5 Preprocessing of Text Corpus

Relevant information in the natural language texts is not solely characterized by lexical expressions. Looking for linguistic regularities in the expression of certain information the syntax and morphology of the language are often even more important than the exact phrases that contain it. Besides, text layout and original markup of a document can provide additional features of extracted information. Therefore linguistic patterns capture all sources of metadata describing the actual text that may be helpful in identifying relevant content.

To obtain the metadata original texts have to be analyzed by linguistic tools. This chapter describes the phase of linguistic preprocessing that precedes the induction of extraction rules and the actual extraction. Different stages of linguistic analysis and the employed tools are presented. Furthermore we introduce a uniform XML format for annotated documents that incorporates all obtained information and serves as the basis for learning and application of extraction rules.

The preprocessing stage is concluded by a semantic preprocessing that aims at the recognition of synonymy, which plays an important role in the generalization of extraction rules. While in the linguistic analysis we utilize tools provided by a third party for the semantic preprocessing we developed a statistical method based on a lexical graph. The vertices of the graph represent words and the length of edges between them reflects their statistical cooccurrence. We propose special synonymy metrics and evaluate them on a test set of 200 synonym sets.

5.1 Linguistic preprocessing

Beside the semantics of words, phrases and sentences text structure and linguistic information can play an important role in identifying relevant content. Structural and linguistic analysis offers additional features for characterization of extractions and their context. To recognize structure and linguistic properties of texts we employ external tools.

XML and HTML documents usually contain explicitly encoded structural and layout information so that no additional structural analysis is required. The system can capture the layout elements directly as additional features when building extraction rules. The semantics of the elements is not supposed to be known, since they are captured on a syntactic level. A presence or absence of certain layout elements or recurring XML structure in context of extracted fragments

are reflected by the extraction patterns.

Although plain texts do not include explicit structuring markup, they feature implicit structure that can vary depending on the text kind. Apart from possible division of the text in chapters and sections structuring of smaller text parts is especially valuable for IE. Headings, paragraphs and enumerations can be distinguished by specific layout, position relatively to other text parts, indentation etc. Emphasized text can be recognized by usage of special characters or capitalization. We use *txt2html* [txt] for analysis of the implicit text structure. Recognized layout elements are encoded as HTML tags so that eventually HTML documents with explicit structural information are generated that correspond to the original plain texts. They serve as the input for consecutive linguistic analysis.

Linguistic knowledge can be successfully utilized for the purpose of information extraction. Since many facts are distinguished by a certain syntactic structure, morphological features or represented by a named entity, the additional information about the linguistic properties of the text is essential for their identification. Linguistic preprocessing is accomplished by *TreeTagger* [Tre04] and includes

Tokenization: Starting with a sequence of characters the goal is to identify the elementary parts of natural language: words, punctuation marks and separators. The resulting sequence of meaningful tokens is a base for further linguistic and any text processing.

Sentence Splitting: Sentences are one of the most important elements of the natural language for structured representation of the written content. Binding interrelated information they are the smallest units for expression of completed thoughts or events. Since the extraction rules match and extract information from sentences, the correct recognition of the sentence borders is crucial for the quality of extractions. The task would be trivial if the punctuation marks were not ambiguously used. Correct representation of a text as a sequence of sentences is utilized for syntactic parsing.

Morphological Analysis: Certain facts are typically expressed by certain parts of speech (e.g. names). Part of speech of every token is determined assigning a category from *Penn Treebank Tag Set* [Mar93] in case of texts written in English or *Stuttgart-Tübingen Tagset* [Stu95] in case of German texts. Both tag sets provide a good coverage of morphological diversity accounting, for instance, for mood, number and tense of verbs. Segmentation of compounds, recognition of stem, flection forms and consecutive normalization disclose further important morphological features.

Chunk Parsing: While deep sentence parsing is still the subject of current research, there are lightweight alternatives that provide limited syntactic knowledge and are more reliable. Chunk parsing – a shallow syntactic analysis of the sentence fragments performed on phrasal level – identifies low-level syntactic constituents (e.g. noun, verb, prepositional phrases) without considering deeper syntactic dependencies between them. However, shallow parsing is sufficient for the purpose of IE because extracted information is often completely included in a noun, verb or prepositional phrase that build the most relevant context for its recognition.

Basic Named Entity Recognition Named entities are one of the most often extracted types of tokens. *TreeTagger* utilizes a combined approach to NER based on lookup in predefined lists (e.g. of geographic locations, company names) and decision trees. However, *TreeTagger*'s NER is limited to mere

```
<HTML>
   ...
   <BODY>
   ...
      <P>
         <SENT>
            <CONST TYPE="NC">
               <POS TYPE="CD" NORMAL="2">2</POS>
               <POS TYPE="NE" NORMAL="FEB">FEB</POS>
               <POS TYPE="CD" NORMAL="@card@">89</POS>
            </CONST>
            <POS TYPE=":" NORMAL="–">–</POS>
            ...
            <CONST TYPE="NC">
               <POS TYPE="CD" NORMAL="seven">SEVEN</POS>
               <POS TYPE="NNS" NORMAL="soldier">SOLDIERS</POS>
            </CONST>
            <CONST TYPE="VC">
               <POS TYPE="VBD" NORMAL="be">WERE</POS>
               <POS TYPE="VVN" NORMAL="kill">KILLED</POS>
            </CONST>
            <CONST TYPE="ADJC">
               <POS TYPE="CC" NORMAL="and">AND</POS>
            </CONST>
            <CONST TYPE="NC">
               <POS TYPE="JJ" NORMAL="several">SEVERAL</POS>
            </CONST>
            <CONST TYPE="VC">
               <POS TYPE="VBD" NORMAL="be">WERE</POS>
            </CONST>
            <CONST TYPE="NC">
               <POS TYPE="VVN" NORMAL="wounded">WOUNDED</POS>
            </CONST>
            <CONST TYPE="PC">
               <POS TYPE="IN" NORMAL="by">BY</POS>
               <CONST TYPE="NC">
                  <POS TYPE="DT" NORMAL="a">A</POS>
                  <POS TYPE="NN" NORMAL="bomb">BOMB</POS>
               </CONST>
            </CONST>
            ...
            <CONST TYPE="NC">
               <POS TYPE="NE">ARAUCA</POS>
               <POS TYPE="NN" NORMAL="department">
                  DEPARTMENT
               </POS>
            </CONST>
            <POS TYPE="SENT" NORMAL=".">.</POS>
         </SENT>
      </P>
   </BODY>
</HTML>
```

Figure 5.1: Sample text enriched with structural and linguistic information

identification of named entities without determining their types and is essentially an extended POS tagging.

The results of linguistic analysis are integrated in the processed documents as XML annotations. These documents already contain HTML/XML markup since they either have been augmented with structural information by *txt2html* or originally have been in HTML or XML format. Therefore the linguistic annotations are merged with the existing XML structure of a document resulting in a single XML document, which contains complete information about its linguistic and structural properties and offers a uniform interface for further processing. Figure 5.1 depicts the preprocessed text from the example in fig. 4.3.

In this connection the important role of XML should be underlined: XML is widely accepted as a meta-language that is very suitable for text annotations. We exploit its ability to easily combine and integrate annotations from different sources to generate a single source combining the textual content and all describing information. To avoid the problem of overlapping annotations, standoff annotations can be used alternatively (refer to sec. 6.1). We deliberately chose the merged format because it offers an optimal platform for learning linguistic patterns since patterns comprise every kind of information about the extracted fragments including linguistic, structural, lexical etc. information, which should be reflected in the document format. XML documents are therefore the basis for the rule learning: the extraction rules are generated from and operate not on original plain texts, but on XML documents enriched with structural and linguistic information.

5.2 Semantic Preprocessing: Recognition of Synonyms

Diversity of the natural language is one of the major hurdles for successful information extraction. Especially expression variety is challenging to capture by extraction patterns in order to cover possibilities of information communication written natural language offers. To cope with the language diversity on the word level semantic relations between words such as synonymy and hypernymy can be utilized. In spite of using similar syntactic structures information is often expressed differently replacing certain words by their synonyms. Extraction patterns that are generated from training examples can therefore be generalized replacing a concrete lexical item by a synonym pattern that matches not only a concrete word but also any of its synonyms. E.g. while the pattern *"murder" [PC: "of" [NC]=:victim_target]* matches the text fragment *Murder of General Bustillo*, extending it by the synonym pattern *Synset:"murder" [PC: "of" [NC]=:victim_target]* allows to identify the victim also in the text fragments *Assassination of General Bustillo* and *Homicide of General Bustillo*. The abstraction of extraction patterns by using synonym patterns instead of lexical elements increases the coverage of patterns without significantly extending their semantics because of similar word senses of synonym words.

Semantic relations between words are usually comprised by thesauri that now serve as an important source of semantic and lexical information for automatic text processing. The electronic online thesauri such as WordNet [Wor05] and OpenThesaurus [Ope05] have been increasingly employed for many IR and NLP problems. However, considerable human effort is required to keep up with the evolving language and many subdomains are not sufficiently covered [Tur01]. Many domain-specific words or word senses are not included; inconsistency and

bias are often cited as further major deficiencies of hand-made thesauri [Cur02], [Sen03]. Besides, certain synonym relations that play an important role in one domain are not essential or even wrong in another because of different word senses that are typical for a domain. For example, consider the domain of MUC corpus dealing with terrorist acts where the words *killing* and *assassination* are synonymous and the domain "operating systems" where the word killing is exclusively reserved for denoting termination of programs and processes and the mentioned synonymy is extraneous. For these reasons we abstained from embedding an existing thesaurus and developed an adaptive method for automatic recognition of synonymy [Sin06b], which detects relations that are relevant and useful in a particular domain relying on domain text corpus.

Our method is based on a lexical graph. The graph is generated from a text corpus by embedding linguistically preprocessed sentences in the graph structure. The vertices of the graph are lexical items (words), their connection follows the syntactic structure of a sentence. The structure of the graph and distances between vertices can be utilized to define metrics for identification of semantic relations. The approach has been evaluated on a test set of 200 German synonym sets. Influence of size of the text corpus, word generality and frequency has been investigated. Conducted experiments for synonyms demonstrate that the presented methods can be extended to other semantic relations.

5.2.1 Approaches to Recognition of Synonymy

Identification of semantic relations has been approached by different communities as a component of a knowledge management system or application of a developed NLP framework. Many approaches are guided by the assumption that similar terms occur in similar context and obtain a context representation of terms as attribute vectors or relation tuples [Cur02], [Rug97], [Lin98]. A similarity metric defined on the context representations is used to cluster similar terms (e.g. by the nearest neighbor method). The actual definitions of context (whole document [Che92], textual window, some customized syntactic contexts, cf. [Sen03]) and similarity metric (cf. [Man99], [Cur02]) are the essential distinguishing features of the approaches.

A pattern-based method is proposed by Hearst [Hea98]. Existing relations in the WordNet database are used to discover regular linguistic patterns that are characteristic for these relations. The patterns contain lexical and syntactic elements and are acquired from a text corpus by identifying common context of word pairs for which a semantic relation holds. Identified patterns are applied to a large text corpus to detect new relations. The method can be enhanced by applying filtering steps and iterating over new found instances [Phi02].

Lafourcade and Prince base their approach on reduction of word semantics to conceptual vectors (vector space is spanned by a hierarchy of concepts provided by a thesaurus, [Laf01b]). Every term is projected in the vector space and can be expressed by the linear combination of conceptual vectors. The angle between the vectorial representations of two terms is used in calculation of thematic closeness [Laf01c]. The approach is more closely related to our approach since it offers a quantitative metric to measure the degree of synonymy between two lexical items.

In contrast, Turney [Tur01] tries to solve a quite simpler "TOEFL-like" task of selecting a synonym to a given word from a set of words. Mutual information related to the co-occurrence of two words, which is based on conditional probabilities of co-occurrence obtained by the information retrieval, is used to assess the

degree of their statistical independence. The least independent word is regarded synonymous.

Blondell et al. [Blo04] encode a monolingual dictionary as a graph and identify synonyms by finding subgraphs that are similar to the subgraph corresponding to the queried term.

The common evaluation method for similarity metrics is comparing their performance on the same test set with the same context representations with some manually created semantic source as the gold standard [Cur02]. Abstracting from results for concrete test sets, Weeds et al. [Wee04] try to identify statistical and linguistic properties on that the performance of similarity metrics generally depends. Different bias towards words with high or low frequency is recognized as one reason for the significant variance of k-nearest neighbors sets of different similarity metrics.

5.2.2 Construction of the lexical graph

The assumption that similar terms occur in similar context leads to the establishing of explicit context models (e.g. in form of vectors or relation tuples) by most researchers. We build an implicit context representation connecting lexical items in a way corresponding to the sentence structure (as opposed to [Blo04]), where a term is linked to every word in its definition). The advantage of the graph model is its transitivity: not only terms in the immediate context but also semantically related terms that have a short path to the examined term (but perhaps have never occurred in its immediate context) can contribute to identification of related terms. The similarity metric can be intuitively derived from the distance between the lexical vertices in the graph.

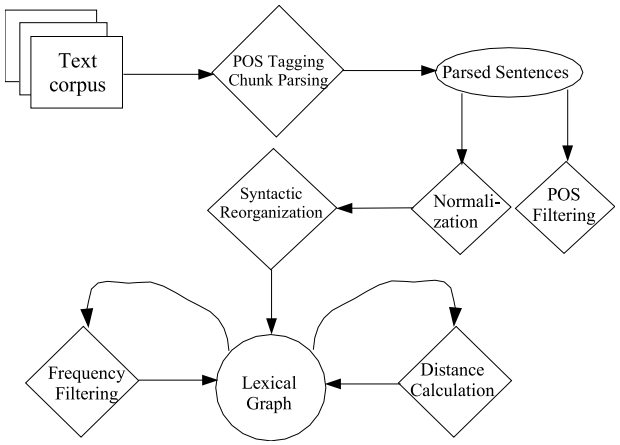

Figure 5.2: Main steps during graph construction

Linguistically annotated documents obtained by linguistic preprocessing are used for construction of the lexical graph. To preserve the semantic structure of the sentences during the graph construction, i.e. to connect words that build the actual statement of the sentence, parsed sentences are modified before being inserted in the graph (fig. 5.2). The punctuation signs and parts of speech that do not carry a self-contained semantics (such as conjunctions, pronouns, articles) are removed in a POS filtering step. Tokenization errors are heuristically removed and the words are replaced by their normal forms (e.g. infinitive form for verbs, nominative singular for nouns). In case of German texts we have to take into account that German grammar is characterized by a very frequent use of auxiliary and modal verbs that in most cases immediately precede or follow the semantically related sentence parts such as direct object or prepositional phrase while the

main verb is often not adjacent to the related parts in a sentence. Since the direct edge between the main verb and non-adjacent related sentence parts cannot be drawn, the sentence is syntactically reorganized by replacing the modal or auxiliary verbs by the corresponding main verb. Another syntactic rearrangement takes place when detachable prefixes are attached to the corresponding main verb. In German some prefixes of verbs are detached and located at the end of the main clause. Since verbs without a prefix have a different meaning, prefixes have to be attached to the verb stem[1]. The reorganized sentence can be added to

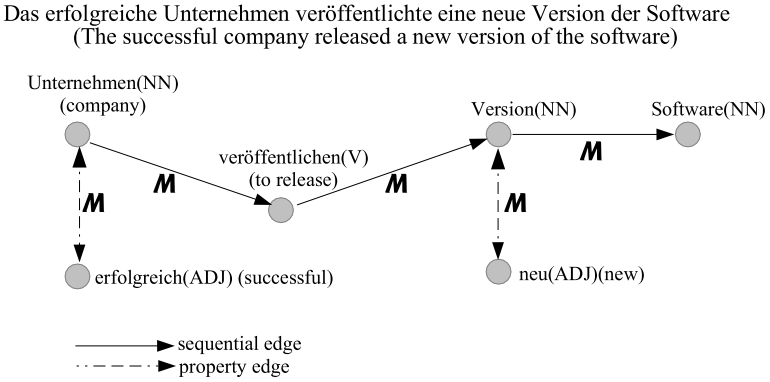

Figure 5.3: An example of a sentence transformed in a lexical graph

the graph inserting the normalized words in a sentence as vertices and connecting the adjacent words by a directed edge. However, some adjacent words are not semantically related to each other, therefore the lexical graph features two types of edges (see an example in fig. 5.3). A *property edge* links the head word of a syntactic chunk (verb or noun phrase) with its modifiers (adverbs or adjectives respectively) that characterize the head word and is bidirectional. A *sequential edge* connects the head words (e.g. main verbs, head nouns) of syntactic chunks reflecting the "semantic backbone" of the sentence.

The length of an edge represents how strong two lexical items are related to each other and depends therefore on the frequency of their co-occurrence. It is initialized with a maximum length M. Every time an existing edge is found in the currently processed sentence, its current length $CurLen$ is modified according to $CurLen = \frac{M}{\frac{M}{CurLen}+1}$; hence the length of an edge is inversely proportional to the frequency of co-occurrence of its endpoints.

After all sentences from the text corpus have been added to the lexical graph, vertices (words) with a low frequency ($\leq \theta$) are removed from the graph to primarily accelerate the distance calculation. Such rarely occurring words are usually proper nouns, abbreviations, typos etc. Because of the low frequency semantic relations for these words cannot be confidently identified. Therefore removing such vertices reduces the size of the graph significantly without performance penalty (the graph generated from 5 journal volumes contained around 300000 vertices and 52191 after frequency filtering with $\theta = 8$). Experimental results feature even a slightly better performance on filtered graphs. To preserve semantic consistency of the graph and compensate removal of existing paths the connections between the predecessors and successors of removed vertices have to be taken into account: the edge length $e(p, s)$ between the predecessor p to the successor s of the removed vertex r can incorporate the length of the path $length(p, r, s)$ from p to s through r by calculating the halved harmonic mean:

[1] this is the only time a language specific heuristic is utilized in GROPUS. Since recognition of synonyms is an optional preprocessing step, the requirement for language independence is fulfilled

$e(p,s) = \frac{e(p,s)*l_{prs}}{e(p,s)+l_{prs}}$. $e(p,s)$ is the more reduced the smaller $length(p,r,s)$ is and if they are equal, $e(p,s)$ is half as long after merging.

Beside direct edges an important indication of semantic closeness is the distance, i.e. the length of the shortest path between two vertices. Distances are calculated by the Dijkstra algorithm with an upper threshold Θ. Once the distances from a certain vertex reach the threshold, the calculation for this vertex is aborted and the not calculated distances are considered infinite. Using the threshold reduces the runtime and space considerably while the semantic relation between the vertices with distances $> \Theta$ is negligible.

The values of M, θ and Θ depend on the particular text corpus and are chosen to keep the size of the graph feasible. θ can be determined experimentally incrementing it as long as the results on the test set are improving.

5.2.3 Identification of synonyms

The lexical graph is conceived as an instrument to identify semantic relations such as synonymy and hypernymy between lexical items represented by its vertices. The goal of semantic preprocessing is finding synonyms albeit some results can be immediately transferred for identification of hyponyms. To provide a quantitative measure of synonymy different similarity metrics (presented below) were defined on the lexical graph. Given a word, the system uses the metric to calculate the closest vertices to the vertex that represents this word. The result is a ranked list of words sorted by the degree of synonymy in descending order. Every metric sim is normalized to provide a confidence measure so that given a vertex v_i the value $sim(v_i, v_j)$ can estimate the confidence of v_j being synonym to v_i. The normalization is performed for each metric sim by the following functions:

$$n_{min}(sim(v_i, v_j)) = \frac{min(sim(v_i,v_1),...,sim(v_i,v_n))}{sim(v_i,v_j)}$$

for metrics that indicate maximum similarity to a vertex v_i by a minimum value and

$$n_{max}(sim(v_i, v_j)) = \frac{sim(v_i,v_j)}{max(sim(v_i,v_1),...,sim(v_i,v_n))}$$

for metrics that indicate maximum similarity to a vertex v_i by a maximum value, where $v_1 \dots v_n$ are the set of graph vertices. In both cases the top-ranked word has the maximum confidence of 1 to be a synonym of v_i. The normalized ranked lists are used for the comparison of different metrics and the evaluation of the approach (see sec. 5.2.4).

A similarity metric is supposed to assess the semantic similarity between two vertices of the lexical graph. Since the distance metric *DistanceM* used for calculation of distances between the vertices in the graph indicates how semantically related two vertices are, it can be used as a similarity metric. As the graph is directed, the distance metric is asymmetric, which implies that different distances between v_j and v_i can be measured depending on, which word is queried. The major drawback of the *DistanceM* is that it takes into account only one path between the examined vertices. Even though the shortest path indicates a strong semantic relation between the vertices, it is not sufficient to conclude synonymy that presupposes similar word senses.

Therefore more evidence for strong semantic relation with the particular aspect of similar word senses should be incorporated in the similarity metric. The *property neighbors* of a vertex v_i (adjacent vertices connected with v_i by the property edge) play significant role in characterizing similar senses. If two terms share many characteristic properties, there is a strong evidence of their synonymy. A

shared property can be regarded as a *witness* of the similarity of two word senses. There are other potential witnesses, e.g. transitive verbs shared by their direct objects; however, we restricted this investigation to the property neighbors as the most reliable witnesses.

The simple method to incorporate the concept of the witnesses into the metric is to determine the number of common property neighbors:

$$NaivePropM(v_i, v_j) = |prop(v_i) \cap prop(v_j)|$$

where $prop(v_i) = \{v_k | e(i,k)\ is\ a\ property\ edge\}$ This method disregards, however, the different degree of correlation between the vertices and their property neighbors that is reflected by the length of property edges. A property is the more significant, the stronger the correlation between the property and the vertex is, that is the shorter the property edge is. The degree of synonymy of two terms depends therefore on the number of common properties and the lengths of paths between these terms leading through the properties. Analogously to the electric circuit one can see the single paths through different shared properties as channels in a parallel connection and path lengths as "synonymy resistances". Since a bigger number of channels and smaller single resistances contribute to the decreasing of the total resistance (i.e. the evidence of synonymy increases), the idea of *WeiPropM* metric is to determine the similarity value analogously to the total resistance in a parallel connection:

$$WeiPropM'(v_i, v_j) = \left(\sum_{k=1}^{n} \frac{1}{length(v_i, p_k, v_j)} \right)^{-1}$$

where $length(v_i, p_k, v_j) = e(v_i, p_k) + e(p_k, v_j)$ is the length of the path from v_i to v_j through p_k and $p_k \in prop(v_i) \cap prop(v_j)$.

Another useful observation is that some properties are more valuable witnesses than the others. There are very general properties that are shared by many different terms and some properties that are characteristic only for certain word senses. Thus the number of property neighbors of a property can be regarded as a measure of its quality (in the sense of characterizing the specific word meaning). *WeiPropM* integrates the quality of a property by weighting the paths leading through it by the number of its property neighbors:

$$WeiPropM(v_i, v_j) = \left(\sum_{k=1}^{n} \frac{1}{(e(v_i, p_k) + e(p_k, v_j)) * |prop(p_k)|} \right)^{-1}$$

where $p_k \in prop(v_i) \cap prop(v_j)$.

WeiPropM measures the correlation between two terms based on the path lengths. Frequently occurring words tend to be ranked higher because the property edge lengths indirectly depend on the absolute word frequency. Because of high absolute frequency of words the frequency of their co-occurrence with different properties is generally also higher and the property edges are shorter. Therefore to compensate this deficiency (i.e. to eliminate the bias discussed in [Wee04]) an edge length from a property to a ranked term $e(p_k, v_j)$ is weighted by the square root of its absolute frequency $\sqrt{freq(v_j)}$. Using the weighted edge length between the property and the ranked term we cannot any longer calculate the path length between v_i and v_j as the sum $length(v_i, p_k, v_j) = e(v_i, p_k) + e(p_k, v_j) * \sqrt{freq(v_j)}$ because the multiplied second component significantly outweighs the first summand. Relative path length can be used instead where both components

are adequately taken into account and added relatively to the minimum of the respective component: let $min1$ be $min(e(v_i, p_a), \ldots, e(v_i, p_n))$ where $p_k \in prop(v_i)$ and $min2 = min(\ldots, e(p_k, v_j) * \sqrt{freq(v_j)}, \ldots)$ where $p_k \in prop(v_i) \cap prop(v_j)$. Relative path length would be $\frac{e(v_i,p_k)}{min1} + \frac{e(p_k,v_j)*\sqrt{freq(v_j)}}{min2}$. Further experimental observation suggests that when searching for synonyms of v_i the connection between v_i and the property is more significant than the second component of the path – the connection between the property and the ranked term v_j. Therefore when calculating the relative path length the first component has to be weighted stronger (the examined ratio was 2:1). The corresponding metric can be defined as follows:

$$FirstCompM(v_i, v_j) = \left(\sum_{k=1}^{n} \frac{1}{RelPathLength(k) * \sqrt{|prop(p_k)|}} \right)^{-1}$$

where $RelPathLength(x) = \frac{2}{3} * \frac{e(v_i,p_x)}{min1} + \frac{1}{3} * \frac{e(p_x,v_j)*\sqrt{freq(v_j)}}{min2}$

As opposed to *NaivePropM* and *WeiPropM* *FirstCompM* is not symmetric because of the emphasis on the first component.

5.2.4 Experiments with Synonym Recognition

The common evaluation method for similarity metrics is comparing their performance on the same test set with the same context representations with some manually created semantic source as the gold standard [Cur02]. We used a large text corpus of 5 volumes of German computer journals to examine the behavior of different metrics and identify other factors that have an influence on the goodness of synonym recognition. The resulting graph generated with $M = 2^{20}$, $\theta = 8$, $\Theta = 60000$ contained 52191 vertices, 4,927,365 edges and 376,000,000 distances. A test set of 200 synonym sets was prepared consulting [Ope05]. The test set consists of 75 everyday words (e.g. “Präsident” (president), “Eingang” (entrance) “Gruppe” (group)), 60 abstract terms (e.g. “Ursache” (reason), “Element”, “Merkmal” (feature)) and 65 domain-specific words (e.g. “Software”, “Prozessor” (CPU)).

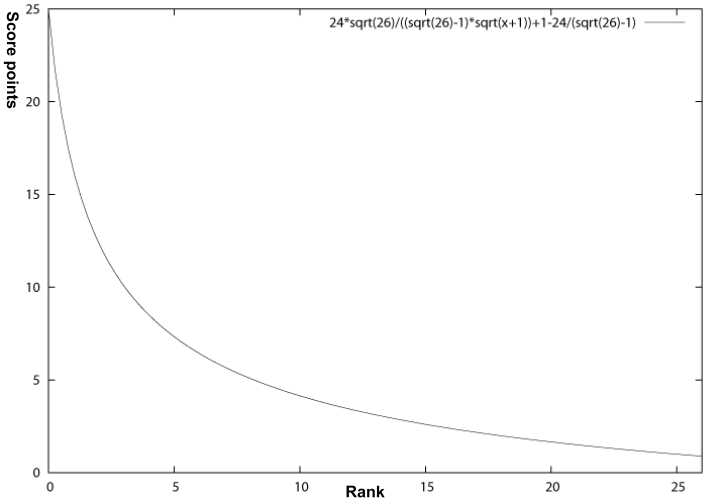

Figure 5.4: Graph of the scoring function

The similarity metrics do not distinguish between different word senses returning synonyms of all senses of the polysemous words in a single ranked list. Therefore the synonym set of a word in the test corpus is the union of synonym sets of its senses. To provide a measure for overall performance and to compare the

Metric	Score	Π(1)	Π(5)	Π(25)	Π(100)
DistanceM	2990.7	0.20	0.208	0.199	0.38
NaivePropM	6546.3	0.415	0.252	0.271	0.440
WeiPropM	9411.7	0.54	0.351	0.398	0.607
FirstCompM	11848	0.575	0.412	0.472	0.637

Table 5.1: Results of different metrics on the test corpus

different metrics a function measuring the *similarity score (SimS)* was defined that assigns a score to a metric for correctly found synonyms among the 25 top-ranked. The function assigns 25 points to the correctly found top-ranked synonym of v_i ($SimS(0, v_i) = 25$) and 1 point to the synonym with the 25^{th} rank ($SimS(25, v_i) = 1$). The rank of a synonym is decreased only by false positives that are ranked higher (i.e. each of correctly identified top n synonyms has rank 0). In order to reward the top-ranked synonyms stronger the scoring function features a hyperbolic descent (its graph is visualized in fig. 5.4). For a synonym of v_i with the rank x:

$$SimS(x, v_i) = \left\{ \begin{array}{c} 0,\ if\ x \notin synset(v_i) \\ \frac{24*\sqrt{26}}{(\sqrt{26}-1)*\sqrt{x+1}} + 1 - \frac{24}{\sqrt{26}-1} \end{array} \right\}$$

To compare performance of different metrics the $SimS$ values of the top 25 words in the ranked list were summed for each word of a test corpus. The total score of a similarity metric Sim is $\sum_{i=1}^{200} \sum_{j=1}^{25} SimS(rank(RankedList(v_i, j)), v_i)$ where $RankedList(v_i, j)$ returns the word at the position j from the ranked list produced by Sim for v_i and $v_1, \ldots, v_{200}$ are the words of the test corpus.

Besides, a combined precision and recall measure Π was used to evaluate the ranked lists. Given the word v_i, we examined the first n words ($n = 1, 5, 25, 100$) of the ranked list returned by a similarity metric for v_i whether they belong to the $synset(v_i)$ of the test corpus. $\Pi(n)$ will measure precision if n is less than the size of the $synset(v_i)$ because the maximum recall can not be reached for such n and recall otherwise because maximum precision cannot be reached for $n > |synset(v_i)|$. The Π values were averaged over 200 words. Table 5.1 presents the result of evaluating the similarity metrics introduced in sec. 5.2.3. The results of *DistanceM* confirm that regarding distance between two vertices alone is not sufficient to conclude their synonymy. *DistanceM* finds many related terms ranking general words with many outgoing and incoming edges higher, but it lacks the features providing the particular evidence of synonymy. *NaivePropM* is clearly outperformed by the both weighted metrics. The improvement relative to the *DistanceM* and acceptable precision of the top-ranked synonyms Π(1) show that considering shared properties is an adequate approach to recognition of synonyms. Ignoring the strength of semantic relation indicated by the graph and the quality of properties is the reason for the big gap in the total score and recall value (Π(100)). Both weighted metrics achieved results comparable with those reported by Curran and Moens in [Cur02] and Turney in [Tur01]. Best results of *FirstCompM* confirm that the criteria identified in sec. 5.2.3 such as generality of a property, abstraction from the absolute word frequency etc. are relevant for identification of synonyms. *FirstCompM* performed particularly better in finding synonyms with the low frequency of occurrence.

In another set of experiments we investigated the influence of the size of the text corpus (cf. fig. 5.5). The plausible assumption is the more texts are processed, the better the semantic connections between terms are reflected by the graph, the more promising results are expected. The fact that the number of vertices

Frequency	9-249	250-499	500-999	1000-1499	1500-2499	2500-3999	4000-5499	5500-7499	>7500
Words/cluster	27	25	44	30	27	15	11	8	13
Aver. score	53.23	51.52	45.80	60.75	56.51	58.75	97.21	106.11	73.85
Π(1)	0.556	0.52	0.432	0.567	0.667	0.667	0.818	0.75	0.615
Π(5)	0.381	0.392	0.342	0.395	0.393	0.413	0.600	0.675	0.503
Π(25)	0.447	0.432	0.446	0.494	0.474	0.419	0.531	0.550	0.600
Π(100)	0.561	0.645	0.618	0.610	0.690	0.623	0.705	0.642	0.748

Table 5.2: Influence of word frequency on the results of FirstCompM *metric*

does not grow proportionally to the size of text corpus can be explained by word recurrence and growing filtering threshold θ. However, the number of edges increases linearly and reflects the improving semantic coverage. As expected, every metric performs considerably better on bigger graphs. While *NaivePropM* seems to converge after three volumes, the both weighted metrics behave strictly monotonically increasing. Hence an improvement of results can be expected on bigger corpora. On the small text corpora the results of single metrics do not differ significantly since there is not sufficient semantic information captured by the graph, i.e. the edge and path lengths do not fully reflect the semantic relations between the words. The scores of both weighted metrics grow, though, much faster than that of *NaivePropM*. *FirstCompM* achieves the highest gradient demonstrating the biggest potential of leveraging the growing graph for finding synonymy.

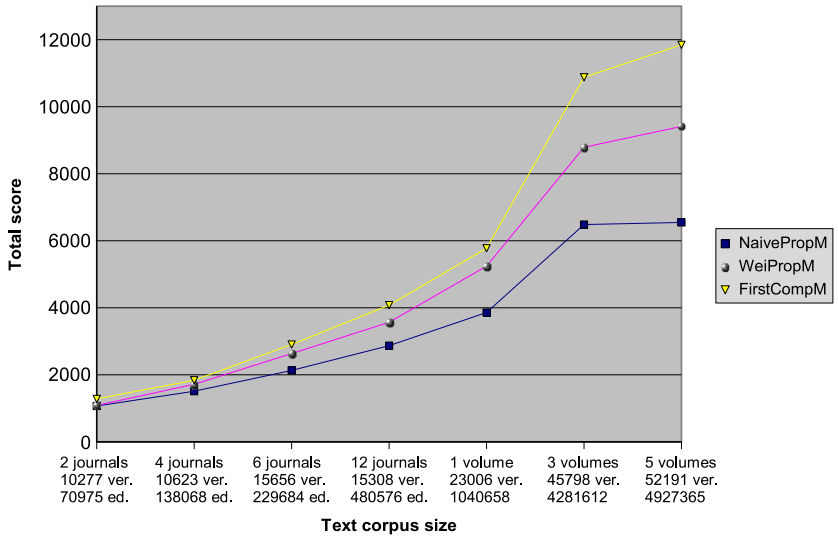

Figure 5.5: Influence of the size of the text corpus.

To examine the influence of the word categories results on the subsets of the text corpus corresponding to a category are compared. All metrics show similar behavior, therefore we restrict the analysis to the Π values of *FirstCompM* (fig. 5.6). Synonyms of domain-specific words are recognized better than those of abstract and everyday words. Their semantics are better reflected by the technically oriented texts. The Π values for abstract and everyday words are pretty similar except for the high precision of top-ranked abstract synonyms. Everyday words suffer from the fact that their properties are often too general to uniquely characterize them, which involves loss of precision. Abstract words can be extremely polysemous and have many subtle aspects that are not sufficiently covered by the texts of computer journals.

To test whether the metrics perform better for the more frequent words the test set was divided in 9 disjunctive frequency clusters (table 5.2). *FirstCompM* achieved considerably better results for very frequently occurring words (≥ 4000 occurrences). This confirms indirectly the better results on the bigger text cor-

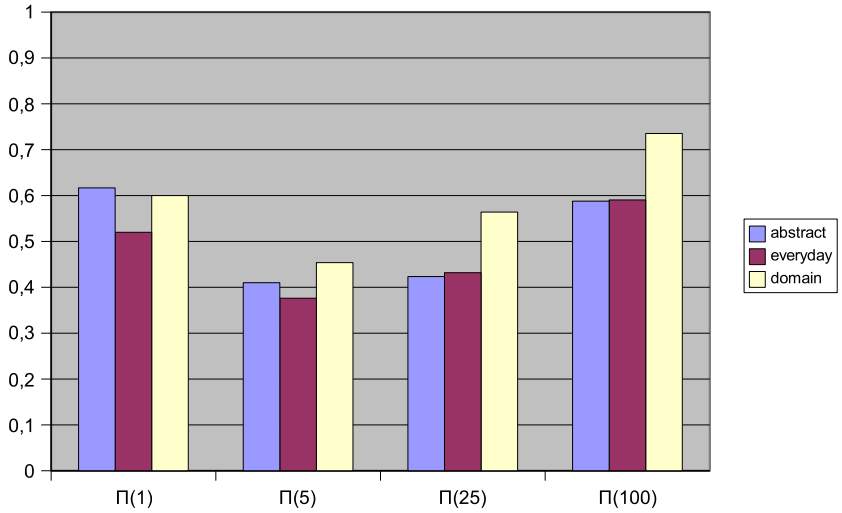

Figure 5.6: Dependency of $\Pi(n)$ on word category (results of FirstCompM *metric)*

pora: while low frequency does not exclude random influence, frequent occurrence involves adequate capturing of the word semantics in the graph by inserting and adjusting all relevant property edges. These results do not contradict the conclusion that *FirstCompM* is not biased towards words with a certain frequency because the mentioned bias pertains to retrieval of synonyms with a certain frequency, whereas in this experiment the performance for different word frequencies of queried words is compared.

5.2.5 Utilization of Synonym Recognition for Abstraction of Extraction Patterns

The empirical investigation demonstrated that the lexical graph can serve as an instrument for finding synonymy between lexical items in natural language corpora. The big advantage of the graph in comparison to other context models is that it captures not only the immediate context but establishes many transitive connections between related terms. Similarity metric *FirstCompM* that best leverages the graph structure achieved the best results confirming the significant role of the number of shared properties, the frequency of their co-occurrence and the degree of their generality for detecting synonymy. Therefore this metric can be used for the determination of a synonym set of a queried word. When abstracting a lexical item during the generalization of extraction patterns, it is submitted as a synonym query to the lexical graph. The ranked list of words is determined by the *FirstCompM* metric. The top five ranked words in the ranked list or any word with the maximum confidence measure are returned as a synonym set, which can be used for abstraction.

The experimental results suggest that a considerable portion of top five ranked words are not synonymous to the queried term (cf. table 5.1). However, we deliberately take these erroneously identified synonyms into account since it is much more unlikely that a non-synonym fits into the context of the queried word in the extraction pattern and causes a wrong match than that a real synonym causes a positive match. In the most cases the extraction patterns with non-synonyms will not match any text fragments having no negative influence on the accuracy of corresponding extraction rule. Besides, only domain words will be queried, for which better synonym recognition than for general words can be expected according to the experimental results.

Since the texts for the graph construction do not have to be annotated by human but only linguistically preprocessed, all texts in a domain can be used to build the lexical graph. Significantly improving results for bigger text corpora and more frequently occurring words indicate that the semantic preprocessing will be more successful for bigger text corpora.

New methods that increasingly exploit the graph structure e.g. regarding the lengths and number of short paths between two terms or extending the witness concept to other morphological types can also contribute to a better recognition accuracy, but have not been investigated within the scope of this research.