesting and useful patterns from textual data. In this paper we discuss several means that introduce background knowledge into unsupervised text mining in order to improve the novelty, the interestingness or the usefulness of the detected patterns. Germane to the different proposals is that they strive for higher abstractions that carry more explanatory power and more possibilities for exploring the input texts than is achievable by unknowledgeable means.

1.1 Introduction

Knowledge discovery is concerned with finding novel, interesting and useful patterns in data. In order to successfully discover such patterns, it is necessary that the data that is investigated is structured in a way accessible to machine learning algorithms. Texts, however, do not show much structure to the eye of the naive machine learning algorithm. Only in the eye of the human beholder, text documents exhibit the rich linguistic and conceptual structures that may let him discover patterns that are not explicit.[1] Based on these considerations we may conjecture that in order to improve the effectiveness and utility of machine learning on texts, we must improve the linguistic and/or the conceptual background knowledge available to machine learning algorithms and we must actively exploit it. We argue that unsupervised machine learning algorithms are greatly handicapped when trying to detect patterns. In contrast to supervised machine learning methods, they

[1] In fact, in the eye of the linguist, texts show a much richer structure than databases because texts are mostly self-explanatory, while databases typically aren't.

cannot even fall back to guidance coming from the training examples when exploring the vast extents of words.

Therefore, we have investigated new methods that apply unsupervised machine learning algorithms and take advantage of linguistic and conceptual background knowledge. In this paper, we focus on the description and application of conceptual background knowledge given by ontologies.

The gist of our approaches can be described as combining shallow information extracting methods in order to map (some) words to their conceptual descriptions, to use the ontology for abstracting text representations to various higher levels of granularity and then to apply conventional machine learning techniques. Finally, the ontologies are also used for communicating and presenting the results to the users.

Organization. The organization of the paper is as follows. First, we provide an introduction of our overall conceptual architecture. We introduce the main components and their relationships. Furthermore, we give a short description of the formal structures we use for defining the ontologies and instances and their interrelationships with the lexicon. Section 1.3 gives an rough overview of the preprocessing component used in all subsequently described proposals. We continue with three different proposals for combining machine learning techniques with ontologies (Section 1.4, viz. document clustering, clustering of content information and discovery of new conceptual relations between concepts. Section 1.5 explains why ontologies provide suitable means for supporting postprocessing and result presentation and gives several examples. Section 1.6 concludes with a summary and provides an outlook to future research challenges.

1.2 OSEM - A Conceptual Architecture for Ontology-based Text Mining

This section introduces OSEM, a conceptual architecture for using background knowledge in the form of ontologies within text mining. The idea behind OSEM is that a domain- and application-specific ontology acts as a backbone for all phases necessary when applying and using text mining in real-world applications. In general we distinguish the following main phases:

- **Preprocessing and Import & Background Knowledge Provisioning** by User: Includes syntactic preprocessing (shallow linguistic processing including tokenizer, morphology, POS-tagger, etc.) as well as semantic preprocessing (viz. assigning concepts to words, defining conceptual relationships between words). This phase is described in detail in section 1.3.
- **Mining**: Mining techniques are applied on properly preprocessed data. In general we distinguish between mining on the document level or mining

on the object (instance) level. Both approaches take advantage of the ontology as structuring background knowledge. This phase is described in detail in section 1.4.

- **Postprocessing & Refinement**: Based on the structural backbone of the ontology, results of the text mining algorithm are preprocessed and refined. E.g. the ontology supports in pruning and focusing on specific results of the text mining algorithm. This phase is described in detail in section 1.5.
- **Presentation**: The postprocessing and refinement phase strongly relates with presentation. Results are presented to the user in an appropriate way, thus, using the ontology as a scaffold for the presentation interfaces.

Figure 1.1 depicts the overall architecture. We consider as input to the overall process textual documents, structured data and existing ontologies.

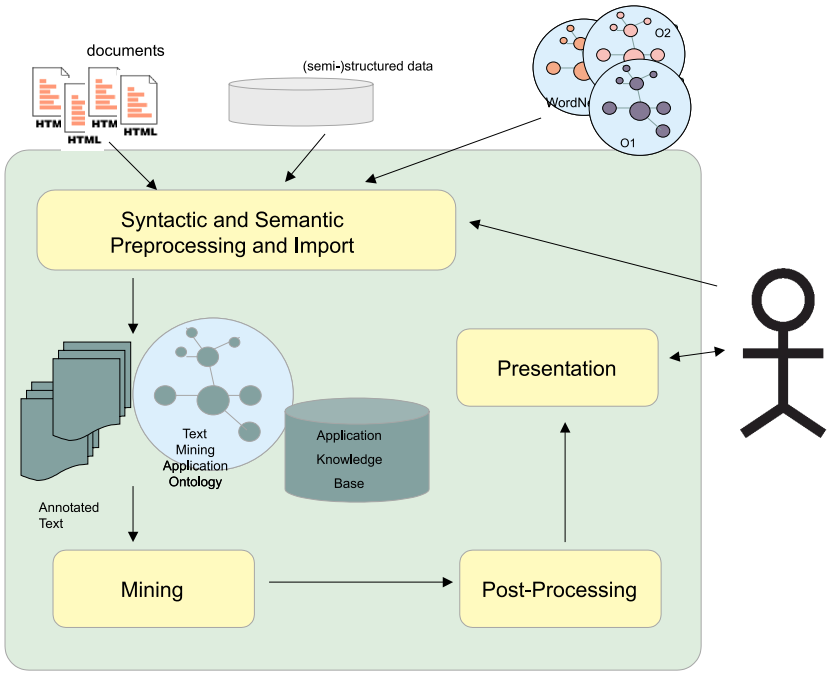

Figure1.1. OSEM Conceptual Architecture

Instantiations of this architecture are described in section 1.4 focusing on three unsupervised text mining techniques:

- Document clustering
- Instance clustering
- Discovery of conceptual relations

As ontologies play a central in OSEM, we provide in the following a definition of what exactly an ontology is constituted of. Furthermore, we describe how instantiations of ontologies look like and give an example for the relationship between ontologies, instances and textual documents.

Ontologies. It has been widely accepted that ontologies and metadata are the core elements for the Semantic Web. In the following we introduce a formal model of our notion of ontologies and metadata, where a specific focus is set on the interaction of ontology and associated metadata with natural language. To this extend, we have developed a layered architecture. We here only present the part of our overall ontology and instance model that is actually used by the text mining module.

Definition 1 (Ontology Layer). An ontology structure is a 6-tupel $\mathcal{O} := \{\mathcal{C}, \mathcal{P}, \mathcal{A}, \mathcal{H}^{\mathcal{C},\mathcal{P}}, prop, att\}$, consisting of two disjoint sets $\mathcal{C}$ and $\mathcal{P}$ whose elements are called concept and relation identifiers (URIs) , respectively, a **concept hierarchy** $\mathcal{H}^{\mathcal{C}}$: $\mathcal{H}^{\mathcal{C}}$ is a directed relation $\mathcal{H}^{\mathcal{C}} \subseteq \mathcal{C} \times \mathcal{C}$ which is also called taxonomy. $\mathcal{H}^{\mathcal{C}}(C_1, C_2)$ means that C_1 is a sub-concept of C_2, a **function** $prop : \mathcal{P} \rightarrow \mathcal{C} \times \mathcal{C}$, that relates conceptes non-taxonomically (The function dom: $\mathcal{P} \rightarrow \mathcal{C}$ with $\text{dom}(P) := \Pi_1(rel(P))$ gives the domain of P, and range: $\mathcal{P} \rightarrow \mathcal{C}$ with $\text{range}(P) := \Pi_2(rel(P)$ give its range. For $prop(P) = (C_1, C_2)$ one may also write $P(C_1, C_2)$). The **relation hierarchy** $\mathcal{H}^{\mathcal{P}}$: $\mathcal{H}^{\mathcal{P}}$ is defined analogously to the concept hierarchy. Thus, a directed relation $\mathcal{H}^{\mathcal{P}} \subseteq \mathcal{P} \times \mathcal{P}$ exists, where $\mathcal{H}^{\mathcal{P}}(P_1, P_2)$ means that P_1 is a sub-relation of P_2. The **function** $att : \mathcal{A} \rightarrow \mathcal{C}$ relates concepts with literal values (this means $\text{range}(A) := \text{STRING}$)

As the text mining process typically operates on natural language documents, the core ontology layer presented above is augmented with a lexical layer that facilitates the linking of textual documents to ontological entities (that is concepts and relations).

Definition 2 (Lexical Layer for the Ontology). A lexicon for the core ontology structure $\mathcal{O}$ is a 6-tupel $\mathcal{L} := \{\mathcal{L}^{\mathcal{C}}, \mathcal{L}^{\mathcal{P}}, \mathcal{L}^{\mathcal{A}}, \mathcal{F}, \mathcal{G}, \mathcal{J}\}$ consisting of three sets $\mathcal{L}^{\mathcal{C}}$, $\mathcal{L}^{\mathcal{P}}$ and $\mathcal{L}^{\mathcal{A}}$, whose elements are called **lexical entries** for concepts, relations and attributes, respectively, and three relations $\mathcal{F} \subseteq \mathcal{L}^{\mathcal{C}} \times \mathcal{C}$, $\mathcal{G} \subseteq \mathcal{L}^{\mathcal{P}} \times \mathcal{P}$ and $\mathcal{J} \subseteq \mathcal{L}^{\mathcal{A}} \times \mathcal{A}$ called **references** for concepts, relations and attributes, respectively. Based on $\mathcal{F}$, let for $L \in \mathcal{L}^{\mathcal{C}}$, $\mathcal{F}(L) = \{C \in \mathcal{C} | (L, C) \in \mathcal{F}\}$ and for $\mathcal{F}^{-1}(C) = \{L \in \mathcal{L}^{\mathcal{C}} | (L, C) \in \mathcal{F}\}$. $\mathcal{G}$, $\mathcal{G}^{-1}$, $\mathcal{J}$ and $\mathcal{J}^{-1}$ are defined analogously.

The definition allows n:m-relations between lexical entries and ontological entities, that is a lexical entry may refer to several concepts or relations and one concept or relation may be referenced by several lexical entries.

Definition 3 (Instance Layer). A metadata structure is a 6-tupel $\mathcal{MD} := \{\mathcal{O}, \mathcal{I}, \mathcal{L}, inst, instr, instl\}$, that consists of an ontology $\mathcal{O}$, a set $\mathcal{I}$ whose elements are called instance identifiers (correspondingly C, P and I are disjoint), a set of literal values L, a function $inst : \mathcal{C} \rightarrow 2^{\mathcal{I}}$ called **concept instantiation** (For $inst(C) = I$ one may also write $C(I)$), and a function $instr : \mathcal{P} \rightarrow 2^{\mathcal{I} \times \mathcal{I}}$ called **relation instantiation** (For $inst(P) = \{I_1, I_2\}$ one may also write $P(I_1, I_2)$). The **attribute instantiation** is described via the function $instl : \mathcal{P} \rightarrow 2^{\mathcal{I} \times \mathcal{L}}$ relates instances with literal values.

Again, we also define a lexical layer for instances.

Definition 4 (Lexicon for the instance structure). A lexicon for the instance structure $\mathcal{KB} := \{\mathcal{O}, \mathcal{I}, inst, instr\}$ is a tupel $\mathcal{L}^{\mathcal{MD}} := (\mathcal{L}^{\mathcal{I}}, \mathcal{J})$ consisting of a set $\mathcal{L}^{\mathcal{I}}$ whose elements are called **lexical entries** for instances, respectively, and a relation $\mathcal{J} \subseteq \mathcal{L}^{\mathcal{I}} \times \mathcal{I}$ **reference** for instances, respectively. Based on $\mathcal{J}$, let for $L \in \mathcal{L}^{\mathcal{I}}$, $\mathcal{J}(L) = \{I \in \mathcal{I} | (L, I) \in \mathcal{J}\}$ and for $\mathcal{J}^{-1}(I) = \{L \in \mathcal{L}^{\mathcal{I}} | (L, I) \in \mathcal{J}\}$.

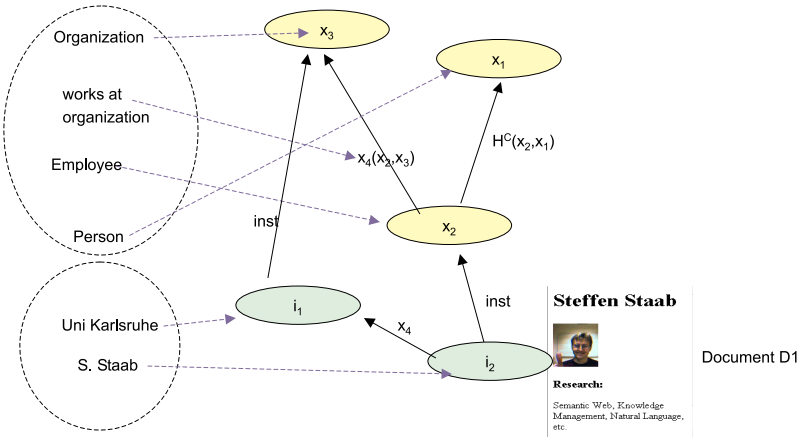

Figure1.2. Example of an instantiated ontology and instance structure

An Example. Let us consider a short example of an instantiated ontology and instance structure as depicted in figure 1.2. Here $\mathcal{C} := \{x_1, x_2, x_3\}$, $\mathcal{R} := \{x_4\}$, and the relation $x_4(x_2, x_3)$ with its domain/range restrictions are defined. The lexical layer is given by $\mathcal{L}^{\mathcal{C}} = \{$"Organization", "Employee", "Person"$\}$ and $\mathcal{L}^{\mathcal{R}} = \{$"works at organization"$\}$. The function $\mathcal{F}$ and $\mathcal{G}$ map the lexical entries to the concepts and relations of the ontology. $\mathcal{F}$ is applied as follows: $\mathcal{F}($"Organization"$) = x_3$, $\mathcal{F}($"Employee"$) = x_2$, $\mathcal{F}($"Person"$) = x_1$ and $\mathcal{G}($"works at organization"$) = x_4$. Based on this ontology, the following instances may be defined: Assume $\mathcal{I} := \{i_1, i_2\}$. $inst$ is applied as following: $inst(i_1) =$

$x_3, inst(i_2) = x_2$. The two instances are related by $x_4(i_2, i_1)$. Similarly to the lexical entries of concepts and relations the lexical entries of instances may have values, e.g. in this example $\mathcal{L}^{\mathcal{I}} := \{$"Uni Karlsruhe", "S. Staab"$\}$. $\mathcal{J}$ is applied as follows: $\mathcal{J}$("Uni Karlsruhe") $= i_1$ and $\mathcal{J}$("S.Staab") $= i_2$.

1.3 Preprocessing Towards a Conceptual Representation

In order to be able to exploit conceptual background knowledge, the preprocessing step requires a conceptual representation of the input texts. For this purpose, we employ a common shallow preprocessing of input texts that maps texts into semantic structures.

1.3.1 Shallow syntactic preprocessing

The mapping of terms to concepts in our approach relies on some modules from third parties, e.g. SMES (Saarbrcken Message Extraction System), a shallow text processor for German (cf. [11]).[2] SMES components we exploit comprise a *tokenizer* based on regular expressions and a *lexical analysis* component including a *word* and a so-called *domain lexicon* (the domain specific part of the lexicon partially defines $\mathcal{F}$).

The tokenizer scans the text in order to identify boundaries of words and complex expressions like "$20.00" or "United States of America", and to expand abbreviations. The word lexicon contains more than 120,000 stem entries. Lexical analysis uses the word lexicon, *(i)*, to perform morphological analysis of terms, i. e. the identification of the canonical common stem of a set of related word forms and the analysis of compounds and, *(ii)*, to recognize named entities. Thus, $\mathcal{L}$ as described in Definition 1 is a set defined by the tokenizer, the word lexicon and the analysis procedures of the lexical analysis component. The domain lexicon contains the mappings from word stems to concepts, i.e. together with the other modules it represents the function $\mathcal{F}$ as defined in Definition 1. By this way, e.g., the expression "Hotel Schwarzer Adler" is associated with the concept HOTEL. During the mapping process we do not resolve ambiguities of terms. This means, if we find several concepts with the same lexical entry we map the term to all related concepts.

1.3.2 Concept Vector Representation

Based on the syntactic input, one subsequent text mining variant represents each document as a vector of concept instantiations. Each entry of each vector specifies the frequency that a concept occurs in the document including the frequency that subconcepts occur.

[2] We also use a simple full form lexicon of our own, which we have derived from WordNet, and GATE [10].

1.3.3 Instance Representation

Frequently, atomic documents do not constitut the right level of granularity to base the text mining algorithm on. Therefore, as an alternative, we also directly exploit the appeareance of instances and their semantic relationships.

In order to derive semantic relationships between instances found in the document we use two strategies:

- Either a finite state machine has the semantic relationship hard-wired into very specific linguistic constructs (e.g. useful for processing of dictionaries).
- Or the establishment of a general syntactic relation triggers the search for a corresponding semantic relation. The background knowledge is then used to check the general availability of such a semantic relationship (cf. [12,?]).

1.3.4 A Glimpse onto KAON

Ontologies as well as Vector and instance representations may be stored in RDF and, hence, are accessible through our KAON framework [2] or directly by SQL queries to a proprietary database. The core idea of the common framework is that conceptual structures of different resources (such as different ontology resources) are integrated into a single framework and, thus, easily re-usable for different text mining algorithms. In this sense, we do not explicitly distinguish between data and text mining mechanisms.

1.4 Mining Component in OSEM

OSEM focuses on the application of unsupervised text mining techniques to the two different data representation layers introduced earlier. In the following section, we first describe a document clustering approach that is based on the usage of a simple, core ontology for generating alternative representations of the given document set such that from the various representations multiple clustering result may be derived by the standard K-Means algorithm. Second, we present an instance clustering approach that takes metadata statements as higher level input for clustering. Third, we present an association rule approach working on the conceptual level using the an ontology at various stage within the mining process.

1.4.1 Document Clustering

A classical unsupervised text mining task is document clustering. With the abundance of text documents available through the Web or corporate document management systems, the dynamic partitioning of document sets into

previously unseen categories ranks high on the priority list for many applications, like business intelligence systems. However, current text clustering approaches tend to neglect several major aspects that greatly limit their practical applicability.

First, text document clustering is mostly seen as an *objective* method, which delivers one clearly defined result, which needs to be "optimal" in some way. This, however, runs contrary to the fact that different people have quite different needs with regard to clustering of texts because they may view the same documents from completely different perspectives (e.g., a business view vs. a technical view). Thus, what is needed are document clustering methods that provide multiple *subjective* perspectives onto the same document set.

Second, text document clustering typically is a machine learning task taking place in a *high-dimensional space* of word vectors, where each word, i.e. each entry of a vector, is seen as a potential attribute for a text. Empirical and mathematical analysis, however, has shown that — in addition to computational ineffiencies — clustering in high-dimensional spaces is very difficult because every data point tends to have the same distance from all other data points (cf. [3]).

Third, text document clustering *per se* is often rather useless, unless it is combined with an *explanation* of why particular texts were categorized into a particular cluster. I.e. one output desired from clustering in practical settings is the explanation of why a particular cluster result was produced besides of the result itself. A common method for producing explanations is the learning of rules based on the cluster results. Again, however, this approach suffers from the high number of features chosen for computing clusters.

Though there are of course different approaches for clustering, simple ones like K-Means or sophisticated ones (like [4]), based on the consideration just mentioned we found that virtually all algorithms working on large feature vectors will eventually face the same principal problems regarding *high-dimensional space* without really approaching the matters of *subjectivity* and *explainability*. Therefore, our objective has been the consideration of different views of the data, i.e. different representations[3] of the same set of text documents, from which alternative clustering results may be derived.

The principal idea of our approach, COSA (Concept Selection and Aggregation), is based on the usage of a simple, core ontology for generating alternative representations of the given document set such that from the various representations multiple clustering result may be derived by the standard K-Means algorithm. The single representations are construed by aggregating the original word vector representation in various ways. More precisely, we have compiled a heterarchy of concepts[4]. The heterarchy is navigated top-

[3] Motivated by the database point of view, we also call derived text representations "aggregations".

[4] A heterarchy of concepts is a kind of "taxonomy" where each term may have multiple parents and — of course — multiple children.

down by COSA in order to select document features (i.e. concepts) for an aggregated vector representation. Thereby, COSA considers that features are neither too frequent (i.e. COSA would split them into their subconcepts) nor too infrequent (i.e. COSA would abandon them in favor of more frequent ones) to be meaningful for clustering.

Thus, a set of clustering results is produced without interaction by a human user of the system. The user may then decide to prefer the one over the other clustering result based on the actual concepts used for clustering as well as on standard quality measures (such as the silhouette measure [7]).

Let us work through a detailed example to show you the problems and to give you an intuition for the proposed solution. In Table 1.1 you find a sample of (abbreviated) concept vectors representing the web pages. In Figure 1.3 one may recognize the corresponding concepts in an excerpt of the ontology. Our simplifying example shows the principal problem of vector representations of documents: The tendency that spurious appearance of concepts (or terms) rather strongly affects the clustering of documents. The reader may bear in mind that our simplification is so extensive that practically it does not appear in such tiny settings, but only when one works with large representations and large document sets. In our simplifying example the appearance of concepts HOTEL, PREMIERE, and CONCERT is spread so evenly across the different documents that all document pairs exhibit (more or less) the same similarity. Corresponding squared Euclidian distances for the example document pairs (1,2), (2,3), (1,3) leads to values of 2, 2, and 2, respectively, and, hence, to no clustering structure at all.

Document #	1 ("Musical")	2 ("Sport Hotel")	3 ("Conference Hotel")
HOTEL	0	1	1
PREMIERE	2	2	1
CONCERT	1	0	1

Table1.1. Concept vector representations

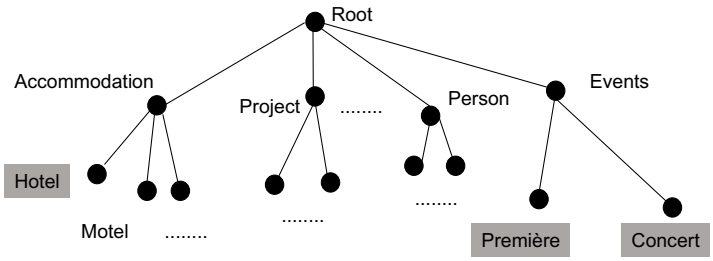

Figure1.3. A sample ontology

When one reduces the size of the representation of our documents, e.g. by projecting into an subspace, one focuses on particular concepts and one may focus on the significant differences that documents exhibit with regard to these concepts. For instance, when we project into a document vector representation that only considers the two dimensions HOTEL and PREMIERE, we will find that document pairs (1,2), (2,3), (1,3) have squared Euclidean distances of 1, 1, and 2. Thus, axis-parallel projections like in this example may improve the clustering situation. In addition, we may exploit the ontology. For instance, we select features according to the taxonomy, choosing, e.g., EVENTS instead of its subconcepts PREMIERE and CONCERT to built our aggregation. Then, the entries for PREMIERE and CONCERT are added into one vector entry resulting in squared Euclidean distances between pairs (1,2), (2,3), (1,3) of 2, 0, and 2, respectively. Thus, documents 2 and 3 can be clustered together, while document 1 falls into a different cluster.

The algorithm *GenerateConceptViews* described in [6] acts as a preprocessing step for clustering. *GenerateConceptViews* chooses a set of interpretable and ontology-based aggregations leading to modified text representations. Conventional clustering algorithms like K-Means may work on these modified representations producing improved clustering results. Because the size of the vector representation is reduced, it becomes easier for the user to track the decisions made by the clustering algorithms. Because there are a variety of aggregations, the user may choose between alternative clustering results. For instance, there are aggregations such that event pages are clustered together and the rest is set aside or aggregations such that web pages about PREMIERES are clustered together and the rest is left in another cluster. The choice of concepts from the taxonomy thus determines the output of the clustering result and the user may use a view like Figure 1.3 in order to select and understand differences between clustering results.

1.4.2 Instance Clustering

In this subsection we present an instance clustering approach that takes instances and instance relations extracted from documents as higher level input for clustering objects. This approach pursues the idea introduced in [5], where information extraction is considered as a preprocessing step before applying text mining. This approach may be more suited for documents containing many links between them or documents that adhere to a fixed structure that is not exploited by the document clustering described algorithm above. In the following we will show exemplary how such a instance structure could be used for the mining step in OSEM.

Measuring Similarity on Ontology-based Instsances As mentioned earlier, clustering of objects requires some kind of similarity measure that is computed between the objects. In our specific case the objects are described via ontology-based instance that serve as input for measuring similarities. Our

approach is based on similarities using the instantiated ontology structure and the instantiated instance structure as introduced earlier in parallel.

Definition 5 (Instance Similarity).

$$sim : (\mathcal{I}, \mathcal{I}) \rightarrow [0, 1]$$

Within the overall similarity computation approach, we distinguish the following three dimensions:

- **Taxonomy similarity:** Computes the similarity between two instance instances on the basis of their corresponding concepts and their position in $\mathcal{H}^{\mathcal{C}}$.
- **Relation similarity:** Compute the similarity between two instances on the basis of their relations to other objects.
- **Attribute similarity:** Computes the similarity between two instances on the basis of their attributes and attribute values.

Taxonomy Similarity. The taxonomic similarity computed between instance instances relies on the concepts with their position in the concept taxonomy $\mathcal{H}^{\mathcal{C}}$. The so-called upwards cotopy (SC) is the underlying measure to compute the semantic distance in a concept hierarchy.

Definition 6 (Upwards Cotopy (UC)).

$$\mathrm{UC}(C_i, \mathcal{H}^{\mathcal{C}}) := \{C_j \in \mathcal{C} | \mathcal{H}^{\mathcal{C}}(C_i, C_j) \vee C_j = C_i\}.$$

The semantic characteristics of $\mathcal{H}^{\mathcal{C}}$ are utilized: The attention is restricted to super-concepts of a given concept C_i and the reflexive relationship of C_i to itself. Based on the definition of the upwards cotopy (UC) the concept match (CM) is then defined:

Definition 7 (Concept Match).

$$\mathrm{CM}(C_1, C_2 := \frac{|(\mathrm{UC}(C_1, \mathcal{H}^{\mathcal{C}}) \cap (\mathrm{UC}(C_2, \mathcal{H}^{\mathcal{C}}))|}{|(\mathrm{UC}(C_1, \mathcal{H}^{\mathcal{C}})) \cup (\mathrm{UC}(C_2, \mathcal{H}^{\mathcal{C}})|}.$$

Example A small example is given for computing CM based on a given concept hierarchy $\mathcal{H}^{\mathcal{C}}$. Figure 1.4 depicts the example scenario graphically. The upwards cotopy $\mathrm{UC}(\mathrm{CHRISTIANISM}, \mathcal{H}^{\mathcal{C}})$ is given by $(\mathrm{UC}((\{\mathrm{CHISTIANISM}\}), \mathcal{H}^{\mathcal{C}})) = \{\mathrm{CHRISTIANISM}, \mathrm{RELIGION}, \mathrm{ROOT}\}$. The upwards cotopy $\mathrm{UC}((\{\mathrm{MUSLIM}\}), \mathcal{H}^{\mathcal{C}})$ is computed by $\mathrm{UC}((\{\mathrm{MUSLIM}\}), \mathcal{H}^{\mathcal{C}}) = \{\mathrm{MUSLIM}, \mathrm{RELIGION}, \mathrm{ROOT}\}$.

Based on the upwards cotopy one can compute the concept match CM between two given specific concepts. The concept match CM between MUSLIM and CHRISTIANISM is given as $\frac{1}{2}$.

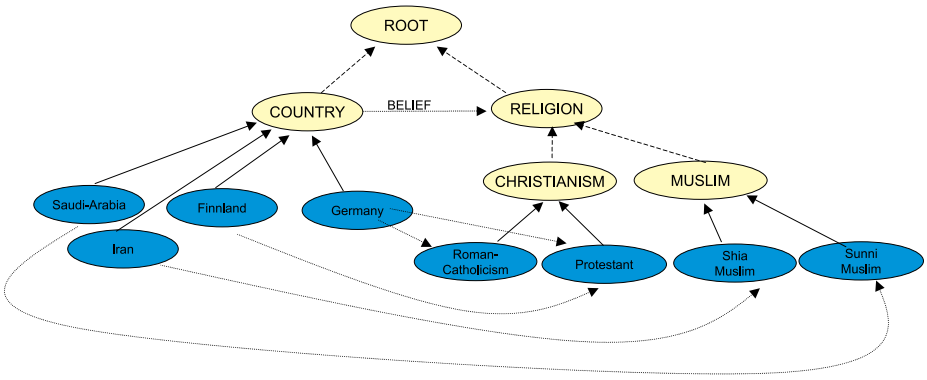

Figure1.4. Example for computing similarities

Definition 8 (Taxonomy Similarity).

$$\mathrm{TS}(I_1, I_2) = \begin{cases} 1 & \text{if } I_1 = I_2 \\ \frac{CM(C(I_1),C(I_2))}{2} & \text{otherwise} \end{cases}$$

Thus, the taxonomy similarity between SHIA MUSLIM to PROTESTANT results in $\frac{1}{4}$.

Relation Similarity. It is pretty save to assume that if two instances have the same relation to a third instance, they are more likely similar than two instances that have relations to totally different instances. We incorporated this observation into our algorithm by changing the similarity of two instances depending on the similarity of the instances they have relations to. The similarity of the referred instances is once again calculated using taxonomic similarity. For example, assuming we are given two concepts COUNTRY and RELIGION and a relation BELIEVE(COUNTRY, RELIGION). The algorithm will infer that specific countries believing in catholizism and protestantism are more similar than either of these two compared to hinduism because more countries have both catholics and protestants than a combination of either of these and hindis.

After this overview, let's get to the nitty gritty of really defining the similarity on relations. We are comparing two instances I_1 and I_2, $I_1, I_2 \in \mathcal{I}$. From the definition of the ontology we know that there is a set of relations P_1 that allow instance I_1 either as domain, as range or both (Likewise there is a set P_2 for I_2). Only the intersection $P_{\mathrm{CO}} = P_1 \cap P_2$ will be of interest for relation similarity because differences between P_1 and P_2 are determined by the taxonomic relations, which are already taken into account by the taxonomic similarity.

The set P_{CO} of relations is differentiated between relations allowing I_1 and I_2 as range - $P_{\mathrm{CO-I}}$, and those that allow I_1 and I_2 as domain - $P_{\mathrm{CO-O}}$.

Definition 9 (Incoming $P_{\mathrm{CO-I}}$ and Outgoing $P_{\mathrm{CO-O}}$ relations).

for an ontology $\mathcal{O} := \{\mathcal{C}, \mathcal{P}, \mathcal{A}, \mathcal{H}^{\mathcal{C},\mathcal{P}}, prop, att\}$ and Instances I_1 and I_2

$$H^{trans} := \left\{(a,b) : (\exists a_1 ... a_n \in C : H^C(a,a_1) ... H^C(a_n,b))\right\}$$

$$P_{\text{co–Ii}}(I_i) := \left\{R : R \in \mathcal{P} \wedge ((C(I_{\text{i}}), \text{range}(R)) \in H^{trans})\right\}$$

$$P_{\text{co–Oi}}(I_i) := \left\{R : R \in \mathcal{P} \wedge ((C(I_{\text{i}}), \text{domain}(R)) \in H^{trans})\right\}$$

$$P_{\text{co–I}}(I_i, I_j) := P_{\text{co–Ii}}(I_i) \cap P_{\text{co–I}}(I_j)$$

$$P_{\text{co–O}}(I_i, I_j) := P_{\text{co–Oi}}(I_i) \cap P_{\text{co–O}}(I_j)$$

In the following we will only look at $P_{\text{co–O}}$, but everything applies to $P_{\text{co–I}}$ as well. Before we continue we have to note an interesting aspect: For a given ontology with a relation P_x there is a minimum similarity greater than zero between any two instances that are source or target of an instance relation - $\text{MinSim}_{s(P_x)}$ and $\text{MinSim}_{t(P_x)}$[5]. Ignoring this will increase the similarity of two instances with relations to the most different instances when compared to two instances that simply don't define this relation. This is especially troublesome when dealing with missing values.

For each relation $P_n \in P_{\text{co–O}}$ and each instance I_i there exists a set of instance relations $P_n(I_i, I_x)$. We will call the set of instances I_x the associated instances A_s.

Definition 10 (Associated Instances).

$$A_s(P, I) := \{I_x : I_x \in \mathcal{I} \wedge P(I, I_x)\}$$

The task of comparing the instances I_1 and I_2 with respect to relation P_n boils down to comparing $A_s(P_n, I_1)$ with $A_s(P_n, I_2)$. This is done as follows:

Definition 11 (Similarity for one relation).

$$\text{OR}(I_1, I_2, P) = \begin{cases} \text{MinSim}_{t(P)} & \text{if } A_s(P, I_1) = \emptyset \vee A_s(P, I_2) = \emptyset \\ \left(\frac{\sum_{(a \in A_s(P, I_1))} \max\{sim(a,b) | b \in A_s(P, I_2)\}}{|A_s(P, I_1)|}\right) & \text{if } |A_s(P, I_1)| \geq |A_s(P, I_2)| \\ \left(\frac{\sum_{(a \in A_s(P, I_2))} \max\{sim(a,b) | b \in A_s(P, I_1)\}}{|A_s(P, I_2)|}\right) & \text{otherwise} \end{cases}$$

Finally, the results for all $P_n \in P_{\text{co–O}}$ and $P_n \in P_{\text{co–I}}$ are combined by calculating their arithmetic mean.

Definition 12 (Relational Similarity).

$$RS(I_1, I_2) := \frac{\sum_{p \in P_{\text{co–I}}} \text{OR}(I_1, I_2, p) + \sum_{p \in P_{\text{co–O}}} \text{OR}(I_1, I_2, p)}{|P_{\text{co–I}}| + |P_{\text{co–O}}|}$$

The last problem that remains is the recursive nature of process of calculating similarities that may lead to infinite cycles, but it can be easily solved by imposing a maximum depth for the recursion. After reaching this maximum depth the arithmetic mean of taxonomic and attribute similarity is returned.

[5] Range and domain specify a concept and any two instances of this concept or one of its sub-concepts will have a taxonomic similarity bigger than zero

Example. Figure 3 gives an ontology and a set of instance instances that we can use for an example of relational similarity. Assuming we compare FINNLAND and GERMANY, we see that the set of common relations only contains the BELIEF relation. As the next step we compare the sets of instances associated with GERMANY and FINNLAND through the belief relation - that's {ROMAN-CATHOLICISM, PROTESTANT} for GERMANY and PROTESTANT for FINNLAND. The similarity function for PROTESTANT compared with PROTESTANT returns one because they are equal, but the similarity of PROTESTANT compared with ROMAN-CATHOLICSM once again depends on their relational similarity.

If we we assume the the maximum depth of recursion is set to one, the relational similarity between ROMAN-CATHOLICSM and PROTESTANT is 0.5[6]. So finally the relational similarity between FINNLAND and GERMANY in this example is 0.75.

Attribute Similarity. Attribute similarity focuses on similar attribute values to infer the similarity between two instances. As attributes are very similar to relations, most of what is said for relations also applies here.

Definition 13 (Compared attributes for two instances).

$$P_A i(I_i) := \{A : A \in \mathcal{A}\}$$

$$P_A(I_i, I_j) := P_A i(I_i) \cap P_A i(I_j)$$

Definition 14 (Attribute values).

$$A_s(A, I_i) := \{L_x : L_x \in \mathcal{L} \wedge A(I_i, L_x)\}$$

Only the members of the sets A_s defined earlier are not instances but literals and we need a new similarity method to compare literals. Because attributes can be names, date of birth, population of a country, income etc. comparing them in a senseful way is very difficult. We decided to try to parse the attribute values as a known data type (so far only date or number)[7] and to do the comparison on the parsed values. If it's not possible to parse all values of a specific attribute, we ignore this attribute. But even if numbers

[6] The set of associated instances for PROTESTANT contains FINNLAND and GERMANY, the set for ROMAN-CATHOLICISM just GERMANY.

[7] For simple string data types one may use a notion of string similarity: The *edit distance* formulated by Levenshtein [8] is a well-established method for weighting the difference between two strings. It measures the minimum number of token insertions, deletions, and substitutions required to transform one string into another using a dynamic programming algorithm. For example, the edit distance, ed, between the two lexical entries "TopHotel" and "Top_Hotel" equals 1, ed("TopHotel", "Top_Hotel") = 1, because one insertion operation changes the string "TopHotel" into "Top_Hotel".

are compared, translating a numeric difference to a similarity value $[0, 1]$ can be difficult. For example comparing the attribute population of a country a difference of 4 should yield a similarity value very close to 1, but comparing the attribute "average number of children per woman" the same numeric difference value should result in a similarity value close to 0. To take this into account, we first find the maximum difference between values of this attribute and then calculate the the similarity as $1 -$ (Difference/ max Difference).

Definition 15 (Literal similarity).

$$slsim(\mathcal{A}, \mathcal{A}) \rightarrow [0, 1]$$

$$mlsim := \max \{slsim(A_1, A_2) : A_1 \in \mathcal{A} \wedge A_2 \in \mathcal{A}\}$$

$$lsim(A_i, A_j, A) := \frac{slsim(A_i, A_j)}{mlsim(A)}$$

And last but not least, unlike for relations the minimal similarity when comparing attributes is always zero.

Definition 16 (Similarity for one attribute).

$$\mathrm{OA}(I_1, I_2, A) := \begin{cases} 0 & \text{if } A_s(A, I_1) = \emptyset \vee A_s(A, I_2) = \emptyset \\ \left(\frac{\sum_{(a \in A_s(A, I_1))} \max\{lsim(a,b,A) | b \in A_s(A, I_2)\}}{|A_s(A, I_1)|} \right) & \text{if } |A_s(A, I_1)| \geq |A_s(A, I_2)| \\ \left(\frac{\sum_{(a \in A_s(A, I_2))} \max\{lsim(a,b,A) | b \in A_s(A, I_1)\}}{|A_s(A, I_2)|} \right) & \text{otherwise} \end{cases}$$

Definition 17 (Attribute Similarity).

$$AS(I_1, I_2) := \frac{\sum_{a \in P_{\mathrm{A}}(I_1, I_2)} \mathrm{OA}(I_1, I_2, a)}{|P_{\mathrm{A}(I_1, I_2)}|}$$

Combined Measure. The combined measure uses the three dimensions introduced above in a common measure. This done by calculating the weighted arithmetic mean of attribute, relation and semantic similarity.

Definition 18 (Similarity Measure).

$$sim(I_i, I_j) := \frac{t \times TS(I_i, I_j) + r \times RS(I_i, I_j) + a \times AS(I_i, I_j)}{t + r + a}$$

The weights may be adjusted according to the given data set the measures should be applied, e.g. within our empirical evaluation we used a weight of 2 for relation similarity, because most of the overall information of the ontology and the associated instance was contained in the relations.

The similarity measures introduced above allow to compute similarities between a set of instances. We consider this step as a specific form of preprocessing to generate a similarity matrix that may serve as input for a hierarchical clustering algorithm, e.g. as described in [9].

Empirical Evaluation We have empirically evaluated our approach for clustering ontology-based instances based on the different similarity measures and the clustering algorithm introduced above. We used the well-known CIA world fact book data set as input[8]. The data set is available in many different forms as MONDIAL databases[9]. Due to a lack of currently available ontology-based instance on the Web, we converted a subset of MONDIAL in RDF and modeled a corresponding RDF-Schema for the databases (on the basis of the ER model also provided by MONDIAL). It has to be noted that the MONDIAL database has a lot of missing and even wrong values. Our subset of the MONDIAL database contained the concepts COUNTRY, LANGUAGE, ETHNIC-GROUP, RELIGION and CONTINENT. Relations contained where

- SPEAK(COUNTRY,LANGUAGE),
- BELONG(COUNTRY, ETHNIC-GROUP),
- BELIEVE(COUNTRY,RELIGION),
- BORDERS(COUNTRY,COUNTRY) and
- ENCOMPASSES(COUNTRY,CONTINENT).

We also converted the attributes infant mortality and population growth of the concept COUNTRY.

The task was now to calculate the hierarchical cluster structure for countries using the data set introduced above. As there is no pre-classification of countries, we decided to empirically evaluate the cluster against the country clusters we know and use in our daily live (like european countries, scandinavian countries, arabic countries etc). Sadly there is no further taxonomic information for the concepts RELIGION, ETHNIC–GROUP or LANGUAGE available within the data set. Thus, the taxonomic similarity measure could not be applied within this evaluation study. We first feed the clustering algorithm with a similarity matrix that has been generated using only relation similarity measures, than with a similarity matrix that has been generated using only attribute similarity measures and finally with a similarity matrix using a the combined relational and attribute similarity measure. For our experiments we used the already introduced bottom-up clustering algorithm with a single linkage computation strategy using cosine measure.

Using only relation similarity. Using only the relations of countries for measuring similarities we got clusters resembling many real world country clusters, like the european countries, the former soviet republics in the caucasus or such small cluster like {AUSTRIA, GERMANY}. A particular interesting example is the cluster of scandinavian countries depicted in Figure 1.5 because

[8] http://www.cia.gov/cia/publications/factbook/
[9] http://www.informatik.uni-freiburg.de/ may/Mondial/

our data nowhere contains a value like "scandinavian language" or a ethnic group "scandinavian".[10]

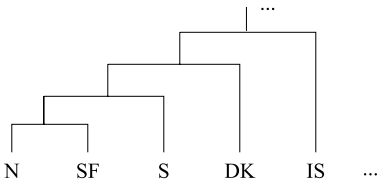

Figure1.5. Example Clustering Result – Scandinavian Countries

Figure 1.6 shows another interesting cluster of countries that we know as the Middle East[11]. The politically interested reader will immediately recognize that Israel is missing. This can be easily explained by observing that Israel, while geographically in the middle east is in terms of language, religion and ethnic group a very different country. More troublesome is that Oman is missing too and this can be only explained by turning to the data set used to calculate the similarities, where we see that Oman is missing many values, for example any relation to language or ethnic group.

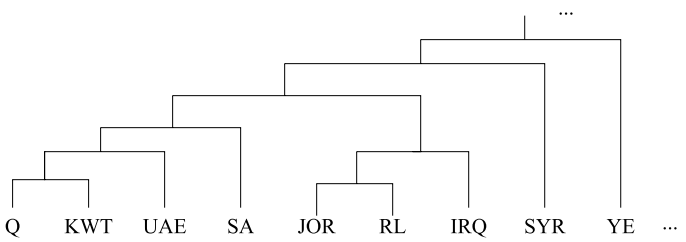

Figure1.6. Example Clustering Result – Middle East

Using only attribute similarity. When using only attributes of countries for measuring similarities we had to restrict the clustering to infant mortality and population growth. As infant mortality and population growth are good indicators for wealth of a country, we got cluster like industrialized countries or very poor countries.

[10] The meaning of the acronyms in the picture is: N:Norway, SF: Finnland, S: Sweden, DK: Denmark and IS:Island.

[11] The meaning of the acronyms used in the picture is: Q:Quatar, KWT: Kuwait, UAE: United Arab Emirates, SA: Saudi Arabia, JOR: Jordan, RL: Lebanon, IRQ: Iraq, SYR: Syria, YE, Yemen.

Combining relation and attribute similarity. At first surprisingly the clusters generated with the combination of attribute and relation similarity closely resemble the clusters generated only with relation similarity. But after checking the attribute values of the countries it actually increased our confidence in the algorithm, because countries that are geographically close together, and are similar in terms of ethnic group, religion and language are almost always also similar in terms of population growth and infant mortality. In the few cases where this was not the case the countries where rated far apart, for example Saudi Arabia and Iraq lost it's position in the core middle east cluster depicted because of their high infant mortality[12].

Summarization of results. Due to the lack of pre-classified countries and due to the subjectivity of clustering in general, we had to restrict our evaluation procedure to an empirical evaluation of the cluster we obtained against the country clusters we know and use in our daily live. It has been seen that using our attribute and relation similarity measures combined with a hierarchical clustering algorithm results in reasonable clusters of countries taking into account the very different aspects a country may be described and classified.

1.4.3 Association Rules

Association rules have been established in the area of data mining, thus, finding interesting association relationships among a large set of data items. Many industries become interested in mining association rules from their databases (e.g. for helping in many business decisions such as customer relationship management, cross-marketing and loss-leader analysis. A typical example of association rule mining is market basket analysis. This process analyzes customer buying habits by finding associations between the different items that customers place in their shopping baskets. The information discovered by association rules may help to develop marketing strategies, e.g. layout optimization in supermarkets (placing milk and bread within close proximity may further encourage the sale of these items together within single visits to the store). In [1] concrete examples for extracted associations between items are given. The examples are based supermarket products that are included in a set of transactions collected from customers' purchases. One of the classical association rule that has been extracted from these databases is that "diaper are purchased together with beer".

For the purpose of illustration, an example is provided to the reader. The example is based on actual experiments. A text corpus given by a WWW

[12] It may be surprising for such a rich country, but according to the CIA world fact book the infant mortality rate in Saudi Arabia (51 death per 1000 live born children) much closer resembles that of sanctioned Iraq (60) than that of much poorer countries like Syria (33) or Lebanon (28)

provider for tourist information has been processed[13]. The corpus describes actual objects referring to locations, accomodations, furnishings of accomodations, administrative information, or cultural events, such as given in the following example sentences.

(1) a. "Mecklenburg's" schönstes "Hotel" liegt in Rostock. ("Mecklenburg's" most beautiful "hotel" is located in Rostock.)
b. Ein besonderer Service für unsere Gäste ist der "Frisörsalon" in unserem "Hotel". (A "hairdresser" in our "hotel" is a special service for our guests.)
c. Das Hotel Mercure hat "Balkone" mit direktem "Strandzugang". (The hotel Mercure offers "balconies" with direct "access" to the beach.)
d. Alle "Zimmer" sind mit "TV", Telefon, Modem und Minibar ausgestattet. (All "rooms" have "TV", telephone, modem and minibar.)

Processing the example sentences (1a) and (1b) the dependency relations between the lexical entries are extracted (and some more). In sentences (1c) and (1d) the heuristic for prepositional phrase-attachment and the sentence heuristic relate pairs of lexical entries, respectively. Thus, four concept pairs – among many others – are derived with knowledge from the lexicon.

Table1.2. Examples for Linguistically Related Pairs of Concepts

L_1	$a_{i,1}$	L_2	$a_{i,2}$
"Mecklenburgs"	AREA	*hotel*	HOTEL
"hairdresser"	HAIRDRESSER	*hotel*	HOTEL
"balconies"	BALCONY	*access*	ACCESS
"room"	ROOM	*TV*	TELEVISION

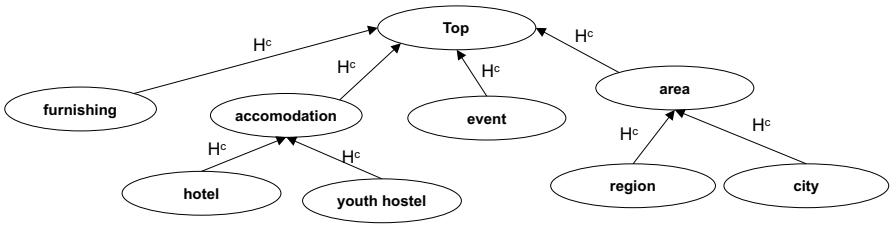

Figure1.7. An Example Concept Taxonomy as Background Knowledge for Non-Taxonomic Relation Extraction

The algorithm for learning generalized association rules uses the concept hierarchy, an excerpt of which is depicted in Figure 1.7, and the con-

[13] A detailed description of the text corpus and the overall application and evaluation study is provided in section **??**.

cept pairs from above (among many other concept pairs). In our actual experiments, it discovered a large number of interesting and important non-taxonomic conceptual relations. A few of them are listed in Table 1.3. Note that in this table we also list two conceptual pairs, viz. (AREA, HOTEL) and (ROOM, TELEVISION), that are not presented to the user, but that are pruned. The reason is that there are ancestral association rules, viz. (AREA, ACCOMODATION) and (ROOM,FURNISHING), respectively with higher confidence and support measures.

Table1.3. Examples of Discovered Non-Taxonomic Relations

Discovered relation	Confidence	Support
(AREA, ACCOMODATION)	0.38	0.04
~~(AREA, HOTEL)~~	~~0.1~~	~~0.03~~
(ROOM, FURNISHING)	0.39	0.03
~~(ROOM, TELEVISION)~~	~~0.29~~	~~0.02~~
(ACCOMODATION, ADDRESS)	0.34	0.05
(RESTAURANT, ACCOMODATION)	0.33	0.02

1.5 Postprocessing and Presentation

Even without background knowledge a major component of unsupervised Text Mining is the postprocessing and presentation component. The reasons essentially are that

- No algorithm can reliably predict what is novel, interesting and useful;
- The results are typically so complex that they cannot be represented in a short phrase or formula or a small picture;
- Even fairly understandable results must be made digestable for a more naive user who is typically not an expert in statistics or data mining, but has an application background.

Background knowledge *per se* does not diminish any of these three problems. However, ontologies add one (or several) additional dimensions that allow the user to explore the results in a way that corresponds to his way of thinking in the application domain. For instance, in the skill management scenario it is not very intuitive for the naive user to think in terms of distance to a hyperplane in some high-dimensional space, but it is quite easy for him to understand an explanation that says "these two clusters are the same on all attributes, but this group of people also has foreign language skills".

Techniques. Technically speaking, this sort of explanation is realized by some core means:[14]

- Navigation. Either navigation in the ontology may be used to select data mining results or data mining results may be selected in order to focus on some ontology parts.
- Exploiting hierarchical relationships of an ontology, e.g. the taxonomic relation $\mathcal{H}^{\mathcal{C}}$ or some part-whole hierarchy (e.g., car-body being part of a car). Different results may be sorted according to the branching of attribute values into different parts of such a hierarchy. In addition, several dimensions may be constructed from such attribute values resulting in a lattice of different results.
- Focusing on the parts of the ontology that best explain a particular result. For instance, when clustering concept vectors one finds that not all vector entries are of similar value for producing the final clustering results. Thus, the explanation can be focused to focal parts of the ontology. At the same time hierarchies in the ontology may be split or aggregated depending on which view is more promising to the user who explores the result.

While we have expanded only on a few mechanisms (e.g., navigation, zooming into particular concepts, selection of data mining results), there is plenty of work in visualizing ontologies and corresponding database entries that may be reused for visualizing knowledgeable text mining results.[15]

1.6 Conclusion

Text Mining is about discovering novel, interesting and useful patterns from textual data. In this paper we have discussed several means that introduce background knowledge into unsupervised text mining in order to improve the novelty, the interestingness or the usefulness of the detected patterns. Germane to the different proposals is that they strive for higher abstractions that carry more explanatory power and more possibilities for exploring the input texts than is achievable by unknowledgeable means.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining Associations between Sets of Items in Massive Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 688–692. ACM Press, 1993.
2. Alexander Maedche, Steffen Staab, Rudi Studer, York Sure, and Raphael Volz. Seal - tying up information integration and web site management by ontologies. In *IEEE Data Engineering Bulletin*, volume 25, March 2002.

[14] The means, of course, depend to some extent on the data mining techniques exploited.

[15] Cf. http://www.aidministrator.nl for some commercial tools.

3. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is 'nearest neighbor' meaningful. In *Proc. of ICDT-1999*, pages 217–235, 1999.
4. P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. of KDD-1998*, pages 9–15. AAAI Press, August 1998.
5. Ronen Feldman, Yonatan Aumann, Moshe Fresko, Orly Lipshtat, Binyamin Rosenfeld, and Yonatan Schler. Text mining via information extraction. In *Proceedings of PKDD-99, Prague, 1999.* Springer, 1999.
6. A. Hotho, A. Maedche, and S. Staab. Ontology-based text clustering. In *Proceedings of the IJCAI-2001 Workshop "Text Learning: Beyond Supervision", August, Seattle, USA*, 2001.
7. L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, New York, 1990.
8. I. V. Levenshtein. Binary Codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.
9. C.D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, Massachusetts, 1999.
10. D. Maynard, V. Tablan, H. Cunningham, C. Ursu, H. Saggion, K. Bontcheva, and Y. Wilks. Architectural elements of language engineering robustness. *Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data*, 2002.
11. G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. An information extraction core system for real world german text processing. In *In Proceedings of ANLP-97*, pages 208–215, Washington, USA, 1997.
12. S. Staab, C. Braun, A. Düsterhöft, A. Heuer, M. Klettke, S. Melzig, G. Neumann, B. Prager, J. Pretzel, H.-P. Schnurr, R. Studer, H. Uszkoreit, and B. Wrenger. GETESS — searching the web exploiting german texts. In *CIA'99 — Proceedings of the 3rd Workshop on Cooperative Information Agents. Upsala, Sweden, July 31-August 2, 1999*, LNCS 1652, pages 113–124, Berlin, 1999. Springer.