Learning a Domain Ontology from Hierarchically Structured Texts

Pavel Makagonov MPP@MIXTECO.UTM.MX
Alejandro Ruiz Figueroa FIGUEROA@NUYOO.UTM.MX
Mixteca University of Technology, Huajuapan de León, Oaxaca, 69000, Mexico

Konstantin Sboychakov KSBOYCHAKOV@YANDEX.RU
Russian National Public Library for Science and Technology

Alexander Gelbukh GELBUKH@GELBUKH.COM
Center for Computing Research, National Polytechnic Institute, 07738, DF, Mexico

Abstract

Any scientific or technical document is organized hierarchically: some sections of the text (such as the abstract or conclusions) summarize the contents of the main text; sections have titles describing their contents in general words; chapter titles describe the contents of a set of sections; book title describes the contexts of all chapters, etc. Moreover, whole collections of scientific documents are usually organized hierarchically: e.g., papers are organized in journals, conferences, etc., which in turn have their own titles. We exploit this hierarchical structure to learn a lexical ontology, in which subordination relationships roughly mirror those between the texts and titles in which these words occur: words occurring in more general titles subordinate the words occurring in the texts described by these titles.

1. Introduction

Our initial motivation was to develop a methodology for quantitative and qualitative comparison of the state of a field of science in certain periods of time, which allows for detecting trends in its development and predicting its future state (Kuhn, 1970; Makagonov & Ruiz Figueroa, 2004).

To portrait the state of a field of science (in a certain time span), we use its ontology: an account of which words describe its large branches in a general way and which words describe the concepts or sub-branches within each branch. The observation on which the work presented here is based is that large branches of a field roughly correspond to the titles of journals, books, or conferences devoted to it; its sub-branches correspond to paper titles published in these journals or conferences, etc.

There exist a variety of methods for automatically extract ontologies from texts. E.g., Hearst (1992) looks for patterns such as *France and other European countries* to extract the fact that *France* is a *European country* from texts. Paşca (2004) reports an experiment on application of a similar method to a corpus of 500 million web pages. However, the amount of texts available on a narrow technical domain is insufficient for such methods. Our approach, on the contrary, can cope with a relatively small amount of data. On the other hand, existing methods for extracting narrow-domain ontologies either require manual effort (Martins *et al*., 2004) or can extract a very small number of words. Our observation on the usefulness of the hierarchical text structure provides an alternative (or—as a future work—an additional) source of information on relations between words.

Thus the aim of this work is to construct a hierarchy of topics and subtopics in a given domain by extracting it from a hierarchically structured text corpus so that the subordination relationships between words in the ontology mirror those between the text segments in the text hierarchy where the given words occur. In particular, the levels in the constructed ontology are in a one-to-one correspondence with the levels in the text hierarchy.

Though the idea looks quite simple, there are a number of details to consider in order for this idea to work in practice. In what follows we define the main notions used in our algorithm: Section 2 describes the input data used for ontology construction and Section 3 the data structure that the algorithm constructs. In Section 4 we describe the algorithm itself, and in Section 5 we present the experimental results. Section 6 concludes the paper and discusses future work directions.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

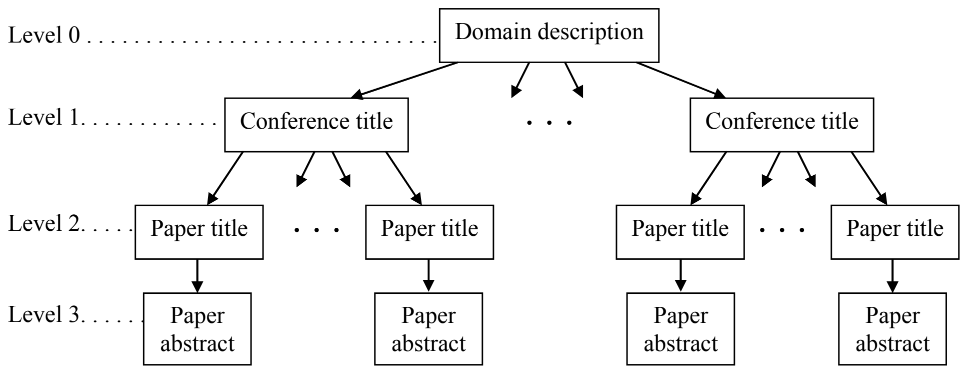

Figure 1. Hierarchy of text segments in our corpus.

2. Input Data: Text Corpus

Most of existing texts are hierarchically structured in such a way that some their segments (which can be very short, e.g., titles) are marked as "more general" than others. We build our ontology hierarchy using such a text, as well as some sources of statistical information, as described below.

Hierarchically structured text is any text partitioned into segments that are arranged in a tree (or, more generally, in an acyclic directed graph) so that the "upper-level" segments contain meta-information with respect to their subordinated text segments. For simplicity we also refer to such text segments as texts.

For example, in a book, the root is the book title (which is a short text); the intermediate levels are formed by part titles, then chapter titles, section titles, etc.; finally, the full text of each section is a leaf below the corresponding section title.

Another example is any HTML-encoded webpage partitioned into subordinated text segments (titles and main text) by the <h1> to <h6> tags that mark the headers, sub-headers, sub-sub-headers, etc., with the main text (text outside of such tags) corresponding to the leaves of the tree.

In the experiments reported in this paper we used a collection of abstracts of papers presented at conferences on a specific domain, namely, on parallel, concurrent, distributed, and simultaneous computing. Note that we did not have access to the full texts of the papers but only to the abstracts; this is because unlike the full texts, the abstracts are freely available for download from the Internet. Thus the collection had a three-level tree structure, with the root level added for convenience, see Figure 1. In the figure, the domain description consists of its name: *Parallel, concurrent, distributed, and simultaneous computing.*

Domain is a narrow technical topic, which in our case was parallel computing. Narrow-domain texts include three types of words:

– General-lexicon words appearing in any text. Such words include articles, prepositions, or the words like *have*, *give*, *see*, *number*, etc. Any information on such words can be easily found in existing dictionaries.
– Technical terms used in a wider area of knowledge and not only in the given narrow domain. For the narrow domain of parallel computing these are words used also in texts on other areas of computing, such as *program* or *execution*.
– Terms specific for the selected domain, such as *clock*, *monitor*, or *semaphore* for the parallel computing domain.

In domain ontology we are only interested in the latter type of words: domain-specific terms. These terms have the following special properties as compared with the general words or wider-domain terms:

– They are not homonymous (polysemous), i.e., they are unambiguous. While in general-topic text the word *parallel* can refer to two straight lines or two similar ideas, in the chosen narrow domain it only can mean *simultaneous*.
– They usually are not synonymous. While in general-topic text *clock* can be substituted with *watch*, *chronometer*, or *timer*, in the chosen domain no word other than *clock* can be used to refer to this device. However, some synonyms still exist in narrow-domain texts, e.g., *parallel*, *concurrent*, *distributed*, and *simultaneous* computing are near-synonyms. Though our algorithm does not detect such rare cases of synonymy explicitly, it handles them almost correctly (see below).

While our method can easily be used to build a general-topic ontology, these properties of domain-specific text greatly alleviate linguistic problems in their automatic statistical analysis and allow us to omit many of such complications both in our implementation in the discussion presented here. In particular, they allow us to avoid addressing word sense disambiguation (WSD) and detection and handling synonymy—the two main problems preventing from obtaining accurate statistics on the word usage in general texts. However, we have to deal with other linguistic phenomena such as morphology and multi-word expressions, as described in Section 3.

Data sources used in our algorithm are: a hierarchically structured domain corpus, a corpus of a wider area, and a frequency dictionary for the given language. The latter two sources are used only to provide negative examples for the classification of words into domain-specific and non-domain-specific terms.

As a domain corpus, we have constructed a corpus of conference papers belonging to a quite narrow domain of parallel computing, see Figure 1. As a source of negative examples for selecting domain-specific words, we used a corpus on software engineering. The frequency dictionary is needed only because the wider-domain corpus may not contain reliable statistics for some words, while collecting a very large corpus of open-topic texts is relatively easy. Thus we used a large general-topic corpus to learn the frequencies of words in general language use.

3. Output Structure: Ontology

The ontology consists of concepts (roughly, words) of a chosen domain, interrelated to form a hierarchy (roughly, a tree). Below we discuss these notions in more detail.

Relationship reflected in our ontology is, informally, subordination between topics and subtopics in a given domain. Since such topics and subtopics are described through words, so finally what are arranged into a hierarchy are words.

The resulting relationship between words differs from the *is-a* relationship (*Socrates* is a *man*) often considered in ontologies in that the corresponding words neither substitute each other in a context (*Socrates died* ⇒ *a man died*) nor inherit properties of others (*men* are *mortal* ⇒ *Socrates* is *mortal*). Instead, the relationship we are interested in here can be called *involvement*. It resembles the holonym / meronym (whole / part) relationship: higher-level words describe situations that involve—include as their (possible) parts—the situations described by the lower-level words. For example, *evaluation* involves *measurement* (which in turn involves *value* and *calculation*) as a part of the process; *processor* involves *memory* as part of its typical functioning, and involves *register* as its physical part.

Word is a semantically meaningful token in the text. The most relevant linguistic phenomena related to words are morphological variation and multiword expressions.

As to morphological variation (e.g., *do*, *did*, *does*, *done*, *doing*), we consider all such variants as representatives of one and the same unit (lexeme), which we hereafter refer to simply as a word. Thus, the above list simply mentions the word *do* five times, even if in different morphological variants.

Multiword expressions are defined as word combinations (roughly, frequent word bigrams) referring to a single entity that cannot be referred to with only one of the words, e.g., *mutual exclusion*, cf. *hot dog*, *New York*. Unlike multiword expressions, other types of word combinations refer to hyponyms of one of the words: *mutual love* is *love* which is *mutual*; *little dog* is a *dog* which is *little*. Multiword terms are even more frequent in narrow-domain texts than in general-topic ones. Accordingly, in our algorithm such expressions are identified and treated in the same way as single words. For simplicity, we will also refer to such expressions as (compound) words.

Concept is an elementary entry in our ontology. It is an individual word or a cluster of words that are very frequently used in the same text, e.g., {*clock*, *mutual exclusion*, *monitor*} (recall that these words are unambiguous within our narrow domain and note that multiword expressions such as *mutual exclusion* are considered in the same way as single words). Note that by definition, each word belongs to only one concept (cluster); this is possible because we consider all words unambiguous.

Indeed, all statistical properties of such frequently co-occurring words are practically identical, so there is no point to distinguish them in an automatically constructed ontology. On the other hand, grouping together highly co-occurring words reduces the dimensionality of the task and thus improves statistical significance of the results, though at the cost of their granularity.

There are other dimensionality reduction methods, such as Latent Semantic Indexing (LSI). However, in LSI the resulting units are linear combinations difficult to understand by the human users of the ontology and with unclear linguistic meaning. This is why we opted for simple word clusters. In the future we can, though, consider weighting the words in a cluster in a way similar to LSI.

The representation of an elementary entry of an ontology as a set of words resembles the familiar WordNet synsets. However, there is a crucial difference between synsets and co-occurrence clusters. The words combined in a synset are different variants of referring to the same entity or idea (interchangeable in every context). Thus, a text mentioning n_1 times the first word in the synset, ..., n_k

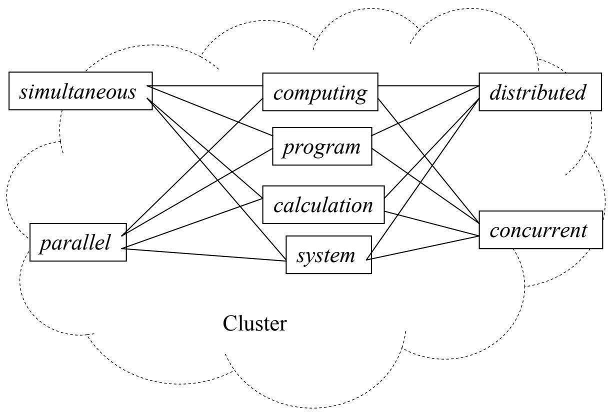

Figure 2. Synonyms are mapped in one cluster.

times the last word in the synset, can be considered as mentioning the corresponding entity $n = n_1 + ... + n_k$ times, i.e., the frequency of the entry (synset) in a text is the *sum* of the frequencies of its member words.

The words included in our clusters are, on the contrary, the names of different participants of the same situation (such as parts of the same device, parameters and tools used in the same measurement, etc.). A text mentioning one of them would mention, or at least imply, all others: e.g., a text on parallel computing mentioning *mutual exclusion* of processes would necessarily mention (or imply) the presence of a *monitor* for its handling and of a *clock* used by the latter. Thus, all k words in the cluster refer to the corresponding situation only once, each one predicting the presence of the rest. In such ideal case the frequency n of the concept (cluster) would be the frequency n_1 of one its representative, with all the other words being ignored. Since in practice some of the words can be implied, we define the frequency of the cluster in a text as the average frequency of its member words: n = $(n_1 + ... + n_k) / k$.

Note that the claim on predictability of words in the cluster is valid only for narrow-domain texts, where the words unambiguously refer to a standard situation and thus clearly predict the presence of each other. What is more, since synonymy nearly does not exist in narrow-domain text, we ignore its possible presence. To handle it seriously (e.g., in open-topic texts) we would need to use synsets instead of words:

{(*clock*, *watch*),
(*mutual exclusion*, *avoidance*),
(*monitor*, *supervisor*)}, (1)

summing up the word frequencies within synsets and averaging the results over all synsets. However, in this paper we do not consider such complications.

Those (rare) synonyms that do exist in narrow-domain texts happen to be mapped into the same concept. The reason for this is that they co-occur with the same words in texts. Since concepts are clusters of highly co-occurring words, synonyms are mapped to the same cluster because they all stick to the same words with which they co-occur, even if they do not co-occur with each other, see Figure 2. Note that unlike (1), the structure of the cluster is flat: {*clock*, *watch*, *mutual exclusion*, *avoidance*, *monitor*, *supervisor*}, so that it is not known which word is a synonym of which. This leads to incorrect calculation of frequencies: frequencies of all words in the cluster are averaged instead of summing up the frequencies of synonyms. However, we do not consider this a big problem, since (a) there are few synonyms in narrow-domain texts and (b) they nearly never appear in the same text together since one author usually uses only one variant of the term.

Topic is a cluster of concepts discussed above, grouped according to their co-occurrence in texts: e.g, {{*clock*, *mutual exclusion*, *monitor*}, {*thread*, *semaphore*}}. The co-occurrence frequency threshold for clustering in this case is lower than for the case of clustering words into concepts, so such clusters are fuzzier than concepts—that is, they correspond to broader ideas than concepts, which refer to one specific situation or thing. This allows for reliable statistics even over very short texts.

Note that such topics are not just "fuzzier" variants of concepts. Indeed, clustering concepts (word clusters) is not the same as clustering individual words. Recall that the frequency of occurrence of a concept in a text is the average of the frequencies of its member words: a text mentioning *clock* and *monitor* contains only one occurrence of the concept {*clock*, *mutual exclusion*, *monitor*}.

Thus, finding such topics is a two-step process: first, individual words are clustered into concepts (with a high co-occurrence threshold), and then the resulting units are clustered again, but with lower threshold. The practice shows that the resulting grouping differs from a one-step grouping with the lower threshold. The clusters at the second step are identified more reliably due to dimensionality reduction resulting from the first step. For the reasons discussed above, the result of such a two-step clustering procedure is linguistically more plausible.

Again, by definition each concept (and thus each word) belongs to only one topic.

Ontology is an acyclic directed graph with one source (root)—in practice, almost a rooted tree, with rare nodes having multiple parents. We maintain a layered structure in the graph: if a node has multiple parents, all of them belong to the same level (i.e., at the same distance from the root).

The leaf nodes of the graph are concepts described above. However, non-terminal nodes are topics. This is because of two reasons. One is that since the ontology is (almost)

```
Pre-process the text in the corpus
Create the root of the hierarchy
For each level of the hierarchy from root until leaves do
    For each node t at this level do
        Select texts containing the words from t
        Select words specific for the topic t
        Cluster these words to form concepts
        Select frequent concepts from next level text fragments
        If they are not leafs then
            Cluster these concepts to form topics
        Add these topics (or concepts) to hierarchy as sons of t
```

Figure 3. Algorithm for constructing the ontology.

a tree, for its usability we need fewer nodes at higher levels of the tree. The second reason is purely technical: reliability of statistics of relationships between N nodes is $O(1/N^2)$; thus with many nodes at the lower levels and many nodes at the higher levels it is difficult to gather reliable statistics of their relationships. Considering fewer (though fuzzier) clusters at the higher levels leads to desirable dimensionality reduction while still preserving detailed classification at the lower levels.

The subordination relationship between nodes is interpreted as the involvement (topic/subtopic) relationship as described above, while the layers of nodes correspond to intuitively perceived levels of abstractness of topics. The latter intuition is expressed in the layered structure of the hierarchical text corpus shown in Figure 1.

4. Algorithm

For each node of the ontology hierarchy, our algorithm recursively constructs the subordinated nodes of the words used at the next level of the text hierarchy that frequently co-occur with the words of the given node. Figure 3 outlines the algorithm; the details are explained below.

Corpus pre-processing. Using the keywords assigned to the root, we search in the Internet for texts containing these words in their titles (in our case, these were conferences; then we downloaded the abstracts of the papers presented at each found conference). The obtained texts are stemmed (we used the stemmer described by Gelbukh *et al.*, (2004); Porter stemmer (Porter, 1980) can also be used) and multiword expressions are identified (we identified them as the most frequent bigrams of content words) (Makagonov *et al.*, 2000). In what follows the term *word* refers to stemmed words or multiword expressions.

Root formation. The words identifying the domain of interest in the most abstract way (in our case, *parallel computing, concurrent computing, distributed computing, simultaneous computing*) are assigned to the root of the ontology as the most general topic of the domain. The root is at level $L = 0$ of the hierarchy.

Recursion by nodes. The following steps are performed for each non-terminal node. The process starts with the root node just mentioned, and is performed for each newly constructed node, except for the nodes of the leaf level. The aim of these steps is, given a topic (node) t at a level $L - 1$ in the ontology, to construct its subordinated nodes (of level L). The recursion must be organized level-by-level (all nodes of a level L are constructed before we proceed with constructing the nodes of level $L + 1$), since at each step we need to know whether a concept belongs to a higher-level topic.

Sub-corpus selection. Given a topic t of level $L - 1$, all texts of the next level containing this topic are selected. By a text of level L, we mean a hood in the text hierarchy (Figure 1) rooted by a fragment of level L (a *hood* rooted by a node x is the set of nodes directly or indirectly subordinated to x, including x itself). For example, if t belongs to level 1 (conference titles in Figure 1), then as texts we consider all individual papers, i.e., a concatenation of an abstract with its title. We say that a text contains a topic if it contains any word belonging to this topic.

Vocabulary selection. Only words whose frequency in the selected sub-corpus is $k = 3$ times higher than their frequency in the general language use, in the wider-topic corpus, or in the rest of the initial corpus (outside the sub-corpus just built) are selected as topic-specific terms. All other words are ignored and do not participate in any further operations for the given topic t. We also exclude from consideration all words already assigned to the nodes of higher levels (less than L).

Document aggregation. For calculating word co-occurrences in documents at the next step, we could consider each text fragment at each level of the text hierarchy (Figure 1) as individual document. However, such “documents” at the higher levels of the hierarchy are very short. To reduce sparseness, we concatenate all sibling short texts. For example, all conference titles in our experiments were merged into a single “document”; similarly, all paper titles within each conference were merged together. Note that each paper abstract was thus considered as an individual document, which is not a problem because the abstracts are longer than titles.

Concept formation. Given a sub-corpus of N documents containing W different words (types), we consider an $N \times W$ term frequency matrix $F = |f_{dw}|$, where f_{dw} is the number of occurrences of the word w in the document d,

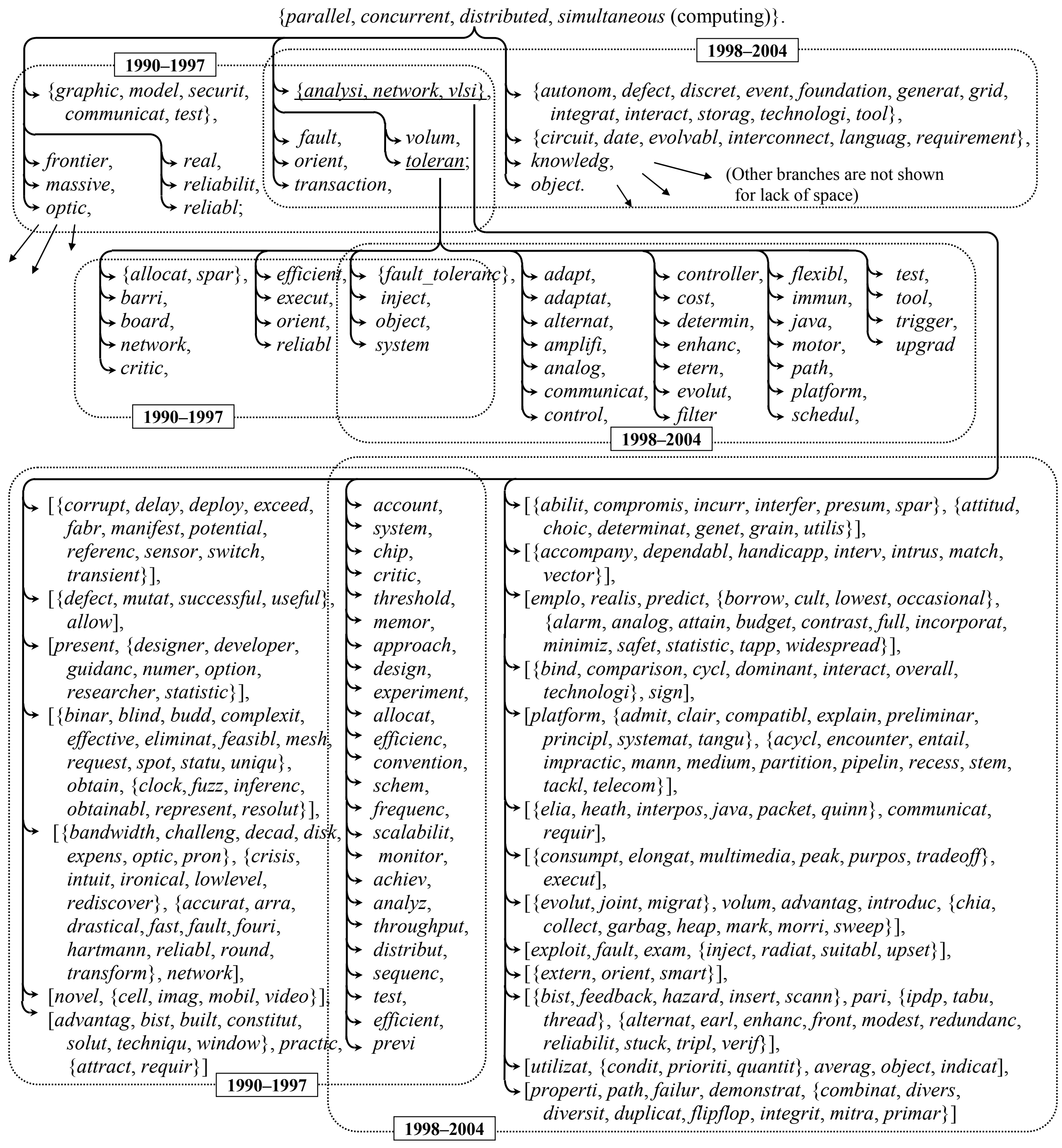
{parallel, concurrent, distributed, simultaneous (computing)}.
1990–1997
1998–2004
{graphic, model, securit, communicat, test},
frontier, massive, optic,
real, reliabilit, reliabl;
{analysi, network, vlsi},
fault, orient, transaction,
volum, toleran;
{autonom, defect, discret, event, foundation, generat, grid, integrat, interact, storag, technologi, tool},
{circuit, date, evolvabl, interconnect, languag, requirement},
knowledg,
object.
(Other branches are not shown for lack of space)
{allocat, spar}, barri, board, network, critic,
efficient, execut, orient, reliabl
{fault_toleranc}, inject, object, system
adapt, adaptat, alternat, amplifi, analog, communicat, control,
controller, cost, determin, enhanc, etern, evolut, filter
flexibl, immun, java, motor, path, platform, schedul,
test, tool, trigger, upgrad
1990–1997
1998–2004
[{corrupt, delay, deploy, exceed, fabr, manifest, potential, referenc, sensor, switch, transient}],
[{defect, mutat, successful, useful}, allow],
[present, {designer, developer, guidanc, numer, option, researcher, statistic}],
[{binar, blind, budd, complexit, effective, eliminat, feasibl, mesh, request, spot, statu, uniqu}, obtain, {clock, fuzz, inferenc, obtainabl, represent, resolut}],
[{bandwidth, challeng, decad, disk, expens, optic, pron}, {crisis, intuit, ironical, lowlevel, rediscover}, {accurat, arra, drastical, fast, fault, fouri, hartmann, reliabl, round, transform}, network],
[novel, {cell, imag, mobil, video}],
[advantag, bist, built, constitut, solut, techniqu, window}, practic, {attract, requir}]
1990–1997
account, system, chip, critic, threshold, memor, approach, design, experiment, allocat, efficienc, convention, schem, frequenc, scalabilit, monitor, achiev, analyz, throughput, distribut, sequenc, test, efficient, previ
[{abilit, compromis, incurr, interfer, presum, spar}, {attitud, choic, determinat, genet, grain, utilis}],
[{accompany, dependabl, handicapp, interv, intrus, match, vector}],
[emplo, realis, predict, {borrow, cult, lowest, occasional}, {alarm, analog, attain, budget, contrast, full, incorporat, minimiz, safet, statistic, tapp, widespread}],
[{bind, comparison, cycl, dominant, interact, overall, technologi}, sign],
[platform, {admit, clair, compatibl, explain, preliminar, principl, systemat, tangu}, {acycl, encounter, entail, impractic, mann, medium, partition, pipelin, recess, stem, tackl, telecom}],
[{elia, heath, interpos, java, packet, quinn}, communicat, requir],
[{consumpt, elongat, multimedia, peak, purpos, tradeoff}, execut],
[{evolut, joint, migrat}, volum, advantag, introduc, {chia, collect, garbag, heap, mark, morri, sweep}],
[exploit, fault, exam, {inject, radiat, suitabl, upset}],
[{extern, orient, smart}],
[{bist, feedback, hazard, insert, scann}, pari, {ipdp, tabu, thread}, {alternat, earl, enhanc, front, modest, redundanc, reliabilit, stuck, tripl, verif}],
[utilizat, {condit, prioriti, quantit}, averag, object, indicat],
[properti, path, failur, demonstrat, {combinat, divers, diversit, duplicat, flipflop, integrit, mitra, primar}]
1998–2004

Figure 4. Examples of the obrtained ontology for two periods.

and a $W \times W$ term co-occurrence matrix $C = |c_{ij}|$, where c_{ij} is a cosine measure between the corresponding rows of F (mutual information can also be used as a co-occurrence measure). To form concepts, the words are clustered according to the measure c_{ij} using any algorithm that allows for a threshold $\alpha = 0.9$ on the inter-cluster relatedness; see details in (Makagonov and Figueroa, 2005). Note that most of the clusters consist of only one word, which is not a problem.

Concept selection. We consider all concepts present in the highest-level fragments of the texts in the sub-corpus (in the previous examples these are the titles of the papers but not their abstracts). Again, we say that a concept is present in a text if at least one of its words is present in this text. To reduce noise, of these concepts only those that occur more than $\lambda_t = \ln N_t$ times are selected.

Topic formation. Unless L is the leaf level, the selected concepts are clustered into topics. This is done in the same way as words are clustered into concepts, but with a lower threshold $\beta = 0.5$ (it can be manually adjusted). Trivial clusters (those of only one element) are not allowed here. As a corpus to measure co-occurrence between concepts, we use the same corpus as we used for concept creation above (and not the sub-corpus used in the previous step). As a number of occurrences of a concept in a document (cf. the matrix C above) we consider the average number of occurrences of its member words; cf. the discussion on the difference between a synset and a concept in Section 3. Each newly constructed topic (or concept in case of leaf level) becomes a new node subordinated to the topic t.

5. Experimental Results

Our initial motivation was to detect trends in computer science development over time. Accordingly, we experimented with two sets of abstracts of conference proceedings (Figure 1) corresponding to the periods of 1990–1997 and 1998–2004. Figure 4 shows some examples from the two constructed ontologies. The words are represented by stems (not by normal form), for example, *toleran* corresponds to *tolerance* and *tolerant*. Non-trivial concepts are shown using {...}, and non-trivial topics using [...].

Because of lack of space in the paper, we only show a very small excerpt of the total ontology. Namely, we show the root, the first level below the root, and the nodes located in the tree below two selected nodes of the first level: the node {*analysi*, *network*, *vlsi*} and the node *toleran*. Other nodes have a similar number of sons (not shown here).

In our experiments, 80% of concept clusters consisted of only one word; 25% of all words belonged to such trivial clusters, while the remaining non-trivial clusters contained 75% of all words; the number of clusters was ca. 20% of that of words.

While the analysis of the trends in the development of the corresponding areas over time will be presented in a future publication, these trends are clearly visible—which can be considered an evidence of usefulness of our approach, in addition to intuitive appropriateness of the words listed in the ontology.

For evaluation of the obtained ontology, we compared it with the structure of existing textbooks on the corresponding topics (the textbooks with the title corresponding to a concept). We noted that the concepts below the given one in the tree matched well the chapters and sections of the textbooks. Note that we could do this only with an ontology corresponding to a period in a rather remote past for which textbook already exist, while for the currently active research areas the textbooks will appear only in the future (actually, as we have seen, our ontology reflects the structure of such a future textbook). However, more rigorous evaluation is still a topic of our future work.

6. Conclusions and Future Work

The hierarchical structure of technical documents is useful for automatic learning of a narrow-domain ontology from a relatively small corpus of scientific papers, as a sole or an additional source of information. The method presented here can potentially be applied to any hierarchically structured texts, for example, HTML web pages.

Experimental results show that the constructed ontology is meaningful. Specifically, it can be used for comparative analysis of the state and development of a branch of science over different time spans; we will report the results of such analysis elsewhere. However, the most probable use of the method, as that of many other automatic ontology learning methods, is rapid prototyping of an ontology, with manual post-editing for higher-quality results (IRBIS; Makagonov and Figueroa, 2005).

There are many possible future work directions, most of which have been mentioned in the text. To improve fully automatic functioning of the method, proper handling of synonymy and, probably, homonymy (word sense disambiguation) can be useful. On the other hand, integration with a visual user interface will be useful for manual correction of parameters (such as thresholds) in semi-automatic mode. Finally, we plan to develop on our idea of the two-step clustering of words into concepts and of concepts into topics, which in our opinion can be used in many clustering and natural language processing applications other than ontology construction.

Acknowledgments

This work was partially supported by Mexican Government (CONACyT, SNI, CGPI).

References

Gelbukh, A., M. Alexandrov, S.Y. Han. Detecting Inflection Patterns in Natural Language by Minimization of Morphological Model. In: *Lecture Notes in Computer Science* **3287**, Springer-Verlag, 2004, p. 432–438.

Hearst, M. Automatic acquisition of hyponyms from large text corpora. In: *Proc. 14th International Conference on*

Computational Linguistics (COLING-92), 1992, p. 539–545.

IRBIS Library Automated System; www.gpntb.ru.

Kuhn, Thomas S. *The Structure of Scientific Revolutions*. The University of Chicago Press, 1970.

Makagonov, P., A. Ruiz Figueroa. Study of Knowledge Evolution in Parallel Computing by Short Texts Analysis. In: *Progress in Pattern Recognition, Image Analysis and Applications, CIARP-2004*. *Lecture Notes in Computer Science* **3287**, Springer, 2004.

Makagonov, P., Alejandro Ruíz Figueroa. A Method of Rapid Prototyping of Evolving Ontologies. In: *Lecture Notes in Computer Science* **3406**, Springer-Verlag, 2005.

Makagonov, P., M. Alexandrov, K. Sboychakov. A toolkit for development of the domain-oriented dictionaries for structuring document flows. In: H. A. Kiers et al. (Eds.), *Data Analysis, Classification, and Related Methods. Studies in classification, data analysis, and knowledge organization*, Springer, 2000, pp. 83–88.

Martins, A., H. S. Pinto, A. L. Oliveira, Towards Automatic Learning of a Structure Ontology for Technical Articles. In: *Proc. Workshop on the Semantic Web at SIGIR-04*, U.K., 2004.

Paşka, M. Acquisition of Categorized Named Entities for Web Search. In: *Proc. 2004 ACM CIKM Intern. Conf. on Information and Knowledge Management*. ACM 2004.

Porter, M. An algorithm for suffix stripping. *Program* **14**, 1980, pp. 130–137.