Domain Taxonomy Learning from Text: The Subsumption Method versus Hierarchical Clustering

Abstract

This paper proposes a framework to automatically construct taxonomies from a corpus of text documents. This framework first extracts terms from documents using a part-of-speech parser. These terms are then filtered using domain pertinence, domain consensus, lexical cohesion, and structural relevance. The remaining terms represent concepts in the taxonomy. These concepts are arranged in a hierarchy with either the extended subsumption method that accounts for concept ancestors in determining the parent of a concept or a hierarchical clustering algorithm that uses various text-based window and document scopes for concept co-occurrences. Our evaluation in the field of management and economics indicates that a trade-off between taxonomy quality and depth must be made when choosing one of these methods. The subsumption method is preferable for shallow taxonomies, whereas the hierarchical clustering algorithm is recommended for deep taxonomies.

Keywords: Ontologies, text mining, clustering, classification, and association rules

1. Introduction

In the past, data were stored physically, not digitally, and were often structured manually so that the desired information could be found easily. Today, data are often stored digitally and are usually unstructured, as in documents. Manually structuring documents is time consuming, which makes it interesting to investigate possibilities to automatically organize documents. This could be performed by automatically generating a concept taxonomy from a document corpus [1].

A taxonomy can be defined as a specific form of an ontology, which is a formal, explicit specification of a shared conceptualization [2], containing type-of relations. Ontologies are more complex than taxonomies. Not only do they consider type-of relations, but they also consider other relations, including part-of or domain-specific relations. A taxonomy provides users with insights into the type-of relations between concepts in a certain domain. These taxonomies are usually manually created, although researchers have proposed several techniques that support automatic taxonomy construction [3, 4]. These techniques could serve as a first step for ontology learning. Automatically constructed taxonomies generally have more errors than manually created taxonomies, but constructing a taxonomy manually might be time consuming, especially when a large taxonomy is desired. Researchers thus aim to automatically create taxonomies that approach the quality of manually created taxonomies. Moreover, although automatic construction still involves a large amount of engineering, this construction must be performed only once, whereas for manual construction, a taxonomy must be built manually for each new domain. Examples of applications which could benefit from an automatic development of taxonomies are monitoring the most important concepts and their evaluation in a certain field, allowing users to refine their queries by selecting more specific or more general concepts (when the previous queries returned too many or too few answers, respectively), contributing as a first step to the development of domain ontologies for improved application inter-operability, etc.

Several approaches that target building domain taxonomies for text corpora are available [1, 3, 5, 6]. These approaches extract terms from a document set. Extracting terms, however, is rather complex because texts contain many words that are not all relevant. Researchers have defined methods that handle these difficulties [7, 8, 9]. The extracted terms should not be too general, e.g., in the case of prepositions and articles, and they must be relevant to the target domain. If the terms are irrelevant, the taxonomy would be a collection of general words without any specific domain meaning. It is thus necessary to find a method that considers these factors. The extracted terms are considered concepts in a domain. Concepts here have shallow semantics in which relations such as part-of and synonymy are not defined.

There is also a need for a technique that generates hierarchical type-of relations between concepts. To create these relations, a method is required that can analyze specific properties of concepts in a data set. Based on these properties, this method should be able to form taxonomic relations between concepts. The properties we consider are mainly co-occurrences of concepts in documents, which makes this method data-driven.

In our current work, we present a framework for automatically constructing a domain taxonomy from text corpora. We call this framework Automatic Domain Taxonomy Construction from Text (ADTCT). This framework extracts concepts from text and arranges these concepts in a type-of hierarchy. We use a filtering method to extract terms from documents. Because many types of semantics can be captured with statistics [10], hierarchical relations are created using the statistics-based methods of subsumption and hierarchical clustering algorithms. For processing times, our statistics-based methods offer better scaling performance than their alternatives, i.e., the relatively heavy-weight semantics-based approaches, which are generally time consuming when disambiguating and reasoning.

We have implemented our approach in economics and management, a domain that has been rarely targeted for automatic taxonomy construction. We extracted thousands of abstracts and titles from RePub[1] and RePEc.[2] RePub is the repository of publications from the Erasmus University Rotterdam, and RePEc is a collection of economic articles maintained by a group of hundreds of volunteers in 70 countries. The document set has been divided into domain and contrastive sets. The domain set contains documents belonging to economics and management, including all RePEc documents and those RePub documents classified as belonging to economics and management. The contrastive set contains all RePub documents that do not belong to economics and management (e.g., medicine, sociology, or psychology).

Compared with existing approaches that focus on different domains, use semantics (e.g., by consulting online resources [11]), or concentrate on general ontology construction instead of taxonomy construction specifically, our contributions are fivefold. First, we compare the hierarchical clustering algorithm and the concept subsumption method in taxonomy extraction from text corpora. Second, we optimize the parameters for these methods. Third, we elaborate an implementation of this approach in economics and management. Although taxonomy construction has been applied to many domains, including finance and tourism [1], economics and management is relatively unexplored and thus is examined in our efforts. Fourth, we define guidelines for choosing the proper method for taxonomy extraction based on a trade-off between taxonomy quality and depth. Last, we refine the existing state-of-the-art subsumption algorithm by considering the position of the ancestors of the current node when constructing the taxonomy, and show that this delivers a superior performance with respect to the taxonomic F-measure.

The remainder of the paper is structured as follows. First, Section 2 reviews related work in the area of automatic taxonomy construction from text. Sections 3 and 4 introduce the ADTCT framework and its implementation. Then, Section 5 evaluates the taxonomy built using our ADTCT implementation. Fi-

[1] `http://repub.eur.nl/`

[2] `http://repec.org/`

nally, we summarize our research and provide future work directions in automatic taxonomy construction in Section 6.

2. Related Work

In this section, we discuss the current body of literature in automatic taxonomy construction. Three taxonomy construction aspects are addressed. First, methods that extract - and possibly filter - terms from a text corpus are described. Next, techniques that can be used to construct taxonomic relations between concepts are discussed, followed by a review of methods used for taxonomy evaluation.

2.1. Term Extraction and Filtering Techniques

The related literature discusses several term extraction methods. These techniques can be classified into three general categories. In this paper, we distinguish among linguistic, statistical, and hybrid methods.

Linguistic methods investigate the linguistic function of a word in a sentence, and terms are extracted based on this function. However, linguistic methods cannot define the importance of a word well. Linguistic methods use natural language processing (NLP) [7]. A common technique is part-of-speech tagging, which assigns a tag to a word in a sentence to indicate a part of speech (e.g., nouns, verbs, adjectives) [7]. Nouns or adjectives are often selected as terms when part-of-speech tagging is used [12]. Using lexico-syntactic patterns, it is possible to extract terms and their relations (including the type-of relation) from text [13].

Statistical methods use probabilistic techniques to extract terms. These methods are generally good for finding the importance of a word, but they lack techniques that handle the grammatical function of a word, as explained above. A popular statistical method is Term Frequency Inverse Document Frequency (TF-IDF) [8]. TF-IDF computes a value for a certain term, with a higher value indicating a more important term. The TF-IDF value is higher for terms that appear frequently in one document, but it is lower for terms that appear frequently in multiple documents. This can be balanced by adding a manual non-stop list to this method [12]. This list contains words that should not be filtered out. However, the non-stop list adds a non-statistical element to this technique, and this method is thus no longer a pure statistical method.

Hybrid methods combine linguistic and statistical techniques to overcome flaws in both methods. This section discusses several hybrid methods. One method is the filtering method of Sclano and Velardi [9, 14]. This method first employs linguistic methods to extract terms, as previously presented. The importance of these terms must then be determined by filtering the terms through five filters. These filters are primarily based on the term frequencies in a domain and a contrastive corpus. Based on the different filters' values, a score is defined that determines the importance of a term. Only the terms with the highest scores remain in the eventual term set.

Another method is the χ^2 method [15]. In this method, named entities are extracted using a linguistic processor, and occurrences of these entities are placed in a contingency table, which contains information about the term frequencies in the domain or contrastive corpus. Based on this table, the χ^2 value is calculated, and terms are selected if their χ^2 value is above a certain threshold value.

The last hybrid method discussed is the C/NC-Value [12]. The C-value is first calculated, which aims to improve the compound term extraction. The terms with the highest C-value are selected, and these terms undergo additional filtering with the NC-value. The NC-value adds a context weighting factor to the C-value, that is, it considers context words, which are words surrounding the selected term. These context words provide additional information about the selected term.

2.2. Creation of Hierarchical Relations

There are several methods for creating hierarchical relations between concepts. In hierarchical clustering, taxonomic relations between concepts are created by clustering terms hierarchically [16]. At the start of this algorithm, all terms are individual clusters; the distances between these clusters are calculated for each cluster. The clusters that are closest to each other are merged. This process continues until one cluster remains. Although hierarchical clustering methods have several drawbacks such as high time and memory complexities, and inefficient and inaccurate cluster validations, previous research has shown that these can be alleviated [17, 18].

Hierarchical clustering methods employ several distances. Distances can be measured using similarity in a semantic lexicon [4, 19] or in a term co-occurrence analysis [4]. One co-occurrence similarity is co-occurrence of terms in a document. The window-based co-occurrence measure, which analyzes the co-occurrences of words in windows of words, can also be used [20].

Sanderson and Croft [5] proposed the subsumption method to derive a term hierarchy from a set of documents. In the subsumption method, a given term subsumes another term (i.e., an ancestor) if the documents in which the latter term occurs are a subset of the documents in which the given term occurs [5, 21]. The advantages of the subsumption method are that it is easy to implement and it makes labeling concepts easy. The disadvantages are that with this method, it is difficult to classify terms that do not co-occur frequently and it requires a large data set to work well. Some extensions have been proposed, for example, subsumption in a fuzzy domain [22]; however, the position of the ancestors with respect to a concept is not considered during the taxonomy construction process, as proposed in this paper. We hypothesize that being able to discount the ancestors contribution to the score of a parent node can enable us to produce more accurate taxonomic relations.

2.3. Classification Methods

Classification methods can also be used to create taxonomic relations. By combining a domain corpus, a general corpus, and a named-entity tagger, terms are extracted and arranged in a taxonomy using additional sources, such as an online encyclopedia or dictionary that provide more information, e.g., synonyms, homonyms, etc. [3]. A tree-descending algorithm adds terms to a hierarchy by descending a tree or hierarchy from the root to the leaves. At every node, the child node most similar to the current node is chosen as the next node in the path to a leaf. Once this method has reached a leaf, the new word is assigned to the most similar node in the tree [23]. The tree-ascending algorithm uses a combination of similarity measures and taxonomic similarities to give a node a certain number of votes, and nodes are placed below the node with the highest number of votes [23]. Classification methods provide good results, but they require a large training set to work properly, and it is thus difficult to use these methods with a small training set.

Formal concept analysis is a conceptual clustering technique that groups objects and their attributes based on similarities. In this method, objects, attributes, and their relations form a formal concept. By linking the attributes of an object with other objects, taxonomic relations can be found. Applications of this method occur in information extraction by linking verbs with terms [1, 24]. Another method, lexical-syntactic patterns, is a linguistic technique that creates relations by analyzing textual patterns. This technique links named entities with noun phrases, and these links form taxonomic relations [13]. For example, if the noun phrase 'Portfolio management' appears with the named entity 'Markowitz,' then 'Markowitz' is extracted, and a taxonomic relation is added because it appears with the noun phrase.

2.4. Evaluation Methods

Previous studies have discussed several evaluation methods [1, 5, 25, 26, 27]. These evaluation methods often involve a comparison with a golden taxonomy, which is a taxonomy created by experts in a certain

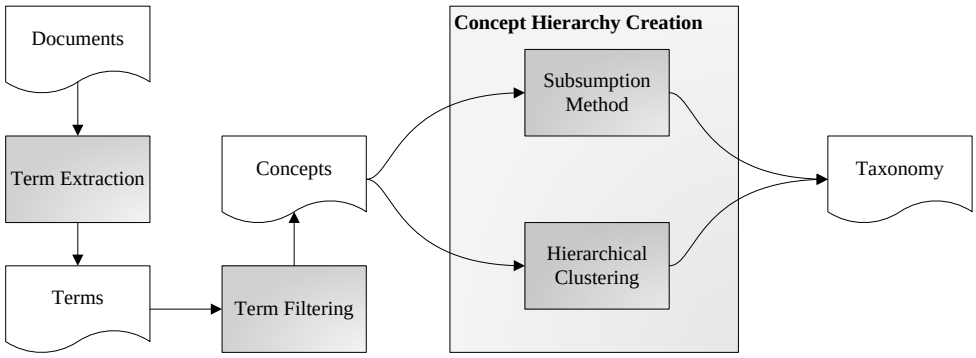

Figure 1: ADTCT framework architecture

domain. The relevance of terms in a certain domain can be determined using lexical recall and precision. The semantic cotopy [25] is used only to consider the super- and sub-concepts of nodes. The common semantic cotopy is similar, but this cotopy only considers nodes that both taxonomies have in common [27]. Both cotopies can be used to compute the taxonomic precision and recall, which reflect the taxonomic relation quality in the taxonomy. It is also possible to evaluate a taxonomy without a golden taxonomy by asking several experts to rate the taxonomy; these judgments are then averaged [5].

3. ADTCT Framework

In this section, we discuss our framework for automatically constructing a taxonomy for a specific domain. Our framework - Automatic Domain Taxonomy Construction from Text (ADTCT) - comprises several steps. First, terms are extracted from documents. Next, these terms are processed in the term filtering step, which filters them on domain pertinence, lexical cohesion, domain consensus, and structural relevance. The resulting term set contains the terms with the highest scores for several filters. These terms are stored as concepts, which are used in the taxonomy construction process. For these concepts, the hierarchical relations between them are created using either the subsumption method or the hierarchical clustering algorithm.

Figure 1 depicts the ADTCT framework architecture. As input, we have a document set. Terms are extracted from the documents in the Term Extraction step. These terms are filtered on several aspects in the Term Filtering step, so only the most relevant terms remain, which our taxonomy considers concepts. These concepts can be hierarchically arranged in the Concept Hierarchy Creation step, using either the Subsumption Method or Hierarchical Clustering. After hierarchically arranging these concepts, we have created a Taxonomy.

3.1. Term Extraction

The terms must first be extracted from text corpora. We use a part-of-speech parser [7], which processes a sentence into a tree structure. In this tree structure, part-of-speech tags are assigned to each node in the tree. All individual words are leaves on the tree, and the internal nodes are tags that determine the grammatical function of several words in sequence. The parser is used to find nouns, which are among the leaves of the tree, and noun phrases, which are internal nodes that group words (appearing in a sequence) with one meaning.

3.2. Term Filtering

In the Term Filtering step, terms are processed using several filters. This allows us to select terms based on various criteria. The method used is an adapted version of a previous filtering method [9]. We first filter the terms using domain pertinence and lexical cohesion. We then determine a domain score, which is based on domain pertinence, domain consensus, and structural relevance. The terms with the highest scores are selected as concepts.

The domain pertinence DP of domain corpus D_i (DP_{D_i}), first introduced in [28], is a filter that checks whether a term is relevant for the target domain. It is defined as a function of term t:

$$DP_{D_i}(t) = \frac{freq(t/D_i)}{\max_j(freq(t/D_j))} , \tag{1}$$

where D_j is a contrastive corpus and $freq(t/D_i)$ is the number of times term t appears in domain D_i. In this equation, the term frequency in the domain corpus is divided by the maximum term frequency in a contrastive corpus. A term that appears more frequently in the domain corpus than in the contrastive corpus has a higher DP than a term that appears less frequently in the domain corpus than the contrastive corpus. Terms with high DP values are thus likely to be more important in the target domain. To emphasize the importance of this filter and accelerate the computing time, we remove the bottom 30% of term DP values.

The lexical cohesion LC of domain D_i (LC_{D_i}), first defined in [28], is used to measure the cohesion among words in compound term t. It is defined as a function of term t:

$$LC_{D_i}(t) = \frac{n \cdot freq(t/D_i) \cdot \log(freq(t/D_i))}{\sum_{w_j \in t} freq(w_j/D_i)} , \tag{2}$$

where n is the number of words in compound term t, $freq(t/D_i)$ is the frequency of the term in domain corpus D_i, and $freq(w_j/D_i)$ is the frequency of word j from term t in domain D_i. A compound term with a high frequency that approaches the frequencies of the individual words in the compound term has a high value for lexical cohesion. To ensure that only relevant compound terms are selected, we remove the 30% compound terms with the lowest values for LC.

The domain consensus DC, first introduced in [29], is used to determine term importance by analyzing term frequencies in documents. This measure, which is a function of term t, is defined thus:

$$DC_{D_i}(t) = -\sum_{d_k \in D_i} norm\_freq(t, d_k) \cdot \log(norm\_freq(t, d_k)) , \tag{3}$$

where $norm\_freq(t, d_k)$ is the normalized frequency of term t in document d_k, which is a document in domain corpus D_i. Normalization occurs by dividing the calculated frequency by the maximum frequency of term t in any document in the domain corpus. This procedure is defined thus:

$$norm\_freq(t, d_k) = \frac{freq(t, d_k)}{max_{d \in D_j}(freq(t, d))} . \tag{4}$$

Terms that appear in the title are likely to be more important than terms that do not appear in the title. We need to consider this information. We thus add a constant value k to terms that appear in the title for the domain score.

After calculating all filter values, we combine these values to compute a final domain score:

$$score(t, D_i) = \alpha \cdot \frac{DP_{D_i}(t)}{max_{t \in D_i}(DP_{Di}(t))} + \beta \cdot \frac{DC_{D_i}(t)}{max_{t \in D_i}(DC_{Di}(t))} + k , \tag{5}$$

where α and β are weights that emphasize $DP_{D_i}(t)$ or $DC_{D_i}(t)$, respectively. Furthermore, $max_t(DP_{Di}(t))$ and $max_t(DC_{Di}(t))$ are the highest values for domain pertinence and consensus in domain corpus D_i. The latter two values are used to normalize the domain consensus and pertinence, thus allowing a relative comparison. Normalizing these values has not been performed in the previous literature. Terms that appear in a title receive a value of k added to their score. The α, β, and k parameters can have any value between 0 and 1.

3.3. Concept Hierarchy Creation

In our framework, we distinguish two methods to determine hierarchical relations between concepts. First we discuss the subsumption method [5, 21], and then we discuss the hierarchical clustering algorithm [16].

We improve the subsumption method introduced in [5] and [21] by considering the ancestors' position with respect to the current node. The subsumption algorithm is based on concept co-occurrences. The co-occurrences for this method are based on document co-occurrence, which means that we analyze concept co-occurrences in different documents. For each concept, potential parent concepts or subsumers are determined. Concept x potentially subsumes concept y if:

$$P(x \mid y) \geq t, P(y \mid x) < t \ , \tag{6}$$

where t is a co-occurrence threshold. If x appears in at least proportion t of the documents in which y also appears and y appears in less than proportion t in which x appears, then x is a potential parent of y. Multiple potential parents might exist, so we must choose one potential parent to maintain the hierarchical tree structure. This decision is based on a score calculated for each potential parent. The score is defined thus:

$$score(p, x) = P(p \mid x) + \sum_{a \in A_p} w(a, x) \cdot P(a \mid x) \ , \tag{7}$$

where p is the potential parent node of x and A_p is the list of ancestors of p. In the equation, $w(a, x)$ is a weight that is multiplied by the co-occurrence probability of ancestor a given concept x. In this equation, the weight is defined thus:

$$w(a, x) = \frac{1}{d(a, x)} \ , \tag{8}$$

where $d(a, x)$ is the path length between node x and ancestor a. By applying this weight, concept nodes that are closer to the parent node have more influence on the final score calculation. In the end the potential parent with the highest score based on (7) is chosen as the parent of node x.

Hierarchical agglomerative clustering is a clustering method that begins with all terms as separate clusters and the nearest clusters are progressively combined until one cluster remains that comprises all terms [16]. In general, we can describe the algorithm as follows.

1. Start with n clusters (each term is a cluster).
2. Compute the distances between clusters.
3. Merge the two nearest clusters into one cluster. Return to step 2 if more than one cluster remains; otherwise, the algorithm has finished.

We implemented several measures to define the distances between clusters: minimum distance/single linkage, maximum distance/complete linkage, and average distance/average linkage. The minimum and maximum distances are calculated by taking the distances between a concept in cluster A and a concept in cluster B that are the shortest and longest distances between concepts in clusters A and B, respectively.

The average distance is calculated by taking the average distance of all of the distances between concepts in clusters A and B.

To calculate the distances between the clusters, the distances between individual concepts must be calculated. We have defined two distance measures for this purpose: document co-occurrence similarity and window-based similarity. The document co-occurrence similarity is determined by computing the number of documents in which two terms co-occur [4]. The similarity is defined thus:

$$sim(t_1, t_2) = \frac{2df(c_1, c_2)}{df(c_1) + df(c_{2)}} , \tag{9}$$

where $df(c_1, c_2)$ is the total number of documents in which c_1 and c_2 co-occur. Furthermore, $df(c_1)$ is the total number of documents in which concept c_1 appears and $df(c_2)$ is the total number of documents in which concept c_2 appears. Concepts that co-occur frequently in documents thus have higher values for this similarity.

The window-based similarity is analogous to the document co-occurrence similarity, but it analyzes the co-occurrences of concepts in user-specified word windows instead of examining co-occurrences in documents [4]. This similarity is defined thus:

$$sim(c_1, c_2) = \frac{2wf(c_1, c_2)}{wf(c_1) + wf(c_{2)}} , \tag{10}$$

where $wf(c_1, c_2)$ is the total number of times that c_1 and c_2 co-occur in one window, $wf(c_1)$ is the total number of times that concept c_1 appears in windows, and $wf(c_2)$ is the number of times that concept c_2 appears in windows. Concepts that often co-occur in windows thus have higher values for this similarity.

The windows are created for each document based on window size. Suppose that we have a document with four concepts: 'Ad,' 'Bert,' 'Cees,' and 'Dirk.' If the window size is 2, the following windows are created for this document: {Ad}, {Ad, Bert}, {Bert, Cees}, {Cees, Dirk}, and {Dirk}.

One disadvantage of this method is that it is difficult to label clusters because the algorithm only clusters concepts together and does not create labels for these clusters. Therefore, we need an appropriate labeling method. In our case, we label concepts by calculating the distances between all of the concepts in the cluster. The centroid, the concept in the center of the cluster, is then chosen as the cluster label. When there are only two concepts in a cluster, the concatenation of these two concepts is chosen as the cluster label because this cluster does not have a center.

4. ADCTC Implementation

We implemented the ADTCT framework as a Java application. We chose Java because many Java libraries are available that support the methods proposed in our framework. To parse nouns, we used the Part-of-Speech Parser created by Stanford University [7]. RDF representations are manufactured using the Jena framework [30]. The created taxonomies are exported as RDF files with a SKOS vocabulary [31]. We can thus compare our generated taxonomy with manually created taxonomies, which are usually also stored as RDF files with an SKOS vocabulary.

Figure 2 presents a screen shot of our program implementation. This screen shows the output of a computed taxonomy created using the hierarchical clustering algorithm.

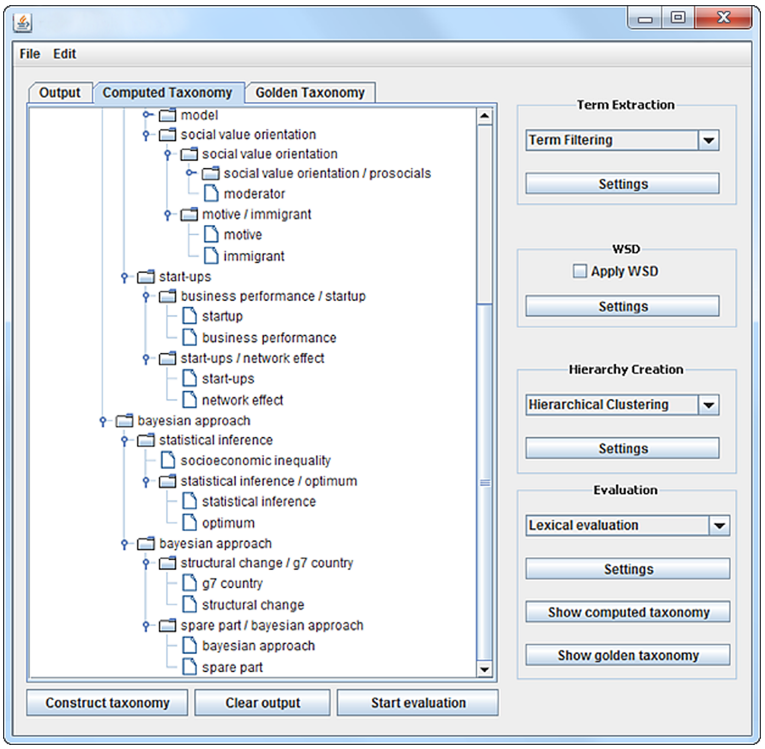

Figure 2: ADTCT implementation

4.1. Term Extraction

In this step, terms are extracted from the domain corpus. This corpus contains RePEc documents and RePub documents that belong to the economics and management domain. The contrastive corpus contains documents from other domains, including law and health. The total corpus size is 25,000 documents, of which 20,000 documents are in our domain and 5,000 documents belong to the contrastive domain.

We use a part-of-speech parser [7] to parse a sentence and extract the nouns and noun phrases from this sentence. A part-of-speech parser is slower than a part-of-speech tagger, but the results are more accurate, especially for compound nouns [32].

4.2. Term Filtering

The term filtering step is used to select the most relevant terms. The extracted terms are processed through several filters before the most important terms are selected. The terms are processed through five filters sequentially, and a score is eventually calculated based on some of these filters, as explained in our framework. A higher score indicates that a term is more relevant for our domain. The most relevant terms are stored as concepts.

The domain pertinence filter assigns a domain pertinence value to each term in the potential term set. This value indicates the relevance of a term to a certain domain. To emphasize the term relevance in our domain, we have filtered out the bottom 30% of the terms' domain pertinence values.

The lexical cohesion filter assigns a lexical cohesion value to all compound terms (terms represented by noun phrases). This value reflects the relevance of a compound term. We use this filter to eliminate

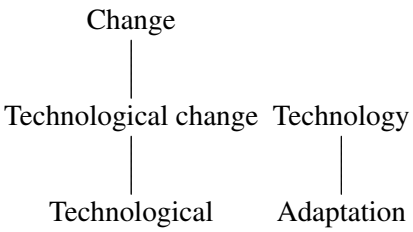

Figure 3: Potential parents of 'Technology adaptation' in a tree structure

compound terms that are less relevant. Therefore, 30% of the compound terms with the lowest value for lexical cohesion are removed from the list of potential terms.

The domain consensus filter assigns a domain consensus value to all of the remaining potential terms. The domain consensus reflects the term frequencies in the domain documents, and it thus reflects the relevance of terms for the taxonomy. This filter value is used only to calculate the final score.

This domain consensus filter assigns an additional k value to potential terms that appear in the title of a document. Terms that appear in the title are most likely more important than terms that do not; we account for this effect by adding a constant of k to the potential terms that appear in the title.

After all filters have processed the potential terms, we can compute a domain score for these terms. A higher score indicates that a term is more relevant for our domain. The score is based on domain pertinence, lexical cohesion, domain consensus, and structure relevance, as explained in the framework.

To find the optimal domain score, we evaluated the taxonomy for values of α, β, and k ranging between 0 and 1 on a set of 25,000 documents. The full analysis of this evaluation can be found in our evaluation section. Based on this evaluation, we found the optimal values, using a discretization step of 0.05, for α, β, and k to be 0.95, 0.05, and 0.5, respectively.

4.3. Concept Hierarchy Creation

We implemented two methods to create a concept hierarchy: the subsumption and hierarchical clustering methods. With the subsumption method, we can create taxonomic relations between concepts by applying a subsumption algorithm. This algorithm is executed as explained in our framework.

Let us consider the concept 'Technology adaptation' with three potential parents: 'Technology,' 'Technological,' and 'Adaptation,' as in Fig. 3. Scores can be calculated using threshold t, which equals 0.4 in this example. Because 'Technology' has no potential parents, we calculate $P(p \mid x)$ using (6), yielding 0.4. For the ancestors of 'Technological,' which are 'Technological change' and 'Change,' we calculate $P(p \mid x)$, which is 0.4. Subsequently, we calculate $P(a \mid x)$ for both parents, yielding 0.2 and 0.05 for 'Technological change' and 'Change', respectively. The influence of the parents is considered by weighing the scores with $\frac{1}{d(a,x)}$. The final score is a sum of $P(p \mid x)$ and all weighted ancestor scores $\frac{1}{d(a,x)}P(a \mid x)$, that is, $0.4 + \frac{1}{2}0.2 + \frac{1}{3}0.05 \approx 0.52$. Similarly, the score for 'Technology adaptation' can be calculated. The parent of 'Technological adaptation' is 'Adaptation' and the parent of 'Adaptation' is 'Technology.' Based on this hierarchy we can compute the score: $0.6 + \frac{1}{2}0.35 + \frac{1}{3}0.4 \approx 0.93$. All scores have been summarized in Table 1.

Based on these scores, 'Adaptation' is chosen as the parent of 'Technology adaptation.' The quality of the created taxonomy depends on the value of threshold t. A higher threshold means that the results are more reliable but the taxonomy is shallower. We have evaluated the subsumption algorithm for t-values ranging from 0.2 to 0.8 with incremental steps of 0.05. Based on these evaluations, we find that $t = 0.75$ has

Potential Parent Concept	Score
Technology	0.40
Technological	0.52
Adaptation	0.93

Table 1: Potential parents of 'Technology adaptation' and their scores

the best results according to the taxonomic F-value but the generated taxonomy has a low average depth. A t-value of 0.25 has a deeper taxonomy, and the results are only 3.2% worse than $t = 0.75$ in the taxonomic F-value. We must make a trade-off between tree quality and depth. More information about this process can be found in the evaluation section of this paper. Figure 4 presents an example of part of a taxonomy constructed using the subsumption method.

The agglomerative hierarchical clustering algorithm is another method used to create taxonomic relations. This method is more complex than the subsumption algorithm, and it requires more computing time. Hierarchical clustering first defines every concept as a cluster and groups these clusters until there is only one cluster remaining. The distances between clusters are based on single, full, or average linkages, which define the distances between clusters based on the distances between concepts in these clusters. The distances between these concepts are calculated using the document co-occurrence or window co-occurrence similarities.

Suppose that we have two concepts, 'System' and 'Process,' and we want to calculate the similarity between them. We have found that 'System' appears in documents {1,3,6,8} and windows {1,5,10,14,18,20,28}; 'Process' appears in documents {1,3,6,12} and windows {1,5,12,14,18,25,30}. Both terms thus co-occur three times in documents and four times in windows. Hence, we can compute the document similarity as $\frac{2 \cdot 3}{4+4} = 0.75$ and the window similarity as $\frac{2 \cdot 4}{7+7} \approx 0.57$.

After calculating the distances between concepts, we can compute the distances between the clusters. Suppose that we have the clusters defined in Table 2, where the similarities are converted to distances by applying the following equation to the similarities:

$$d(t_1, t_2) = \frac{1}{sim(t_1, t_2)} \, . \tag{11}$$

We merge the clusters that are closest to each other. For minimum linkage, these are 'Money' and '(Return, Trader)' because the minimum distance between the clusters is 0.09, which is the lowest distance in the

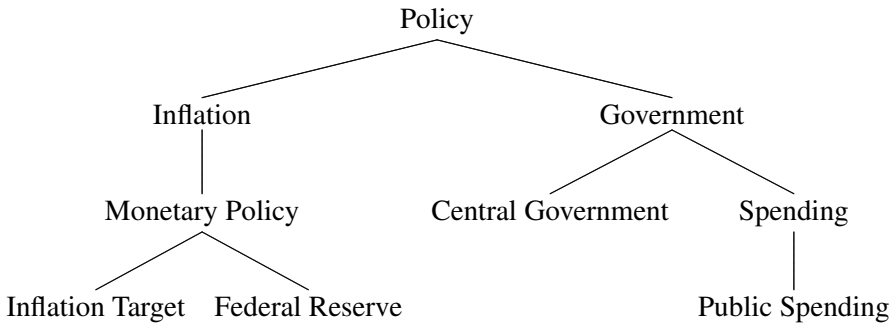

Figure 4: Part of a taxonomy created using the subsumption method for economics and management

Concept / Concept	(Return, Trader)	Organization	Real Option	Money
(Return, Trader)	0.00			
Organization	{0.20,0.10}	0.00		
Real option	{0.55,0.40}	0.55	0.00	
Money	{0.09,0.28}	0.70	0.21	0.00

Table 2: Distance matrix after (Return, Trader) concatenation

distance matrix. For maximum linkage, this connection is '(Return, Trader)' with 'Organization' because the maximum distance is 0.20, which is the lowest distance in the distance matrix. For average linkage, we use '(Return, Trader)' with 'Organization' because the average distance is 0.15, which is also the lowest distance in this distance matrix.

The newly created cluster must be labeled. Suppose that we have used complete linkage, and '(Return, Trader)' is merged with 'Organization.' Table 3 shows the distances between these concepts. The average distance from 'Return' to other concepts is $\frac{(0.20+0.06)}{2} = 0.13$, the distance from 'Trader' to other concepts is $\frac{(0.06+0.10)}{2} = 0.08$, and the distance from 'Organization' to other concepts is $\frac{(0.20+0.10)}{2} = 0.15$. 'Trader' has the lowest average distance, which implies that 'Trader' is most likely the centroid. 'Trader' is thus chosen as the cluster label.

After two clusters have been merged, the algorithm continues to merge clusters until one cluster remains. We evaluated this algorithm for complete, single, and average linkages. Based on our evaluation, we found that complete linkage gives the most accurate results. We also evaluated the different window sizes, ranging from 4 to 17, and we found an ideal window size of 11. We found that window-based similarity has better results than document-based similarity. More information about our evaluation can be found in our evaluation section. Figure 5 presents an example of part of a taxonomy constructed with the hierarchical clustering algorithm.

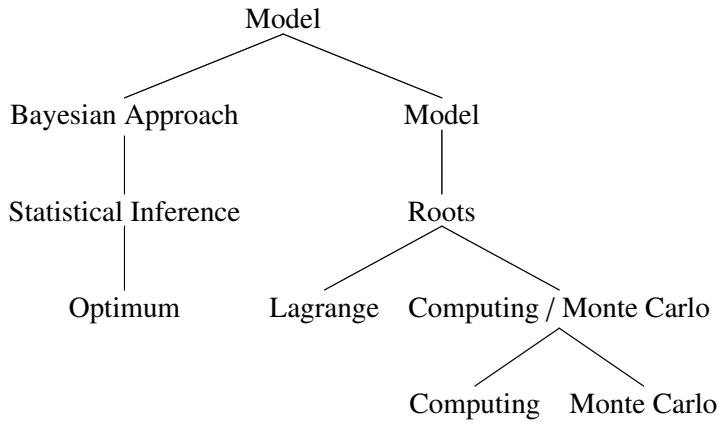

Figure 5: Part of a taxonomy created using hierarchical clustering for economics and management

Concept / Concept	Return	Trader	Organization
Return	0.00		
Trader	0.06	0.00	
Organization	0.20	0.10	0.00

Table 3: Distance matrix of concepts in a cluster

5. Evaluation

In this section, we evaluate the taxonomy for the economics and management domain built based on the settings used in the previously discussed implementation of our ADTCT framework. We first describe the framework of our evaluation approach and report and discuss the evaluation results.

5.1. Experimental Setup

Previous studies [1] and [27], have discussed several evaluation methods for domain taxonomies. Based on these research results, we defined our evaluation framework. In this framework, we compare our automatically created taxonomy with a taxonomy created by experts in economics and management. The golden taxonomy used for this purpose is a pre-processed version of the STW Thesaurus.[3] This pre-processed version does not include concepts belonging to other domains, and approximately 2% of the concepts were translated from German into English. This detailed taxonomy contains approximately $4,000$ concepts.

First, we define a measurement that analyzes the relevance of found terms. The measurement used is the lexical precision, defined thus:

$$LP(O_C, O_R) = \frac{|C_C \cap C_R|}{|C_C|} , \tag{12}$$

where O_C is the core ontology, our computed taxonomy, and O_R is the reference ontology, our golden taxonomy. Furthermore, C_C are concepts in the core ontology, and C_R are concepts in the reference ontology. This method thus divides the number of concepts in the intersection of concepts in both taxonomies by the number of concepts in the core ontology. This measure gives a good indication of the relevance of the found terms. A similar measurement, lexical recall, is also used in the literature. This method is unsuitable for our purposes because our golden taxonomy is much larger than our computed taxonomy.

After defining a method to compute term relevance, we need a method to compute the quality of the relations between concepts. We must first define the foundations of these measurements. We can define the common semantic cotopy of a node, which considers only the super- and sub-concepts that appear in both in taxonomies:

$$csc(c, O_1, O_2) = \{c_i \in C_1 \cap C_2 | (c_i \leq_{C_1} \ c \vee c \leq_{C_1} \ c_i)\} , \tag{13}$$

where O_1 and O_2 are two ontologies, c is the concept being analyzed, c_i is a common concept, and $\leq_{C_1}$ is the partial order induced by type relationships in ontology O_1.

With this definition, we can define local measurements that compute the local taxonomic relation qualities using the local taxonomic precision:

$$tp_{csc}(c, O_1, O_2) = \frac{|csc(c, O_1, O_2) \cap csc(c, O_2, O_1)|}{|csc(c, O_1, O_2)|} . \tag{14}$$

[3] http://zbw.eu/stw/

This measurement uses the intersection of the two common semantic cotopies in which the input ontologies are swapped. This intersection is then divided by the first of these common semantic cotopies. This indicates how many common relations are found in O_1.

Using this local measurement definition, we can define a global measurement, for which we use the taxonomic precision and recall, formulated as:

$$TP_{csc}(O_C, O_R) = \frac{1}{|C_C \cap C_R|} \sum_{c \in C_C \cap C_R} tp_{csc}(c, O_C, O_R) \,, \tag{15}$$

$$TR_{csc}(O_C, O_R) = TP_{csc}(O_R, O_C) \,, \tag{16}$$

where O_C is the core ontology, our computed taxonomy, and O_R is the reference ontology, our golden taxonomy. The taxonomic precision reflects how similar the relations in the intersection of both ontologies are to the core ontology; the taxonomic recall captures the similarity of the relations in the intersection of the ontologies to the reference ontology. Both measurements must be considered; we use a measure for the overall quality of the relations in the taxonomy, the taxonomic F-measure (TF):

$$TF(O_C, O_R) = \frac{2 \cdot TP_{csc}(O_C, O_R) \cdot TR_{csc}(O_C, O_R)}{TP_{csc}(O_C, O_R) + TR_{csc}(O_C, O_R)} \,, \tag{17}$$

where TF is the harmonic mean of the taxonomic recall and precision. This measurement obtains a value between 0 and 1; higher values indicate that the relations between common concepts in the taxonomies are more similar.

5.2. Experimental results

We evaluated the parameters of the term filtering algorithm using a brute force approach by incrementing α, β, and k with a step size of 0.05 in the range between 0 and 1 on a set of 20,000 domain and 5,000 contrastive documents. We extracted 2,000 concepts, which we placed in the taxonomy. Table 4 shows a selection of the results.

Table 4 gives the optimal setting for term extraction for the considered discretization step size and range. The weights are $\alpha = 0.95$ (domain pertinence), $\beta = 0.05$ (domain consensus) and $k = 0.5$ (structural relevance). The values for these weights can be explained by the importance of domain pertinence. Domain pertinence considers the appearance of terms in domain documents relative to contrastive documents, whereas domain consensus analyzes only the appearance of terms in domain documents. More specific terms can be found by assigning a high weight to the domain pertinence filter. A high k value indicates that important domain terms often appear in term titles.

We evaluated the subsumption algorithm for t ranging from 0.2 to 0.8 with a step size of 0.05. We also added the results of a subsumption algorithm that does not consider the position of the ancestors with respect to the current node. The results are presented in Table 5 and Figs. 6 and 7. Table 5 also includes the results of the improved and original versions of the subsumption algorithm.

Table 5 demonstrates that the improved subsumption algorithm has the same or better TF results for all t-values. This is especially true for a low t because this value generates more potential parents and our method improves the parent selection process from a set of potential parents. Furthermore, $t = 0.75$ provides the highest TF value, and the relations for this t-value are the best; however, the average depth for this t-value is low. It is possible to achieve a deeper taxonomy with minimal quality loss by lowering the threshold value because the depth increases significantly more than the quality decreases. For very low threshold values, however, the quality decreases too much, and the concept relations are poor. We must

generally consider two aspects in our trade-off decision. A user might want a minimal taxonomic relation quality and taxonomy depth. We can formulate this trade-off decision mathematically as:

$$TaxonomyScore(t) = \gamma \cdot \frac{TF(t)}{max_t(TF(t))} + \lambda \cdot \frac{Depth(t)}{max_t(Depth(t))} \quad \text{w.r.t.} \tag{18}$$
$$TF(t) \geq q\ , Depth(t) \geq\ d\ , \gamma + \lambda =\ 1\ , \gamma \geq\ 0\ , \lambda \geq\ 0\ ,$$

where $TF(t)$ is the TF measure depending on threshold t, $Depth(t)$ is the average depth depending on threshold t, and γ and λ are weights that emphasize quality and depth. The t-value that maximizes the $TaxonomyScore$ is chosen as the best value, which must adhere to the above mentioned constraints. In these constraints, q is the minimal required taxonomy quality, and d is the minimal required taxonomy depth.

Suppose that we want to develop a taxonomy with a minimal average depth of 3 and a minimal quality of 0.60. Only $t = 0.20$,$t = 0.25$, and $t = 0.30$ obey these constraints. Then, suppose that we place more emphasis on depth, so that $\gamma = 0.40$ and $\lambda = 0.60$. For $t = 0.20$, the computed score is $0.40 \cdot \frac{0.6002}{0.6401} + 0.60 \cdot \frac{5.07}{5.07} \approx 0.975$; for $t = 0.25$ the computed score is $0.40 \cdot \frac{0.6182}{0.6401} + 0.60 \cdot \frac{4.14}{5.07} \approx 0.878$; and for $t = 0.30$ the

α	β	k	LP
1.00	0.00	0.00	0.1579
1.00	0.00	0.05	0.1744
1.00	0.00	0.10	0.1744
1.00	0.00	0.50	0.1749
1.00	0.00	1.00	0.1749
0.95	0.05	0.00	0.1614
0.95	0.05	0.05	0.1764
0.95	0.05	0.10	0.1764
0.95	0.05	0.50	0.1769
0.95	0.05	1.00	0.1769
0.90	0.10	0.00	0.1624
0.90	0.10	0.05	0.1759
0.90	0.10	0.10	0.1759
0.90	0.10	0.50	0.1759
0.90	0.10	1.00	0.1759
0.85	0.15	0.00	0.1589
0.85	0.15	0.05	0.1734
0.85	0.15	0.10	0.1734
0.85	0.15	0.50	0.1734
0.85	0.15	1.00	0.1734
0.10	0.90	0.00	0.1409
0.10	0.90	0.05	0.1489
0.10	0.90	0.10	0.1494
0.10	0.90	0.50	0.1494
0.10	0.90	1.00	0.1494

Table 4: Evaluation results of term extraction

	ADTCT		Original	
t	TF	Avg. Depth	TF	Avg. Depth
0.20	0.6002	5.07	0.5887	5.52
0.25	0.6194	4.15	0.6148	4.34
0.30	0.6294	3.29	0.6268	3.35
0.35	0.6323	2.95	0.6319	2.97
0.40	0.6327	2.81	0.6320	2.82
0.45	0.6385	2.64	0.6379	2.65
0.50	0.6399	2.59	0.6397	2.59
0.55	0.6399	2.59	0.6387	2.50
0.60	0.6391	2.47	0.6391	2.47
0.65	0.6399	2.43	0.6399	2.43
0.70	0.6399	2.42	0.6399	2.42
0.75	0.6401	2.39	0.6401	2.39
0.80	0.6392	2.38	0.6392	2.38

Table 5: Evaluation results of the subsumption method

computed score is $0.40 \cdot \frac{0.6282}{0.6401} + 0.60 \cdot \frac{3.29}{5.07} \approx 0.782$. With these preferences, we obtain $t = 0.20$ as the preferred t-value.

In our work, we also implemented the hierarchical clustering algorithm. The computing time of this algorithm is much longer than before, and the algorithm evaluation is based on a smaller set than the subsumption algorithm. To ensure that both sets are similar, we calculated the average number of nouns per document for every set with their standard deviations, as in Table 6. The table underlines that the used data sets are comparable because the averages and standard deviations of the different sets are similar. Labels may appear multiple times in the clustering evaluation. Approximately 11% to 40% of the concepts have double labels, and these values can differ depending on the algorithm parameters and input.

We evaluated the cluster linkages on a set of $5,000$ domain and $2,500$ contrastive documents with $1,000$ extracted concepts. The distance used is the document co-occurrence similarity. Table 7 presents the evaluation results. As demonstrated by the table, the complete linkage method provides the best results. Many relations formed with clustering are appropriate. This evaluation method has not, however, considered the labeling issue. A concept often appears multiple times in the taxonomy because the centroids of some clusters might be the same. In our evaluation framework, we considered every label as a separate concept.

For hierarchical clustering, we can use two distance measures: the document co-occurrence or window co-occurrence similarities. We evaluated both distance measures on a set of $2,500$ domain and $2,500$ contrastive documents with complete linkage and $1,000$ extracted terms. Table 8 summarizes the results, based on which we can conclude that the window co-occurrence similarity outperforms the document co-occurrence similarity and the optimal window size is 11.

To compare the hierarchical clustering and subsumption algorithms, we evaluated two document subsets, containing $2,500$ domain and $2,500$ contrastive documents, for both algorithms. For the subsumption algorithm, the thresholds $t = 0.25$, $t = 0.50$, and $t = 0.75$ were evaluated to reflect different user preferences. For the hierarchical clustering algorithm, the optimized settings found in the previous experiments were used. Table 9 presents the results, indicating that the subsumption algorithm outperforms the hierarchical clustering algorithm for $t = 0.75$ and $t = 0.50$. However, for $t = 0.25$, the hierarchical clustering algorithm performs better. The taxonomic relations formed in the subsumption algorithm are more accurate

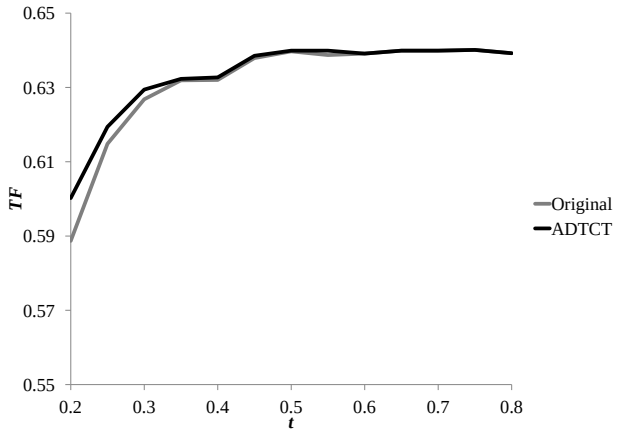

Figure 6: TF in relation to t

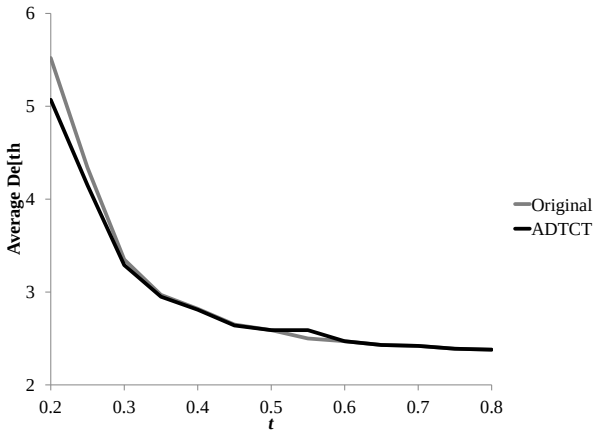

Figure 7: Average depth in relation to t

for higher thresholds than those formed in the hierarchical clustering algorithm. The hierarchical clustering is interesting mainly for users who want to obtain a deep taxonomy. For deeper taxonomies, the optimized hierarchical clustering algorithm settings perform better, but the subsumption method performs better for shallower taxonomies.

The greater depth of the hierarchical clustering algorithm can easily be explained. The depth is greater because the clustering algorithm clusters all $1,000$ concepts until one cluster remains that contains all concepts. When many concepts are clustered, the paths from the leaf nodes to the root node tend to be long, which increases the path length. The taxonomy has one root node and many leaf nodes. The paths to the leaf nodes vary in length: some are long, and some are short, and long paths often have many repeated labels. The relations in the taxonomy are, however, precise. Taxonomy interpretability and quality are especially good near leaf nodes because these nodes are most similar to each other and there are few repeated labels at this level.

Data set	Size	Avg. Nouns	St. Dev.
Domain	20,000	36.23	20.16
	5,000	36.35	18.30
	2,500	36.88	20.28
Contra	5,000	60.62	18.30
	2,500	60.97	20.28

Table 6: Average nouns per document per data set

Cluster linkage	TF
Average	0.5432
Minimum	0.5508
Maximum	0.6210

Table 7: Evaluation results of cluster linkages

Distance	TF
Window (size: 4)	0.5839
Window (size: 5)	0.5708
Window (size: 6)	0.6163
Window (size: 7)	0.6153
Window (size: 8)	0.6367
Window (size: 9)	0.6046
Window (size: 10)	0.6110
Window (size: 11)	0.6509
Window (size: 12)	0.6303
Window (size: 13)	0.6404
Window (size: 14)	0.6495
Window (size: 15)	0.6417
Window (size: 16)	0.6429
Window (size: 17)	0.6196
Document	0.5700

Table 8: Evaluation results of distance measures

Method	TF	Avg. Depth
Subsumption (t = 0.25)	0.6296	4.837
Subsumption (t = 0.5)	0.6887	2.9585
Subsumption (t = 0.75)	0.7022	2.4378
Hierarchical clustering (window size: 11)	0.6509	237.9040

Table 9: Subsumption and clustering compared

6. Conclusions and future work

In this paper, we have presented a framework to automatically extract a taxonomy from text corpora. In our framework, Automatic Domain Taxonomy Construction from Text (ADTCT), we extract terms from the text corpora employing a part-of-speech parser. The nouns extracted by this parser are filtered using domain pertinence, domain consensus, lexical cohesion, and structural relevance. After filtering these terms, only the most important terms remain, which we consider domain concepts.

In our framework, concepts (taxonomy terms) are hierarchically arranged using two methods: the subsumption method and a hierarchical clustering algorithm. The subsumption method is based on document co-occurrence: in this method, a concept subsumes another concept if the occurrence of the second concept is always in conjunction with the occurrence of the first concept, but not the other way around. The subsumption algorithm labels concepts and arranges them in a hierarchy. We refined the algorithm so that it considers the position of the ancestors with respect to a concept when creating a taxonomy. The taxonomy created by this method is often shallow, but its depth can be influenced by changing the threshold parameter of this method. A lower threshold creates a deeper taxonomy, although the relation qualities in this taxonomy are usually worse than those of a shallower taxonomy.

The hierarchical clustering algorithm creates a hierarchy by first assuming that all concepts are clusters. The most similar clusters are merged until one cluster remains. The hierarchical clustering algorithm provides good results for the created taxonomic relations. However, labels may appear multiple times when the hierarchical clustering algorithm is employed because of the labeling method used. Approximately 11% to 40% of the concepts have the same labels, and these values differ depending on the algorithm parameters and input.

We used several evaluation methods for our framework. To determine the relevance of the found concepts, we used the lexical precision (LP), which expresses the relevance of the found terms in the computed taxonomy. To evaluate taxonomic relations in the taxonomy, we used the taxonomic F-measure (TF), which is based on the taxonomic recall and precision.

We found that the optimized settings in the term extraction step require that the domain pertinence filter has a higher weight than the other filters. The optimized settings for the subsumption method depend on the user's preferences regarding the depth and quality of the relations in the taxonomy. A taxonomy with greater depth has more information about the taxonomic relations, but these taxonomic relations are less reliable. We have defined a decision-support criterion based on taxonomy quality and average depth that considers the user-preferred settings. For hierarchical clustering, complete linkage provides better results than single and average linkage. The best-performing distance measure is the window co-occurrence similarity with a window size of 11. We compared the subsumption method to the hierarchical clustering algorithm. Users who are primarily interested in the quality of the taxonomic relations should apply the subsumption algorithm with a high threshold value. Users interested in a deep taxonomy should apply the hierarchical clustering algorithm with the previously determined optimized settings.

For future work, we would like to evaluate other term extracting techniques, including the C/NC-Value or TF-IDF. It could also be useful to develop better labeling methods for the hierarchical clustering algorithm, for instance by using Web-based approaches such as using highly ranked search result snippets [11]. Alternatively, we would like to investigate how to label concepts by analyzing the relations between concepts in a cluster using a semantic lexicon (e.g., WordNet). Our current approach does not consider concepts that have the same meaning, but that are lexically different, or concepts that share the same lexical representation, but have different meanings, to be equal. This limitation can be addressed by employing a word sense disambiguation step in the taxonomy construction process.

Future work could also examine the possibilities of using external sources and evaluate the impact of these external sources on automatic taxonomy creation. For example, the distance measures in the hierarchical clustering algorithm could be based on external sources, such as a semantic lexicon [19]. Nevertheless, in order to be able to apply these distance measures, the previously identified word sense disambiguation procedure needs to be performed.

Our framework could also be used with a different method to create taxonomic relations. We could use formal concept analysis or classification methods to create taxonomic relations. Experimenting with these techniques in our framework will result in a better overview of the advantages and disadvantages of methods that can be applied to create taxonomic relations from text corpora. Additionally, we could refine our subsumption method by analyzing concept co-occurrences in user-defined word windows instead of whole documents, similar to how we applied the hierarchical clustering algorithm. We would also like to apply our framework to other domains such as law, physics, and chemistry.

References

[1] P. Cimiano, A. Hotho, S. Staab, Learning Concept Hierarchies from Text Corpora Using Formal Concept Analysis, Journal of Artificial Intelligence research 24 (2005) 305–339.

[2] T. R. Gruber, A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition 5 (1993) 199–220.

[3] N. Weber, P. Buitelaar, Web-based Ontology Learning with ISOLDE, in: Workshop on Web Content Mining with Human Language Technologies collocated with the 5th International Semantic Web Conference (ISWC 2006).

[4] M. Neshati, A. Alijamaat, H. Abolhassani, A. Rahimi, M. Hosseini, Taxonomy Learning Using Compound Similarity Measure, in: 6th ACM International Conference on Web Intelligence (WI 2007), IEEE Computer Society, 2007.

[5] M. Sanderson, B. Croft, Deriving Concept Hierarchies from Text, in: 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999), ACM, 1999, pp. 206–213.

[6] B. Fortuna, D. Mladenič, M. Grobelnik, Semi-Automatic Construction of Topic Ontologies, in: M. Ackermann, B. Berendt, M. Grobelnik, A. Hotho, D. Mladenič, G. Semeraro, M. Spiliopoulou, G. Stumme, V. Svátek, M. van Someren (Eds.), Semantics, Web and Mining, Joint International Workshops, EWMF 2005 and KDO 2005, volume 4289 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 121–131.

[7] D. Klein, C. D. Manning, Fast Exact Inference with a Factored Model for Natural Language Parsing, in: S. Becker, S. Thrun, K. Obermayer (Eds.), Neural Information Processing Systems (NIPS 2002), volume 15 of *Advances in Neural Information Processing Systems*, MIT Press, 2002, pp. 3–10.

[8] G. Salton, M. J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, Inc., 1986.

[9] F. Sclano, P. Velardi, TermExtractor: a Web Application to Learn the Shared Terminology of Emergent Web Communities, in: R. Dieng-Kuntz, C. Enguehard (Eds.), 7th Conference on Terminology and Artificial Intelligence (TIA 2007), Presses Universitaires de Grenoble, 2007.

[10] A. Halevy, P. Norvig, F. Pereira, The Unreasonable Effectiveness of Data, IEEE Intelligent Systems 24 (2009) 8–12.

[11] S.-L. Chuang, L.-F. Chien, Taxonomy Generation for Text Segments: A Practical Web-Based Approach, ACM Transactions on Information Systems 23 (2005) 363–396.

[12] K. T. Frantzi, S. Ananiadou, J. ichi Tsujii, The C-value/NC-value Method of Automatic Recognition for Multi-Word Terms, in: 2nd European Conference on Research and Advanced Technology for Digital Libraries (ECDL 1998), Springer, 1998, pp. 585–604.

[13] M. A. Hearst, Automatic Acquisition of Hyponyms from Large Text Corpora, in: 14th Conference on Computational Linguistics (COLING 1992), volume 2, pp. 539–545.

[14] B. Fortuna, N. Lavrac, P. Velardi, Advancing Topic Ontology Learning through Term Extraction, in: T. B. Ho, Z.-H. Zhou (Eds.), 10th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2008), volume 5351 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 626–635.

[15] K. Pearson, On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is such that it can be Reasonably supposed to have arisen from Random Sampling, Philosophical Magazine Series 5 50 (1900) 157–175.

[16] G. N. Lance, W. T. Williams, A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems, The Computer Journal 9 (1967) 373–380.

[17] M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast Hierarchical Clustering and Its Validation, Data & Knowledge Engineering 44 (2003) 109–138.

[18] M. Dash, S. Petrutiu, P. Scheuermann, pPOP: Fast yet Accurate Parallel Hierarchical Clustering using Partitioning, Data & Knowledge Engineering 61 (2007) 563–578.

[19] T. Pedersen, S. Patwardhan, J. Michelizzi, WordNet::Similarity - Measuring the Relatedness of Concepts, in: 19th National Conference on Artificial Intelligence (AAAI 2004), MIT Press, 2004, pp. 1024–1025.

[20] K. W. Church, P. Hanks, Word Association Norms, Mutual Information, and Lexicography, Computational Linguistics 16 (1990) 22–29.

[21] P. Schmitz, Inducing Ontology from Flickr Tags, in: Workshop on Collaborative Web Tagging (CWT 2006) collocated with the 15th World Wide Web Conference 2006 (WWW 2006), pp. 206–209. From: `http://www.semanticmetadata.net/hosted/taggingws-www2006-files/22.pdf`.

[22] R. Lau, D. Song, Y. Li, T. Cheung, J.-X. Hao, Towards A Fuzzy Domain Ontology Extraction Method for Adaptive e-Learning, IEEE Transactions on Knowledge and Data Engineering 21 (2009) 800–813.

[23] V. Pekar, S. Staab, Taxonomy Learning - Factoring the Structure of a Taxonomy into a Semantic Classification Decision, in: 19th International Conference on Computational Linguistics (COLING 2002). From: `http://acl.ldc.upenn.edu/C/C02/C02-1090.pdf`.

[24] G. Stumme, R. Taouil, Y. Bastide, L. Lakhal, Conceptual Clustering with Iceberg Concept Lattices, in: Treffen der Fachgruppe maschinelles Lernen der Gesellschaft für Informatik. From: `http://www-ai.cs.uni-dortmund.de/EVENTS/FGML2001/FGML2001-Paper-Stumme_etal.pdf`.

[25] A. Maedche, S. Staab, Measuring Similarity between Ontologies, in: 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web (EKAW 2002), Springer, 2002, pp. 251–263.

[26] P. Cimiano, A. Hotho, S. Staab, Comparing Conceptual, Divise and Agglomerative Clustering for Learning Taxonomies from Text, in: R. L. de Mántaras, L. Saitta (Eds.), 16th European Conference on Artificial Intelligence (ECAI 2004), IOS Press, 2004, pp. 435–439.

[27] K. Dellschaft, S. Staab, On How to Perform a Gold Standard Based Evaluation of Ontology Learning, in: I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, L. Aroyo (Eds.), 5th International Semantic Web Conference (ISWC 2006), volume 4273 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 228–241.

[28] Y. Park, R. J. Byrd, B. K. Boguraev, Automatic Glossary Extraction: Beyond Terminology. Identification, in: 19th International Conference on Computational Linguistics (COLING 2002). From: `http://acl.ldc.upenn.edu/C/C02/C02-1142.pdf`.

[29] R. Navigli, P. Velardi, Semantic Interpretation of Terminological Strings, in: 6th International Conference on Terminology and Knowledge Engineering (TKE 2002), Springer, 2002, pp. 95–100.

[30] J. Carrol, B. McBride, The Jena Semantic Web Toolkit, Public API, HP-Labs, Bristol, 2001. From: `http://jena.sourceforge.net/`.

[31] S. Bechhofer, A. MIles, SKOS Simple Knowledge Organization System Reference – W3C Recommendation 18 August 2009, Technical Report, W3C, 2009. From: `http://www.w3.org/TR/2009/REC-skos-reference-20090818/`.

[32] K. Toutanova, C. D. Manning, Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger, in: Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC 2000), ACM, 2000, pp. 63–70.