Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für verteilte Systeme
Fachgebiet Wissensbasierte Systeme

Forschungszentrum L3S

Diploma thesis

Discovery of Descriptive Words for Sets of Semantically Related Nouns

Author:
Thomas Theuerkauf

Examiner: **Prof. Dr. techn. Wolfgang Nejdl**
Second examiner: **Prof. Dr. Heribert Vollmer**

Supervisors:
M.Sc. Nina Tahmasebi
Dr. Thomas Risse

Hanover, 11th June 2010

Abstract

Word sense discrimination algorithms extract clusters from text where each cluster represents a single word sense or semantic meaning. However, these algorithms have shortcomings in labelling their clusters with descriptive words. Knowing these descriptive words can help to improve search or automatic detection of language evolution. In this thesis, an approach to label clusters, especially clusters extracted from historic document collections, is proposed. The approach uses semantic relations, in particular hyponymy relations, which are extracted from text with lexico-syntactic patterns. A *relation graph* is built iteratively for each cluster by extending the graph with nouns which participate in these semantic relations. Two quality aspects are defined for descriptive words, *coverage* and *specificity*. Scores are assigned to the vertices in the relation graph w.r.t. these aspects. Vertices with the highest score are chosen as the most appropriate descriptive words for the cluster. Experiments were conducted on subsets of Wikipedia. 500 clusters and their descriptive words were manually evaluated, also with descriptive words discovered with relations extracted from WordNet as a comparison. The evaluation showed that the approach provides good results as long as the amount and quality of semantic relations is sufficient.

Zusammenfassung

Um die verschiedenen Bedeutungen eines Wortes voneinander abzugrenzen, werden Algorithmen benutzt, die Wortgruppen aus freiem Text extrahieren (Word Sense Discrimination). Die Wörter jeder dieser Gruppen, auch Cluster genannt, haben eine gemeinsame semantische Bedeutung. Allerdings sind diese Algorithmen oft nicht in der Lage, ihren Clustern Wörter zuzuweisen, die diese Bedeutung beschreiben. Solche beschreibenden Wörter können zusammen mit ihren Clustern zum Beispiel dazu benutzt werden, um die Suche in Textarchiven zu verbessern oder um die Entwicklung bzw. Evolution einer Sprache automatisiert zu erkennen. Gegenstand dieser Arbeit ist das Finden solcher beschreibenden Wörter, insbesondere für Cluster, die aus historischen Textarchiven extrahiert werden. Dazu wird eine Methode vorgeschlagen, die semantische Relationen benutzt, welche mit Hilfe von Textmustern ("lexico-syntactic patterns") aus dem zugrundeliegenden Text gewonnen werden. Mit diesen Relationen wird für jeden der Cluster ein Graph (*relation graph*) erstellt, indem dieser iterativ mit Wörtern erweitert wird, die in passenden Relationen vorkommen. Es werden zwei Qualitätsaspekte für beschreibende Wörter definiert, Abdeckung (*coverage*) und Spezifität (*specificity*). Jeder Knoten des Graphen bekommt, bezogen auf diese Qualitätsaspekte, eine Punktzahl zugewiesen, die die Eignung als beschreibendes Wort für den Cluster ausdrückt. Die Knoten mit der höchsten Punktzahl werden ausgewählt, um den Cluster zu beschreiben. Experimente wurden auf Untermengen der Wikipedia durchgeführt. 500 Cluster und ihre beschreibenden Wörter wurden manuell evaluiert. Als Vergleich wurden außerdem beschreibende Wörter evaluiert, die mit aus WordNet extrahierten Relationen gefunden worden sind. Die Evaluation hat gezeigt, dass die vorgeschlagene Methode gute Resultate liefert, solange genügend Relationen mit ausreichend hoher Qualität verfügbar sind.

Contents

List of Figures

List of Algorithms

List of Tables

1. Introduction

1.1. Motivation

Semantic classes of nouns contain members which describe different objects of the same kind. For example, the semantic class *fruit* contains members such as *banana*, *apple* or *orange*. Sets which contain nouns of the same semantic class are useful for many tasks in natural language processing (NLP). This includes applications such as question answering, information retrieval and word sense disambiguation. One method to gather such sets is to use unsupervised word sense discrimination approaches. These approaches take raw text as input, detect co-occurrences of terms and apply clustering algorithms to discriminate the terms among their semantic meanings (Dorow [Dor07, DW03], Schütze [Sch98] and Pantel and Lin [PL02]). Because such sets are a result of clustering, they are also referred to as clusters.

Although these approaches are able to extract such clusters, they have shortcomings in mapping them to their associated semantic classes. For example, these methods are able to extract clusters such as {*banana*, *apple*, *orange*} but provide no information that this cluster is associated to the semantic class *fruit*. This task is also referred to as labelling of extracted clusters with the name of their associated semantic classes (Pantel and Ravichandran [PR04]). Knowing the associated classes for these clusters assist for example in building or extending resources such as semantical lexicons or ontologies for nouns. Both resources can be used to improve results of NLP applications.

Clusters may not only contain nouns of the semantic class (e.g. {*banana*, *apple*, *orange*}) but also nouns which are semantically related in another way instead. For example, the members of the cluster {*rock*, *guitar*, *concert*} are semantically related since all of them belong to *music*. However, they do not belong to the same semantic class since they are not objects of the same kind. The noun *rock* is a musical genre, *guitar* is a musical instrument and *concert* is a musical event. Clusters of these and other different types are therefore subsumed under the more

general notion **sets of semantically related nouns** in this thesis. Since members in such clusters might not belong to a common semantic class, they can not be labelled with the name of a semantic class. Instead, labels which describe the members of such clusters are better suitable. In this thesis, these labels are referred to as **descriptive words**. However, most research has been performed to label clusters containing nouns of the same semantic class with their class name. Labelled clusters have been used to derive hyponymy relations ("*class-member* IS-A *class-label*").

Previously proposed approaches to label clusters fall into two different categories. Approaches of the first category use additional information gathered from the text which has already been used to extract clusters. Approaches of this category (Pantel and Ravichandran [PR04], Jickels and Kondrak [JK06]) parse the text to extract grammatical relationships for members of the cluster (e.g. *verb-object* where a member of the cluster is a *subject*). Common affiliates of the members in these relationships are considered as possible labels (e.g. all nouns which occur as *object* for the previous example). An approach which has been proposed by Caraballo [Car99] clusters nouns in a bottom-up fashion by calculating the similarity of nouns. These clusters are labelled afterwards by exploiting the hierarchical relations between clusters and hyponymy relations extracted from text.

Proposed approaches of the second category use external knowledge sources. Most of these approaches use lexical databases such as WordNet [WN] (e.g. Widdows [Wid03], Resnik [Res99]). WordNet provides a taxonomy of nouns defined by hyponymy relations. The members of a class are identified in the taxonomy and a subsuming noun is taken as label.

This thesis concerns in particular with the task of labelling extracted clusters from historic document collections with descriptive words. These collections are a result of the evolving trend to digitize paper based documents, e.g. newspaper archives. Extracted and labelled clusters can be used to improve the accessibility of information in these archives. If found, the labelled clusters can be used to improve search for content.

One way to improve search is to give additional information about nouns in search results to users. For a noun in such results, all clusters which contain this noun can be determined. The descriptive words for these clusters can be returned to users to give a hint for possible semantic meanings of the associated noun. This method gives useful hints because each cluster represents a certain semantic meaning for its members. A descriptive word for this cluster expresses in particular this semantic

meaning. For example, a user could search for musical instruments. In the list of results, the word *rock* might appear but is not known by the user. The descriptive words which are given as additional information could be

- *music* for the cluster {*rock*, *guitar*, *concert*},
- *musical genre* for the cluster {*pop*, *rock*, *jazz*} or
- *material* for the cluster {*water*, *wood*, *rock*}.

In combination with the the original intention to search musical instruments, the user can derive that *musical genre* is the semantic meaning of *rock* in its context.

This method of giving hints to the user is especially useful for long term archives. A word can have different meanings in different periods of time. For example, the noun *rock* might have the semantic meaning *material* in all periods of time, but its meaning *musical genre* first emerged in 20th century. Over the time, meanings for a word can appear, disappear or change. This is referred to as terminology evolution. By extracting clusters for different periods of time, the returned descriptive words for search results may differ due to the time of appearance of the results. While improving search is a direct approach to improve the accessibility of long term archives, descriptive words may also be useful to track evolution in terminology.

Unsupervised word sense discrimination was successfully used by Tahmasebi et al. [TNTR10] to extract clusters from historic document collections. Tahmasebi et al. parsed 201 years of newspaper articles from The Times Archive [Tim]. Overall, Tahmasebi et al. showed that the chosen algorithm for unsupervised word sense discrimination from Dorow [Dor07, DW03] works well over the entire collection. Therefore, this thesis addresses to find descriptive words for clusters extracted using the same approach. Tahmasebi et al. also showed that WordNet covers as a dictionary on average only 53% of all terms contained by The Times Archive (without stopwords).

On the contrary, the performance of state of the art approaches to label clusters has not been investigated on historical collections. Because they have been developed for modern text, written in today's language, they can be less suitable for the task of labelling clusters extracted from historical document collections. Since external information sources such as WordNet do not cover outdated terms and domain-specific knowledge contained by long term archives, approaches which use external knowledge sources are not best suited. Approaches which belong to the first category avoid this issue by using the text which has already been used to extract clusters. However, previous approaches used either sophisticated grammat-

ical parsers to extract grammatical relations or additional information gathered during the clustering process. Approaches using such parsers should be avoided since it is not known if these parsers work well on long term archives. The approach of Caraballo [Car99] uses hyponymy relations but relies on hierarchical structures of extracted clusters to assign labels. Since the approach of Dorow [Dor07, DW03] does not provide such information, this approach is also not suitable.

1.2. Contribution of the thesis

In this thesis, an alternative graph-based approach to label clusters is proposed. Especially clusters extracted with unsupervised word sense discrimination are addressed. This approach does not rely on external information sources like WordNet or sophisticated grammatical parsers. Instead, semantic relations extracted with lexico-syntactic patterns from raw text are used. Therefore, it is better suited for historic document collections such as long term archives.

Firstly, the challenges for discovering descriptive words are identified and described. This includes different types of clusters and their quality which may affect any approach. Secondly, different types of semantic relations are described. Methods for extracting such relations from raw text with patterns are given afterwards. Thirdly, the approach itself is described. This includes the algorithm for building a so called relation graph and calculating scores for vertices in this graph. Quality aspects for descriptive words, measures for these aspects and weighting functions to combine these measures are given afterwards. Finally the approach is discussed regarding to the identified challenges.

To investigate the performance of the approach, experiments are performed on subsets of the English Wikipedia [Wik]. By using Wikipedia, it can be investigated if the approach provides good results without, or only slightly, being affected by errors in raw text. These errors can be caused by Optical Character Recognition which is used to digitize paper-based archives. Tahmasebi et al. [TNTR10] already addressed the effect of such errors to unsupervised word sense discrimination. A manual evaluation is performed for 500 extracted clusters from five subsets, each related to a category of Wikipedia. An additional experiment is performed to investigate, if the patterns also provide relations applied on text in long term archives. Therefore, every fifth year of The Times Archive [Tim], a collection of newspaper articles from 1785-1985, is parsed with the patterns to gather first statistics about extracted relations.

1.3. Outline of the thesis

This thesis continues with the following chapters:

Chapter 2 introduces basic graph-theoretic definitions and algorithms. It also introduces the unsupervised word sense discrimination approach which is used to find sets of semantically related nouns. The chapter concludes with describing the challenges which any approach to discover descriptive words has to tackle.

Chapter 3 describes how the approach proposed in this thesis tries to discover descriptive words for sets of semantically related nouns. It describes the types of semantic relations which are used as well as how they are extracted from text. Afterwards, the algorithm for building a so called relation graph is explained. The chapter continues with the calculation of scores for each vertex in the relation graph and which vertices are chosen as descriptive words. The last section of this chapter discusses how the challenges given in Chapter 2 are tackled by the proposed approach.

Chapter 4 describes the experiments which have been performed on Wikipedia and on The Times Archive. This includes how subsets of category-related pages are extracted, how appropriate clusters extracted from these subsets are chosen for manual evaluation as well as the method used for the evaluation and it's results. Additionally, it presents first statistics about the performance of the used patterns on The Times Archive. The chapter concludes with a discussion of the results and a conclusion if the proposed approach is suitable to be applied on long term archives.

Chapter 5 describes related work.

Chapter 6 provides a summary of the thesis. Afterwards, it gives a conclusion and ideas for future research.

2. Definitions and challenges

The aim of this thesis is to discover descriptive words for sets of semantically related nouns. These sets are extracted from raw text by unsupervised word sense discrimination. Because the approach proposed in this thesis and the used word sense discrimination approach both use graph theory, the first section of this chapter introduces needed graph-theoretical definitions and algorithms. Section 2.2 continues with the description of the used unsupervised word sense discrimination approach. Afterwards, this chapter concludes with the definition of descriptive words and the description of challenges for an approach to find such words in Section 2.3.

2.1. Graph theory definitions and algorithms

This section starts with giving graph-theoretic definitions in Section 2.1.1. These definitions are used by the subsequent section which describes graph-theoretic algorithms which are applied in Section 3.

2.1.1. Definitions

Each of the following definitions is based on the reference given in front of its text. The first definitions (2.1 - 2.4) concern undirected graphs which are needed in particular for the description of the used unsupervised word sense discrimination approach in Section 2.2. The subsequent definitions for directed graphs, networks and flows in networks are needed to describe the proposed graph-based approach in this thesis.

Definition 2.1 (Graph)

[BJG07, pp.18-20] An **undirected graph** (or a **graph**) $G = (V, E)$ consists of a non-empty finite set of elements $V = V(G)$ (or V_G) called **vertices** and a finite set $E = E(G)$ (or E_G) of *unordered* pairs of distinct vertices called **edges**. $V(G)$ is called **vertex set** and $E(G)$ the **edge set** of G. In other words, an edge $\{x, y\}$ is a 2-element subset of $V(G)$.

For $\{x, y\} \in E(G)$, the vertices x and y are said to be **adjacent**, i.e. x is adjacent to y and y is adjacent to x. The vertices x and y of $\{x, y\} \in E(G)$ are also called **end-vertices**. In this definition, **loops** (i.e. pairs consisting of the same vertex) or parallel edges (i.e. multiple pairs with the same end-vertices) are not allowed.

The **neighbourhood** $N_G(v)$ of a vertex $v \in V_G$ is the set of vertices adjacent to v. The **degree** $\deg_G(v)$ of a vertex v is the number of edges having v as an end-vertex, in other words the number of neighbours of v.

Definition 2.2 (Subgraph)

[Die05, p.3] A graph H is a **subgraph** of a graph G if $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$.

Definition 2.3 (Weighted graph)

[Dor07, p.23] A **weighted graph** is a graph $G = (V, E)$ with a weight assigned to each edge. This can be seen as a mapping $c : E(G) \to \mathbb{R}$. Thus, a weighted graph is a triple $G = (V, E, c)$. For $e = (x, y) \in E$, $c(e)$ is called the weight of the edge e.

Definition 2.4 (Curvature)

[Dor07, p.31],[WS98] Let $v \in V$ be a vertex of the graph $G = (V, E)$ and $N_G(v)$ the set of v's neighbours in G. There are at most

$$k_{\max} = \frac{\#N_G(v) * (\#N_G(v) - 1)}{2}$$

edges between the vertices in $N_G(v)$. Furthermore, the number of actual edges between the neighbours of v is denoted as k_{linked}. The fraction of actual edges

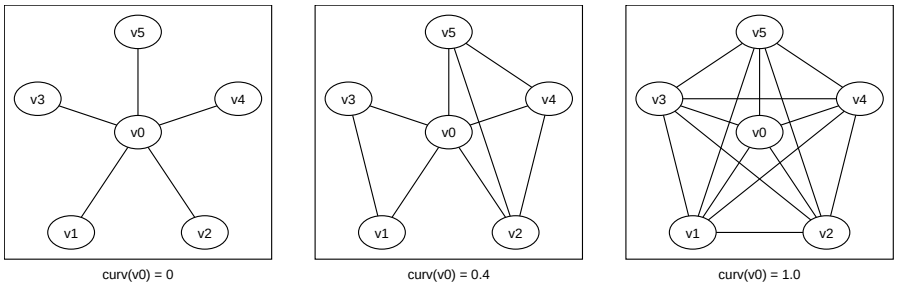


Figure 2.1.: Example for curvature. The vertex *v0* with five neighbours. Number of connected neighbours of *v0* increases from the leftmost diagram to the rightmost diagram, [Dor07, p.33].

among the neighbours of v out of the maximum possible number of such edges is called **curvature** (or clustering coefficient) and is denoted as

$$\mathrm{curv}(v) = \frac{k_{\mathrm{linked}}}{k_{\max}}.$$

Since every connected pair of neighbours of v forms a triangle together with v, the curvature can be expressed equivalently by

$$\mathrm{curv}(v) = \frac{\#(\text{triangles } v \text{ participates in})}{\#(\text{triangles } v \text{ could possibly participate in})}.$$

An example of curvature can be seen in Figure 2.1. The vertex *v0* in the leftmost diagram of Figure 2.1 has five neighbours but none of them are connected. Hence, the curvature value of *v0* is 0. In the diagram in the middle, four edges between neighbours exists out of ten possible edges which causes the curvature value of *v0* to be 0.4. In the rightmost diagram all neighbours are connected. Thus, *v0* participates in all possible triangles and hence its curvature value is 1.0.

Definition 2.5 (Directed Graph)

[BJG07, pp.2-5] A **directed graph** (or just **digraph**) $D = (V, A)$ consists of a non-empty finite set of elements $V = V(D)$ (or V_D) called **vertices** and a finite set $A = A(D) \subseteq V \times V$ (or A_D) of *ordered* pairs of distinct vertices called **arcs** instead of unordered pairs called edges for undirected graphs. $V(D)$ is called **vertex set** and $A(D)$ the **arc set** of D.

For $(x, y) \in A(D)$, x and y are said to be **adjacent**, i.e. x is adjacent to y and y is adjacent to x. The vertices x and y of $(x, y) \in A(D)$ are also called **end-vertices**. The first vertex x is the **tail** of the arc (x, y) and the second vertex y is its **head**. A vertex x is called **incident** to an arc $a \in A(D)$ if x is the head or tail of a.

The above definition of a digraph implies that arcs with the same end-vertices (for example, (x, y) and $(y, x) \in A(D)$) are allowed. Parallel arcs (i.e. pairs of arcs with the same tail and the same head) and loops (i.e. arcs whose head and tail coincide) are not allowed.

For a vertex $v \in V(D)$, the following notation for types of neighbourhood is used:

$N_D^+(v) = \{w \in V \setminus v \mid (v, w) \in A\}$	is called **out-neighbourhood**,
$N_D^-(v) = \{u \in V \setminus v \mid (u, v) \in A\}$	is called **in-neighbourhood**,
$N_D(v) = N_D^+(v) \cup N_D^-(v)$	is called **neighbourhood**.

According to the previous three definitions, types of degree are denoted as:

$\deg_D^+(v) = \lvert N_D^+(v)\rvert$	is called **out-degree**,
$\deg_D^-(v) = \lvert N_D^-(v)\rvert$	is called **in-degree**,
$\deg_D(v) = \lvert N_D(v)\rvert = \lvert N_D^+(v)\rvert + \lvert N_D^-(v)\rvert$	is called **degree**.

Definition 2.6 (Subdigraph)

[BJG07, p.5] A digraph H is a **subdigraph** of a digraph D if $V(H) \subseteq V(D)$, $A(H) \subseteq A(D)$ and every arc in $A(H)$ has both end-vertices in $V(H)$.

Definition 2.7 (Weighted directed graph)

[BJG07, p.6] A **weighted directed graph** is a directed graph $D = (V, A)$ along with a mapping $c : A(D) \to \mathbb{R}$. Thus, a weighted directed graph is a triple $D = (V, A, c)$. For $a \in A(D)$, $c(a)$ is called the **weight** of the arc a. For a set $B \subseteq A(D)$ of arcs the weight is defined as follows: $c(B) = \sum_{a \in B} c(a)$. Therefore, the **weight of a subdigraph** H of a weighted directed graph D is the sum of the weights of the arcs in H.

Definition 2.8 (Walk in directed graphs)

[BJG07, pp.11-13] A **walk** in a directed graph D is an alternating sequence $W = x_1a_1x_2a_2x_3 \dots x_{k-1}a_{k-1}x_k$ of vertices x_i and arcs a_j from D such that the tail of a_i is x_i and the head of a_i is x_{i+1} for every $i \in \{1, 2, \dots, k\}$. A walk W is **closed** if $x_1 = x_k$ and **open** otherwise. W is said to be a walk **from** x_1 **to** x_k, a walk **between** x_1 **and** x_k or an $(\mathbf{x_1}, \mathbf{x_k})$**-walk**. The set of vertices $\{x_i \mid i \in \{1, 2, \dots, k\}\}$ is denoted by $V(W)$; the set of arcs $\{a_j \mid j \in \{1, 2, \dots, k-1\}\}$ is denoted by $A(W)$. If W is open, then x_1 is called **initial** vertex and x_k is called **terminal** vertex of W. A vertex y is **reachable** from a vertex x in D, if D has an (x, y)-walk. The **length** of a walk is the number of its arcs.

Definition 2.9 (Trail, path and cycle in directed graphs)

[BJG07, pp.12-13] A **trail** W in a directed graph D is a walk in which all arcs are distinct. A trail can be **identified** by the directed graph $(V(W), A(W))$ which is a subdigraph of D. If the vertices of W are distinct, W is a **path**. If the vertices $x_1, x_2, \dots, x_{k-1}$ are distinct, $k \geq 3$ and $x_1 = x_k$, W is a **cycle**.

Since paths and cycles are special cases of walks, the **length** of a path and a cycle is already defined. The same remark is valid for other notions, e.g., an $(\mathbf{x_1}, \mathbf{x_k})$**-path**. When W is a cycle and x is a vertex of W, W is said to be a cycle **through** x.

Definition 2.10 (Shortest path)

[BJG07, pp.88-89] Let $D = (V, A, c)$ be a weighted directed graph without negative cycles. A **negative cycle** in D is a cycle W whose weight of its identifying subdigraph $c(W)$ is negative. The **length** of a walk W in a weighted graph D is defined as its weight with respect to c, i.e. the sum of the weights of the arcs of W while walking through D. The **distance from** x **to** y in D, denoted by $\text{dist}(x, y)$, is the minimum length of an (x, y)-walk, if y is reachable from x. Otherwise, $\text{dist}(x, y) = \infty$. Since D has no negative cycles, it follows that $\text{dist}(x, x) = 0$ for every vertex $x \in V$. If y is reachable from x, there are shortest (x, y)-walks (because D has no negative cycles). At least one of these shortest walks is a path (following Proposition 1.4.1 in [BJG07, pp.13-14]). This path is called the **shortest path**.

The definition of length and shortest paths are applicable to unweighted directed graphs as well. If the weight of each arc of the unweighted graph is taken equal

to 1, the length of a walk in the "weighted" and "unweighted" cases coincide. Therefore, a shortest path in an unweighted directed graph is in other words a path with the minimum number of edges.

Definition 2.11 (Network)

[BJG07, pp.127-128] A **network** extends a directed graph $D = (V, A)$ associated with the following functions on $V \times V \to \mathbb{R}$: a **lower bound** $l_{x,y} \geq 0$, a **capacity** $u_{x,y} \geq l_{x,y}$ and a **cost** $c_{x,y}$ for each $(x, y) \in V \times V$. These functions satisfy the following requirement for arcs which are not contained by D:

$$\forall (x, y) \in V \times V : (x, y) \notin A \Rightarrow l_{x,y} = u_{x,y} = 0$$

A network is denoted as $\mathcal{N} = (V, A, l, u, c)$. Since applying costs to edges is not used in this thesis, the parameter c can be omitted.

Definition 2.12 (Flow)

[BJG07, pp.128-129] A **flow** in a network $\mathcal{N} = (V, A, l, u)$ is a function $f : A \to \mathbb{R}^+$ on the arc set of $\mathcal{N}$. A value of f on the arc (x, y) is denoted as $f_{x,y}$. An **integer flow** is a flow f such that $f_{x,y} \in \mathbb{Z}$ for every arc (x, y). For a given flow f in $\mathcal{N}$ the **balance function** of f is the following function b_f on the vertices:

$$b_f(x) = \sum_{(x,y) \in A} f_{x,y} - \sum_{(w,x) \in A} f_{w,x}$$

In other words, the balance function is the difference between the flow on arcs with tail x and the flow on arcs with head x. The vertices V can be classified according to their balance values. A vertex x is a **source** if $b_f(x) > 0$, a **sink** if $b_f(x) < 0$ and otherwise x is **balanced** ($b_f(v) = 0$). A flow is **feasible** if $l_{x,y} \leq f_{x,y} \leq u_{x,y}$ for all $(x, y) \in A$.

Definition 2.13 (Residual Network)

[BJG07, pp.130-131] For a given flow f in a network $\mathcal{N} = (V, A, l, u, c)$, the **residual capacity** $r_{x,y}$ from x to y is defined as follows:

$$r_{x,y} = (u_{x,y} - f_{x,y}) + (f_{y,x} - l_{y,x}).$$

The **residual network** $\mathcal{N}(f)$ with respect to f is defined as $\mathcal{N}(f) = (V, A(f), \tilde{l} \equiv 0, r, c)$, where $A(f) = \{(x, y) \mid r_{x,y} > 0\}$. The cost function is the same as for $\mathcal{N}$ when assuming that $c_{y,x} = -c_{x,y}$. All lower bounds are zero.

The arcs of the residual network have a natural interpretation. If $(x, y) \in A$ and $f_{x,y} = 5 < 7 = u_{x,y}$, then f may be increased by up to $r_{x,y} = 2$ units on the arc (x, y) at the cost of $c_{x,y}$ per unit. Furthermore, if $l_{x,y} = 2$, then f can be decreased by up to 3 units along the arc (x, y). The cost of this decrease x is exactly $c_{y,x} = -c_{x,y}$ per unit. The decrease of a flow along the arc (x, y) can also be thought of as sending a flow in the opposite direction along the residual arc (y, x).

Definition 2.14 ((s, t)-flow)

[BJG07, p.132] Let s, t be distinct vertices of a network $\mathcal{N} = (V, A, l \equiv 0, u)$. An $(\mathbf{s}, \mathbf{t})$-flow is a flow f satisfying the following for some $k \in \mathbb{R}$:

$$b_f(v) = \begin{cases} k & \text{if } v = s, \\ -k & \text{if } v = t, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, the flow f is a flow with only one source and one sink. The **value** of an (s, t)-flow f is denoted by $|f|$ and is defined by

$$|f| := b_f(s).$$

Definition 2.15 (The maximum flow problem)

[BJG07, p.140] Let s, t be distinct vertices of a network $\mathcal{N} = (V, A, l \equiv 0, u)$. A **maximum flow** f is a feasible (s, t)-flow in $\mathcal{N}$ which has the highest possible value of all (s, t)-flows in $\mathcal{N}$. The problem of finding a maximum flow from s to t in a network with a specified source s and sink t is known as the **maximum flow problem**.

2.1.2. Algorithms

The following three algorithms are used in Chapter 3. The **Breadth-First Search** (BFS) algorithm (Bang-Jensen [BJG07, pp.92-93]) allows to find all reachable vertices from an initial vertex s in an unweighted directed graph. It also determines the distances between s and each of its reachable vertices. The **Ford-Fulkerson** algorithm (Bang-Jensen [BJG07, pp.142-143]) is an abstract approach to find maximum flows in a network. An implementation of the Ford-Fulkerson algorithm is the **Edmonds-Karp** algorithm (Bang-Jensen [BJG07, p.148]) which makes use of the BFS algorithm. In the following, $n = |V|$ denotes the number of vertices in the examined directed graph $D = (V, A)$ and $m = |A|$ denotes the number of arcs.

Breadth-First Search (BFS)

Breadth-First Search (BFS) can be used to find all reachable vertices from a given vertex s in an unweighted directed graph $D = (V, A)$. It also provides the determination of distances from s to each vertex $v \in V$. BFS is based on the following idea. Starting with the vertex s, each vertex in the out-neighbourhood of s is visited. For each of the vertices x in the out-neighbourhood the predecessor of x is set to s ($\text{pred}(x) = s$). The distance from s to x is set to 1. In the next step each vertex y in the out-neighbourhood of vertices x which have not yet been visited are visited. The predecessor of these vertices y is set to x ($\text{pred}(y) = x$). The distance from s to y is set to 2 ($\text{dist}(s, y) = \text{dist}(s, x)+1$). The algorithm continues this way until all reachable vertices from s have been visited. For all vertices z which are not reachable from s, the distance from s to z is set to infinity ($\text{dist}(s, z) = \infty$) and the predecessor is set to **nil** ($\text{pred}(v) =$ **nil**). A more formal definition of BFS is given by Algorithm 2.1. The proof for the correctness of the algorithm is given by induction and can be taken from Bang-Jensen [BJG07, pp.92-93]. Step 1 requires $O(n)$ time. The time to perform step 3 is $O(m)$ because output-neighbours of every vertex are considered only once and $\sum_{x \in V} \deg_D^+(x) = m$ (proof given by Proposition 1.2.1 in Bang-Jensen [BJG07, p.5]). Therefore, the complexity of the algorithm is $O(m + n)$.

Definition 2.16 (BFS tree)

[BJG07, p.93] Let U be the set of vertices reachable from s. Then (U, B), where $B = \{(pred(v), v) : v \in U \setminus s\}$, is a tree with root s (follows from the definition of BFS). This tree is called **Breath-First Search tree from s**.

Algorithm 2.1 Breadth-First Search algorithm

Require: Digraph $D = (V, A)$
Require: Vertex $s \in V$

```
// Step 1
for all v ∈ V do
   dist(s, v) ← ∞; pred(v) ← nil;
end for

// Step 2
dist(s, s) ← 0;
Queue Q = new Queue(); Q.enqueue(s);

// Step 3
while Q not empty do
   Vertex u ← Q.dequeue();
   for all v ∈ N_D^+(u) do
      if dist(s, v) = ∞ then
         dist(s, v) ← dist(s, u) + 1;
         pred(v) ← u;
         Q.enqueue(v);
      end if
   end for
end while

return dist(s, v) and pred(v) for all v ∈ V
```

Ford-Fulkerson

The **Ford-Fulkerson** algorithm (Bang-Jensen [BJG07, pp.142-143]) is an abstract approach to find a maximum flow, i.e. an (s,t)-flow f with the maximum value $k = |f|$, in a network $\mathcal{N} = (V, A, l \equiv 0, u)$ with the source s and the sink t. The idea is based on the *Max-Flow Min-Cut* theorem ([BJG07, pp.92-93], which states the following equivalence (i) $\Leftrightarrow$ (ii) for $\mathcal{N}$ and a feasible flow f:

(i) The flow f is a maximum (s,t)-flow.
(ii) There is no (s,t)-path in the residual network $\mathcal{N}(f)$.

Algorithm 2.2 Ford-Fulkerson algorithm

Require: Network $\mathcal{N} = (V, A, l \equiv 0, u)$;
Require: Source $s \in V$;
Require: Sink $t \in V$;

Flow $f \equiv 0$;
while $\exists$ $P = (s,t)$-path in $\mathcal{N}(f)$ **do**
  Augment f along P;
end while

return f

The algorithm starts with $f \equiv 0$. This is a feasible flow since $0 \equiv l_{x,y} \leq u_{x,y}$ for all arcs $(x, y) \in A$. The algorithm tries to find an (s,t)-path P in $\mathcal{N}(f)$. If there is such a path P, the flow f cannot be a maximum flow according to the Max-Flow Min-Cut theorem. Remembering that the residual network describes the residual capacities of the arcs of $\mathcal{N}$, the flow f can be augmented by $\delta(P)$ units along the path P. $\delta(P)$ is the capacity of the path P in the residual network $\mathcal{N}(f)$, in other words:

$$\delta(P) = \min \{r_{x,y} \mid (x, y) \text{ is an arc of } P\}$$

The exact way to augment a flow along a path is given in Bang-Jensen [BJG07, p.141]. The flow f is augmented continuing this way until there is no (s,t)-path left in the residual network $\mathcal{N}(f)$. The final flow f is a maximum flow according to the Max-flow Min-cut theorem. The Ford-Fulkerson algorithm is given in a more formal description by Algorithm 2.2.

If in the network $\mathcal{N} = (V, A, l \equiv 0, u)$ all capacities are integers, then the Ford-Fulkerson algorithm finds a maximum (s,t)-flow in $O(m|f^*|)$ time, where f^* is a maximum (s,t)-flow (see theorem 4.5.4 in Bang-Jensen [BJG07, p.143]). The factor $|f^*|$ is caused by the fact that the value of the current flow increases at least by 1 since all capacities are integers. Hence there can be no more than $|f^*|$ iterations of the search for an (s,t)-path in $\mathcal{N}$. The proof given in Bang-Jensen [BJG07, p.143] uses a labelling algorithm with complexity $O(n + m)$ to find such an (s,t)-path. Together with the assumption that $n = O(m)$, which holds for all interesting networks according to Bang-Jensen [BJG07, p.128], the overall complexity follows. Furthermore, if all capacities are integers, then the maximum (s,t)-flow is an integer flow (see theorem 4.5.5 and its proof in Bang-Jensen [BJG07, p.144]).

Edmonds-Karp Algorithm

Since no concrete method of finding an (s, t)-path in the residual network is specified for the Ford-Fulkerson algorithm, the algorithm may chose paths with only a low increase of the flow value compared to other possible paths. Thus, the complexity of the Ford-Fulkerson algorithm is not bounded by a polynomial in the size of the input but the value of a maximum flow. To overcome this issue, the Ford-Fulkerson algorithm can be modified in various ways to ensure that it becomes a polynomial algorithm.

One way is to use Breadth-First Search to find an (s, t)-path in the residual network. The flow is therefore augmented along shortest paths in the residual network with respect to the number of arcs on the path. This modification is sufficient to get a polynomial algorithm with complexity $O(nm^2)$, which has been proven by Jack Edmonds and Richard M. Karp (see theorem 4.6.4 in Bang-Jensen [BJG07, p.148]). The resulting algorithm can be seen as an implementation of the Ford-Fulkerson algorithm and is called **Edmonds-Karp** algorithm.

2.2. Unsupervised word sense discrimination

The aim of word sense discrimination is to find different senses, i.e. semantic meanings, of words present in a collection, like large text corpora. Therefore, the words present in a collection are clustered into coherent sets of semantically related words where each set represents one word sense or meaning. These sets are returned as output of word sense discrimination. Unsupervised approaches for word sense discrimination have been proposed before, for example by Dorow. [Dor07, DW03], Schütze [Sch98] and Pantel and Lin [PL02].

For this thesis, the approach proposed by Dorow and Widdows [DW03] is used. The first step is to find co-occurrences of words in a large text corpora. These occurrences are then used to build a co-occurrence graph with words as vertices and co-occurrences as edges. The graph is afterwards being clustered to derive sets of words which represent a semantic meaning. The clustering is done using the curvature values of vertices in the co-occurrence graph. The discovery of descriptive words is investigated in particular for these sets.

This section gives an overview over the approach of Dorow and Widdows and how it has been implemented. Section 2.2.1 describes how the text of a given text corpus is cleaned and lemmatized. Section 2.2.2 describes how the co-occurrence graph is build by parsing the lemmatized text for co-occurrences. Finally, Section 2.2.3 describes how the co-occurrence graph is clustered by curvature to extract sets of semantically related nouns.

2.2.1. Cleaning and lemmatizing the text

Since raw text may contain non-letter characters which may affect the following methods, they have to be filtered out in a first preprocessing step. The filtering is implemented using regular expressions. All non-regular characters are removed, except commas and dots. Commas and dots are needed for the next step of lemmatizing the text and building a dictionary.

The next step is to extract nouns out of the cleaned text by natural language processing. Therefore, the text is given to the part-of-speech tagger TreeTagger [Sch94]. This tagger is used to lemmatize the text, e.g. nouns given in plural are transformed to their lemma in singular, and to extract nouns. Extracted nouns are added to a **dictionary** containing nouns together with their associated count of occurrences in the text. The lemmatized text is afterwards given to another part-of-speech tagger, namely Lingua::EN::Tagger [Cob], to extract further nouns. This second tagger also allows to search for noun phrases, which are not used in this thesis. Nouns extracted by this second tagger are added to the dictionary as well. The lemmatized text is kept to be parsed to build the co-occurrence graph.

For the first experiments performed for this thesis, nouns with upper capital letters, such as proper nouns like person or city names, and unique nouns were filtered out of the dictionary. This was done because clusters for example containing person names are not suitable for the performed experiments in this thesis because they require a high amount of specific knowledge. Unique nouns are likely to be spelling errors and are therefore also not suitable.

2.2.2. The co-occurrence graph

The **co-occurrence graph** is a weighted graph $G = (V, E, c)$. The vertices represent terms, i.e. nouns or noun phrases, contained by the dictionary described in

the previous section. The edges represent co-occurrences in the lemmatized text. Two terms are considered to co-occur, if they are both part of a list of terms. Lists are text phrases of terms separated by an "and", an "or" or a comma. For example, the three co-occurrences *apple-orange*, *orange-banana* and *apple-banana* can be derived from the list "apple, orange and banana". Since co-occurrences are assumed to be symmetric, the co-occurrence graph is undirected. The edge weights correspond to the frequency of co-occurrences of the terms represented by the two adjacent vertices of an edge.

To build the co-occurrence graph, the lemmatized text is parsed for lists of terms contained by the dictionary. For each found list, co-occurrences are derived. Each of the derived co-occurrences is added to the graph. If an associated edge for a co-occurrence already exists, its weight is increased by 1. This is continued until the complete text has been parsed. The graph can be build simultaneously with parsing the text.

Before the graph is clustered, it is filtered by a threshold applied to the edge weights of the co-occurrence graph. Edges with a weight lower or equal to this threshold are removed. The threshold for experiments performed in this thesis has been set to 2 which has shown to provide good results in preliminary experiments (see Tahmasebi [TNTR10]).

2.2.3. Clustering by curvature

The idea of the approach of Dorow and Widdows [DW03] is to cluster vertices in the co-occurrence graph using the curvature values of vertices in the co-occurrence graph. The curvature value of a vertex can be seen as a measure for the interconnectedness between its neighbours and has been proposed by Dorow [Dor07].

Terms co-occurring in lists are assumed to share a common semantic meaning. This is represented in the co-occurrence graph by subgraphs containing vertices with a high interconnectedness among their neighbours. Ambiguous terms, i.e. terms with different semantic meanings occur in lists representing different semantic meanings. Since other terms of these different lists are not assumed to co-occur, ambiguous terms are represented by vertices with a low interconnectedness between their neighbours in the co-occurrence graph, i.e. low curvature value.

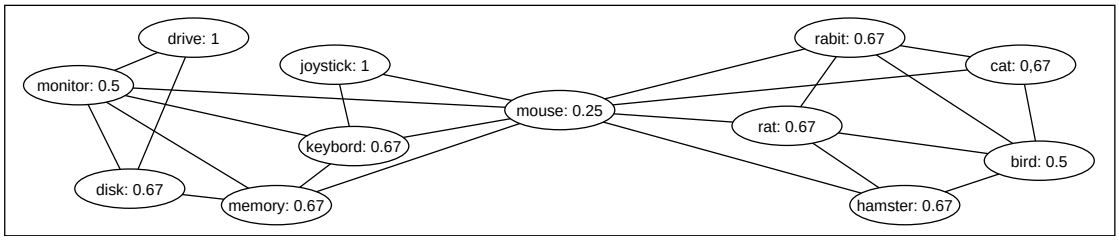


Figure 2.2.: Example for a co-occurrence graph, Dorow [Dor07, p.51]. Vertices are labelled with <*vertex name*>:<*curvature value*>. The vertex *mouse* has the lowest curvature value of 0.25. If clustering with the curvature threshold of 0.3, the graph falls apart into 2 clusters representing two different semantic meanings of mouse, namely *electronic device* and *animal*.

Since curvature can be seen as a measure for interconnectedness, vertices with a low curvature in the co-occurrence graph are assumed to represent ambiguous terms. These vertices are likely to connect parts of the graph that would otherwise not be connected. By removing vertices with a curvature value lower than a certain curvature threshold, the graph falls apart into subgraphs, which are considered to be clusters, containing vertices with a high interconnectedness. Each of these clusters is considered to represent another word sense. An example is given in Figure 2.2. When restricting the curvature threshold to 0.3, the vertex *mouse* is removed because its curvature value is only 0.25. Therefore, the co-occurrence graph falls apart into two clusters, representing the two semantic meanings *electronic device* and *animal*.

The used clustering algorithm implements this idea. As curvature threshold, the value 0.3 was taken. This threshold provided good results in preliminary experiments on long term archives (see Tahmasebi [TNTR10]). After the graph is clustered, each cluster is enriched with the nearest neighbours of its vertices. This way the clusters also capture the previously removed ambiguous terms.

2.3. Challenges for an approach to discover descriptive words

This section presents challenges involved in discovery of descriptive words for clusters containing semantically related nouns. A descriptive word for a cluster is a

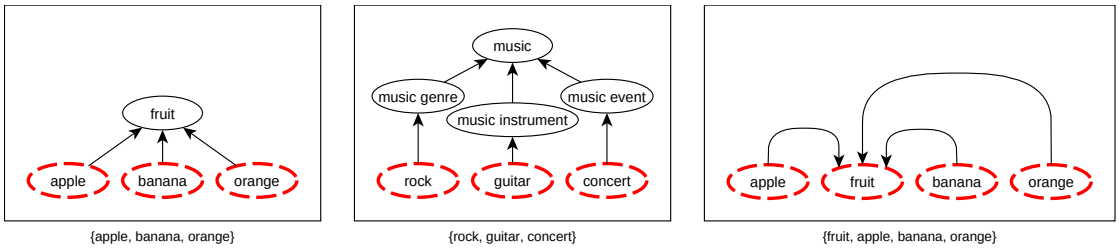


Figure 2.3.: Abstract examples for three different types of cluster. Each noun in the cluster is highlighted with dotted borders. Arcs have the meaning "belongs to semantic class". The left diagram is an example for clusters consisting of nouns being in the same semantic class; the diagram in the middle is an example for clusters with a common semantic meaning being not in the same semantic class; the rightmost diagram is an example for clusters which already contain a descriptive word.

word which "describes" the nouns in this clusters. For a cluster containing nouns of the same semantic class, a descriptive word could be the name of this class. Descriptive words can also be seen as labels for their associated clusters expressing the common semantic meaning which is shared by the nouns in a cluster.

To develop an appropriate approach to find descriptive words for a cluster, especially by using semantic relations, several challenges have to be tackled. These challenges are:

1. Discovery of descriptive words for different types of clusters.
2. Achieving best-possible appropriateness for discovered descriptive words.
3. Handling of low-quality input.
4. Keeping the computational effort reasonable.

1. Discovery of descriptive words for different types of clusters

As described in Section 1.1, there are different types of clusters which leads to the first challenge. Preliminary experiments discovered three of these types:

a. Clusters consisting of nouns being in the same semantic class,
b. clusters with a common semantic meaning not in the same semantic class,
c. clusters which already contain a descriptive word.

An example for the first type is given in the leftmost diagram in Figure 2.3. All nouns in the cluster {*apple*, *banana*, *orange*} belong to the same semantic class *fruit*. This is illustrated by an arc from each noun in the cluster to *fruit*. An arc can be seen as with the meaning "belongs to semantic class" in all three diagrams of the figure. The diagram in the middle of Figure 2.3 describes the second type of clusters. Each of the nouns in the cluster {*rock*, *guitar*, *concert*} belongs to a separate semantic class but all of them have the common semantic meaning *music*. The rightmost diagram is an example for the third type. It is possible that the descriptive word is already contained by the cluster, for example {*fruit*, *apple*, *banana*, *orange*} in this rightmost diagram. The approach should select this noun in the cluster as a descriptive word or at least provide a higher probability that this noun is selected. Giving this noun a higher probability is reasonable considering that this noun already contributes to define the meaning of the cluster and if it is exchanged by another noun, the meaning of the cluster can change completely. Exchanging the noun *fruit* to *produce* in the cluster {*fruit*, *apple*, *banana*, *orange*} of the rightmost diagram changes its meaning from an ecological to an economical meaning.

2. Achieving best-possible appropriateness for discovered descriptive words

The second challenge concerns about the appropriateness of descriptive words. Obviously, an approach should provide best-appropriate descriptive words. This thesis assumes that the appropriateness of a descriptive word depends on two quality aspects, namely *coverage* and *specificity*. An approach should consider these aspects, for example by implementing measures for them. In the following, the two quality aspects are described.

A given cluster may contain semantically unrelated nouns. Therefore, discovered descriptive words may only be descriptive for a certain portion of nouns in the cluster. To have a high appropriateness for a descriptive word, it has to "cover" as many nouns in the cluster as possible. This is the first quality aspect and is called **coverage**. To give an example, the cluster

$$\{apple, banana, orange, car\}$$

contains three semantically related nouns *apple*, *banana* and *orange* as well as the semantically unrelated noun *car*. The noun *fruit* covers three of the four nouns in the cluster and is therefore more appropriate as descriptive word than

the noun *vehicle* which only covers *car*. A low coverage of a descriptive word is not only caused by semantically unrelated nouns in the cluster. The noun phrase *round fruit* is descriptive for the nouns *apple* and *orange* but not for *banana* and *car*. This also shows, that a noun (respectively noun phrase) which is more abstract subject to the cluster probably covers more nouns in the cluster. The other way around, a noun which is more specific probably covers less nouns. However, the coverage and specificity have to be treated independently. For example, the cluster $\{apple, banana, orange\}$ has the appropriate descriptive word *fruit*. The noun *object* covers the same portion, i.e. all nouns of the cluster, but is too abstract (not specific enough) to be appropriate as descriptive word. Therefore, the second quality aspect is the **specificity** of a descriptive word.

Comparing the noun *fruit* and the noun phrase *round fruit* to be appropriate for the cluster $\{apple, banana, orange\}$ makes clear, that there is a trade-off between the coverage and the specificity. If requiring a higher coverage the need to allow a lower specificity is obvious. The other way around, if a higher specificity is required, a lower coverage has to be accepted. The weighting of the two quality aspects is an essential problem which an approach also has to tackle. However, nouns with a high coverage and a high specificity are assumed to have a high appropriateness.

3. Handling of low-quality input

The statement of the third challenge is that an approach has to handle low quality of given input, i.e. clusters and semantic relations. As described in Section 2.2, clusters are extracted by unsupervised word sense discrimination. This extraction method has the advantage of extracting clusters without human assistance. A disadvantage of this extraction method is however that clusters can contain semantically unrelated nouns since there is no quality check performed by human assessors (e.g. a cluster $\{apple, banana, orange, car\}$).

For clusters with a small portion of semantically unrelated nouns, the approach should still be able to discover descriptive words. Furthermore, since the approach for the discovery of descriptive words is also unsupervised, it should have a sufficient fault tolerance for erroneous relations. However, an approach of any kind is

not supposed to be able to find descriptive words for every cluster, since one or a mixture of the following three issues can be true for an examined cluster:

- A high portion of semantically unrelated nouns in the cluster,
- only erroneous semantic relations for nouns in the cluster,
- no or too few available semantic relations for nouns in the cluster.

4. Keeping the computational effort reasonable

Due to the possibility of a high number of nouns in a cluster and of having a high amount of semantic relations extracted from text, the computational effort has to be kept as low as possible. An approach has to limit the semantic relations used to a reasonable amount. Besides the limitation for the amount of relations to use, it has to keep the computational complexity as low as possible but without lowering the quality of discovered descriptive words.

3. Discovery of descriptive words

After getting familiar with graph-theoretical definitions and algorithms as well as the challenges for finding descriptive words for clusters, this chapter proposes a graph-based approach for the discovery of descriptive words. Therefore, the first section describes the idea of the approach and gives an overview over the complete approach. In the subsequent section the used semantic relations to build a so called relation graph and their extraction is described. Section 3.3 describes afterwards how this graph is build. In the following section, the way of calculating scores for vertices in the relation graph is described. This chapter concludes with a discussion of the approach, regarding the four challenges given in Section 2.3.

Because in this thesis in particular the discovery of descriptive words for sets of semantically related nouns extracted with the curvature curvature clustering approach given in Section 2.2 is investigated, the terms *set of semantically related nouns* and *cluster* are used interchangeably from here on. Like seen in Section 2.2, clusters can also contain noun phrases with a higher length than 1, i.e. compound nouns such as "sign of the zodiac" with length 4, if also noun phrases are added to the dictionary. However, this thesis focusses only on clusters with noun phrases of length 1 but remarks are given for the use of noun phrases for used methods. The proposed graph-based approach determines descriptive words for a single given cluster of nouns, i.e. noun phrases of length 1, in the notation:

$$C = \{noun_1, noun_2, \ldots, noun_n\}.$$

3.1. Idea of the approach and overview

In this section, the idea of the graph-based approach is given in Section 3.1.1. Section 3.1.2 gives afterwards an overview over all steps performed for this thesis and by the proposed approach on a dataset to extract clusters and descriptive words for these clusters.

3.1.1. Idea of the approach

The idea of the approach is as follows: A directed graph is build starting with the nouns of the given cluster C as vertices. New vertices and arcs are added iteratively by using semantic relations (binary, directed and transitive) extracted from raw text. Each new vertex represents a noun, respectively noun phrase. A semantic relation is represented by an arc. The size of the graph is kept reasonable by using a limit for steps of the algorithm and the out-degree of vertices. Since the out-degree is limited, relations for a noun are sorted by frequency and are used starting with the highest frequent relation until the maximum out-degree of the vertex representing this noun has been reached.

Each vertex in the graph represents a possible candidate as descriptive word. Therefore, measures for the two quality aspects *coverage* and *specificity* are applied for each vertex, including the vertices representing nouns which participate in the cluster. The values for these measures are combined by a weighting function to get a score for each vertex. The score, which describes for each vertex the appropriateness as descriptive word, is used to apply a ranking over all vertices in the graph. Vertices with the highest score are taken as descriptive words. A normalization is applied to the scores to get comparable scores over different clusters. Hence, in future applications a threshold for a minimum score is imaginable.

Since every vertex is examined separately, the vertex under examination is called the **examined vertex**. Vertices which represent nouns of the cluster are referred to as **cluster vertices**. Respectively, a single vertex representing a noun in the cluster is called **cluster vertex**. For an examined vertex, the notion **covered vertex** describes a cluster vertex which is covered by the examined vertex. This notion only concerns cluster vertices and none of the other vertices added while extending the graph. Therefore, the number of covered vertices of an examined vertex is the number of nouns in the cluster which are covered by the noun represented by the examined vertex. If a cluster vertex is in scope, the vertices in the graph which cover this cluster vertex are called **covering vertices** of the cluster vertex.

3.1.2. Overview

The complete sequence of steps done by this approach and preliminary steps for a given dataset, which is for example a subset of an examined long term archive, is

given below. The sequence already includes the discovery of descriptive words for all clusters extracted from the dataset.

1. Lemmatize the given dataset and build a dictionary of nouns / noun phrases (Section 2.2.1).
2. Extract co-occurrences of nouns / noun phrases and semantic relations out of the lemmatized text supported by the dictionary (Section 2.2.2 and 3.2.1).
3. Build the co-occurence graph (done simultaneously with step 2).
4. Extract clusters using curvature clustering applied on the co-occurrence graph (Section 2.2.3).
5. Do the following steps for each cluster:

    5.1 Build the relation graph for the cluster with the extracted relations from the dataset (Section 3.3).

    5.2 Calculate a coverage value for each vertex in the relation graph (Section 3.4.1).

    5.3 Calculate a specificity value for each vertex in the relation graph (Section 3.4.2).

    5.4 Calculate scores for each vertex in the relation graph by applying a weighting of coverage and specificity (Section 3.4.3).

    5.5 Order all vertices by their scores and select the top scored vertices. (Do a ranking of vertices)

    5.6 If more than one vertex has the highest score, take only the vertices with the highest coverage.

    5.7 If the found vertices are equal to the cluster then no descriptive word is found. Else save the residual vertices as found descriptive words.

The proposed approach can be seen as performing the steps 5.1 to 5.7 for a single given cluster C with semantic relations extracted in step 2.

3.2. Semantic relations

The first step of the proposed graph-based approach is building the so called **relation graph**. This is a directed graph which contains the nouns of the given cluster and is extended by other nouns which are semantically related to them. To build the graph, binary semantic relations which have a *direction* and *transitive property* are used. These directed and transitive semantic relations can be written as a pair of nouns (X, Y) with the direction from X to Y. The set of all these pairs, i.e. relations, is from here on referred to as the relations R. Additionally, the set of all nouns provided by the English language is from here on named L_{nouns}.

This way, R can be written as a subset of the cartesian product:

$$R \subseteq L_{\text{nouns}} \times L_{\text{nouns}}$$

This thesis focusses on two types of these semantic relations. These types are **hyponymy** ($Hyp \subseteq L_{\text{nouns}} \times L_{\text{nouns}}$) and **meronymy** ($Mer \subseteq L_{\text{nouns}} \times L_{\text{nouns}}$) relations which are both added to R ($R = Hyp \cup Mer$). The following descriptions are based on Miller et al. [MBF+90], Lyon [Lyo77a, pp. 291-295] and Cruse [Cru86, pp. 157-180]. A **concept** refers to a set of synonyms for a certain word meaning.

Hyponymy relations are "IS-A"-relations between two nouns. A hyponymy relation between the nouns X and Y exists, if native speakers of English accept the phrase "X is a Y". For these relations denoted as ordered pair (X, Y), the noun X is called **hyponym** and the noun Y is called **hypernym**. Y can be seen as a more general noun for X or as a class of X. The set of hyponymy relations is denoted as $Hyp \subseteq L_{\text{nouns}} \times L_{\text{nouns}}$.

Examples for hyponymy relations are

- "banana IS-A fruit",
- "red IS-A color" or
- "car IS-A vehicle"

also written as

$$(banana, fruit), (red, color) \text{ and } (car, vehicle).$$

Because nouns can have different semantic meanings they can be a part of more than one concept. Thus, they can participate in more than one hyponymy relation, i.e. they can have more than one hypernym. The relations $(rock, music)$ and $(rock, material)$ can be taken as example.

Hyponymy relations are *transitive* and *asymmetrical* [Lyo77a, p. 292]. To give an example for transitivity, the hyponymy relation $(banana, food)$ can be derived from the hyponymy relations $(banana, fruit)$ and $(fruit, food)$. Asymmetry can be given in the more formal description

$$\forall(X, Y) \in Hyp \Rightarrow (Y, X) \notin Hyp.$$

An example for asymmetry is a the hyponymy relation $(banana, fruit)$. The relation $(fruit, banana)$ is obviously not a hyponymy relation since native speakers of English would not accept the phrase "fruit is a banana". If nouns are differentiated among their concepts ("lexemes" in [Lyo77a]) hyponymy relations can be used to build a hierarchical semantic structure for concepts.

Meronymy relations are "PART-OF"-relations between two nouns. Similar to hyponymy relations, a meronymy relation between the nouns X and Y exists, if native speakers of English accept the phrase "X is a part of Y" or "Y has a|an X (as a part)". For these relations denoted as ordered pair (X, Y), the noun X is called **meronym** and Y is called **holonym**. The set of meronymy relations is denoted as $Mer \subseteq L_{\text{nouns}} \times L_{\text{nouns}}$. The relations

- "wheel PART-OF car",
- "musician PART-OF concert" or
- "player PART-OF team"

can be taken as examples, again also written as

$$(wheel, car),\ (musician, concert) \text{ and } (player, team).$$

Meronymy relations are also *transitive* and *asymmetrical* if qualified by their concepts (respectively their functional domain or specific context in [Cru86, p. 164]), but building a hierarchical semantic structure is more challenging since a concept can be a meronym of more than one holonym concept, which is not the case for hyponymy relations over concepts. To give an example for transitivity, the two meronymy relations $(wheel, axis)$ and $(axis, car)$ can be taken to derive the meronym relation $(wheel, car)$. The

relation (*wheel*, *car*) is also an example for asymmetry since a car is not the part of a wheel.

However, for the approach it is not necessary to build a complete hierarchical structure of the extracted semantic relations. The approach just uses the assumption that relations between nouns which are representatives of different concepts, reach over different values of abstractness. The proposed approach assumes that the more relations have to be used together with the transitive property to get from a noun $noun_1$ to another noun $noun_2$, the higher is the value of abstractness of $noun_2$ subject to $noun_1$. This becomes clear considering the chain

$$\text{banana} \rightarrow \ldots \rightarrow \text{food} \rightarrow \ldots \rightarrow \text{object}$$

In this chain, *food* and *object* are both hypernyms for *banana*, but *object* is much more abstract than *food*. To address abstractness, hypernyms and holonyms are also called **superordinates** while hyponyms and meronyms are called **subordinates** in the following text. In other words, the noun X is called subordinate of the superordinate Y for the pair (X, Y). Since building of a hierchical structure is not necessary, both relation of both types can be combined. The relations $R = Hyp \cup Mer$ can be seen as semantic relations between subordinates and superordinates. A benefit of this combination is that chains like described above can be build as a combination of hyponymy and meronymy relations.

It has to be mentioned, that this graph-based approach assumes these relations to be reflexive and that each noun in L_{nouns} is related to itself. In other words:

$$\forall X \in L_{\text{nouns}} : (X, X) \in R$$

Contrary to the assumption that both of these types of relations are asymmetric, the approach also allows symmetric relations. This is reasonable because relations are extracted from a text corpus consisting of raw text. Therefore, directed cycles of these directed relations can occur, especially caused by erroneous extracted relations. Considering the two correct relations $(banana, fruit)$ and $(fruit, food)$ together with the wrong relation $(food, banana)$, the relation $(fruit, banana)$ can be derived using the transitive property applied on $(fruit, food)$ and $(food, banana)$. Since it is not possible to determine if a relation is wrong due to lack of knowledge, especially in long term archives, the approach has to handle symmetric relations and cycles.

All definitions from above which are related to single nouns are also true for noun phrases. To give an example, the relation (*lion*, *sign of the zodiac*) is accepted by native speakers of English as a hyponymy relation, the text phrase "*lion* is a *sign of the zodiac*" holds. Therefore, hyponymy and meronymy relations can also be used for clusters consisting of noun phrases with length ≥ 1 (or also $= 1$).

The next step is to extract the hyponymy and meronymy relations from information sources. Section 3.2.1 describes a pattern-based extraction method running on raw text in large text corpora, while Section 3.2.2 deals with the extraction of relations from WordNet. Using relations from WordNet is contrary to the requirement of avoiding external information sources. But it is a way to investigate if the proposed approach provides useful results, because WordNet contains only manually added nouns and relations while automatically extracted relations can contain noise, i.e. erroneous relations, which may affect the descriptive words.

3.2.1. Relations extracted from large text corpora

Extracting semantic relations from raw text in a given text corpus has two main advantages:

1. The inclusion of domain-specific language and
2. the inclusion of outdated terms existing in documents especially contained by long term archives.

However, it has to be investigated if the method of extracting semantic relations with text patterns provides enough relations and if they are applicable to build an appropriate relation graph. For the task of finding semantic relations, the six slightly modified lexico-syntactic patterns found by Marti A. Hearst [Hea92] and one additional pattern are used in this thesis. Caused by the assumption that these patterns are affected by the evolution in terminology, a less strict placement of commas and conjunctions "and" and "or" is allowed. For example, the placement of a comma before an "and" or an "or" in a list of nouns (e.g. "*fruits* such as *bananas*, *apples*, and *oranges*") is allowed but not mandatory for all modified patterns which contain such lists. This is not allowed for the original patterns which contain such lists. All relations which can be extracted by the original patterns are extracted by the modified patterns as well.

The primary intention of using these patterns is to find hyponymy relations. However, these patterns do not only extract hyponymy relations but also a small

portion of meronymy relations (detected in preliminary experiments for this thesis) besides other semantic relations like metonymy relations [Hea92] and noise. Metonymy is a figure of speech similar to a metaphor [Lyo77b, p. 548] where the name of a concept is exchanged by another close related concept, for example (*king*, *institution*) [Hea92].

For the following six patterns used in this thesis, the original version is added below each pattern if it has been modified. The variables "NP_{hyper}" and "NP_{hypo_i}" represent hypernym, respectively hyponym noun phrases. To achieve a higher quality of resulting relations only single nouns are considered. Therefore, each of the noun phrases has length 1.

1. "NP_{hyper} {,} **such as** {NP_{hypo_1}, NP_{hypo_2} . . . (, | {,} and | {,} or)} NP_{hypo_n}"

   Original: "NP_{hyper} such as {NP_{hypo_1}, NP_{hypo_2} . . . , (and | or)} NP_{hypo_n}"

   Examples: ". . . fruits such as bananas, apples and oranges . . ."
   ". . . fruits, such as bananas, apples, or oranges . . ."
   ". . . fruits such as bananas or apples . . ."

2. "**such** NP_{hyper} **as** {NP_{hypo_1}, NP_{hypo_2} . . . (, | {,} and | {,} or)} NP_{hypo_n}"

   Original: "NP_{hyper} as {NP_{hypo_i}, }* {(and | or)} NP_{hypo_n}"

   Examples: ". . . such fruits as bananas, apples and oranges . . ."
   ". . . such fruits as bananas, apples, or oranges . . ."
   ". . . such fruits as bananas . . ."

3. "NP_{hypo_1} {, NP_{hypo_i}}* {,} **or other** NP_{hyper}"

   Examples: ". . . bananas, apples, oranges or other fruits. . ."
   ". . . bananas, apples, oranges, or other fruits. . ."
   ". . . bananas or other fruits . . ."

4. "NP_{hypo_1} {, NP_{hypo_i}}* {,} **and other** NP_{hyper}"

   Examples: ". . . bananas, apples, oranges and other fruits. . ."
   ". . . bananas, apples, oranges, and other fruits. . ."
   ". . . bananas and other fruits . . ."

5. "NP_{hyper} {,} **including** {NP_{hypo_1}, NP_{hypo_2} . . . (, | {,} and| {,} or)} NP_{hypo_n}"

   Original: "NP_{hyper} {,} including {NP_{hypo_i}, }* {(and | or)} NP_{hypo_n}"

   Examples: ". . . fruits, including bananas, apples, and oranges . . ."
   ". . . fruits including bananas, apples or oranges. . ."
   ". . . fruits, including apples . . ."

6. "NP_{hyper} {,} **especially** {NP_{hypo_1}, NP_{hypo_2} . . . (, | {,} and|| {,} or)} NP_{hypo_n}"

   Original: "NP_{hyper} {,} especially {NP_{hypo_i}, }* {(and | or)} NP_{hypo_n}"

   Examples: ". . . fruits, especially bananas, apples, and oranges . . ."
   ". . . fruits especially bananas, apples or oranges. . ."
   ". . . fruits, especially apples . . ."

Additionally, the following own pattern is used:

7. "NP_{hypo_1} **is** (**a** | **an**) NP_{hyper}"

   Examples: ". . . orange is a fruit . . ."
   ". . . water is a liquid . . ."
   ". . . zeppelin is an airship . . ."

Since the raw text given as input is lemmatized as described in Section 2.2.3, also the patterns have to be in a lemmatized form. This is already the case for the patterns 1-4 and 6, while pattern 5 is extended by "(include|including)" instead of "including" and pattern 7 by "(be|is|are)" instead of "is".

It can be assumed that all nouns in matching text phrases for a pattern are given in singular since the input is lemmatized. Thus, nouns can be taken to create relations without modification. Hence, if a text phrase matches one of the patterns above, one hypernym ("NP_{hyper}") and a set of hyponyms ($\{NP_{\text{hypo}_i} \mid 1 \leq i \leq n\}$) can be derived according to the position of the noun phrases in the matching text phrase and the matched pattern. All possible combinations of the hypernym with

a hyponym in $\{NP_{\text{hypo}_i} \mid 1 \leq i \leq n\}$ can be added to the relations R. To give an example, the following lemmatized text phrase matches pattern 1:

"... fruit such as banana , apple and orange ..."
"... NP_{hyper} such as NP_{hypo_1}, NP_{hypo_2} and NP_{hypo_3} ...".

The following relations can be added to R:

$$(\textit{banana}, \textit{fruit}),\ (\textit{apple}, \textit{fruit}),\ (\textit{orange}, \textit{fruit})$$
$$(NP_{\text{hypo}_1}, NP_{\text{hyper}}),\ (NP_{\text{hypo}_2}, NP_{\text{hyper}}),\ (NP_{\text{hypo}_3}, NP_{\text{hyper}}).$$

To find matchings for the patterns in the given text, word based deterministic finite state machines (DFAs), one for each pattern, are used. Every read word is given to each of the DFAs before proceeding with the next word. The computational complexity of this approach is

$$O(\#\text{patterns} * n) = O(n), \quad n = \#\text{words} \quad \text{(approach takes linear time)}$$

since a DFA works in linear time $O(n)$. The method of giving a read word to each DFA before proceeding has the advantage that this word has to be read only once. Considering that reading the text from the files in a given text corpus takes most of the parsing time, this is an appropriate method to parse large text corpora. In addition to the fast processing, the possibility to add or remove patterns just by implementing another DFA and integrating it parallel to the other DFAs is another advantage.

To give an example, Figure 3.1 presents the state diagram for the DFA which is used to implement the search for text phrases matching pattern 3 ("or other"). The DFA is defined as a 5-tupel $(Q, \Sigma, \delta, q_0, F)$ consisting of

- the states $Q =$ { START,
  LIST_NOUN_READ,
  LIST_COMMA_READ,
  OR_AFTER_LIST_READ,
  OTHER_AFTER_LIST_READ,
  ACCEPT }

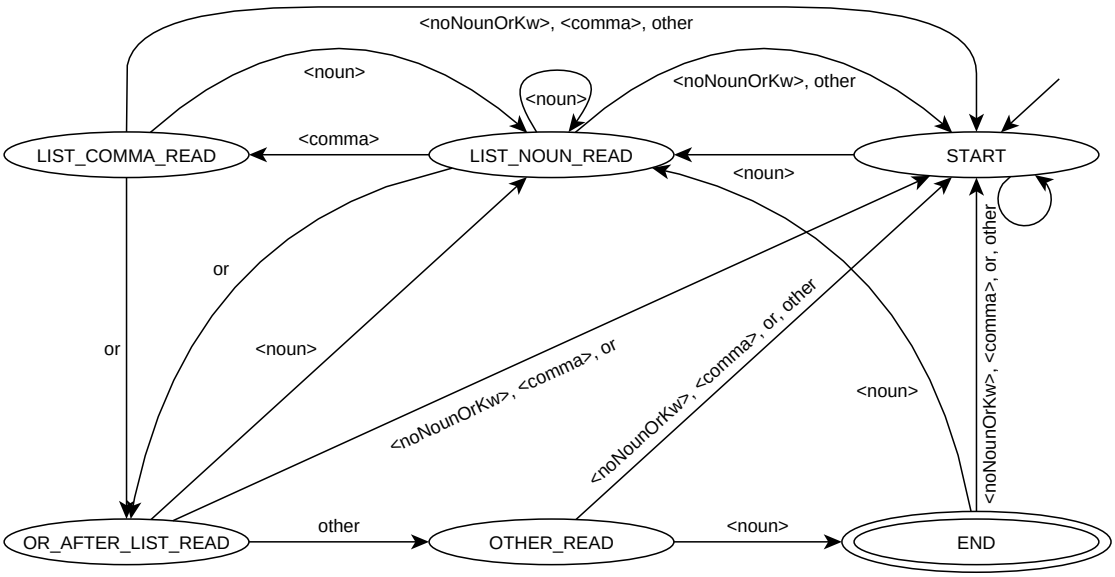


Figure 3.1.: DFA for pattern 3 ("or other"): "NP_{hypo_1} {, NP_{hypo_i}}* {,} or other NP_{hyper}". The states are represented by vertices given as ellipses, the end state with a double ellipse. A single arc without a tail marks the start state $q_0 = \text{START}$. The transition function δ is given by the arcs.

- the input symbols $\Sigma = \{$ <noun>,
  <noNounOrKw>,
  <comma>,
  or,
  other $\}$

- the transition function $\delta : Q \times \Sigma \to Q$, defined by the state diagram in Figure 3.1

- the start state $q_0 = \text{START}$

- the end state $F = \text{ACCEPT}$

While reading the text, the dictionary created when lemmatizing the text is used to identify the last word (or words for noun phrases with length ≥ 1) as a noun. If it is a noun, the input symbol "<noun>" is given as input symbol to the DFA. If the noun is part of the list in front of the phrase "or other" it is saved in a list

of possible hyponyms, which is later used to create semantic relations. Otherwise, if the noun is behind the phrase “or other” and the list of possible hyponyms is not empty, hyponymy relations are created like described above by combining the last read noun as hypernym with each hyponym in the list of possible hyponyms. This happens every time the DFA reaches its end state “ACCEPT”.

If the read word is not a noun, it is checked for being a comma, “or” or “other”. In this case, it is directly given as input symbol to the DFA. Otherwise, the input symbol “<noNounOrKw>” is given to the DFA. Because the keywords “or” and “other” can erroneously be contained by the dictionary, each assumed noun is additionally checked. If it is one of the keywords, it is given as keyword to the DFA, not as a noun. The same method is applied for all patterns based on keywords.

It has to be pointed out, that the given lemmatized text is not preprocessed or modified before it is parsed. Modifications according to Marti A. Hearst [Hea92] could be the removal of adjectival quantifiers like “some” or “other” especially before hyponym noun phrases, for example the quantifier “some” in

“... eggs, flour, *some* milk or other ingredients ...”

and comparatives such as “important” or “smaller”, for example “important” in

“... flour and other *important* ingredients ...”.

The amount of possible modifications depends amongst other things on the domain of the given text. To remove the comparatives like “important” before “ingredient” is less critical in the domain of cooking than removing such comparatives in a biological domain. To give an example, the noun phrase *lesser panda* describes an animal which has not the same mammal genus as *giant panda*. Because it can not be assumed that the domain is known and the possible coverage of several different domains by documents in long term archives lead to the decision to parse only the unmodified text.

From another point of view, such modifications would allow more text phrases to match with one of the patterns, but probably with a lower quality of resulting relations. This describes a trade-off between the quantity and the quality of found relations. Since a vast amount of text is available, the method above extracts less but hopefully enough relations with a higher quality. This trade-off is also the reason for the decision to consider only noun phrases of length 1. The extraction of noun phrases with the length ≥ 1 can result in a higher portion of extracted

erroneous relations. This is caused by more erroneous noun phrases in the generated dictionary during the lemmatizing step. On the other side, a lot of possible pattern matchings are missed out due to the restriction of noun phrases to the length 1.

Since parsing for co-occurrences can be done in the same way as parsing the text for the seven patterns from above, this is done at the same time with another DFA for the pattern "NP_1 {, NP_i} {(and|or)NP_n}".

3.2.2. Relations extracted from WordNet

Unsupervised extraction of semantic relations from a given text corpus can result in erroneous relations. To investigate the quality of the proposed approach, there is a need for testing the approach with relations of a high quality. Therefore, the proposed approach is also tested with relations extracted from WordNet [Fel98] for which nouns and relations were added manually. Thus, a high quality of these relations is assured.

Nouns contained by WordNet are organized in so called synonym sets (**synsets**), representing a certain word meaning. These synsets can also be seen as the concepts mentioned in Section 3.3. Each noun can participate in several synsets due to its possible polysemy. These synsets are ordered from most to least frequently used, with the most common sense numbered 1. This ordering is determined by frequency of use in semantically tagged corpora (Fellbaum [Fel98, 112]). Hyponymy and meronymy relations are both defined to be semantic relations between two synsets.

For the experiments, WordNet 3.0 is used. This version of WordNet contains 117.798 nouns assigned to 82.115 synsets. Since every noun can participate in more than one synset, there are 146.312 noun-synset pairs (noun-sense pairs) available (statistics contained by the documentation in [WN]). For these synsets, 97.666 hyponymy relations and 22.187 meronymy relations are available. Statistics are given by the user's guide of the used Java WordNet interface [Fin], the number of hyponymy relations is the sum of the pointers "hypernym" and "hypernym_in", for the number of meronymy relations "holonym_mem", "holonym_prt" and "holonym_sub".

To extract hypernyms and holonyms for a given noun from WordNet, the first step is to find all synsets in which this noun participates. The determined synsets are

already ordered by their frequency of use. For these ordered synsets, hypernym and holonym synsets are extracted from WordNet. From each of these hypernym and holonym synsets, the first noun is taken as representative of the synset to be the hypernym, respectively holonym to the given noun. This is reasonable when considering that all words in a synset are synonyms and have the same word meaning.

For example, the noun *banana* participates in the synsets

$$\{banana\} \text{ and } \{banana, banana\_tree\}.$$

The first synset is the most frequent synset. Thus, hypernym and holonym synsets are extracted for this synset first. Such a hypernym synset is $\{edible\_fruit\}$. From the extracted synset, the noun *edible_fruit* is taken as a representative of the extracted synset and is used to build the hyponymy relation $(banana, fruit)$. This is done for all extracted hypernym and holonym synsets of $\{banana\}$. Afterwards, hypernym and holonym synsets of the less frequent synset $\{banana, banana\_tree\}$ are extracted. Such a synset is $\{herb, herbaceous\_plant\}$ which results in the hyponymy relation $(banana, herb)$ using the first noun *herb* as representative of the extracted cluster.

To build the relation graph, the relations are extracted from WordNet while building the graph and not forehand. With the extraction method described in the last two paragraphs, WordNet can directly be taken as the relations R. Because synsets are extracted in order of their frequency, together with the assumption that more frequent nouns participate in more instances of semantic relations, the frequencies for semantic relations are given implicitly. For example, the relation $(banana, edible\_fruit)$ is assumed to have a higher frequency than the relation $(banana, herb))$ since the synset $\{banana\}$ is more frequent than $\{banana, banana\_tree\}$. It is not necessary to know the exact value for the frequency since the algorithm to build the relation graph which is described in Section 3.3 (Algorithm 3.1) only needs an ordering for the relations.

It should be pointed out that only nouns are added as hypernyms or meronyms to the relation graph, not whole synsets. For every step of Algorithm 3.1 and for a hypernym which has been added to the relation graph in the last step, all synsets for this hypernym are extracted from WordNet and hyponym as well as holonym synsets for all of these extracted synsets are considered. If whole synsets would be added, then only hyponym and holonym synsets for a last added synset would be considered. This would result in a stricter hierarchy in the relation graph but uses

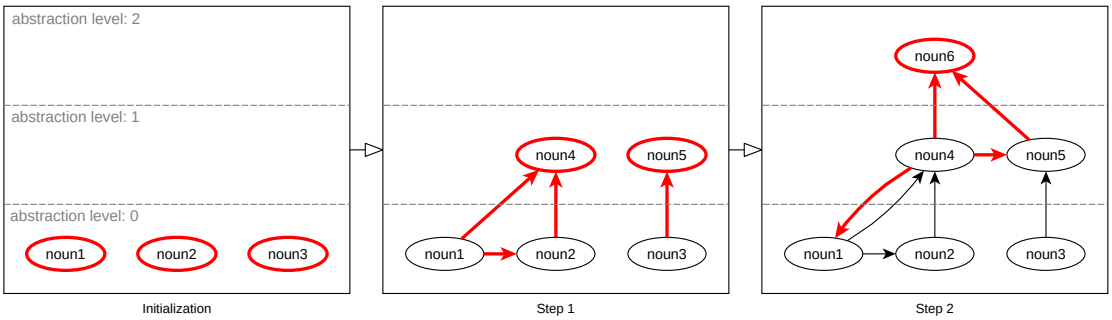


Figure 3.2.: Illustration of the algorithm for building the relation graph with initialization and two steps of the algorithm (see Algorithm 3.1). New vertices and edges added in the current step of the algorithm are highlighted.

the information of the semantic meaning of a subordinate, i.e. hyponym synset or meronym synset. The semantic meaning of nouns is not available for nouns participating in relations extracted with text patterns from a given text corpus. The information in which synsets a noun participates, i.e. its semantic meaning, and the disambiguation of nouns subject to these synsets is not investigated in this thesis. Therefore, the information contained by the hierarchical structure of synsets can be seen as external information which makes the approach with using WordNet relations less comparable to the intended approach with using relations extracted from the given text corpus.

3.3. Building the relation graph

After specifying the types of semantic relations to be used (hyponymy and meronymy relations) and how they are extracted, the procedure of building the relation graph can be described. Algorithm 3.1 explains this procedure in pseudo code, while Figure 3.2 illustrates the initialization of the graph ("Initialization") and two steps of the graph building procedure of the algorithm ("Step 1" and "Step 2"). New vertices and arcs are highlighted.

The algorithm first puts the nouns of the given cluster C as vertices into a new created directed graph D. In the following steps, the relations given by R are used to extend the graph. For each noun in the graph which has not been processed in a previous step, the relations R are being browsed for relations containing this noun as subordinate, i.e. hyponym or meronym. The relations found are sorted

Algorithm 3.1 Algorithm to build the relation graph

Require: Cluster $C = \{noun_1, noun_2, \ldots, noun_n\}$;
Require: Relations $R = \{(noun_\mathrm{X}, noun_\mathrm{Y}) \mid noun_\mathrm{X}$ is related to $noun_\mathrm{Y}\}$;
Require: Frequencies F =Map<$R, \mathbb{R}^+$>;
Require: *maxAS* "Maximum number of steps of the algorithm"
Require: *maxODV* "Maximum out-degree of vertices in the graph"

```
// Initialize G with nouns from C as first vertices
Vertices V_G = C;
Arcs A_G = ∅;
Directed graph D = new Graph(V_G, A_G);

// Extend D by adding related nouns to V_G
for transitionLevel = 1 to maxAS do
  V_lastAdded ← {noun ∈ V_G | noun ∉ processedVertices};
  for all noun_X ∈ V_lastAdded do
    N_related ← {noun_Y | (noun_X, noun_Y) ∈ R, noun_X ≠ noun_Y};
    N_sorted ← sortByFrequencyDesc(N_related, F);
    for all noun_Y ∈ N_sorted do
      if deg+_D(noun_X) ≥ maxODV then
        break;
      end if
      if noun_Y ∉ V_G then
        V_G ← V_G ∪ {noun_Y};
      end if
      if (noun_X, noun_Y) ∉ A_G then
        A_G ← A_G ∪ {(noun_X, noun_Y)};
      end if
    end for
    processedVertices ← processedVertices ∪ {noun_X};
  end for
end for

// Return the relation graph
return D;
```

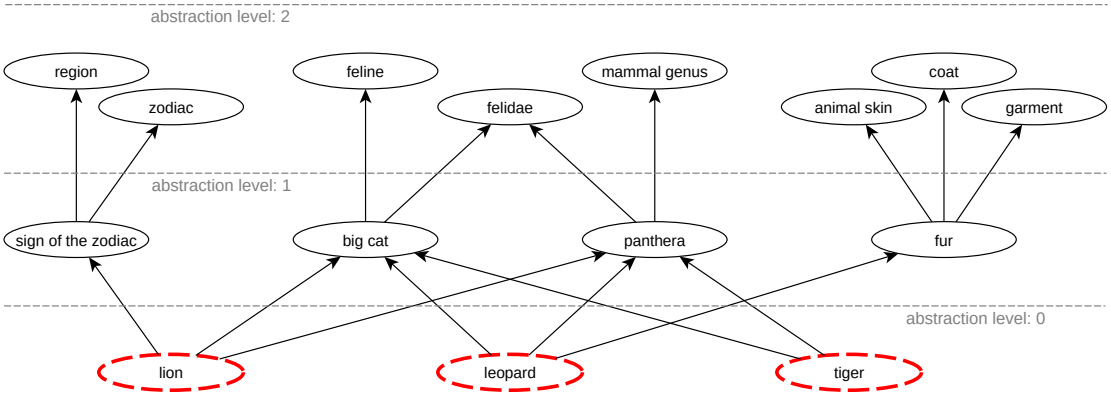


Figure 3.3.: Shortened relation graph example for the cluster $\{lion, leopard, tiger\}$. Build with Algorithm 3.1 and semantic relations extracted from WordNet 3.0 [Fel98, WN]. $maxAS = 2$; $maxODV = 3$.

by their frequencies F. The frequencies are derived by counting their occurrences in the extraction source. Symmetric relations are removed to prevent unnecessary loops in the graph. From these sorted relations a maximum number, given by $maxODV$, or less if not enough relations are available, is taken to extend the graph. This is done by adding the superordinates of these relations as vertices to the graph if the vertices are not already contained by D. Arcs are added from the subordinates to their related superordinates if they are not already connected in the same direction. Together with the limitation of steps given by $maxAS$, the result is the finite directed relation graph D.

When presenting a relation graph like in Figure 3.2, nouns in the cluster are drawn as vertices on the baseline while other nouns are drawn on higher levels associated to the number of steps the algorithm needs to add them. These levels are called **abstraction levels** and are characterized by the length of the shortest path in the relation graph between a cluster vertex and a vertex on the given abstraction level. Nouns on a higher abstraction level are assumed to be more abstract than nouns on lower levels.

An example of a shortened relation graph build with relations of WordNet 3.0 [Fel98, WN], two algorithms steps ($maxAS = 2$) and a maximum out-degree for vertices of 3 ($maxODV = 3$) is given in Figure 3.3. The nouns of the cluster $\{lion, leopard, tiger\}$ are highlighted in the graph. It can be seen that there are a few nouns / noun phrases, i.e. vertices, in the relation graph which are appropri-

ate as descriptive words for the cluster. These vertices are *big cat* and *panthera* on abstraction level 1 and their three superordinates *feline*, *felidae* and *mammal genus* on abstraction level 2. The appropriateness of these five nouns / noun phrases is expressed by the fact that all of them cover all nouns in the cluster. All other vertices cover only one noun in the cluster which indicates that these other vertices are only descriptive for one noun in the cluster. Another example is given by Figure A.4, which presents a relation graphs with relations extracted from Wikipedia and a relation graph with relations extracted from WordNet for the cluster {*spandex*, *nylon*, *lycra*}.

3.4. Calculation of scores for vertices in the relation graph

The next step after building a relation graph for a given cluster is calculating scores for the appropriateness for each vertex in the graph according to Section 3.1. Section 3.4.1 discusses the measure for coverage and how it is calculated. A measure for specificity is given in Section 3.4.2 and discussed in the same way. After the calculation of values for coverage and specificity with these measures, they have to be combined by a weighting function. Section 3.4.3 proposes two different weighting functions which can be adjusted by weighting parameters.

3.4.1. Measuring the coverage of a vertex

The coverage of a descriptive word indicates the proportion of nouns in the cluster which are "covered" by this descriptive word. The notion "covered" means formally that a directed chain of relations from the noun in the cluster to the descriptive word can be build with relations in R. In other words, the direct relations from the noun in the cluster to the descriptive word is in the transitive closure of R. Therefore, the coverage of a descriptive word expresses for how many nouns in the cluster the descriptive word is descriptive.

As described in Section 3.1.1, the notion of coverage is reduced to the relation graph. Due to the possibly vast amount of relations in R, the reason for the reduction to the relation graph is the high computational effort to build the transitive closure of R. With this reduction, the coverage is defined against the cluster C and its associated relation graph D with its set of vertices V_D and $C \subseteq V_D$.

An examined vertex in the relation graph covers a cluster vertex if there is a path from the cluster vertex to the examined vertex. Therefore, the notation in textual form is:

$$v_1 \in V_G \text{ covers } v_2 \in C \text{ in (the relation graph) } D$$

This notation includes the limitation of the coverage notion to cluster vertices. A vertex $v \in V_D \backslash C$ can not be covered. Therefore, a covered vertex is always also a cluster vertex. Furthermore, a cluster vertex always covers itself. This is implied by the assumption that every noun is related to itself, described as reflexivity of the chosen types of semantic relations in Section 3.3.

Because the coverage describes the proportion between the covered nouns and all nouns in the cluster, the measure for coverage is given as the percentage of covered nouns, denoted as **coverage function**

$$\begin{aligned} &\text{cov} : V_G \to \mathbb{R}^+ \cap [1/\#C,\ 1]\,, \\ &v \mapsto \text{cov}(v) = \frac{\#\left\{v_C \in C \mid v \text{ covers } v_C \text{ in } D\right\}}{\#C} \end{aligned} \tag{3.1}$$

where $\#C$ is the number of participating nouns in C. An example is illustrated with the left graph in Figure 3.4. In the given example, the vertices *vegetable*, *fruit* and *food* have the following coverage values:

$$\begin{aligned} \text{cov}(\textit{vegetable}) &= \frac{2}{5} = 0.4 \quad \text{for the vertex } \textit{vegetable}, \\ \text{cov}(\textit{fruit}) &= \frac{3}{5} = 0.6 \quad \text{for the vertex } \textit{fruit} \text{ and} \\ \text{cov}(\textit{food}) &= \frac{2+3}{5} = 1.0 \quad \text{for the vertex } \textit{food}. \end{aligned}$$

Furthermore, each cluster vertex has the coverage value

$$\frac{1}{5} = \frac{1}{\#C} = 0.2$$

since each cluster vertex only covers itself. The vertex *food* is the vertex with the highest coverage value in this example, because it covers all cluster vertices. The paths in the relation graph from the cluster vertices to the vertex *food* are routed over the vertices *fruit* and *vegetable*.

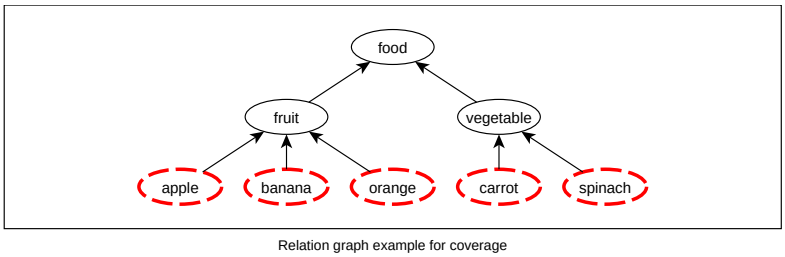


Figure 3.4.: A simple example relation graph for coverage. The vertex *food* has the highest coverage value 5/7; the vertex *fruit* has the coverage value 3/5; the vertex *vegetable* has the coverage value 2/5.

The definition of the relation graphs implies, that *every* vertex has at least the coverage value $1/\#C$. Each vertex is either a cluster vertex covering itself or has been added to the graph because it is related to at least one noun in cluster over one or more semantic relations in R, which are added as arcs to D. Because of this, the coverage function (3.1) has a range from $1/\#C$ to 1.0.

Cluster vertices can also have a higher coverage value than $1/\#C$. This is the case if there is at least one path in the relation graph from one cluster vertex to another cluster vertex. A higher coverage for cluster vertices can be seen in the example given by the left diagram in Figure 3.5. In this more complex example, the cluster vertex *vertex1* has the coverage value 3/5 since it covers the vertices *vertex2* (path: *vertex2* $\rightarrow$ *vertex6* $\rightarrow$ *vertex1*), *vertex3* (path: *vertex3* $\rightarrow$ *vertex6* $\rightarrow$ *vertex1*) and itself.

To calculate the coverage value of an examined vertex with the coverage function (3.1), the number of cluster vertices and the number of covered nouns of the examined vertex is needed. While the first number is given implicitly, the determination of the latter number is more complicated. This is caused by possible intersections of subordinates of two superordinates. In a hierarchy without intersections, the number of covered nouns could be calculated recursively by simply summing up the number of covered words like seen in the relation graph given by Figure 3.4.

As defined in Section 3.3, symmetric relations between two nouns and directed cycles in the relation graph are allowed. Furthermore, according to Algorithm 3.1 no loops, i.e. reflexive relations, are contained by the relation graph. An abstract relation graph example with cycles and intersections of subordinates is given in the left diagram of Figure 3.5. In the right hand diagram presenting the same graph, the examined vertex *vertex10* and its covered vertices are highlighted. Paths from

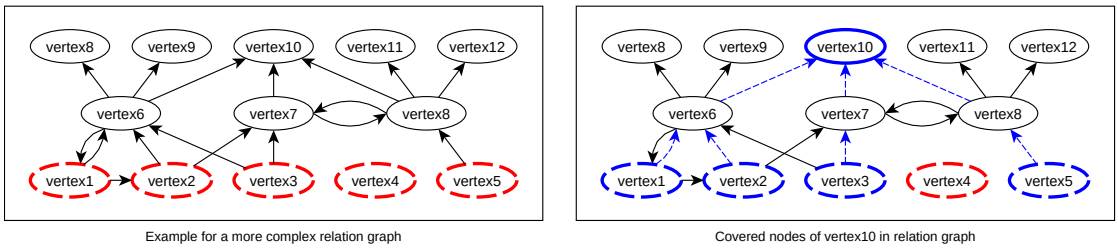


Figure 3.5.: Complex relation graph with cycles and intersection of subordinates. In the left diagram, cluster vertices are highlighted. In the right hand diagram, the covered vertices of vertex "vertex10" are highlighted. Paths between covered vertices and "vertex10" are highlighted with dashed lines.

cluster vertices to the examined vertex are highlighted, too. A directed cycle exists between the vertices *vertex1*, *vertex2* and *vertex6* in a counter-clockwise direction. The vertex *vertex2* is an example for a subordinate which has more than one, namely the two superordinates *vertex6* and *vertex7*. Hence, the number of covered words for the vertex *vertex10* as a superordinate of *vertex6* and *vertex7* can not be calculated by summing up the number of covered vertices of each subordinate *vertex6*, *vertex7* and *vertex8*. This example for a relation graph is used to illustrate the following approaches and their algorithms. It is also used in later chapters as an example for specificity and calculated vertex scores.

This thesis proposes two approaches to determine the number of covered vertices to calculate the coverage. Both approaches are appropriate to determine the covered vertices and not only their number for an examined vertex in the relation graph. They differ in their computational complexity and flexibility for future purposes. The approaches are

- Calculation of covered vertices by creating a Breadth-First Search tree
- Calculation of covered vertices by finding a maximum flow

The first approach is perfectly suited to determine the covered vertices with respect to the computational complexity and is described in this section. The second approach has a higher computational complexity and requires therefore more computational effort. However, this approach has a higher flexibility for future purposes. Therefore, it is described in Section A.1. It is not possible to derive cov-

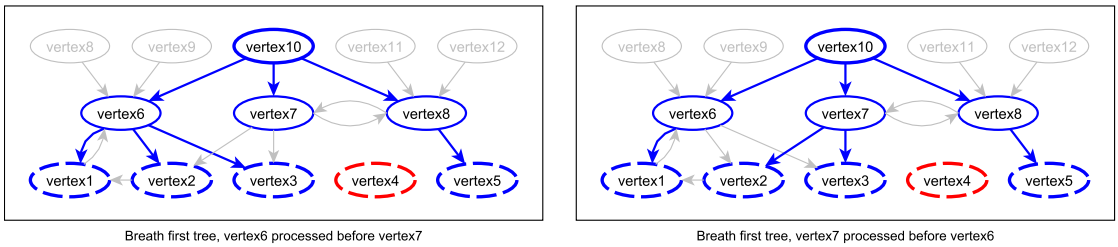


Figure 3.6.: Possible Breadth-First Search trees for the relation graph example in Figure 3.5. For the tree in the left diagram, the vertex *vertex6* has been visited before *vertex7*. The other way around is given in the right hand diagram.

ered vertices by generating this information during the graph building procedure. The reason for this is given in Section A.2.

Calculation of covered vertices by creating a Breadth-First Search tree

This approach to calculate the number of covered words is fast and at the same time very simple. It runs on the already build relation graph. The only needed modification of the relation graph D is inverting its arcs, i.e. for each arc the head and the tail are exchanged. The result of this modification is the **modified relation graph** D'. Afterwards, a Breadth-First Search tree is build starting with an examined vertex as root in D'. The Breadth-First Search algorithm (Algorithm 2.1), which is used to build the Breadth-First Search tree, terminates for every vertex in the relation graph as root vertex, because there are only finitely many vertices in the relation graph and each vertex is visited one time at most.

For this approach using a Breadth-First Search tree, the inversion of arcs is mandatory. Starting without the inversion of arcs and a cluster vertex as root vertex, the resulting Breath-First Search tree would contain all covering vertices of the cluster vertex in the relation graph. On the contrary, the tree built out of the modified relation graph with the examined vertex as root vertex contains all covered cluster vertices of the examined vertex. Additionally, it contains the vertices on shortest paths between the examined vertex and the cluster vertices in the modified relation graph (see definition of Breath-First Search in Section 2.1.2). Therefore, the covered vertices of the examined vertex can be determined by calculating the intersection of cluster vertices and vertices in the breath first search tree.

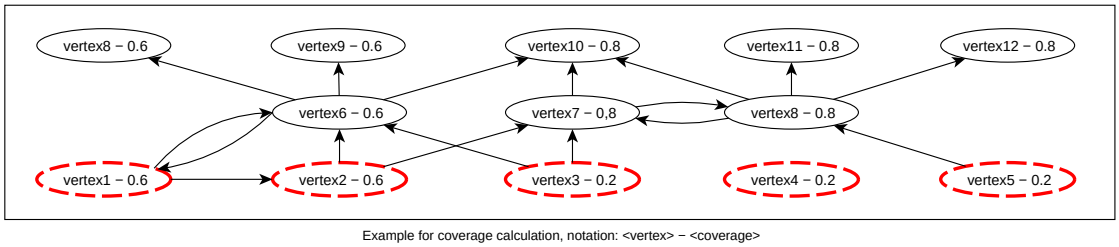


Figure 3.7.: Example for calculated coverage values of the previously introduced example for a relation graph in Figure 3.5. Vertices are labelled with <*vertex name*> - <*coverage value*>.

Since a Breadth-First Search tree can be build in $O(m+n)$ (see Section 2.1.2), it provides results faster than the maximum flow approach described in Section A.1. As a secondary result, the Breadth-First Search always returns a tree with the fewest number of arcs between the root vertex and all other vertices which are reachable in the modified relation graph. Furthermore, there is no need to count the arcs to determine this fewest number of arcs because the distance between the root vertex and each reachable vertex is already determined by the Breadth-First Search algorithm.

Built Breadth-First Search trees are not unique. A different tree structure is caused by different possible processing-orders of vertices. In Figure 3.6 two possible breadth-first trees are presented. While *vertex6* is processed before *vertex7* for the left tree, the other way around is the case for the right hand tree. However, building the tree with Breadth-First Search ensures that paths from the root vertex to each other vertex in the tree are possible shortest paths in the modified relation graph.

While this approach with Breadth-First Search is perfectly suited for a relation graph without weighted edges, it is not as flexible for future purposes as the maximum-flow approach. However, for

- clusters containing a high number of nouns,
- a high number of abstraction levels in the relation graph and
- an unlimited out-degree of vertices in the relation graph

it supports a good method to derive covered words and shortest-path lengths.

Figure 3.7 contains all coverage values for the example relation graph introduced in Figure A.1. It can be seen that more than one vertex has the maximum coverage value of 0.8 over all vertices in the relation graph, namely the vertices *vertex7*, *vertex8*, *vertex10*, *vertex11* and *vertex12*. This points out again that the coverage alone is not sufficient. The vertices *vertex3*, *vertex4* and *vertex5* have the lowest coverage value 0.2 because they only cover themselves.

3.4.2. Measuring the specificity of a vertex

The calculation of a reasonable value for the specificity of an examined vertex in the relation graph is challenging. This is especially caused by technical issues like cycles in the relation graph and different distances between the examined vertex and its covered vertices. To encounter these issues, an appropriate method is to define a measure defined on the shortest-path lengths from the examined vertex to its covered vertices. These shortest-path lengths have either already been calculated when determining the coverage value for the examined vertex or have to be calculated in this step.

If the Breadth-First-Search approach has been used to derive a coverage value for the examined vertex, the shortest paths length are already known. Distances, i.e. shortest-path lengths, between the examined vertex and all covered vertices are already determined by the Breadth-First algorithm. For the maximum-flow approach, shortest paths for the examined vertex are already contained by the routing of the flow when using appropriate algorithms like the Edmonds-Karp algorithm. The shortest-path lengths between the examined vertex and a covered vertex can be determined by counting arcs while tracing back the path between them in the routing of the flow.

If the shortest paths are not available, they can be determined by a slightly modified Breadth-First Search with the covered vertex as root vertex and the examined vertex as searched vertex (or the other way around if the arcs are inverted, i.e. if a modified relation graph similar or the same as the one for approaches to determine a coverage value is used). The modified search terminates if the searched vertex has been visited. An alternative is using more sophisticated algorithms like the algorithm of Dijkstra [BJG07, pp.94-95]. Using this algorithm would allow to assign weights to the arcs, for example depending on frequencies of associated relations.

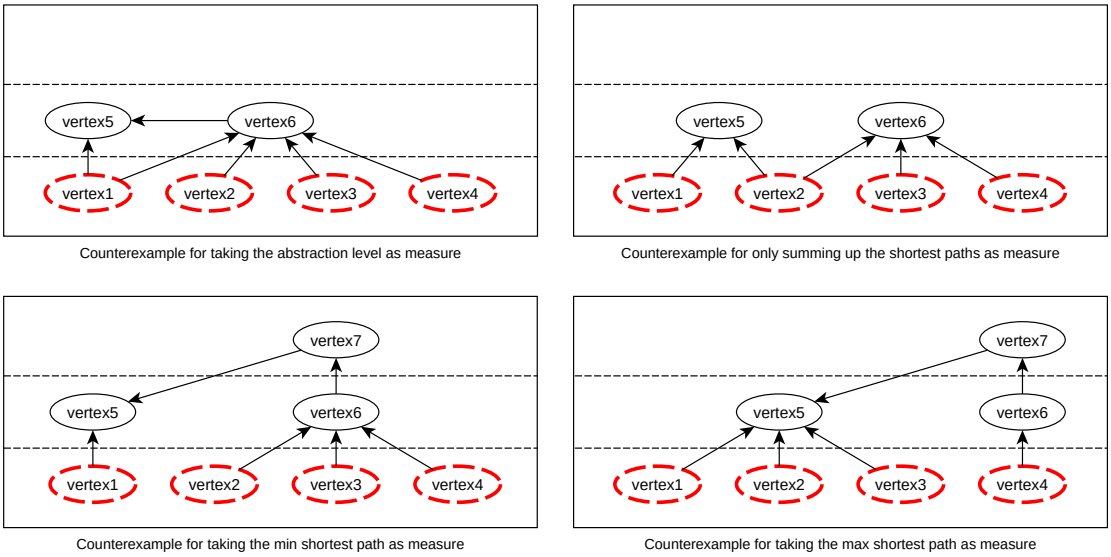


Figure 3.8.: Challenges for the specificity measure. Four different counterexamples for the first four approaches to determine a specificity value.

The shortest-path lengths can be used in different approaches. The following five possible approaches are discussed in the following text and the last of them is taken as the best suited one. These are

1. taking the abstraction level as a measure,
2. taking the shortest path with minimum length as a measure,
3. taking the shortest path with maximum length as a measure,
4. taking the sum of shortest paths as a measure,
5. taking the average of shortest paths as a measure.

The notion of specificity is strongly related to the notion of abstractness. This thesis assumes, that the specificity increases exactly as much as the abstractness decreases. Values between 0 and 1 are taken to express the level of specificity and abstractness. The following equation describes the relationship between specificity and abstractness:

$$specificity = 1 - abstractness \tag{3.2}$$

1. Taking the abstraction level as a measure

The most obvious approach is to take the abstraction level of the examined vertex and to normalize it by division by the highest possible number of abstraction levels defined by the maximum number of steps of the algorithm to build the relation graph. An abstractness value is the result, which can be transformed into the specificity with equation (3.2). The disadvantage of this approach is that all vertices on the same abstraction level have the same specificity value. Therefore it is not detailed enough to be a good measure. An example is given in the upper left diagram of Figure 3.8. In this example, the vertices *vertex5* and *vertex6* would have the same specificity value, but *vertex5* is connected over *vertex6* to the cluster vertices (except to *vertex1* which causes *vertex5* to be on the first abstraction level) and should have a higher specificity than *vertex6*.

2. Taking the shortest path with minimum length as a measure

Another approach is to take the shortest path with minimum length of the examined vertex over all paths to covered vertices as a measure. This measure describes how near the examined vertex is to the cluster. This is the same as taking the abstraction level as a measure, because the length of a shortest path with minimum length for a vertex on an abstraction level is exactly the number of the abstraction level, i.e. the step of the algorithm to build the relation graph in which all vertices on this level have been added. Therefore, it has the same disadvantage to be not detailed enough. An additional example is given in the bottom left diagram of Figure 3.8. It can be seen that the majority of covered vertices of *vertex5* have shortest paths of the length 3 and only one covered vertex, namely *vertex1*, which is covered over the shortest path with minimum length 1. Therefore it is also not appropriate.

3. Taking the shortest path with maximum length as a measure

Instead using the approach of preferring vertices on low abstraction levels, another approach is to penalize vertices with long shortest paths. This can be done by taking the maximum length over all shortest paths as a measure instead of taking the shortest path with minimum length like it has been done in the previous two approaches. This approach is also not suitable caused by the same reason as for the last approach. If the majority of shortest paths is short, taking the shortest path

with maximum length does not reflect the complete situation. An example is given in the bottom right diagram of Figure 3.8. Measuring *vertex5* by its shortest path with maximum length 3 is not appropriate due to the other shortest-path lengths of 1. Besides this, a disadvantage is, that the length of the maximum shortest path can not be normalized easily. This is caused by allowing arcs to connect vertices on the same abstraction level or even from a higher abstraction level to a lower one. Hence, there can be shortest paths with maximum length which have a higher number of participating arcs than the maximum number of abstraction levels.

4. Taking the sum of shortest paths length as a measure

The previous inappropriate approaches lead to the conclusion that considering only one of the shortest paths cannot provide a good measure. Therefore, another possible approach is to sum up the lengths of all shortest paths of the examined vertex. This approach is also not appropriate due to the different number of covered vertices for different examined vertices. An example is given in the top right diagram of Figure 3.8. The two vertices *vertex5* and *vertex6* should have the same specificity. By summing up the shortest paths, the vertex *vertex6* gets another specificity than the vertex *vertex5*.

5. Taking the average of shortest paths length as a measure

This last approach combines the need to consider all shortest paths with considering the proportions of different shortest paths lengths. It sums up all shortest-path lengths and neutralises its disadvantage described for approach 4, caused by different numbers of covered vertices for different examined vertices, by dividing this sum by the number of covered vertices. Therefore, a high number of shortest paths with a low length can compensate a few shortest paths with a high length and the other way around. The resulting value describes the not normalized abstractness of the examined vertex and can be transformed after being normalized into the value for specificity by applying equation (3.2).

The normalization of the resulting average to a value between 0 and 1 is difficult due to the consideration made for the third approach above. Shortest paths can be longer than the number of abstraction levels. One way to normalize is to divide the sum by the the absolute maximum length of a shortest path. This maximum length depends on the size of the relation graph, especially the number of ver-

tices. Since every vertex participates only one time at most in a shortest path, the maximum length of a shortest path is $|V_G| - 1$ at most. This normalization has a disadvantage which this thesis wants to avoid. It depends on the number of vertices in the relation graph which is different for each cluster. For this reason, calculated normalized specificity values are not comparable over two different clusters anymore.

Another way of normalizing is dividing by the maximum number of abstraction levels, i.e. maximum number of steps performed by the algorithm to build the relation graph (Algorithm 3.1), first. This number is identical for all clusters and dividing by the same number keeps the specificity values comparable. To take the maximum number of abstraction levels is reasonable because vertices on the highest abstraction level have only shortest paths with the minimum length of the number of abstraction levels. Hence, for all vertices on this abstraction level, the normalized abstractness value is ≥ 1. To keep the abstraction level ≤ 1, the approach assumes that all of these vertices on the highest abstraction level and vertices with an even higher abstractness value are already too abstract to be considered and sets the upper bound to 1. Besides, the according specificity value would be negative if higher abstractness values would be allowed (see Equation 3.2).

The disadvantage of this approach is that vertices on the highest abstraction level are lost by assuming that they are already to abstract. This can be compensated by increasing the maximum number of abstraction levels, i.e. maximum number of steps performed by the algorithm to build the relation graph, by 1.

The **specificity function** which is using this approach can be denoted as follows. The maximum number of abstraction levels, i.e. maximum number of steps performed by the algorithm to build the relation graph $maxAS$, is given.

$$\text{spec} : V_G \to \mathbb{R}^+ \cap [0, 1]$$

$$\begin{aligned} v \mapsto \text{spec}(v) &= 1 - \min\left(1, \frac{\varnothing(\text{Shortest path lengths for all cov. vertices})}{maxAS}\right) \\ &= 1 - \min\left(1, \frac{\sum_{u \in \text{covVertices}(v)} \text{shortestPathLength}(u, v)}{\#\text{covVertices}(v) \cdot maxAS}\right). \end{aligned} \quad (3.3)$$

It has to be mentioned, that cluster vertices are assumed to be have the highest possible specificity for themselves. Therefore, the length of the shortest path

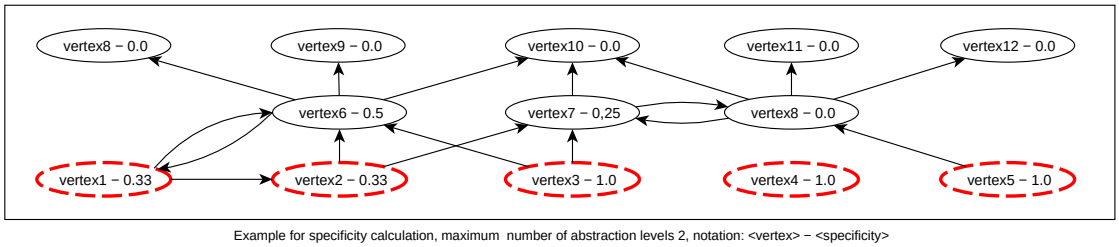


Figure 3.9.: Example for calculated specificity values. Vertices are labelled with <*vertex name*> - <*specificity value*>.

shortestPathLength(v, v) for $v \in C$ is defined to be 0 (also according to Section 2.1.1, definition for shortest path) which evaluates to a specificity value of 1.0 (see Equation 3.2) if the cluster vertex only covers itself.

Figure 3.9 shows the introduced example from Section 3.4.1 with calculated specificity values. When considering the vertices *vertex2* and *vertex7* it becomes clear again, that also the specificity alone is not sufficient. The vertex *vertex2* has a higher specificity and covers two cluster vertices, namely *vertex2* and itself, while the vertex *vertex7* has a lower specificity and covers four cluster vertices. Caused by the low maximum number of abstraction levels, the vertex *vertex8* has a specificity value of 0.0, since all of its three shortest paths have a higher or equal length than the number of maximum abstraction levels.

The calculation of the specificity for some selected vertices in Figure 3.9 with a maximum number of abstraction levels of 2 is as follows:

$$
\begin{aligned}
\text{spec}(\textit{vertex2}) &= 1 - min\left(1, \frac{(0+1+3)/3}{2}\right) &&= 0.33 \\
\text{spec}(\textit{vertex4}) &= 1 - min\left(1, \frac{0/1}{2}\right) &&= 1.0 \\
\text{spec}(\textit{vertex7}) &= 1 - min\left(1, \frac{(2+1+1+2)/4}{2}\right) &&= 0.25 \\
\text{spec}(\textit{vertex12}) &= 1 - min\left(1, \frac{(4+3+3+2)/4}{2}\right) &&= 0.0
\end{aligned}
$$

3.4.3. The scoring function

The last two sections described the two used measures for the quality aspects coverage and specificity for each vertex in the relation graph. Both of them are not sufficient alone to measure the appropriateness of vertices, they need to be combined. A method used for combination needs to consider that there is a trade-off between the two quality aspects. If a high coverage is required, then a lower specificity has to be accepted, because more abstract vertices cover more cluster vertices. The other way around, if a higher specificity is required, then a lower coverage has to be accepted. An appropriate descriptive word should be sufficient enough for both aspects. Depending on whether the coverage or specificity is more important, a specific weighting function is needed to handle this trade-off.

In general, each possible **weighting function** is defined as a 2-dimensional function with a range between 0 and 1. It takes the normalized coverage and normalized specificity as arguments:

$$\begin{aligned} &\text{weighting} : [0,1] \times [0,1] \to [0,1] \\ &(\mathit{coverage}, \mathit{specificity}) \mapsto \text{weighting}(\mathit{coverage}, \mathit{specificity}). \end{aligned} \tag{3.4}$$

This thesis proposes two different concrete weighting functions, each with two additional parameters $w_{\text{cov}}, w_{\text{spec}} \in \mathbb{R}^+$ to weight the given values for coverage and specificity:

1. Weighting given by power and product

$$\text{weightingPP}_{w_{\text{cov}},w_{\text{spec}}}(\mathit{cov}, \mathit{spec}) = \mathit{cov}^{w_{\text{cov}}} \cdot \mathit{spec}^{w_{\text{spec}}} \tag{3.5}$$

2. Weighting given by weighted arithmetic mean

$$\text{weightingAM}_{w_{\text{cov}},w_{\text{spec}}}(\mathit{cov}, \mathit{spec}) = \frac{w_{\text{cov}} \cdot \mathit{cov} + w_{\text{spec}} \cdot \mathit{spec}}{w_{\text{cov}} + w_{\text{spec}}} \tag{3.6}$$

The scoring function can be seen as an overall quality measure for both quality aspects. For each of the vertices in the relation graph the score is calculated with this function. The last step of the approach is to order the vertices by their score and choosing the vertices with the highest score. The **scoring function** is defined

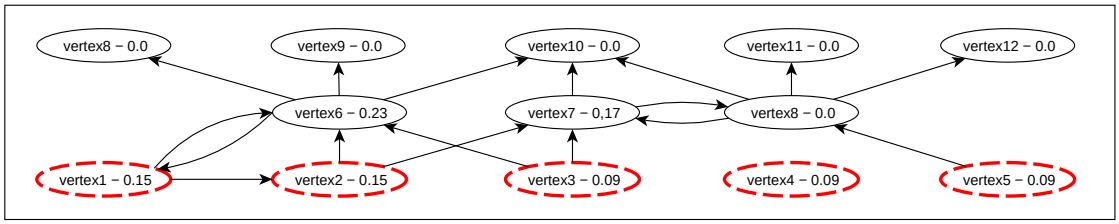


Figure 3.10.: Example for calculated scores for the previously introduced example. Vertices are labelled with <*vertex name*> - <*score*>.

as function with a range between 0 and 1, taking a vertex as argument and using a weighting function to calculate the score:

$$\begin{aligned} &\text{score} : V_G \to [0,1] \\ &v \mapsto \text{score}(v) = \text{weighting}(\text{cov}(v), \text{spec}(v)). \end{aligned} \tag{3.7}$$

In Figure 3.10 the previously introduced example is given with calculated scores. To calculate the scores, the weighting function weightingPP 3.5 with weighting parameters $w_{\text{cov}} = 1.5$ and $w_{\text{spec}} = 1.0$ has been taken. In preliminary experiments, these parameter values provided the best results. Comparing the vertices *vertex6* an *vertex7* illustrates the trade-off between coverage and specificity. While *vertex7* covers four cluster vertices, *vertex6* only covers three. Caused by the lower specificity (see Figure 3.9) of *vertex7*, the vertex *vertex6* has nevertheless a higher score. The calculation of the scores of some selected vertices is as follows:

$$\begin{aligned} \text{score}(\textit{vertex2}) &= (3/5)^{1.5} \cdot (0.33)^1 &&= 0.15 \\ \text{score}(\textit{vertex4}) &= (1/5)^{1.5} \cdot (1.0)^1 &&= 0.09 \\ \text{score}(\textit{vertex6}) &= (3/5)^{1.5} \cdot (0.5)^1 &&= 0.23 \\ \text{score}(\textit{vertex7}) &= (4/5)^{1.5} \cdot (0.25)^1 &&= 0.17 \\ \text{score}(\textit{vertex10}) &= (4/5)^{1.5} \cdot (0.0)^1 &&= 0.0 \end{aligned}$$

3.5. Discussion of the approach

In the last section of this chapter, the four challenges for an approach to discover descriptive words, which have been given in Section 2.3, can be discussed.

This proposed approach tackles the first challange of finding descriptive words for different types of clusters in the following way:

a. Clusters consisting of nouns being in the same semantic class.

   For this type of clusters, all cluster vertices are assumed to be connected with its semantic class, i.e. representing vertex, by only one arc. This vertex is supposed to have a high coverage and specificity and therefore a high score. Therefore, this vertex is chosen by the ranking which is performed after all scores have been calculated.

b. Clusters with a common semantic meaning not in the same semantic class.

   Cluster vertices of clusters with a common semantic meaning being not in the same semantic class are connected over one or more arcs to a vertex in the graph, representing the common semantic meaning of the cluster. This vertex has a high coverage and is assumed to have a higher specificity than other vertices with the same number of covered vertices. Therefore, this certain vertex should be one of the vertices in the graph with the highest score.

c. Clusters which already contain a descriptive word.

   The last type of clusters, which already contain a descriptive word, are tackled by allowing paths between two different nouns in the cluster (paths between cluster vertices). Since cluster vertices take also part in the ranking of vertices by score, they can be chosen as descriptive words. Because cluster vertices have the highest possible specificity value for themselves, they are likely to be one of the vertices with the highest score in the ranking because the average shortest-path length is decreased.

The second challenge, achieving the highest appropriateness for discovered descriptive words, is tackled by applying a ranking over the scores of vertices. Since the score is based on measures for the two quality aspects coverage and specificity,

the approach should provide descriptive words with the highest appropriateness respect to the limitations of the relation graph for the number of abstraction levels and out-degree of vertices. If two vertices have the same score, only the vertices with the highest coverage are chosen (step 5.6 in the overview in Section 3.1.2). This is reasonable assuming that the coverage is more important than the specificity after all. Therefore, an additional ranking by the coverage for all vertices with the same highest score is applied. If the discovered descriptive words for a cluster are equal to the cluster, then none of the cluster vertices share a reachable vertex in the relation graph. This can be caused by the three reasons for the disability to find descriptive words for every cluster given in Section 2.3. For example, for a cluster containing only semantically unrelated nouns it is likely that no common reachable vertex can be found till the maximum abstraction level is reached. For this reason, each vertex in the relation graph would have the coverage value $1/\#C$. The vertices only differ in their specificity value. Therefore, only the cluster vertices have the highest score with a specificity value of 1.0 and the approach would deliver the cluster as discovered descriptive words, which is not acceptable. Thus, this case is filtered out by labelling the cluster with the information that no descriptive word has been found (step 5.7 in the overview in Section 3.1.2). However, since not all available semantic relations are used to extend the relation graph and semantic relations can be erroneous, the resulting discovered words may differ from human expectations. This is investigated in Chapter 4.

To be less affected by erroneous semantic relations, semantic relations are used in the order of their frequency. This is reasonable due to the assumption that erroneous relations occur less often than correct relations. Furthermore, semantically unrelated nouns in a given cluster only cause a lower coverage value for all covering vertices of cluster vertices which are semantically related. Therefore handling of low-quality input is considered (third challenge).

The last challenge concerning the computational effort is mainly tackled by keeping the size of the relation graph reasonable. This is done with the limitations for the maximum number of steps allowed for Algorithm 3.1 to be performed and the maximum out-degree of vertices in the graph. Using the Breadth-First Search approach to determine covered vertices enables the approach to discover descriptive words even for clusters with a high number of contained nouns due to its low computational complexity of $O(m + n)$. Because shortest-path lengths have not to be calculated since they have already been determined by the Breadth-First Search, the approach is as fast as it can be for the calculation of the specificity.

4. Experiments and Evaluation

This chapter describes and discusses experiments and evaluations performed for this thesis. Section 4.1 starts with describing the used dataset and with giving statistics for extracted clusters and semantic relations. Afterwards, Section 4.2 describes the performed evaluation and and its results. Additional experiments performed on long term archives are described in Section 4.3. Finally, all results are discussed in the last section of this chapter.

4.1. The dataset

Because of the high quality and vast amount of text contained by articles in the English Wikipedia [Wik], it was chosen as dataset for experiments. The high quality is a result of the collaboration of multiple authors for each article. Due to the vast amount of text, subsets of reasonable size were extracted. Articles in these subsets are related to a combination of categories or their subordinate categories. Subsets for five category combinations (separate categories in a combination are divided by an emphasized "and" in the following list), were extracted, namely

1. Technology *and* Applied sciences,
2. Sports,
3. Religion *and* Belief,
4. Food and drink,
5. Geography.

Clusters and semantic relations were extracted for each subset separately. To find descriptive words for clusters extracted from a subset, only semantic relations of the same subset were used. The method used for the extraction of subsets is given in the next section. Statistics over extracted subsets, clusters and semantic relations are presented in Section 4.1.2.

4.1.1. Extraction of reasonable subsets

A subset consists of articles which are related to one or two categories and their descendants in the category hierarchy of Wikipedia. To extract a subset, a category tree was build using this hierarchy and the chosen categories. For subsets related to two categories ("Technology *and* Applied sciences", "Religion *and* Belief"), a generic root node was added, which subsumes only these two categories. For subsets which related to only one category ("Sports", "Food and drink", "Geography"), the related category itself was used as root node.

Starting from the root node, the category tree was extracted from the category hierarchy of Wikipedia for each subset. All categories with their hierarchy and articles were extracted from the complete dump of the English Wikipedia generated in February 2010. This dump was given as XML-file and was processed with the Perl module Parse::MediaWikiDump [Rid]. Breadth-First Search was used to extract a Breadth-First Search tree from the root node in the category hierarchy. This Breadth-First Search tree was taken as category tree. The top 30 categories for each subset in the order of their visit by the Breadth-First Search algorithm are presented in Table A.1.

Due to possible cycles in the category hierarchy of Wikipedia, the hierarchy is not strict. Therefore, the depths of the category trees were limited. The maximum depth, $maxDepthCategoryTree$, is subset-dependent and one larger for trees starting from a generic root node. Furthermore, categories in the category hierarchy with too many children were ignored, i.e. with a number of subcategories greater than $\theta_{subCats}$. The threshold $\theta_{subCats}$ is also subset-dependent. Categories with a high number of children are in particular lists of subcategories or high level categories in the category hierarchy.

A category in Wikipedia may belong to more than one parent category. Since only categories which are sufficiently related to their root category should be considered, the category tree was filtered by relatedness. The relatedness of a category to its root category was measured in percent. This percentage expresses the portion of parent categories which are contained by the category tree among all parent categories. Categories with a relatedness value lower than a threshold θ_{catRel} (again subset-dependent) were removed from the category tree. Because removing categories from the category tree affects the relatedness measure for other categories, the removal of categories was done iteratively. All categories were examined in each step in the order of their distance to the root category in the category tree.

Subsets / Thresholds	$maxDepthCategoryTree$	$\theta_{subCats}$	θ_{catRel}	θ_{artRel}	$maxNrArticles$
Technology and Applied sciences	6	51	0.3	0.3	150k
Sports	7	51	0.3	0.3	150k
Religion and Belief	4	51	0.3	0.3	150k
Food and drink	6	51	0.2	0.2	150k
Geography	3	unlimited	0.2	0.2	150k

Table 4.1.: Thresholds and limitations to extract subsets from Wikipedia. The subset *Technology and Applied sciences* is a combination of the categories *Technology* and *Applied sciences*, *Religion and Belief* is a combination of the categories *Religion* and *Belief*.

The removal procedure terminated if there were no categories which have been removed in the actual step.

After extracting and filtering the category tree, articles were extracted by examining the categories to which they belong. Each article with a relatedness value above a threshold (θ_{artRel}) was added to the subset. The relatedness value is the percentage of categories which are in the category tree among all categories which the article belongs to. Furthermore, the number of articles for each subset was limited to 150k articles to keep the subsets reasonable in size.

The used thresholds and limitations are given in Table 4.1. They were determined by preliminary experiments. It can be seen that the threshold for the relatedness of categories θ_{catRel} was chosen equal to the threshold for the relatedness of articles θ_{catRel} for all subsets. With this equality, reasonable results were provided. However, the equality of both thresholds is not mandatory.

Before a subset can be parsed for co-occurrences and semantic relations to extract clusters and to discover their descriptive words, the text has to be cleaned from the markup language used in Wikipedia articles. The markup language, namely MediaWiki markup language, was first transformed into HTML by applying the Perl module Text::MediawikiFormat [Pri]. Remaining scattered markup phrases and all HTML-tags and HTML-entities (e.g. "&" for "&") were removed by applying regular expressions. Additionally, all non-letter characters were removed the same way, except numbers and important chars, such as punctuation marks. Finally, all unnecessary white spaces were removed.

Subset / Objective	#Categories in category tree	#Extracted articles	Token count	Physical size
Technology and Applied sciences	30,672	150,000	124,969,992	763M
Sports	47,242	150,000	75,413,063	456M
Religion and Belief	5,564	69,387	52,082,706	316M
Food and drink	4,526	60,312	28,207,474	173M
Geography	2,428	53,707	24,195,138	155M

Table 4.2.: Statistics for extracted subsets from Wikipedia. The subsets are ordered by their physical size. For the two subsets *Technology and Applied sciences* and *Sports*, the maximum number of articles was reached.

4.1.2. Statistics on the subsets

The above presented method to extract subsets provided subsets of reasonable size. Statistics are given in Table 4.2. It can be seen that the number of categories in the category tree differs over the subsets. For example, the subset *Sports* has a number of categories in the category tree which is 19 times higher than the number for the subset *Geography*. This indicates structural differences between category branches in Wikipedia. For example, in the category branch for *Sports*, *tournament/year* categories are created and each player participating in the tournament for this year is assigned to that category.

The subsets differ not only in their structure of categories. When comparing the token counts (the number of terms parted by a space character) it can be seen that the average number of words per article also differs between categories. For example, articles in the subset *Technology and Applied sciences* have an average number of words per article of $124,969,992/150,000 \approx 833$ words while articles in the subset *Sports* have an average number of $75,413,063/150,000 \approx 503$ words.

Since the subsets differ in size, the numbers of extracted relations differ, too. This can be seen in Table 4.3. For example, the subset *Technology and Applied science* has a total of 229,816 relations which is approximately 49,2% of the total number of relations extracted from all subsets. However, it can be seen that the distribution of extracted semantic relations over the different patterns is similar among the subsets. The maximum deviation of a pattern-subset combination from the pattern-total percentage is 6.88 percentage points for Pattern 7 ("is a") and the subset *Sports*. The deviations for other subset-pattern combinations from the percentages for all relations extracted with the associated pattern are smaller.

Subset / Pattern	Pattern 1 "*NP* such as"	Pattern 2 "such *NP* as"	Pattern 3 "or other"	Pattern 4 "and other"
Technology and Appl. sci.	65,755 (28.61%)	2,111 (0.92%)	6,259 (2.72%)	24,546 (10.68%)
Sports	11,463 (20.75%)	682 (1.23%)	699 (1.27%)	3,977 (7.20%)
Religion and Belief	18,898 (23.93%)	1,279 (1.62%)	1,653 (2.09%)	9,841 (12.46%)
Food and drink	17,532 (25.66%)	481 (0.70%)	1,451 (2.12%)	6,459 (9.45%)
Geography	7,899 (22.80%)	383 (1.11%)	617 (1.78%)	3,725 (10.75%)
Totals (patterns)	121,547 (26.03%)	4,936 (1.06%)	10,679 (2.29%)	48,548 (10.40%)

Subset / Pattern	Pattern 5 "including"	Pattern 6 "especially"	Pattern 7 "is a"	**Totals** (for subsets)
Technology and Appl. sci.	38,106 (16.58%)	989 (0.43%)	92,050 (40.05%)	229,816 (100%)
Sports	10,618 (19.22%)	170 (0.31%)	27,625 (50.01%)	55,234 (100%)
Religion and Belief	10,902 (13.88%)	535 (0.68%)	35,880 (45.42%)	78,988 (100%)
Food and drink	11,315 (16.56%)	356 (0.52%)	30,730 (44.98%)	68,324 (100%)
Geography	6,751 (19.48%)	154 (0.44%)	15,123 (43.64%)	34,652 (100%)
Totals (for patterns)	77,692 (16.64%)	2,204 (0.47%)	201,408 (43.13%)	467,014 (100%)

Table 4.3.: Statistics for extracted semantic relations from Wikipedia differentiated among subsets. Percentages describe the portion of semantic relations extracted by the associated pattern amongst all patterns in a particular subset.

There is also a difference in the frequency of extracted relations between different subsets. This is illustrated by Table 4.4. Considering all subsets together, there were approximately 15.3185 semantic relations extracted per 10,000 words. If comparing the total number of relations per 10,000 words for separate subsets, there are great differences. There were more than three times as many relations per 10,000 words extracted for the subset *Food and drink* than for the subset *Sports*. Based on this, it can be assumed, that there is a higher demand to give semantic relations like hyponymy relations for certain category branches. Examples for extracted semantic relations are given by Table A.2.

For the evaluation in the next section, clusters were extracted for each subset separately by the curvature clustering approach described in Section 2.2. In Table 4.5, the size of generated dictionaries in the lemmatizing step, the number of cluster, the average size of clusters and the minimum and maximum size of clusters are given. It can be seen that also the dictionaries related to different subsets differ in their size in proportion to the size of their associated subsets. To give an example, the subset *Sports* has the second highest token count (see Table 4.2) with 75 million tokens, but has a smaller dictionary than the subset *Religion and Belief* with

Subset / Pattern	Pattern 1 "*NP* such as"	Pattern 2 "such *NP* as"	Pattern 3 "or other"	Pattern 4 "and other"
Technology and Applied sciences	5.2617	0.1689	0.5008	1.9642
Sports	1.5200	0.0904	0.0927	0.5274
Religion and Belief	3.6285	0.2456	0.3174	1.8895
Food and drink	6.2154	0.1705	0.5144	2.2898
Geography	3.2647	0.1583	0.2550	1.5396
Totals (patterns)	3.9869	0.1619	0.3503	1.5924

Subset / Pattern	Pattern 5 "including"	Pattern 6 "especially"	Pattern 7 "is a"	**Totals** (for subsets)
Technology and Applied sciences	3.0492	0.0791	7.3658	18.3897
Sports	1.4080	0.0225	3.6632	7.3242
Religion and Belief	2.0932	0.1027	6.8890	15.1659
Food and drink	4.0113	0.1262	10.8943	24.2219
Geography	2.7902	0.0636	6.2504	14.3219
Totals (for patterns)	2.5484	0.0723	6.6064	15.3185

Table 4.4.: Statistics for extracted relations per 10,000 words.

only 52 million tokens. The same inverted proportion can be seen for the number of clusters.

There are 1,905 clusters available for evaluation purposes. Since not all of them are suitable for an evaluation, e.g. clusters with 1,000 or more terms, some clusters were selected for the evaluation. This selection is described in Section 4.2.2. Samples of extracted clusters are given in Table A.4.

4.2. Evaluation

To investigate the quality of descriptive words discovered by the proposed approach, a human evaluation of clusters and their descriptive words was performed. The details for the discovery of descriptive words and related statistics are presented in Section 4.2.1. Selection criteria for choosing clusters for the evaluation are given in Section 4.2.2. In the subsequent section, the method used for the evaluation, i.e. the used interface, is described. Finally, Section 4.2.4 presents the results of the performed evaluation.

Subset / Objective	Size of dictionary (number of nouns)	#Clusters	∅Size of clusters	Min. size	Max. size
Technology and Applied sciences	602,182	745	12.92	3	2,075
Sports	293,033	256	16.43	3	1,542
Religion and Belief	311,230	360	10.90	3	674
Food and drink	217,478	262	13.55	3	1,465
Geography	287,287	282	10.02	3	285
All subsets		1905	12.67	3	2,075

Table 4.5.: Statistics for extracted clusters from subsets of Wikipedia. The statistics for all subsets are based on the union of all clusters. Due to possible intersections, the size of a dictionary for all subsets can not be calculated by summing up the dictionary sizes of all subsets.

4.2.1. Discovered descriptive words for evaluation

To discover descriptive words for the evaluation, Algorithm 3.1 was used with a maximum number of steps $maxAS = 3$ and a maximum out-degree of vertices $maxODV = 6$. The relation graph was build on the one hand with semantic relations extracted from the related subset of this cluster extracted from Wikipedia (in the following text referred to as **WIKI**) and on the other hand by using relations extracted from WordNet (**WN**). As weighting function weightingPP$_{w_{\text{cov}},w_{\text{spec}}}$ (Equation 3.5) was used with $w_{\text{cov}} = 1.5$ and $w_{\text{spec}} = 1$.

The relations WIKI have been filtered before they were used to build the relation graph. Relations with the superordinates *example*, *type*, *list*, *common*, *form* were filtered out. These relations are not appropriate to find descriptive words since they are too abstract and can be used as a hypernym for almost every noun. Hence, relations with this superordinate occur very frequently. Among the most frequently occurring superordinates for all subsets, these five words were manually chosen and removed. The most frequently occurring superordinates for each subset are given in Table A.3.

Table 4.6 illustrates the portion of clusters which have a discovered descriptive word, respectively more than one descriptive word. The last column states the number of clusters having discovered descriptive words on the one hand with WIKI and on the other hand with WN. It can be seen that the approach discovers descriptive words for 83.7% with WIKI and for 74.1% with WN. The higher percentage is reasonable when considering the domain-specific knowledge contained by Wikipedia. Since WordNet has been created manually, it does not cover all con-

Subset / Obj.	#Clusters	#Clusters ≥ 1, WIKI	#Clusters > 1, WIKI	#Clusters ≥ 1, WN	#Clusters > 1, WN	#Clusters ≥ 1, both
Tech. a. App. sci.	745	655 (87.9%)	111 (14.9%)	543 (72.9%)	41 (5.5%)	522 (70.1%)
Sports	256	214 (83.6%)	30 (11.7%)	188 (73.4%)	13 (5.1%)	174 (68.0%)
Religion and Bel.	360	292 (81.1%)	44 (12.2%)	264 (73.3%)	19 (5.3%)	247 (68.6%)
Food and drink	262	231 (88.2%)	31 (11.8%)	208 (79.4%)	14 (5.3%)	197 (75.2%)
Geography	282	203 (72.0%)	26 (9.2%)	208 (73.8%)	15 (5.3%)	174 (61.7%)
Totals	1905	1595(83.7%)	242(12.7%)	1411(74.1%)	102 (5.4%)	1314(69.0%)

Table 4.6.: Statistics for the number of clusters for which descriptive words were discovered. ≥ 1 means that at least one descriptive word was discovered, > 1 respectively more than one descriptive word were discovered. The abbreviation "both" stands for clusters with discovered descriptive words using both WIKI and WN separately.

tent contained by Wikipedia and especially no knowledge of very specific domains. This was checked manually for randomly selected clusters which have descriptive words discovered by WIKI but no discovered descriptive words by WN. Indeed, most of these clusters contain nouns which are not recognized by WordNet. For example, in the cluster $\{html, xml, java, xquery, xslt, xpath\}$ only the nouns *html* and *java* are recognized by WordNet. On the contrary, there are relations available for every noun in the cluster as subordinate in the relations extracted from Wikipedia.

It can also be seen that the approach can not discover descriptive words with relations from Wikipedia for $1,411 - 1,314 = 97$ clusters which have descriptive words discovered with relation from WordNet. Investigating selected clusters from these 97 showed, that for nouns in these clusters too few or no relations are available. For example, in the cluster {*nominative*, *dative*, *ablative*, *locative*, *genitive*, *accusative*} there are only relations with the noun *genitive* as subordinate available.

The approach cannot find descriptive words for 213 clusters neither with WIKI nor with WN due to the same reasons from above. There are too few relations available for WIKI and these clusters contain nouns with too specific knowledge for WN. Besides, there are clusters with spelling mistakes, abbreviations, non-noun words or proper nouns among these 213 clusters. Furthermore, it can also be seen that most of the clusters have exactly one discovered descriptive word for both WIKI and WN (12.7% (> 1) in relation to 83.7% (≥ 1) for WIKI, 5.4% to 74.1% for WN respectively).

Because descriptive words were discovered for all subsets for both WIKI and WN for at least 72% of all clusters it can be assumed that the approach is applicable for different knowledge domains. The portion of clusters with discovered descriptive words depends on the amount of extracted relations for the nouns participating in the clusters as subordinates.

4.2.2. Selection of clusters for evaluation

Clusters extracted from the five subsets given in Section 4.1 have to satisfy selection criteria to be appropriate for evaluation purposes. These criteria for a single cluster are:

1. The cluster has descriptive words discovered with the proposed approach and with relations

   - extracted from Wikipedia (see Sections 3.2.1 and 4.1.2) and
   - extracted from WordNet (see Section 3.2.2).

2. The cluster size has to be lower than a threshold θ_{clSize}.

3. The cluster does not contain

   - nouns which require specific knowledge,
   - person names,
   - a majority of nouns having spelling mistakes or
   - a majority of nouns which are not nouns, e.g. adjectives or verbs.

Appropriate clusters were selected by filtering all clusters (1,905) based on these criteria in the order they are given. After the first step, filtering based on descriptive words for both WIKI and WN, there were 1,314 clusters remaining. These were filtered by size in the second step, i.e clusters with a size below or equal the threshold θ_{clSize} were kept. The threshold θ_{clSize} was restricted to 20, clusters with a greater size are unmanageable for assessors. As a third and last step, the remaining 1,186 clusters were classified into five classes, namely:

- **Appropriate**. Clusters in this class are appropriate for evaluation purposes.

  Example: $\{litre, quart, gallon, pint\}$

Subset / Classes	Appropriate	Specific knowledge	Person names	Spelling mistakes	Other	Totals (for subsets)
Technology and Applied sciences	237	184	32	2	11	466
Sports	51	83	6	0	6	146
Religion and Belief	112	96	9	1	9	227
Food and drink	102	78	0	0	4	184
Geography	74	82	0	0	7	163
Totals	576	523	47	3	37	1,186

Table 4.7.: Classification of clusters for evaluation purposes. Only clusters with a size equal or lower than $\theta_{clSize} = 20$ and discovered descriptive words for both used semantic relation sources were considered.

- **Specific knowledge**. Clusters in this class require specific knowledge from the assessors. Due to very domain-specific knowledge contained by Wikipedia it can be very difficult even for human, non-expert assessors to find a descriptive word.

  Example: $\{galactose, sucrose, glucose, fructose, mannose\}$

- **Person names**. Despite trying to avoid person names in clusters by only allowing lower case words in the dictionary (see Section 2.2.1), there is a small portion of extracted clusters which still contain person names.

  Example: $\{sampson, kennedy, lyndon, truman, charles, \ldots\}$

- **Spelling mistakes**. Clusters with spelling mistakes are assigned to this class.

  Example: $\{munich, tokyo, rnberg, bavaria, germany, nuremberg\}$

- **Other**. Clusters in this class does not contain nouns or are not appropriate for other reasons.

  Example: $\{homeless, artist, poor, hungry\}$

The distribution of clusters over the different classes is given in Table 4.7. It can be seen that most of the non-appropriate clusters contain specific knowledge. Only a small portion of clusters contain person names. An even smaller portion of clusters has spelling mistakes, does not contain nouns or is non-appropriate for other reasons. Therefore, 576 clusters assigned to the class *Appropriate* were

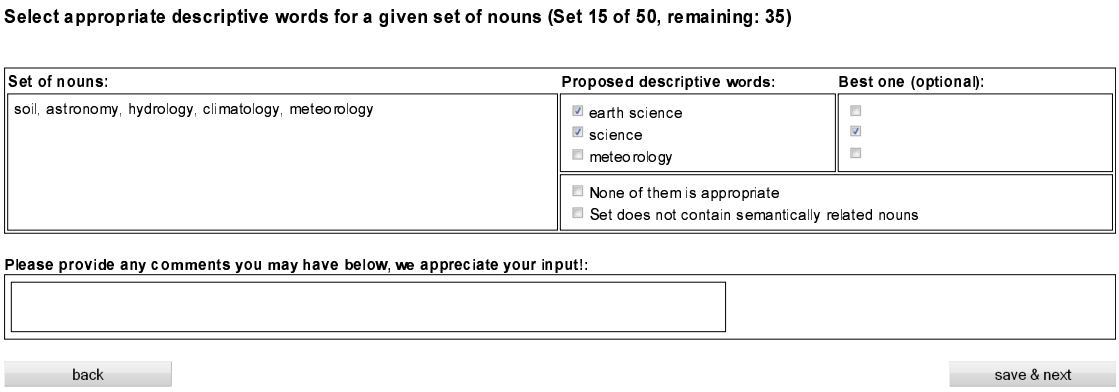


Figure 4.1.: Interface for the evaluation. Guidelines are removed for illustration. The evaluated cluster is {*soil*, *astronomy*, *hydrology*, *climatology*, *meteorology*}. The proposed descriptive words are *earth science* discovered with relations from WordNet, *science* discovered with relations extracted from Wikipedia and the noun *meteorology* in the cluster as baseline.

available for evaluation purposes. For the evaluation, 500 of these clusters were randomly selected.

4.2.3. Method of evaluation

To determine the quality of the discovered descriptive words, i.e. if they are appropriate for their associated clusters, an evaluation by human assessors was performed. 500 randomly selected clusters (see Section 4.2.2) were given to human assessors, together with their discovered descriptive words for both relation sources. Additionally, a noun contained by the cluster was given as a descriptive word to the assessors to have a baseline for the evaluation. This noun was selected by calculating a local curvature value based on the vertices in the cluster choosing the one with the lowest value (more in Section A.3). If a cluster had more than one descriptive word for one relation source, one of these descriptive words was randomly chosen. This is reasonable because all of them have the same score for appropriateness. Hence, each cluster was given with three descriptive words to assessors.

The clusters were parted into ten batches, each with 50 clusters. Each of these batches was evaluated three times by different assessors to get a majority voting.

The interface for the evaluation is presented in Figure 4.1 and can be described as follows:

- The assessors were able to select each of the three proposed descriptive words as appropriate.
- Additionally, if more than one proposal is appropriate, the assessors were able to chose the best proposal.
- If none of the proposals is appropriate, the assessors could alternatively chose "None of them is appropriate".
- The check box "Set does not contain semantically related nouns" enabled the assessor to disqualify the cluster, i.e. stating that the cluster is not appropriate for the evaluation.
- Before continuing with the next cluster to evaluate, an assessor was able to give comments in the comment field.

The order of the proposals was randomly switched for each cluster to ensure that assessors do not know which proposal belongs to which method of discovery (i.e. WIKI, WN or the baseline).

The interface has been implemented based on PHP, HTML and JavaScript. The batches and given answers have been stored in a MySQL-Database accessed by PHP. JavaScript has been used to assist the assessors in enabling the correct combinations of check boxes. For example, enabling the check box "None of them is appropriate" disables all check boxes for the proposed descriptive words. If two proposed descriptive words are equal and an assessor selects one of them as appropriate, the other proposal is also selected as appropriate.

4.2.4. Results of evaluation

In this section the results of the evaluation is given. An assessment which is done for a single cluster by an assessor is from here on referred to as a **vote**. Since each batch was evaluated by three assessors, $3\ \textit{assessors} * 10\ \textit{batches} * 50\ \textit{clusters} = 1,500\ \textit{votes}$ were performed. The approach in combination with semantic relations extracted from Wikipedia is again abbreviated by **WIKI**, respectively **WN**

Possibles Choices/Positive votes	0	1	2	3	≥ 1	≥ 2
Wiki	271	102	62	65	229	**127**
WN	86	109	140	165	414	**305**
MLC (Baseline)	266	137	66	31	234	**97**
None of them is appropriate	334	120	37	9	166	**46**
Set does not contain sem. rel. nouns	450	47	3	0	50	**3**

Table 4.8.: Unfiltered evaluation results. The clusters are parted into their number of positive votes among the three approaches to discover descriptive words.

for relations from WordNet. The baseline approach is abbreviated by **MLC** for Minimum Local Curvature.

Table 4.8 presents the raw (unfiltered) results of the evaluation. It can be seen that only 3 clusters were disqualified by checking "Set does not contain semantically related nouns" by the majority of ≥ 2 assessors. This indicates a high quality of the extracted clusters. To assess the quality of the discovered descriptive words, these three clusters were filtered out. The results after this filtering are given in Table 4.9.

It can be seen that the proposed approach provided the best descriptive words with relations from WN. It provided appropriate descriptive words for 61.37% of the evaluated clusters according to the majority of assessors (≥ 2 assessors). On the contrary, descriptive words discovered by WIKI have a significantly lower appropriateness indicated by the lower percentage of 25.55%. One reason for this is that the quality of relations differs between the relation sources. Since relations from WordNet have a very high quality, the discovered descriptive words are more appropriate.

The baseline (MLC) achieved a percentage of 19.52% which is higher than it was expected before the evaluation. One reason for this is that each proposed descriptive word derived by the baseline is already contained by its associated cluster (see previous section). Observing the performed votes it can be seen that this was for some assessors enough for a proposed word to be descriptive. For example, for the cluster {*lady*, *lion*, *leopard*, *tiger*}, the descriptive word *lion* discovered by MLC was checked to be appropriate by one accessor.

For 9.26% of the clusters, none of the approaches discovered an appropriate descriptive word. The percentage of clusters for which either the descriptive word

Possibles Choices/Number of votes	0	1	2	3	≥ 1	≥ 2
WIKI	268	102	62	65	229	**127**
WN	83	109	140	165	414	**305**
MLC (Baseline)	263	137	66	31	234	**97**
None of them is appropriate	334	117	37	9	163	**46**

Possibles Choices/Number of votes	0	1	2	3	≥ 1	≥ 2
WIKI	53.92%	20.52%	12.47%	13.08%	46.08%	**25.55**%
WN	16.70%	21.93%	28.17%	33.20%	83.30%	**61.37**%
MLC (Baseline)	52.92%	27.57%	13.28%	6.24%	47.08%	**19.52**%
None of them is appropriate	67.20%	23.54%	7.44%	1.81%	32.80%	**9.26**%

Table 4.9.: Filtered evaluation results. Sets with ≥ 2 votes for "Set does not contain semantically related nouns" are filtered out, 497 clusters remain. The upper table presents the number of clusters parted into their number of positive votes among the three approaches to discover descriptive words. The lower table presents the associated percentages referring to the remaining 497 clusters.

discovered by WIKI or by WN (or by both approaches) was checked from a majority as appropriate is 72.03%.

If two or all approaches discovered appropriate descriptive words, assessors were allowed to select one of these words as the best one. Since this selection was defined as optional, all votes are observed without being grouped by their associated clusters. Therefore, all $1,500$ votes are differentiated among their checked descriptive words for the different approaches. The result is presented in Table 4.10. For positive votes for both approaches WIKI and MLC it can be seen that the proposed words determined by WIKI were more often checked to be better than the other way around (17 times to 9). This is also true for WN and MLC (36 times to 29). It can also be seen that WN is more often better than WIKI if both associated proposed descriptive words are appropriate.

It can be observed by examining the votes which have WN checked to be better than WIKI, that the descriptive word discovered by WN is often more specific than the descriptive word discovered by WIKI. For example, WN discovered the descriptive word *internal organ* for the cluster {*heart*, *liver*, *brain*, *lung*, *kidney*}. This descriptive word is more specific than the descriptive word *organ* which was discovered by WIKI. The reason for this is that semantic relations extracted from WordNet are "atomic". This means that they can not be derived by transitivity.

WIKI	WN	MLC	#Votes	WIKI (best)	WN (best)	MLC (best)	none is best
✓	✓	×	142 (9.47%)	34 (23.94%)	72 (50.70%)	0 (0.00%)	36 (25.35%)
✓	×	✓	41 (2.73%)	17 (41.46%)	0 (0.00%)	9 (21.95%)	15 (36.59%)
×	✓	✓	108 (7.20%)	0 (0.00%)	36 (33.33%)	29 (26.85%)	43 (39.81%)
✓	✓	✓	51 (3.40%)	10 (19.61%)	13 (25.49%)	19 (37.25%)	14 (27.45%)

Table 4.10.: Comparison of votes with at least two approaches checked as appropriate. Only clusters which have two different proposed descriptive words checked as appropriate are considered. The percentages for votes refer to the total number of votes (1,500), the percentages for best discovered descriptive words for a certain approach refer to the number of votes for the associated combination of checked proposed descriptive words.

For example, the relations (*heart*, *internal organ*) and (*internal organ*, *organ*) can be extracted from WordNet but the relation (*heart*, *organ*) not. This is not the case for relations extracted from raw text. It is up to the author to write higher level superordinates and lower level subordinates in the same text phrase which is extracted by patterns. The relation (*heart*, *organ*) might be extracted from raw text with patterns since there might be a text phrase like ".. organs, such as heart, liver, ..." in the text.

The approach more often discovered better descriptive words than the baseline (MLC). However, the number of votes which have MLC as the better approach is quite high. One reason is that an assessor chose a proposed descriptive word in the cluster rather than one outside. This is especially true if both proposed descriptive words have the same appropriateness for an assessor. This can also be seen in the fourth line of Table 4.10. This line represents votes with all proposed descriptive words checked as appropriate. At least one of the three proposed descriptive words has to differ for a vote in this line. For these votes, the baseline performed more often better than the other two approaches. For example, the cluster {*mars*, *lander*, *probe*, *orbiter*} has the proposed descriptive words *space* (WIKI), *equipment* (WN) and *orbiter* (MLC). The word *orbiter* was selected as best descriptive word by one assessor. Besides, all three assessors chose *space* to be appropriate but not as best descriptive word.

In Chapter 3 it has been assumed that the score of a descriptive word expresses its appropriateness. With the results of the evaluation, it could be checked if this assumption holds. This was done separately for WIKI and WN. For each of WIKI and WN, the clusters were parted into two classes. The first class contains the clusters which have a descriptive word that is not appropriate (< 2 positive votes

for their associated descriptive word). The other class contains the clusters with ≥ 2 positive votes. For both approaches, a student t-test with $\alpha = 5\%$ showed that the mean score of the associated descriptive words for the former class of clusters is lower than the mean score of the latter class. The mean scores are:

- WIKI: 0.26 (not appropriate, SD: 0.11), 0.35 (appropriate, SD: 0.15).
- WN: 0.27 (not appropriate, SD: 0.15), 0.35 (appropriate, SD: 0.16).

Hence, the mean scores for both approaches and different relation sources are similar. However, there are high standard deviations (SD) for the scores. This implies that the scores are not suitable to make a general statement if a descriptive word is really appropriate. Therefore, applying a threshold for the score to assure the appropriateness of a descriptive word is not reasonable. Nevertheless, the scores are significant to find the best appropriate word for a single cluster. A few clusters with descriptive words and their scores are given in Table A.4.

4.3. First results on a long term archive

All previously described experiments and evaluations were performed on text which is written in today's language and which has a high quality. This was done to investigate if the proposed approach provides reasonable results without being affected by issues like Optical Character Recognition (OCR) errors. Since the approach should be applied on long term archives as an application, experiments were performed on such an archive. The chosen archive is The Times Archive [Tim], which has been generated using OCR on newspaper articles spanning from 1785 to 1985.

Since the quality of discovered descriptive words mainly depends on the amount and the quality of semantic relations extracted from the text in an archive, the first experiments focussed on the extraction of them. It was investigated how the seven patterns (see Section 3.2.1) perform on The Times Archive. Therefore, each fifth year was parsed and the number of relations found for each pattern in each separate year has been counted. The counts were normalized by dividing by the total number of tokens (white-space separated words). The normalized counts are presented in Figure 4.2, scaled in relations per 10,000 tokens. The unnormalized counts can be taken from Figure A.3.

Figure 4.2.: Number of extracted semantic relations from The Times Archive per 10,000 tokens for the patterns from Section 3.2.1. Spanning from 1785 to 1985, each fifth year was parsed for semantic relations. The highlighted graph represents all patterns.

It can be seen that the total number of extracted relations per 10,000 tokens for the complete period of 200 years is below 8 semantic relations. This is rather low compared to over 15 relations per 10,000 tokens for the total number of extracted relations from Wikipedia (see Table 4.4). Two possible reasons for this low total number are the high amount of OCR errors, especially in the early decades, and different writing styles. OCR errors affect the patterns not only by wrongly spelled words but also by wrongly recognized punctuation marks like commas or dots which are needed by the patterns. The writing style in Wikipedia has a "descriptive" character, while newspaper articles have a more "narrative" style. Therefore, it can be assumed that an encyclopaedia like Wikipedia contains more semantic relations, especially hyponymy relations, than a newspaper archive like The Times Archive.

All patterns are subject to fluctuations over the time. For example, there is an increase of extracted semantic relations per 10,000 tokens for each pattern from 1810 to 1815. This is caused by the introduction of a steam press in the end of 1814 [Tim14]. From this event on, due to its more regular type face, OCR methods were better able to digitize articles, resulting in a lower number of OCR errors. Other reasons for fluctuations may be a change in writing style over the years or better paper quality, storage etc. A more detailed discussion of OCR errors and statistics for The Times Archive can be found in Tahmasebi et al. [TNTR10].

However, all patterns provided semantic relations for all years starting after the introduction of the steam press in 1814. The most patterns seem to increase in extracted relations per 10,000 words over the time. This is reflected in the total number of relations per 10,000 tokens. A student t-test with $\alpha = 5\%$ showed that the mean number of total relations per 10,000 tokens for the years spanning from 1785 to 1880 (2.76 relations per 10,000 tokens) is significantly lower than the mean number for the years spanning from 1885 to 1985 (5.95 relations per 10,000 tokens). Besides OCR errors, a reason for the increasing number of extracted realtions could be that the different patterns gain or loose in popularity over the years. In the earlier decades, other patterns could have been used to express semantic relations. For example, the pattern "and other" has great fluctuations over the 201 years but provides about the same number of relations per 10,000 tokens in 1820, 1885, 1895 and 1945. This is not true for other patterns such as "*NP* such as" which provide a steadily increasing number of relations over the time.

When comparing the distributions of semantic relations over their associated patterns of Wikipedia and The Times Archive for 1985, it can be seen that the pat-

Dataset/Pattern	*NP* such as	such *NP* as	or other	and other	including	especially	is a
Wikipedia	121,547 (26.03%)	4,936 (1.06%)	10,679 (2.29%)	48,548 (10.40%)	77,692 (16.64%)	2,204 (0.47%)	201,408 (43.13%)
Times (1985)	4,912 (15.36%)	637 (1.99%)	782 (2.45%)	5,066 (15.84%)	10,926 (34.17%)	407 (1.27%)	9,250 (28.92%)

Table 4.11.: Distribution of relations over patterns compared for Wikipedia and The Times Archive. Percentages refer to the total number of relations extracted by all patterns for the associated dataset.

terns"such *NP* as", "or other" and "especially" provided the smallest portion of semantic relations in both datasets. This is depicted in Table 4.11. The shifting from the patterns "*NP* such as" and "is a" to the patterns "including" and "and other" may be caused by the different writing styles (e.g. "descriptive" and "narrative" from above).

4.4. Discussion

Overall, the proposed graph-based approach seems to be suitable for the discovery of descriptive words. Measuring the "appropriateness" with two quality aspects, coverage and specificity, is shown to work well. However, the approach has only an about 6% percentage points higher mean acceptance by the assessors in comparison to the baseline with extracted relations from text (25.55 to 19.52%). Furthermore, due to the high standard deviations among the scores, a high or low score does not automatically imply acceptance but a higher probability for acceptance.

The appropriateness of discovered descriptive words depends highly on the amount and the quality of available semantic relations. The approach provided much better results with semantic relations extracted from WordNet than with relations extracted from the related dataset, in this case Wikipedia (61.37 to 25.55). When comparing the dictionary size for the text in the extracted subsets of Wikipedia with the number of extracted semantic relations, it becomes clear that the amount of extracted relations is far to small. For example, the subset *Technology and Applied sciences* has a dictionary size of 602,182 terms (nouns) but only 229,816 semantic relations have been extracted. This means, for almost every third noun there exists no semantic relations with the noun as subordinate. The situation becomes even worse considering that a noun can be a subordinate in several rela-

tions. With the low number of relations it is almost surprising that the approach is able to find appropriate descriptive words for over 25% of the extracted clusters. This could be caused by two possible reasons. Firstly, the superordinate nouns of used relations also cover nouns in the associated cluster even if there is no path between them in the relation graph. For example, for the cluster {*lady*, *lion*, *leopard*, *tiger*}, the descriptive word *cat* covers *lion*, *leopard* and *tiger* for assessors, but only two semantic relations (*lion*, *cat*) and (*leopard*, *cat*) were available. The relation (*leopard*, *cat*) was not available. This could also explain the high standard deviations among the scores. Secondly, the lists used to extract semantic relations with the seven patterns were also used to build the co-occurrence graph. Because the clusters are extracted from this co-occurence graph, there could be a higher probability that nouns in the clusters also participate in semantic relations. Hence, the extracted relations could be applicable for nouns but not for nouns outside. It would therefore be interesting if clusters derived by other word sense discrimination methods, not using lists of nouns, have descriptive words discovered by the proposed approach.

Despite the low number of extracted relations, more descriptive words were discovered with relations from Wikipedia than with relations from WordNet. Even if the appropriateness of descriptive words is lower than with relations from WordNet, this shows the potential of the approach regarding the issue of domain-specific knowledge which is not contained by manually created lexical databases, e.g. WordNet. To increase the appropriateness of descriptive words discovered with relations from text, it could be enough to increase the number of extracted relations. The graph building algorithm tries to filter out erroneous relations by using only the most frequent relations. In spite of the low number of extracted relations, this filtering method probably failed often since fewer relations than the maximum out-degree of vertices are available. It is also likely that many valid relations were filtered out, especially considering that in an encyclopaedia each topic occurs only once. Because the approach is already with these small number of relations better than the baseline, an improvement of the extraction method for semantic relations seems to be worth investigating.

Since the used patterns also provided semantic relations on The Times Archive, even for years in the 18th century, the approach including the pattern-based extraction of semantic relations seems to be applicable also for long term archives. However, because of the smaller number of relations per $10,000$ tokens discovered in the last section, an improvement or the discovery of new patterns, which perform better on earlier decades, is even more needed for such an archive.

5. Related work

This chapter is parted into two section. The first section presents related work concerning unsupervised labelling of clusters, in the notion of this thesis discovery of descriptive words. Afterwards, Section 5.2 describes related work concerning automatic extraction of semantic relations from text.

5.1. Unsupervised labelling of clusters

There are three general methods for labelling clusters unsupervised. Which method is suitable depends on the method which was used to extract or generate clusters. An approach for labelling clusters can also be a mixture of these methods. Clusters are also called *sets of semantically related nouns* in the notion of this thesis or *semantic classes*.

The first method of labelling concerns clusters which have been extracted using unsupervised word sense discrimination, for example approaches proposed by Schütze [Sch98], Pantel and Lin [PL02] or Dorow. [Dor07, DW03]. These approaches do not provide a labelling of generated clusters by themselves. This method of labelling uses additional internal information extracted from the underlying text to label clusters. The approach proposed in this thesis belongs to this method if using semantic relations extracted from text.

The second method, also suitable for unsupervised word sense discrimination approaches, uses external knowledge sources such as lexical databases like WordNet. This method has also been used in this thesis to show that the approach provides good results with relations of a high quality extracted from WordNet. However, due to the lack of domain-specific knowledge or outdated terms this method should be avoided.

The third method labels the clusters with information gathered during the generation of clusters. In other words, the approach used for clustering or generation of clusters already provides information which can be used to determine labels. The baseline for the evaluation described in this thesis labels clusters with descriptive words by using the curvature value of its members (see Section 4.2.3). Therefore, it is an implementation of the third method.

Cluster labelling with internal information

Using the first method, Pantel and Ravichandran [PR04] tried to discover new hyponymy relations. Pantel and Ravichandran tried to label clusters which are a result of the CBC (Clustering By Committee) algorithm, proposed by Pantel and Lin [PL02]. Hyponymy relations are derived as relations from each cluster member to its assigned label. To label the clusters, Pantel and Ravichandran used the same methods as the CBC algorithm. Each word in the underlying text is represented by a **feature vector** with feature counts as entries. A **feature** corresponds to a context in which the word occurs, e.g. a verb-object context like "*object-of catch*" where "*catch*" is a verb. In the first phase, a **mutual information vector** is created for each word. This vector represents the association strength between a word and its features. In the second phase, a **committee** for each cluster is determined to detect the most important cluster members. In the third phase, labels are assigned to the cluster. To find a label for a cluster, the feature vectors of the cluster's committee members are averaged. The output is a so called **signature** of the cluster. Four specific semantic relationships are searched in this signature (*apposition*, *nominal subject*, *such as*, *like*). The mutual information scores of terms occurring with committee members in these relationships of the signature are summed up. The term with the highest scoring is taken as a label.

Using this approach, Pantel and Ravichandran were able to label 98.5% of clusters extracted from 3G of newsletter text. To evaluate the quality of their approach, 125 randomly selected clusters were given to judges. Each cluster was given with the top-5 terms as possible labels. For 72.0% percent of the evaluated clusters, the top-1 label has been judged as correct. This is significantly better than the proposed approach in this thesis with relations extracted from Wikipedia (25.55%). As described in Section 4.4, this is caused by the small number of extracted relations. As a baseline, Pantel and Ravichandran took up to five most frequent common ancestors for cluster members from WordNet. With this method, labels could be assigned to 33 of the 125 clusters (26.4%). Using the approach in this thesis and relations extracted from WordNet, almost three times as many clusters

(74.1%) could be labelled (see Section 4.2.1). Unfortunately, there is no percentage for correct labels given for the top-1 labels determined with WordNet which can be compared with the results of the performed evaluation (only the MRR (Mean Reciprocal Rank) score is given). Although the approach of Pantel and Ravichandran is effective, it has the disadvantage that it requires the extraction of sophisticated grammatical relations. Since it can not be assumed that grammatical parsers provide good results on texts in long term archives, this thesis uses the already proposed alternative approach with patterns to extract semantic relations.

Jickels and Kondrak [JK06] proposed an approach which is also based on grammatical relations. The proposed algorithm automatically assigns weights to all features extracted from text, called **feature scores**. This is done by an iteration over all clusters and each of the **feature words** of all cluster members. For example, for the word *banana* which is a member of a certain cluster, the feature "*-fruit:such as:banana*" has the feature word *fruit*. To find a label for a cluster, all feature words of all cluster members are observed by calculating so called **label scores**. A label score for a feature word is the sum over all feature scores whose features contain the feature word. The feature word with the highest score is taken as label. Jickels and Kondrak also proposed further variations of their algorithm. One variation is to take a fixed subset of features like it has been done by Pantel and Ravichandran. Equal feature scores are assigned to these features. A further variation is to intersect the output of the labelling algorithm with all hypernyms for words in a cluster recursively up to the top of the WordNet synset hierarchy (mixture of internal and external information).

To evaluate the quality of the determined labels, eight human participants were asked to label the classes manually. The agreement between labels from different participants was 42.8%, which shows that the resulting labels highly depend on the opinion of each separate participant. There were no clusters with the same assigned label by all participants but there were several clusters with different labels for each pair of participants. The performance of the different methods was measured by computing precision and recall against the labels which were given by the participants. A cluster label is considered to be correct if it has been proposed by at least one participant. Hence, for a clusters more than one label can be correct. Precision is defined as the percentage of generated labels which are correct. Recall is defined as the percentage of clusters that have at least one assigned correct label. For exactly one assigned label per cluster, the variation with intersection provides the highest precision of almost 70% with a recall of approximately 30%. The other variations have a lower precision between

approximately 16% and 23% and a recall between 15% and 22%. With these measures, the top-1 labels of Pantel and Ravichandran have the same recall as the variation with intersection, but a lower precision. But the precision is higher than the precision of the other variations ($\geq 25\%$). However, since the evaluation method of comparing the assigned labels with human labels is different from the evaluation method used in this thesis, the percentages can not be compared with results in this thesis. Because the approach of Jickels and Kondrak is also based on the extration using a grammatical parser it is also not suited for long term archives.

An approach which uses hypernyms to label clusters was presented by Caraballo [Car99]. Caraballo investigated the automatic construction of a hypernym-labelled noun hierarchy from text. Therefore, Caraballo proposed to cluster nouns with a bottom-up approach. The context of a noun is described as a **vector** which stores the count of co-occurrences of this noun with other nouns. Appearances of two nouns in a **conjunction** (e.g. "... $noun_1$ and $noun_2$ ...") or in an **apposition** (e.g. "... $noun_1$, a $noun_2$ ...") are taken as co-occurrences. The **similarity** of two nouns is measured with the **cosine** between their associated vectors. All nouns are added as leaf to a tree. The most similar nouns are clustered by assigning them to a new parent node in a tree. Nodes are further clustered until all nodes are subsumed by a single ancestor. Afterwards, labels are assigned to the inner nodes, i.e. clusters, of the tree. Therefore, the underlying text is parsed for hypernyms with two patterns of Hearst [Hea92], namely the two patterns "*or other*" and "*and other*". The most common hypernym of all leaf nodes, i.e. words, of an internal node in the tree is taken as label for the internal node (cluster). Hyponymy relations between cluster members and cluster labels were manually judged to be correct for 33% of randomly selected hyponyms from a cluster. An evaluation of the appropriateness of labels for their associated clusters has not been performed by Caraballo since the extraction of hypernyms was in focus. The approach of Caraballo is obviously a mixture of internal information (hypernyms from the underlying text) and information gathered during the clustering process (the generated tree structure). Since there is no such tree available, this approach is not suitable for our task.

Cluster labelling with external information

Labelling of clusters can also be tackled using external information sources. Widdows [Wid03] investigated how new words can be added to existing taxonomies such as WordNet. The method which was proposed by Widdows is to search for

semantic neighbours of a new word in a text corpus. The set of neighbours can be seen as the semantic class (or cluster) of the new word. This set of neighbours S is labelled using the hypernym tree of WordNet. The new word can be integrated in WordNet afterwards as a new synset which is a hyponym synset of the label synset as hypernym synset. To determine a label, all hypernym synsets of all neighbours $\mathcal{H}$ are taken from the hypernym tree, i.e. all superordinate hypernym synsets. An **affinity score** is calculated, describing the affinity between a word $w \in S$ and a hypernym $h \in \mathcal{H}$. The **affinity score function** is defined as

$$\alpha(w,h) = \begin{cases} f(dist(w,h)) & \text{if } h \in H(w), \\ -g(w,h) & \text{if } h \notin H(w) \end{cases}$$

where $H(w)$ is the set of hypernyms of only w. The function f is defined as positive, monotonically decreasing function and g is some positive (possibly constant) function. The function *dist* can be any distance measure in the hypernym tree of WordNet. For example, *dist* can be defined as the number of hyponymy relations needed to get from w to h in the hypernym tree. The hypernym which is finally taken as label is h_{max} and has the highest total affinity score summed up over all the members of S:

$$h_{max} = \max_{h \in \mathcal{H}} \sum_{w \in S} \alpha(w,h).$$

This approach is similar to the proposed algorithm in this thesis. The most appropriate class label is the hypernym which subsumes (or in the terminology of this thesis "covers") as many as possible of the members of S as closely as possible (specificity in this thesis). However, the highest total affinity score is not directly comparable with the used scoring function defined in this thesis. For the scoring function, the coverage and the specificity of a vertex are calculated separately. The approach proposed of Widdows sums up the "specificity" of covered members of S, which can be seen as adding positive points. Points are subtracted for each member which is not covered. A disadvantage is that the majority proportions of the distances between hypernyms and their covered members are not considered (see also Section 3.4.2). Besides, the highest total affinity score is not comparable over two clusters because it is not normalized.

Another approach to label clusters with taxonomies like WordNet was given indirectly by Resnik [Res99]. The idea is to take the **most informative subsumer**

in the WordNet hierarchy. The primary goal of Resnik was to find a measure for semantical similarity of two words based on a taxonomy (IS-A hierarchy of nouns). Previous approaches implemented this by counting the edges on the shortest paths between two words in the taxonomy. Resnik stated that there is a high variability in the distance for different sub-taxonomies like biological categories. This is caused by different densities of these sub-taxonomies. Therefore, he proposed an information based similarity measure which is not sensitive to the varying distances. For each concept, i.e. synset for WordNet, in the taxonomy, a probability is assigned. This is done by augmenting the taxonomy by a function $\mathrm{p}(c)$, which is a monotonically nondecreasing function when moving up in the taxonomy. The similarity measure is denoted as

$$\mathrm{sim}(c_1, c_2) = \max_{c \in S(c_1, c_2)} [-\log \mathrm{p}(c)]$$

where $S(c_1, c_2)$ is the set of all subsuming concepts of c_1 and c_2. The negative log likelihood expresses the information content of the concept c. The lowest common subsumer is the concept c for which the maximum is obtained. For two words w_1, w_2 instead of concepts the similarity is denoted as

$$\mathrm{sim}(w_1, w_2) = \max_{c_1, c_2} [\mathrm{sim}(c_1, c_2)] ,$$

where c_1 and c_2 contain w_1 or w_2 respectively. As an application of this similarity measure, Resnik has presented an algorithm to assign word senses to **noun groupings**. This can be seen as a disambiguation of the nouns in the noun grouping. For each noun in a noun grouping (cluster), the best sense (synset for WordNet) with respect to the cluster should be taken. Therefore, the algorithm goes through all possible pairs of nouns in the group and determines the most informative subsumer which is needed for the similarity measure. Although this method takes the nouns pairwise, it should be possible to extend the definition of the most informative subsumer to more than two nouns.

Cluster labelling with information gathered during the generation of clusters

Since there are many approaches for unsupervised word sense discrimination, especially such approaches which derive cluster labels during the clustering process, not all of them are discussed here.

An example for such an approach was proposed from Kulkarni [KP05]. The approach uses **lexical features** to build first and second order representations of contexts. The contexts are clustered using unsupervised methods for both of the two representations. The approach was first used to find clusters around given target words which are located in the centre of their associated clusters. However, it can also be used without a target word. The author refers to this discrimination method without a target word as **headless clustering**. As lexical features which represent contexts, in particular bigrams and co-occurences are used. **Bigrams** are ordered pairs of words that may have intervening words between them in the text, while **co-occurrences** are simply unordered bigrams. To gain second order representations of the context of a certain word, the lexical features of this word are combined to a **context vector**. The discovered clusters are afterwards labelled using the already detected bigrams which are associated to nouns in the cluster. Kulkarni has differentiated between two types of labels which are lists of bigrams, namely **descriptive** and **discriminating** labels. Descriptive labels describe their associated cluster in general while discriminating labels capture the content that separates one cluster from another. Descriptive labels are the top-N bigrams of the contexts of the cluster according to their log–likelihood ratio. Discriminating labels are descriptive labels which are not descriptive labels for other clusters.

A different way to generate semantic classes is to start with a manually created set of seed words of the same semantic class (e.g. “apple” and “orange”). This set is extended iteratively with new seeds like “banana” and further instances of fruits. This was investigated by Riloff and Shepherd [RS97], Roark and Charniak [RC98] and Widdows and Dorow [WD02]. Guggilla et al. [GMV08] proposed a similar approach to generate semantic classes and a way to label them. The **set of seed words** is extended by a statistical method. This method is based on an **Expectation-Maximization** (EM) algorithm. The iterative algorithm calculates the probability of other words to co-occur (e.g. in comma-separated lists) with a single seed word of the seed set in each step and for a set of documents. The word with the highest probability is added to the seed set. This continues for each seed in the seed set until a maximum number of words in the seed set has been reached. To generate a label for a derived set, random seed words are selected from the set. The probability for all randomly selected seed words to co-occur with other words is determined in an additional algorithm step. The word which co-occurs with all selected seed words and has the maximum co-occurrence probability is taken as a label of the set. As an alternative, WordNet is used to label the clusters. Hence, this alternative already belongs to the second method using external information. Therefore, glosses for words which are randomly selected from the seed set are extracted from WordNet. Overlapping terms of these glosses are considered as

labels. If more than one term is overlapping for all seed words, the term which matches most hypernyms of the seed words is taken as class label.

The method for extending a seed set to generate semantic classes is especially useful to build semantic lexicons. New members of a semantic class can be discovered and added to a lexicon. For this task the semantic class, i.e. its name or label, may be known. Kozareva et al. [KRH08] proposed an approach to use this knowledge. The "*NP such as*" pattern proposed by Hearst [Hea92] is used as a double-anchored pattern. A class name and a class member is taken to search for text phrases "... *class name* such as *class member* and * ...". A so called **hyponym pattern linkage graph** is build which represents the frequencies of the class members generating each other. New class members are determined by applying scores for the vertices in the graph. The label for generated semantic classes is the class name which is used to find class members.

5.2. Extracting semantic relations from text corpora

Since the proposed approach in this thesis uses semantic relations (hyponymy and meronymy relations), this section gives a brief overview of methods for extracting semantic relations from text. To extract semantic relations from raw text, there are two basic methods. The first, and most often used method, is to parse the text with patterns. The second method can be referred to as cluster-based. An example for a cluster-based approach is the approach of Pantel and Ravichandran [PR04] which was described in Section 5.1. Cluster which contain nouns of the same semantic class are labelled. Semantic relations can be derived by combining the cluster members with the cluster label. The disadvantage of the cluster-based method is that the used text must be of a sufficient size to generate clusters. This is not the case for pattern-based approaches. Instead, pattern-based approaches have the disadvantage of relation sparseness in the used text. Since it can not be assumed that grammatical parsers provide good results on texts in long term archives (used by Pantel and Ravichandran [PR04] and Jickels and Kondrak [JK06]), a pattern-based approach is used to extract semantic relations for the discovery of descriptive words.

The first approach to extract semantic relations from raw text was proposed by Hearst [Hea92]. Hearst noticed that hypernymy relations can be extracted from raw text with so called **lexico-syntactic** patterns. By observing text manually, Hearst discovered three of these patterns, namely "NP such as", "such NP as"

and "or other" (see Section 3.2.1). To find more patterns, Hearst proposed a bootstrapping algorithm. For relations which are known to hold, the environments of places in the text where two words of a relation occur near to each other are recorded. Commonalities among these environments are searched to discover new patterns. Instances found with new patterns can be used to discover more patterns. With this method, Hearst was able to discover three further patterns, namely "and other", "including", and "especially". Since these six patterns are commonly used in research, perform domain independent and are intended to extract hyponymy relations, they have been chosen to extract semantic relations for the performed experiments. Another reason for using these patterns is that a deep analysis of the text, e.g. part-of-speech analysis or dependency parsing, is not necessary except for the detection of nouns in the text.

Berland and Charniak [BC99] used the bootstrapping algorithm proposed by Hearst to search also for meronym patterns. Again, seed instances of meronymy relations are used to discover five patterns. They were able to detect two patterns which perform well for the task of finding meronymy relations, namely "NP's NP" and "NP of NP". Given a holonym, Berland and Charniak were able to achieve an accuracy of 55% for the 50 top-ranked meronyms with these two patterns. To rank extracted meronyms, a statistical model to measure their quality was introduced.

Girju et al. [GBM06] also used the bootstrapping algorithm of Hearst to detect patterns for meronymy relations. Girju et al. stated that these patterns also encode other semantic relations besides meronymy relations. To discriminate if a pattern occurrence in the text describes a part–whole relation, they employed a machine learning algorithm which uses WordNet to derive classification rules. With these rules, Girju et al. were able to achieve a precision of 80.95% and a recall of 75.91% on their test corpus.

Pantel et al. [PRH04] proposed a pattern learning algorithm which is used to learn **lexico-POS** patterns. Lexico-POS patterns are lexico-syntactic patterns with additional part-of-speech restrictions. The pattern learning algorithm calculates the minimal edit distance between two similar text phrases and retrieves an optimal pattern with POS information. Pantel et al. reported a precision between 38.7% and 55.9% depending on the size of the used text corpus. They also showed that this pattern-based approach achieves a higher recall than their cluster-based approach (Pantel and Ravichandran [PR04]). Part-of-speech information was been used by Mann [Man02] to create a proper noun ontology. Therefore, Mann [Man02] proposed to search for two nouns immediately following after each other. The first

noun is a common noun, followed by a proper noun. Fleischmann et al. [FHE03] extended this approach by adding patterns for appositions. They addidionally used a machine learning filter trained on hand-tagged data to filter extracted relations.

Snow et al. [SJN05] proposed the use of patterns extended with "dependency path" features. Given a training set of text containing known hyponymy relations, new patterns are identified by selecting all sentences for each noun pair which represents a hyponymy relation. These sentences are parsed with a dependency parser, nouns in the resulting dependency trees are removed to get general patterns. These patterns are combined to hypernym classifiers. To accept a noun pair as hyponymy relation, it has to occur in sentences which match all patterns of the classifier.

To improve precision and recall of automatically extracted relations from text, Cederberg and Widdows [CW03] proposed to apply a filter on extracted relations. The proposed filter is based on latent semantic analysis (LSA). It removes incorrect and spurious extracted relations and was able to reduce the rate of error of pattern-based hyponymy extraction by 30%. Cederberg and Widdows also tried to improve the recall by using coordination information to detect semantic relations additional to the relations extracted by patterns. They achieved a fivefold increase in recall with an accuracy of 46%.

6. Summary, conclusion and future work

6.1. Summary

This thesis addressed the task of labelling sets of semantically related nouns (clusters) with descriptive words. An approach which uses semantic relations extracted from raw text was proposed and tested. This approach does not depend on external information sources or sophisticated grammatical parsers but uses simple lexico-syntactic patterns to parse the text for such relations instead. Therefore it is also applicable for text corpora which contain domain-specific knowledge and long term archives. However, also relations from external sources can be used by the approach to discover descriptive words, e.g. semantic relations extracted from WordNet.

To discover descriptive words, hyponymy and meronymy relations were used. These relations were extracted from raw text using six lexico-syntactic patterns discovered by Hearst [Hea92] and one additionally proposed pattern. The extracted relations were used to build a *relation graph* for each cluster, whose vertices represent descriptive word candidates. Therefore, the algorithm which builds the graph makes use of the direction and transitivity of extracted relations. Vertices in the graph were assessed by calculating measures for the two quality aspects *coverage* and *specificity*. Coverage expresses the portion of nouns in the cluster which are covered by the vertex. The specificity on the contrary expresses how specific the vertex is in subject to its covered nouns in the cluster. These measures were combined with a weighting function to provide a score for each vertex. The vertices were ranked by their score to determine the vertices with the highest score. The vertices with the highest coverage among the vertices with the highest score were chosen as descriptive words.

To evaluate the performance of the approach, an evaluation was performed for clusters extracted from Wikipedia. Five subsets were extracted from Wikipedia. For each subset, clusters and semantic relations were extracted. With these relations, descriptive words for the clusters were discovered. 500 of these clusters were given with three proposed descriptive words to human assessors to determine their appropriateness and the performance of the proposed approach. The first one of these descriptive words was derived with relations from Wikipedia while the second one were derived with relations extracted from WordNet to show that the approach works well with semantic relations of a high quality. The third one was a noun contained by the cluster with the minimum local curvature value of all nouns in the cluster as a baseline. To further investigate the suitability of the approach for long term archives, initial experiments were performed on a long term archive, namely The Times Archive.

6.2. Conclusion and future work

Overall, the results of the performed experiments and the evaluation showed that the approach provides good results as long as the amount and quality of semantic relations is sufficient. However, the experiments also showed that the used pattern-based method to extract semantic relations does not provide enough semantic relations. The appropriateness of descriptive words discovered with these relations was only slightly better than the appropriateness of descriptive words proposed by the baseline. Despite the domain-specific knowledge contained by Wikipedia, the approach provided twice as much appropriate descriptive words with relations from WordNet than with relations extracted from text. On the other hand, the approach was able to label more clusters with descriptive words with relations from Wikipedia than with relations from WordNet. Therefore, the best way to label clusters in a fully automatic way seems to be using relations from WordNet and combining them with relations extracted from text.

To increase the appropriateness of descriptive words, the first step is to increase the number of extracted relations. Additional patterns can be used to search also for appositions (e.g. with the pattern "...NP, a NP,...") or meronymy relations (e.g. with patterns such as "NP's NP" and "NP of NP"). Another way is to allow also noun phrases and to ignore adjectival quantifiers (e.g. "some" or "other") or comparatives (e.g. "important" or "smaller") when parsing the text. Because the used patterns provided fewer relations for earlier periods of time in The Times Archive, the discovery of additional patterns is necessary to use the approach also

on long term archives. This could be achieved using methods to learn patterns automatically from text.

An advanced filtering of erroneous relations in an additional step before building the relation graph seems to be necessary. This also includes the automatic filtering of correct but not suitable relations which have been removed manually for the performed experiments. However, such a filtering is reasonable only if more relations can be extracted. Applying a filtering to this small amount of relations would also remove correct relations that are found at most once in the text.

The evaluation also showed that the mean scores of appropriate descriptive words is higher than for descriptive words which are not appropriate. However, the score can not be used to make a clear statement if a descriptive word is appropriate or not. This problem could be solved automatically if more relations become available. Vertices which cover nouns even if there is no path between them in the relation graph would be reduced to a minimum. Hence, this would lead to more meaningful coverage values. Another solution for this problem could be to exchange the weighting function or adjusting its parameters to calculate more explicit scores.

Bibliography

[BC99] Matthew Berland and Eugene Charniak. Finding Parts in Very Large Corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 57–64, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[BJG07] Jørgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer-Verlag, second edition, 2007.

[Car99] Sharon A. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 120–126, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[Cob] Aaron Coburn. Lingua::EN::Tagger - Part-of-speech tagger for English natural language processing. http://search.cpan.org/~acoburn/Lingua-EN-Tagger-0.15/Tagger.pm.

[Cru86] David Allan Cruse. *Lexical Semantics*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, UK, 1986.

[CW03] Scott Cederberg and Dominic Widdows. Using LSA and Noun Coordination Information to Improve the Precision and Recall of Automatic Hyponymy Extraction. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 111–118, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Die05] Reinhard Diestel. *Graph theory*. Springer-Verlag, third edition, 2005.

[Dor07] Beate Dorow. *A Graph Model for Words and their Meanings*. PhD thesis, University of Stuttgart, March 2007.

[DW03] Beate Dorow and Dominic Widdows. Discovering Corpus-Specific Word Senses. In *EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, pages 79–82, Budapest, Hungary, 2003.

[Fel98] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[FHE03] Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. Offline Strategies for Online Question Answering: Answering Questions Before They Are Asked. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 1–7, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Fin] Mark A. Finlayson. MIT Java Wordnet Interface (JWI), version 2.1.5. http://projects.csail.mit.edu/jwi. Released under Creative Commons Attribution-NonCommerical Version 3.0 Unported License.

[GBM06] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Automatic Discovery of Part-Whole Relations. *Computational Linguistics*, 32(1):83–135, 2006.

[GMV08] Chinnappa Guggilla, Sadiq Mohamed, and Sridhar Vardarajan. Extracting semantic sets from the World Wide Web using unsupervised learning. In *JADT 2008: Proceedings of 9th International Conference on Textual Data statistical Analysis*, pages 539–547. Lyon: Presses Universitaires de Lyon (PUL), 2008.

[Hea92] Marti A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[JGT] JGraphT, version 0.8.1. http://www.jgrapht.org. Released under GNU Lesser General Public License Version 2.1.

[JK06] Theresa Jickels and Grzegorz Kondrak. Unsupervised Labeling of Noun Clusters. In *Canadian Conference on AI*, volume 4013 of *Lecture Notes in Computer Science*, pages 278–287. Springer, 2006.

[KP05] Anagha Kulkarni and Ted Pedersen. SenseClusters: Unsupervised Clustering and Labeling of Similar Contexts. In *ACL*. Association for Computer Linguistics, 2005.

[KRH08] Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056, Columbus, Ohio, 2008. Association for Computer Linguistics.

[Lyo77a] John Lyons. *Semantics*, volume 1. Cambridge University Press, New York, 1977.

[Lyo77b] John Lyons. *Semantics*, volume 2. Cambridge University Press, New York, 1977.

[Man02] Gideon S. Mann. Fine-Grained Proper Noun Ontologies for Question Answering. In *COLING-02 on SEMANET*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[MBF+90] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An On-line Lexical Database. *Journal of Lexicography*, 3(4):235–244, 1990.

[PL02] Patrick Pantel and Dekang Lin. Discovering Word Senses from Text. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619, Edmonton, Alberta, Canada, 2002. ACM.

[PR04] Patrick Pantel and Deepak Ravichandran. Automatically Labeling Semantic Classes. In *Proceedings of Human Language Technology/North American chapter of the Association for Computational Linguistics (HLT/NAACL-04)*, pages 321–328, Boston, MA, USA, 2004.

[PRH04] Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. Towards Terascale Knowledge Acquisition. In *Proceedings of the 20th international conference on Computational Linguistics (COLING-04)*, pages 771–777, Geneva, Switzerland, 2004. Association for Computational Linguistics.

[Pri] Derek Price. Text::MediawikiFormat - Translate Wiki markup into other text formats, version 1.0. http://search.cpan.org/~dprice/Text-MediawikiFormat/lib/Text/MediawikiFormat.pm.

[RC98] Brian Roark and Eugene Charniak. Noun-phrase Co-occurrence Statistics for Semi-automatic Semantic Lexicon Construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 1110–1116, Montreal, Canada, 1998.

[Res99] Philip Resnik. Semantic Similarity in a Taxonomy An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[Rid] Tyler Riddle. Parse::MediaWikiDump - Tools to process MediaWiki dump files, version 1.0.4. http://search.cpan.org/~triddle/Parse-MediaWikiDump/lib/Parse/MediaWikiDump.pm.

[RS97] Ellen Riloff and Jessica Shepherd. A Corpus-Based Approach for Building Semantic Lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 109–116, 1997.

[Sch94] Helmut Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, 1994.

[Sch98] Heinrich Schütze. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[SJN05] R. Snow, D. Jurafsky, and A.Y. Ng. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems*, 17:1297–1304, 2005.

[Tim] The Times Archive. http://archive.timesonline.co.uk/tol/archive/.

[Tim14] The Times, November 29, 1814. http://archive.timesonline.co.uk/tol/viewArticle.arc?articleId=ARCHIVE-The_Times-1814-11-29-03-003&pageId=ARCHIVE-The_Times-1814-11-29-03.

[TNTR10] Nina Tahmasebi, Kai Niklas, Thomas Theuerkauf, and Thomas Risse. Using Word Sense Discrimination on Historic Document Collections. In *(to appear in) 10th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, Surfers Paradise, Gold Coast, Australia, June 21-25, 2010.

[WD02] Dominic Widdows and Beate Dorow. A Graph Model for Unsupervised Lexical Acquisition. In *COLING*, 2002.

[Wid03] Dominic Widdows. Unsupervised Methods for Developing Taxonomies by Combining Syntactic and Statistical Information. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, pages 276–283, Edmonton, Canada, 2003.

[Wik] Wikipedia, the free encyclopedia. http://en.wikipedia.org.

[WN] WordNet - a lexical database for English. http://wordnet.princeton.edu.

[WS98] D.J. Watts and S. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.

A. Appendix

A.1. Calculation of covered vertices by finding a maximum flow

This approach is based on finding a maximum flow in the relation graph. The basic idea is to transform the coverage problem into the problem of finding a maximum flow in a modified relation graph D'. Transforming graph theoretical problems into maximum flow problems has been applied to other graph-theoretical problems before, for example for the Marriage theorem (Hall's theorem) to determine if a perfect matching in a bipartite graph exists [BJG07, pp.173-174].

Before finding a maximum flow in the relation graph, it has to be prepared to be applicable for maximum flow algorithms. Therefore it is transformed into a directed graph D'. Also a capacity function $c : A_{D'} \to \mathbb{N}$ is defined to be later combined with D' to define a network. The graph D' is initialized with all vertices contained by D. The first step after initialization of the graph D' is to invert all arcs of D, i.e. for each arc the head and the tail are exchanged, and to add them to the graph D'. This inversion is not mandatory but reduces the computational effort for finding a maximum flow later. The second step is to add a dummy vertex $DUMMY$ to the new graph D' and arcs with from each cluster vertex to the new dummy vertex. The third and last modification step is to create the capacity function c which is defined for all arcs in the graph D'. All arcs which have been added in the second step get a capacity of 1 while for all other arcs the capacity $\#C$ is assigned. This can be written as

$$a = (u, v) \in A_{D'} \mapsto c(a) = \begin{cases} 1 & \text{if } v = \text{DUMMY} \\ \#C & \text{else.} \end{cases}$$

After D' has been build, a network $\mathcal{N}$ can be defined with the directed modified relation graph D' and the capacity function c. Lower bounds are not needed for

Algorithm A.1 Algorithm to determine coverage values for vertices with maximum flow approach

Require: Relation graph $D = (V_D, A_D)$;
Require: Cluster C;

// Create modified relation graph D' and capacity function
Vertices $V'_D = V_D$;
Arcs $A'_D = \emptyset$;
Graph D' = new Graph(V'_D, A'_D);
Capacity function c = new Map<Vertex,integer>;

// — First modification step: Add inverted arcs of D to D'
for all $a = (v_1, v_2) \in A_D$ **do**
  $A'_D \leftarrow A'_D \cup (v_2, v_1)$;
end for
// — Second modification step: Add dummy vertex and edges to it
$V'_D \leftarrow V'_D \cup \{\text{DUMMY}\}$;
for all $v \in C$ **do**
  $A'_D \leftarrow A'_D \cup \{(v, \text{DUMMY})\}$;
end for
// — Third modification step: Define capacity function
for all $a = (v_1, v_2) \in A'_D$ **do**
  if v_2 = DUMMY **then**
    $c(a) \leftarrow 1$;
  else
    $c(a) \leftarrow \#C$;
  end if
end for

// Calculate max. flow for all vertices, initialize storage for covered vertices
Map *CovVerticesOf* = new Map<Vertex,Set<Vertex>>;
for all $v \in D'$ **do**
  Flow f = calcMaxFlow(G', c, v, DUMMY);
  CovVerticesOf$(v) \leftarrow C \cap$ verticesOf(f);
end for

// This function returns all covered vertices for a given vertex
function coveredVertices(*vertex*)
    return *CovVerticesOf*(*vertex*);

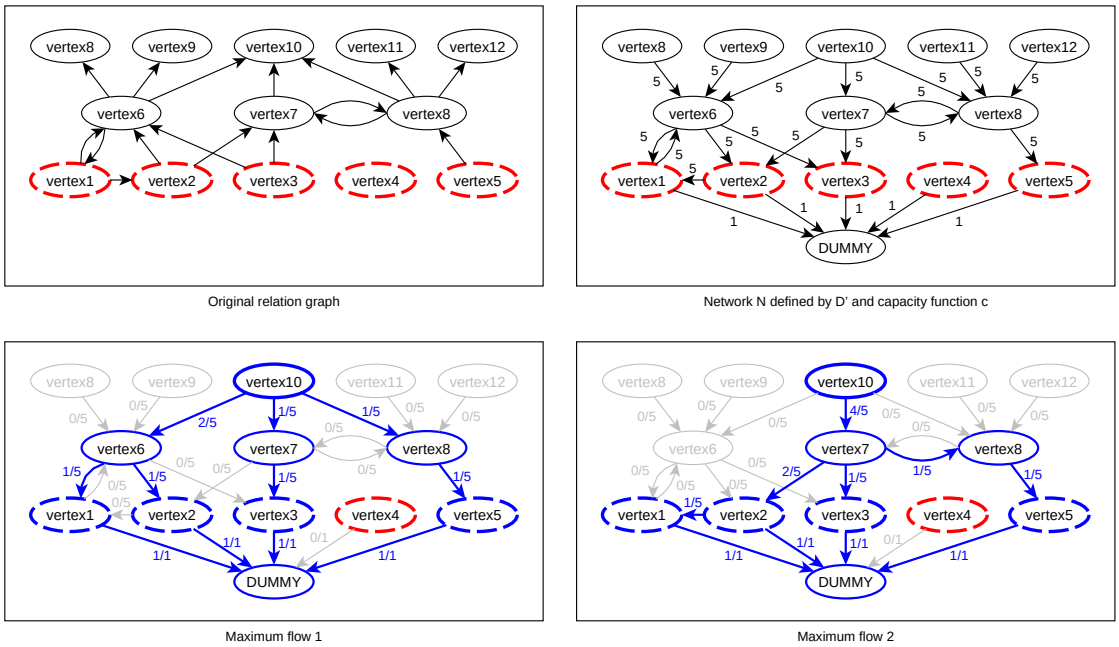


Figure A.1.: Relation graph example (top left diagram) with its associated network (top right diagram) defined by the modified graph D' and the capacity function c. Additionally two possible maximum flows (diagrams on the bottom) with the vertex "vertex10" as source and the vertex "DUMMY" as sink. Arcs are labelled with <*flow value*>/<*capacity*>.

arcs, therefore the function for lower bounds is set to $l \equiv 0$. In the shorthand notation of Section 2.1.1 the network can be written as:

$$\mathcal{N} = (V_{D'}, A_{D'}, l \equiv 0, c).$$

The number of covered words for an examined vertex of the relation graph is the value of the maximum flow from the examined vertex as source and the dummy vertex as sink in the network $\mathcal{N}$. The assignment of the capacity 1 to arcs with the dummy vertex as head and cluster vertices as tail equates with counting the covered vertices. Since all capacities are integers, also the maximum flow is an integer flow (see Section 2.1.2).

To get all covered words of an examined vertex, the tails of all arcs $(v, DUMMY)$ which have the dummy vertex $DUMMY$ as head and which participate in the maximum flow with a value $f_{v,DUMMY} > 0$ can be taken. Alternatively, the intersection of cluster vertices and vertices participating in the maximum flow can been taken

as covered words. The complete procedure of finding covered words is also given in Algorithm A.1. In the upper two diagrams of Figure A.1, the relation graph, which has been introduced in Figure 3.5 of Section 3.4.1, and its associated network $\mathcal{N}$ are given in the two topmost diagrams. It can be seen that the network contains the inverted arcs of D as well as the additional vertex $DUMMY$ which is the head of all additional arcs from cluster vertices to the vertex $DUMMY$. Examples for two possible maximum flows from the vertex *vertex10* to the dummy vertex in the network are given in the lower two diagrams of Figure A.1. The left diagram presents a maximum flow with all paths over covered vertices being shortest paths from the source over a covered vertex to the sink. The right hand graph is also a maximum flow but contains longer paths, namely the paths over *vertex1* and *vertex5*.

Because shortest paths between the examined vertex and its covered vertices are needed for the calculation of the specificity later, it is reasonable to use an algorithm which already supports the calculation of a maximum flow with shortest paths which are routed over the cluster vertices for this type of networks. Therefore, the Edmonds–Karp algorithm which has been described in Section 2.1.2 has been chosen. This algorithm augments an initial flow $f \equiv 0$ along the shortest available paths in the residual network in each step. Assigning the capacity $\#C$ to all arcs which have not the dummy vertex as head ensures that the Edmonts-Karp algorithm is able to find the shortest path for each covered vertex from the examined vertex over the covered vertex to the dummy vertex. The capacity of $\#C$ enables the algorithm to choose all arcs as often as needed to find such a path through the network. At maximum, an arc can be used for each cluster vertex, i.e. all shortest paths are routed over this arc. However, shortest paths does not have to be unique. To give an example, in the left bottom left diagram of Figure A.1 and the path from the source *vertex10* over the covered vertex *vertex2* to the sink $DUMMY$, the arc $(vertex6, vertex2)$ can be changed to $(vertex7, vertex2)$ while the length of the path, which is now routed over *vertex7* instead of *vertex6*, stays the same. But since only the length of these shortest paths is needed later, this is not a problem.

Instead of using the Edmond-Karp algorithm, any other maximum-flow algorithm can be taken. If it does not support the implicit calculation of shortest paths, this can be done in a separate step. Due to the computational complexity of $O(|A_D|^2 \cdot |V_D|)$ and the higher amount of edges than vertices, the Dinic algorithm [BJG07, pp.148-149] would be even better applicable than the Edmonds-Karp algorithm with the complexity $O(|A_D| \cdot |V_D|^2)$. Since the quality is in these first experiments more important than the performance and a well tested imple-

mentation of the Edmond-Karp algorithm is already contained by the used Java library JGraphT [JGT], the algorithms have not been exchanged so far.

The reason for inverting the arcs of D in the first relation graph modification step is, that less vertices have to be considered when calculating a maximum flow. Starting with the dummy vertex as source to the examined vertex as sink would force any chosen algorithm to walk through the whole network, since every vertex is reachable starting from the dummy vertex and each of the vertices could have an arc to the examined vertex. If inverting the arcs, the dummy vertex is reachable from every vertex in the relation graph, including the examined vertex. But there can be other vertices in the relation graph which are not reachable from the examined vertex. An example is given in the upper right diagram of Figure A.1. In this example, the vertex *vertex8* is, amongst other vertices, not reachable for the algorithm starting with *vertex10* as source. Therefore, the algorithm only has to process a smaller subset of vertices in the network and not the whole network.

This approach to determine covered vertices has been implemented first due to the advantage that it performs independent from the structure of the relation graph. Another advantage is the flexibility for future purposes which is given by the possibility to apply other capacities to the arcs in the network. This could be a capacity depending on the frequency of associated semantic relations in R. Another possibility is choosing a capacity depending on the degree of head or tail vertices. For this type of modification it has to be considered that the calculation of shortest paths as a secondary product could be disabled. This type of modification can be seen as adding more restrictions for the algorithm to chose arcs for a maximum flow. For this first implementation, the edge capacities are taken as described to get first results. A further modification could be to extend the network by assigning additional costs to arcs and searching for a maximum flow with minimum costs. With this modification two vertices with the same coverage but different maximum flows could be differentiated by their different costs additional to the specificity.

Algorithm A.2 Erroneous algorithm for the determination of covered vertices during the relation graph building procedure

```
// Initialize the storage of covered vertices for each vertex in the graph
// Given: Cluster C = {noun_1, noun_2, ..., noun_n}
Map CovVerticesOf = new Map<Vertex,Set<Vertex>>;
for all noun ∈ C do
   CovVerticesOf(noun) ← {noun};
end for

// The following code has to be executed every time a new arc is added
if new arc arc = (tail, head) has been added then
   CovVerticesOf(head) ← CovVerticesOf(head) ∪ CovVerticesOf(tail);
end if

// This function returns all covered vertices for a given vertex
function coveredVertices(vertex)
     return CovVerticesOf(vertex);
```

A.2. Calculation of covered vertices during the relation graph building procedure

An alternative for the two approaches building a Breadth-First Search tree and finding a maximum flow seems to be the storing of covered cluster vertices while building the relation graph. Covered vertices could be stored separately during the building procedure. The covered vertices for a certain vertex could be updated every time an edge with the certain vertex as target is added to the relation graph. The target vertex would then inherit all covered vertices from the source vertex. This approach is given in pseudo code by Algorithm A.2.

Although this approach seems to be working and to be the fastest one, it does not provide the right results for a simple reason. This becomes clear considering Figure A.2. In the first step of the algorithm the vertex *vertex3* and the two arcs *a* and *b* are added to the relation graph. The result of Algorithm A.2 depends on the order of adding the arcs. If *a* is added first, *vertex2* is recognized to cover *vertex1* and after adding the arc *b* also *vertex3* is recognized to do this. In the reverse order this is not the case, because the covered vertices for *vertex3* are updated

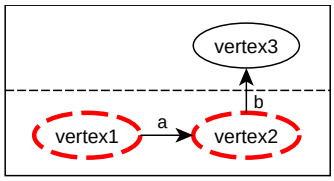


Figure A.2.: Counterexample for the determination of covered words during the building procedure of the relation graph

only when adding the arc *b*. Since the arc *a* has not been added before, it never gets recognized that *vertex3* covers *vertex1*.

The only solution is doing a complete search for covered vertices for all vertices in the relation graph after each completed step of the algorithm or after doing the last step. Since the information of covered vertices in a previous step of the algorithm can not be used for the search for covered vertices in the next step, performing a search in every step is not reasonable. Instead a search after the last step is sufficient, for example implemented by Breadth-First Search as used in the approach using a Breadth-First Search tree.

A.3. Baselines for evaluation

As a baseline for the evaluation, a noun in the cluster has been taken as a proposed descriptive word. The selection of a noun in the cluster can be done in several ways based on information which is already available as a secondary product of the curvature clustering approach to extract clusters. From the co-occurrence graph, related to the subset the cluster has been extracted from, a subgraph can be extracted. This subgraph contains all vertices associated to nouns in the cluster and all edges between them in the co-occurrence graph. This subgraph can be used to calculate a local curvature value and the degree, i.e. the number of neighbours, of all vertices in the subgraph. Based on the local curvature value and the degree, the following baseline approaches have been tested to be suitable in preliminary experiments:

1. Select vertices with the lowest local curvature value.
2. Select vertices with the highest local curvature value.
3. Select vertices with the highest number of neighbours.
4. Select a random vertex from the cluster.

For the first baseline approach, vertices with a local curvature value of 0.0 have been ignored. These vertices are in particular vertices which have only one neighbour. In the preliminary experiments, the first and third approach provided the best results, i.e. vertices representing nouns which are most descriptive for its own cluster amongst all nouns in the cluster. The second approach provided worse results than the first and third approach. The reason for this is that the approach returns in particular vertices with less neighbours. Such vertices are for example vertices which have two neighbours being interconnected. These vertices have the maximum curvature value of 1.0. The last baseline approach of choosing a vertex randomly provided unacceptable results. Since the first approach seems to provide slightly better results than the third one, it has been taken as baseline for the evaluation.

A.4. Additional tables and figures

Figure A.3.: Number of extracted semantic relations from The Times Archive for the patterns from Section 3.2.1. Spanning from 1785 to 1985, each fifth year has been parsed for semantic relations. The highlighted graph represents all patterns.

Subset: **Techology and Applied sciences**
applied sciences, *technology*, computer science, optics, engineering, archaeology, architecture, measurement, health sciences, applied mathematics, food science, forestry, industrial design, applied psychology, industrial processes, ergonomics, forensics, applied genetics, military science, aeronautics, agronomy, horology, sports science, maintenance, metrology, pharmaceutical sciences, ceramic engineering, applied linguistics, wikipedia:books on applied sciences, programming languages, algorithms, data structures, operating systems, computer scientists, artificial intelligence, software engineering, computer graphics, computer security, computer architecture, theoretical computer science
Subset: **Sports**
sports, sportspeople, sports by year, multi-sport events, sports officiating, sporting comparison, sports technology, history of sports, sports culture, sports organisations, sports terminology, sports media, sports-related lists, sport by city, sports records and statistics, sports venues, sports medicine, sport and politics, sports in fiction, sports clubs, sports science, animals in sport, sports law, sports business, sport by continent, violence in sports, sports equipment, sports instruction, sports rules and regulations, sports controversies, years in sport, categories by sport, sports and entertainment skills, set indices on sports, sport by country, sports by country, gaelic games, sexual orientation and sports, wikipedia:books on sports, sports robots
Subset: **Religion and Belief**
religion, *belief*, religious faiths, traditions, and movements, sexuality and religion, deities, spirits, and mythic beings, religious objects, religion and society, religious behaviour and experience, religious architecture, religion-related lists, religious ethics, religious conversion, religion in fiction, religion portals, disengagement from religion, religion and science, religious pluralism, religious controversies, geography of religion, religious occupations, religious demographics, categories by religion, freemasonry and religion, environment and religion, religious literature, opposition to religion, religious events, religious writings, people associated with religion, religious belief and doctrine, religious philosophy, punishments in religion, salvation, religious antisemitism, wikipedia:books on religion, religion and children, gender and religion, religious organizations, glossaries on religion, water and religion
Subset: **Food and drink**
food and drink, cuisine, restaurants, meals, famines, beverages, gustation, foods, people in food and agriculture occupations, organic food, eating behaviors, food safety, food industry, food law, ceremonial food and drink, wikiproject food and drink, food festivals, food and drink portals, food museums, food politics, history of food and drink, nutrition, food and drink terminology, food riots, food and drink preparation, food awards, food and drink in canada, food and drink media, food-related lists, food and drink appreciation, food allergies, environmental issues with food, food culture, foodservice, wikipedia:books on food and drink, food and drink images, food and the environment, hunger, thirst, serving and dining
Subset: **Geography**
geography, geocodes, geography stubs, exploration, geographers, surveying, geographic images, geographical technology, history of geography, geography-related lists, geography portals, geography by place, branches of geography, landscape, geography awards and competitions, geography education, geography by time, geography terminology, geographic literature, geography organizations, geography and place templates, places, music geography, wikipedia:books on geography, vernacular geography, lists of postal codes, country codes, iso 3166, nuts, icao airport designator, iata airport designator, telephone numbers by country, geolocation, fips 10, un/locode, geographic coordinate lists, airport stubs, africa geography stubs, antarctica geography stubs, oceania geography stubs

Table A.1.: Categories in category trees for the different subsets. The first 30 categories are listed for each category tree in order of their visit when extracting the category trees from the complete category hierarchy of Wikipedia. The root categories are emphasized.

Subset: **Techology and Applied sciences**
(*viscoelastic*, *polyurethane*), (*au*, *millionfold*), (*bot*, *extension*), (*ocal*, *abbreviation*), (*atmosphere*, *medium*), (*mictlantecuhtli*, *fictional*), (*macintosh*, *aquarium*), (*terrier*, *dog*), (*product*, *nucleophilic*), (*rakehoe*, *tool*), (*walt*, *scholar*), (*earlham*, *campus*), (*streptomycin*, *protein*), (*integration*, *ideal*), (*autostitch*, *program*), (*leave*, *material*), (*amplitude*, *sound*), (*kiewit*, *sponsor*), (*drakensberg*, *essential*), (*umlaut*, *type*), (*coincidence*, *clf*), (*vgmaps*, *site*), (*ing*, *fine*), (*tenochtitlan*, *city*), (*glac*, *fruit*), (*inhibitor*, *function*), (*lightning*, *cause*), (*jewelry*, *type*), (*revolution*, *book*), (*traffic*, *task*)
Subset: **Sports**
(*jr*, *baseball*), (*rink*, *variant*), (*union*, *minor*), (*regis*, *city*), (*image*, *little*), (*president*, *active*), (*swordsmanship*, *canadian*), (*facility*, *cost*), (*islanders*, *professional*), (*lake*, *facility*), (*toppfotball*, *football*), (*murphy*, *american*), (*title*, *bit*), (*following*, *list*), (*loperamide*, *medication*), (*orey*, *theme*), (*machineguns*, *item*), (*ball*, *ball*), (*canberra*, *team*), (*john*, *coach*), (*reffner*, *veteran*), (*louisville*, *ice*), (*stadium*, *major*), (*united*, *american*), (*passage*, *monica*), (*india*, *subcontinent*), (*kov*, *czech*), (*winn*, *switch*), (*world*, *event*), (*certificate*, *maximum*)
Subset: **Religion and Belief**
(*baptism*, *mystery*), (*maus*, *communications*), (*silence*, *charity*), (*aesthetics*, *philosophy*), (*swami*, *personality*), (*holder*, *chinese*), (*planet*, *solar*), (*includes*, *term*), (*marriage*, *companion*), (*eastern*, *world*), (*kid*, *magic*), (*miravalle*, *professor*), (*term*, *charm*), (*sialkot*, *district*), (*title*, *culture*), (*kosher*, *voluntary*), (*independent*, *term*), (*william*, *reformer*), (*christos*, *student*), (*imamichi*, *supporter*), (*burma*, *majority*), (*detraction*, *sin*), (*feature*, *crucial*), (*epiphenomenon*, *consequence*), (*peanut*, *allergy*), (*dance*, *activity*), (*element*, *government*), (*laddoo*, *sweet*), (*burung*, *alleged*), (*philosophy*, *necessity*)
Subset: **Food and drink**
(*choripan*, *street*), (*raat*, *time*), (*indonesia*, *nation*), (*coxinha*, *chicken*), (*bog*, *temporary*), (*daphnia*, *invertebrate*), (*gate*, *designated*), (*death*, *serious*), (*mutton*, *filling*), (*rum*, *dark*), (*digestive*, *medicinal*), (*deepview*, *program*), (*basic*, *science*), (*fritos*, *example*), (*equation*, *pair*), (*siegfried*, *includes*), (*canopy*, *management*), (*garden*, *specie*), (*international*, *celebration*), (*trawler*, *type*), (*irritability*, *condition*), (*precipitation*, *technique*), (*cisapride*, *parasympathomimetic*), (*tamales*, *treat*), (*beef*, *meat*), (*dam*, *dam*), (*reignition*, *process*), (*erinacea*, *species*), (*inn*, *restaurant*), (*special*, *documentary*)
Subset: **Geography**
(*dam*, *hydroelectric*), (*pacific*, *band*), (*makossa*, *type*), (*lloyd*, *alumnus*), (*buddhism*, *religion*), (*airport*, *feature*), (*track*, *representation*), (*buchanan*, *city*), (*special*, *numbering*), (*royal*, *award*), (*dr*, *conservationist*), (*attack*, *violent*), (*capital*, *mongolia*), (*water*, *volatiles*), (*series*, *apis*), (*classic*, *album*), (*dla*, *pinion*), (*gustaf*, *delegation*), (*wandernadel*, *network*), (*dock*, *fixed*), (*kuwait*, *sovereign*), (*austin*, *neighborhood*), (*music*, *key*), (*conurbation*, *urban*), (*hall*, *national*), (*cleomedes*, *philosopher*), (*island*, *archipelago*), (*tuli*, *reserve*), (*casino*, *shop*), (*paddingtons*, *english*)

Table A.2.: Hypernym relations extracted from Wikipedia subsets, each subset with 30 sample relations. Relations have been extracted with patterns as described in Section 3.2.1 and are choosen randomly from the complete set of relations.

Subset: **Techology and Applied sciences**				
example (2571)	device (1756)	application (1375)	company (1165)	**common** (1098)
type (2284)	feature (1538)	method (1338)	language (1162)	major (947)
system (2088)	**form** (1488)	term (1269)	game (1121)	factor (916)
material (1779)	country (1421)	product (1179)	technique (1098)	service (906)

Subset: **Sports**				
player (1628)	former (969)	event (667)	game (430)	indoor (308)
american (1062)	professional (746)	football (593)	stadium (377)	star (302)
sport (1013)	**list** (736)	annual (535)	australian (368)	canadian (275)
team (981)	club (720)	country (436)	**type** (315)	rugby (274)

Subset: **Religion and Belief**				
form (965)	country (653)	religion (420)	language (352)	christian (332)
term (897)	organization (508)	**type** (375)	**list** (342)	**common** (326)
example (824)	concept (450)	figure (367)	scholar (339)	movement (323)
book (671)	fictional (431)	issue (366)	philosopher (337)	writer (309)

Subset: **Food and drink**				
ingredient (859)	product (730)	popular (634)	**common** (474)	vegetable (408)
type (838)	dish (713)	american (531)	**example** (457)	fish (394)
brand (830)	country (674)	variety (496)	protein (422)	spice (339)
food (796)	species (655)	traditional (477)	specie (415)	item (338)

Subset: **Geography**				
family (1574)	city (432)	**form** (266)	island (202)	medium (193)
list (785)	artist (355)	**example** (259)	major (200)	name (193)
country (585)	region (310)	**type** (217)	international (199)	adaptation (186)
band (439)	instrument (296)	diet (210)	term (195)	shorebirds (183)

Table A.3.: Top 20 most frequent hypernyms extracted from Wikipedia subsets. Hypernyms for which semantic relations with these hypernyms as superordinates have been removed are highlighted.

Subset: **Techology and Applied sciences**			
{zeppelin, aircraft, mail, dirigible, balloon, airship}	aircraft (WIKI, 3, 0.44)	aircraft (WN, 3, 0.42)	airship (MLC, 2, 0.33)
{pen, pastel, drawing, oil, chalk, watercolor, charcoal, pencil, ink}	product (WIKI, 2, 0.42)	writing implement (WN, 3, 0.22)	pastel (MLC, 0, 0.4)
{friend, relative, parent, child, caregiver, patient, loved, family}	factor (WIKI, 0, 0.34)	person (WN, 3, 0.44)	family (MLC, 2, 0.61)

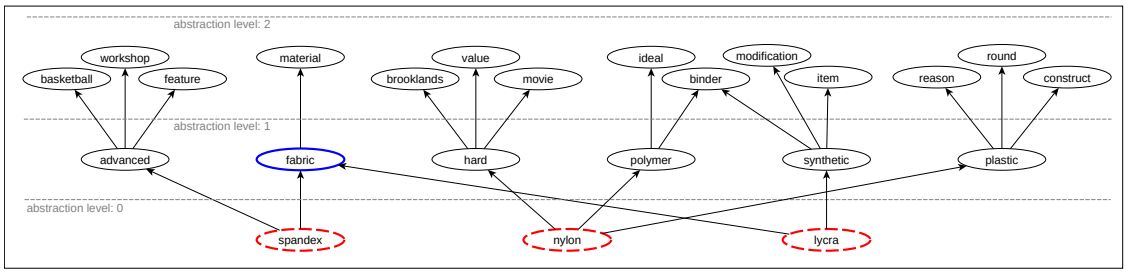
Subset: **Sports**			
{trial, supercross, motocross, racing, enduro, bmx, motorcross, road}	sport (WIKI, 3, 0.41)	experiment (WN, 0, 0.06)	enduro (MLC, 3, 0.33)
{heart, liver, brain, lung, kidney}	organ (WIKI, 2, 0.19)	internal organ (WN, 3, 0.45)	kidney (MLC, 0, 0.33)
{decathlon, sport, jump, heptathlon, pentathlon}	sport (WIKI, 3, 0.4)	athletic contest (WN, 3, 0.19)	pentathlon (MLC, 0, 0.33)

Subset: **Religion and Belief**			
{booklet, newspaper, pamphlet, article, book, child, essay, leaflet}	book (WIKI, 0, 0.47)	writing (WN, 3, 0.34)	pamphlet (MLC, 0, 0.33)
{window, doorway, carving, wall, door, roof}	element (WIKI, 2, 0.3)	building (WN, 3, 0.49)	window (MLC, 0, 0.3)
{impalement, disembowelment, crucifixion}	term (WIKI, 0, 0.34)	change (WN, 3, 0.2)	crucifixion (MLC, 2, 1.0)

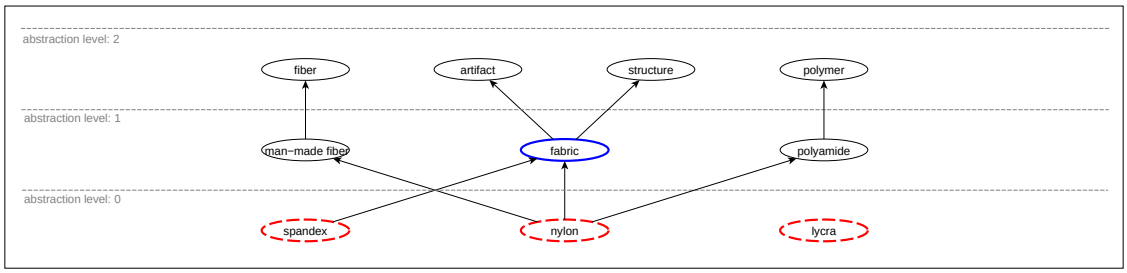
Subset: **Food and drink**			
{dietitian, doctor, physician, nutritionist}	health (WIKI, 1, 0.49)	medical practitioner (WN, 2, 0.5)	nutritionist (MLC, 2, 0.33)
{pen, pastel, drawing, oil, chalk, watercolor, charcoal, pencil, ink}	product (WIKI, 2, 0.42)	writing implement (WN, 3, 0.22)	pastel (MLC, 0, 0.4)
{spaghetti, rigatoni, penne, fusilli }	dish (WIKI, 1, 0.38)	pasta (WN, 2, 0.49)	penne (MLC, 1, 0.33)

Subset: **Geography**			
{mars, lander, probe, orbiter }	space (WIKI, 3, 0.63)	equipment (WN, 1, 0.18)	orbiter (MLC, 1, 0.33)
{overgrazing, desertification, soil, deforestation, water}	issue (WIKI, 0, 0.22)	earth (WN, 2, 0.16)	deforestation (MLC, 1, 0.5)
{independent, punk, historical, rock, alternative, experimental, metal}	music (WIKI, 2, 0.54)	person (WN, 1, 0.12)	alternative (MLC, 0, 0.33)

Table A.4.: Examples for clusters and their discovered descriptive words together with their number of positive votes to be appropriate (3 votes are the maximum) and their score in the format (<Method>, <Positive votes>, <Score>).



Relation graph for {spandex, nylon, lycra} with relations extracted from Wikipedia



Relation graph for {spandex, nylon, lycra} with relations extracted from WordNet

Figure A.4.: Relation graphs for {*spandex*, *nylon*, *lycra*} with relations extracted from Wikipedia in the upper graph and relations extracted from WordNet in the lower graph. Both graphs are build with parameters $maxAS = 2$ and $maxODV = 3$. The approach chooses the vertex *fabric* as descriptive word candidate in both graphs.

Acknowledgements

My special thanks go to my supervisor Nina Tahmasebi for her great support and huge amount of spent time to help me correcting my written English. Great thanks go to Kai Niklas, who wrote an own diploma thesis simultaneously at the same institute. He supported me not only by providing ideas concerning factual issues but also as a friend. Furthermore, I would like to thank all assessors which participated in the evaluation for this thesis. Last but not least, the biggest thanks go to my family, who made my studies possible at all.

We would like to thank Times Newspapers Limited for providing the archive of The Times for our research.

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe.

Thesis Declaration

I hereby certify that this thesis is my own and original work using the sources and methods stated therein.

Thomas Theuerkauf Hannover, den 11.06.2010