Master Thesis Informatics & Economics

Towards Automatic Knowledge Discovery from Scientific Literature

Computer Based Tools for Supporting Scientific Research

Nees Jan van Eck

April 2005

Supervisor:
Jan van den Berg

Econometric Institute
Faculty of Economics
Erasmus University Rotterdam

Acknowledgements

The completion of this master thesis would not have come about without the help and encouragement of several people. They should not be left unmentioned. First of all, I would like to thank Jan van den Berg, the supervisor of this thesis, for his guidance and for the many helpful suggestions and comments he made regarding the thesis. I am also grateful to Ludo Waltman, a very good friend and a co-student, for his interest and for the enjoyable teamwork during our joint research. Finally, I wish to thank my parents for their support and encouragement during my study.

Contents

Chapter 1

Introduction

This chapter gives an introduction to this thesis. The motivation that underlies the research reported in this thesis is discussed in Section 1.1. In Section 1.2, a text data mining approach to knowledge discovery from text documents is briefly sketched. In Section 1.3, the research objective is formulated. Finally, in Section 1.4, an overview which outlines the purpose and content of the different parts of the thesis is given. In this overview, the methodology that is used in the different parts of the thesis is also briefly discussed.

1.1 Motivation

The amount of available information in scientific literature is immense and still growing at an incredible rate. A scientist can only deal with a limited amount of information and can only keep up with a limited amount of new information. As a consequence, no single scientist is capable of studying all available scientific literature. Complete coverage is simply impossible. This forces scientists to specialize in a certain field of science. But even keeping up with all that goes on in a scientist's own field takes too much time. There is, simply said, an overload of information. An illustrative example is the rapidly evolving medical field. Due to the discovery of new diseases, medicines, and treatments, scientific publications appear with high speed in this field. As a consequence, the number of medical journals has doubled every 19 years since 1870 (Wyatt, 1991). To keep up with the 10 leading journals in the medicine subfield alone, it would be necessary for a

scientist working in this subfield to read 200 articles and 70 editorials per month (Sackett et al., 1991). Clearly, due to time constraints it is far from possible for a scientist to read all these publications. The same phenomenon of information overload can be seen in other scientific fields as well, e.g. in the field of economics.

Due to the information overload in scientific fields, for a scientist it is hard to keep an overview of the structure and the development of his field. An effect of information overload is that relevant information is ignored because it is never uncovered or read. This effect brings about some unfavorable consequences. A scientist without knowledge of all relevant information on his research topics probably makes less progress than he would have made if he had all relevant information. Also, different scientists might be solving the same problem without being aware of each others efforts.

In order to tackle the problem of information overload, since the 1940s attempts have been made to create intelligent computer systems for the retrieval of relevant information (e.g. van Rijsbergen, 1979; Baeza-Yates and Ribeiro-Neto, 1999). Traditional information retrieval systems try to find documents in a given document collection that satisfy a given information need of the user. Typically, information retrieval systems operate using a retrieval process which consists of three phases. First, the user of the retrieval system has to express his information need by a query in the language provided by the system. This normally implies specifying a set of words which describe as well as possible the information need. Then, the retrieval system investigates how close the contents each document in the collection satisfies the query of the user. In this phase the retrieval system will estimate the relevancy of each document. Finally, the documents that are considered to be sufficiently relevant are presented to the user. The retrieved documents are usually ranked according to their likelihood of relevance.

Although information retrieval systems help to find relevant documents, they do not contribute directly to obtain an overview of the structure and the development of a scientific field. Scientist have to obtain this overview themselves, because all the documents retrieved using an information retrieval system have to be read to uncover the information that is enclosed in them. This requires a lot of time. Therefore, there is a need for computer

based tools that assist scientists in effectively extracting and reviewing information from documents. Such tools can be seen as a first step towards fully automatic knowledge discovery from documents, which we will discuss in the next section.

1.2 A Text Data Mining Approach to Knowledge Discovery

Text data mining, also known as text mining, is a type of data mining that deals with unstructured textual data. Following Hearst (1999), we define text data mining as the process of discovering heretofore-unknown information from large text collections. The information to be discovered is thus, in other words, currently not present in one single text document. Text data mining is a relatively young field with relations to information retrieval (e.g. van Rijsbergen, 1979; Baeza-Yates and Ribeiro-Neto, 1999), natural language processing (e.g. Jackson and Moulinier, 2002), and data mining (e.g. Fayyad et al., 1996).

Text data mining operations is typically divided into four main steps (Meij and van den Bosch, 2002). We distinguish the following four steps:

1. **Text preprocessing**: the process of preprocessing the textual data in documents and extracting important information to allow faster processing and more precise results.

2. **Document representation**: the process of storing the extracted information into a certain structure.

3. **Information aggregation and analysis**: the process of aggregating the information contained in the document representations and analyzing this aggregated information to discover new information.

4. **Knowledge presentation**: the process of making discovered information accessible using, e.g., information visualization.

In the following paragraphs, we describe the four text data mining steps in more detail.

1.2.1 Text Preprocessing

Text preprocessing is the first step of the text data mining process. During this step, the textual data in documents is preprocessed and important information is extracted to allow faster processing and more precise results. Because textual data consists of natural language, techniques that are developed within the field of natural language processing are used for preprocessing. These techniques are used for tasks like dividing documents into sentences (sentence boundary detection), dividing sentences into individual words (tokenization), identifying the root forms of words (stemming), identifying the lexical syntactical category of words (part-of-speech tagging), recognizing verb phrases and noun phrases (parsing), and recognizing names of people and organizations (named entity recognition) (e.g. Jackson and Moulinier, 2002).

1.2.2 Document Representation

In the text data mining process, the preprocessing step is followed by the document representation step. During the document representation step, the information from a document that has been extracted during the preprocessing step is stored into a certain structure. When representations of multiple documents are made, these representations can subsequently be analyzed through data mining algorithms. The accuracy of such an analysis is dependent on the type and the accuracy of the document representation that is used. Therefore, the type of document representation is an important choice in the text data mining process.

The most commonly used representation is the document vector representation (e.g. Salton, 1989). A disadvantage of this representation is that it does not take into account the order in which words or terms occur in a document and the structure of a document. Other representations are the document matrix representation (e.g. van den Berg et al., 2004) and the semantic network representation. Compared to the document vector representation, these document representations should better represent the structure of a document and, more specifically, the relations between words, terms, or concepts that are discussed in a document.

1.2.3 Information Aggregation and Analysis

During the information aggregation and analysis step of the text data mining process, the information contained in the document representations is aggregated. The aggregated information can subsequently be analyzed to discover new information. The analysis is done using data mining algorithms (e.g. Fayyad et al., 1996). For example, the similarity of concepts or the similarity of documents can be investigated by analyzing document representations using similarity measures. After that, new information may be discovered by analyzing the similarities that were found. In that way, a relationship between two concepts that was not known before may be revealed.

1.2.4 Knowledge Presentation

The last step in the text data mining process is knowledge presentation. During this step, discovered information is made accessible in a usually visual way. Techniques that have been developed within the field of information visualization (e.g. Card et al., 1999) are used. Visual knowledge presentations are particularly useful to provide insight into, for example, the relationships between important concepts or documents. Visual representations range from lists of terms to maps that depict the relationships between key concepts or between documents.

The difference between analysis and knowledge presentation may not always be clear. Displaying the raw results of the information aggregation and analysis step is a form of knowledge presentation.

1.3 Research Objective

Clearly, it appears that it is important for a scientist to keep an overview of the structure and the development of the field of science in which he is operating. As we discussed in Section 1.1, there is a need for computer based tools that assist scientists in effectively extracting and reviewing information that is enclosed in their field's scientific literature. We have seen that such tools are a first step towards fully automatic knowledge discovery

from scientific literature. With the knowledge from Section 1.1 in mind, we can define the objective of our research project.

The objective of our research project is to develop computer based tools that assist scientists in effectively extracting and reviewing information from collections of text documents from a certain scientific field. This is a first step towards the larger objective of developing tools for automatic knowledge discovery from collections of text documents.

To achieve the objective of our research project, we focus on two research questions, namely

1. How can the terms that are relevant for a certain scientific field be extracted from a collection of scientific text documents?

2. How can the associations between terms that are relevant for a certain scientific field be visualized?

The first objective is concerned with step one of the text data mining approach to knowledge discovery (see Section 1.2). The second objective is concerned with step three and four of the text data mining approach to knowledge discovery. Note that, in order to obtain a fully automatic knowledge discovery tool, answering the first research question is a prerequisite for answering the second research question. Of course, the second step in the text data mining process cannot be omitted in a knowledge discovery tool. In this thesis, however, we use existing methods to implement the second step and we do not present any new research concerning this step.

1.4 Outline of the Thesis

This thesis is organized in two main chapters (Chapter 2 and 3), which are concerned with the above two research questions, and a concluding chapter (Chapter 4), which gives conclusions and offers some ideas about the next steps in automatic knowledge discovery from scientific literature. Appendix A and B give information about the data used in the experiments in the two main chapters. Appendix C gives definitions of the most important terms used in this thesis. In the following paragraphs, we briefly discuss the contents of the two main chapters.

1.4.1 Extraction of Domain Relevant Terms

Term extraction is the computer assisted process of extracting terms from text documents. Term extraction can, for example, be used to obtain a terminological resource of a certain scientific field.

In Chapter 2, we present a term extraction system that aims to extract domain relevant terms from text documents. The system makes use of both a linguistic filter and a statistical filter. First, the linguistic filter is used to select words and phrases from text documents that are likely to be terms. The statistical filter is then used to decide whether the selected words and phrases can be considered as candidate terms. In the experiments that we describe in Chapter 2, our term extraction system is applied to abstracts from journals that are representative for the computational intelligence field. In one experiment, candidate terms are extracted for three subfields of the computational intelligence field. In this experiment, it turns out that the quality of the extracted multi-word candidate terms is higher than that of the extracted single-word candidate terms. Furthermore, it turns out that the more abstracts are used as input for the term extraction system, the higher the quality of the extracted candidate terms. In another experiment, two different implementations of our term extraction system are compared with each other. The two implementations make use of different statistical filters. In this experiment, it turns out that the quality of the candidate terms extracted using the two implementations does not differ significantly in terms of recall and precision.

1.4.2 Visualization of Concept Associations

An associative concept space is a map that visualizes the associations between concepts in a scientific field. An associative concept space can be used to obtain an overview a scientific field and to support the discovery of unknown associations between concepts.

In Chapter 3, we propose a novel algorithm for constructing an associative concept space. This algorithm can be seen as an alternative to multi-dimensional scaling, which is typically used in the literature on knowledge domain visualization. We describe experiments in which the proposed al-

gorithm and multidimensional scaling are both used for constructing an associative concept space of the computational intelligence field. It turns out that the associations between concepts in this field are better reflected in the concept space constructed using the proposed algorithm than in the concept space constructed using multidimensional scaling.

Chapter 2

Extraction of Domain Relevant Terms

2.1 Introduction

Scientific research is very dynamic. Scientific fields are continuously evolving, sometimes very rapidly. Also, new fields of science emerge frequently. The dynamic development of scientific fields is accompanied by a vast growth of new terms. For scientists working in a certain field of science, it is important to keep up with all the field's new terms and to use them in a correct way. An illustrative example is the rapidly evolving medical field. Due to the discovery of new diseases, medicines, and treatments, new terms appear with high speed in this field. When these new terms are used in an unclear or incorrect way, this has negative effects on the understanding among scientist and consequently on the research they perform.

Constructing a terminological resource (e.g. a thesaurus or an ontology) is very expensive and very time consuming if it is done manually by human experts. Such an investment is feasible for the core vocabulary of a language but not for each specific domain. The construction of terminological resources by human experts also suffers from problems of bias and lack of coverage and consistency. Therefore, there is a need to automate the extraction of terms from text documents as much as possible.

Automatic term extraction, or term extraction for short, is the computer assisted process of extracting terms from text documents. Terms are words or phrases with a special meaning in a certain subject field. For example,

in the field of artificial intelligence the phrase *computational intelligence* is a term, but the phrase *vast growth* is not a term because it does not have a special meaning. Term extraction is not a fully automatic process because it requires human experts to manually validate whether extracted terms are correct and do indeed have a special meaning in the relevant subject area. Extracted terms that have not yet been manually validated by a human expert are called candidate terms.

Term extraction can be used for a wide range of problems and applications. For example, it can be used to construct a terminological resource for a certain field of science (see, e.g., Appendix B for the construction of a thesaurus of the computational intelligence field). In addition, term extraction can improve the performance of information retrieval by making a document collection searchable for key terms (document indexing) and by adding terms to the query (query expansion).

The problem of extracting terms from text documents has already been studied by several researchers (e.g. Jacquemin, 2001). Two major types of techniques can be distinguished: linguistic techniques and statistical techniques. Linguistic techniques are based on the identification of linguistic patterns and statistical techniques are based on statistical criteria.

In this chapter, we present a term extraction system that aims to extract domain relevant terms from text documents. The system makes use of both a linguistic filter and a statistical filter. First, the linguistic filter is used to select from text documents words and phrases that are likely to be terms. The statistical filter is then used to decide whether the selected words and phrases can be considered as candidate terms. In the experiments that we describe in this chapter, our term extraction system is applied to abstracts from journals that are representative for the computational intelligence field. In one experiment, candidate terms are extracted for three subfields of the computational intelligence field. In this experiment, it turns out that the quality of the extracted multi-word candidate terms is higher than that of the extracted single-word candidate terms. Furthermore, it turns out that the more abstracts are used as input for the term extraction system, the higher the quality of the extracted candidate terms. In another experiment, two different implementations of our term extraction system are compared

with each other. The two implementations make use of different statistical filters. In this experiment, it turns out that the quality of the candidate terms extracted using the two implementations does not differ significantly in terms of recall and precision.

This chapter is organized as follows. Basic term extraction concepts and techniques are discussed in Section 2.2. In Section 2.3 and Section 2.4, our term extraction approach and the resulting term extraction system is presented. The setup and the results of the experiments are described in Section 2.5 and 2.6, respectively. Finally, in Section 2.7, conclusions and future research are discussed.

2.2 Basic Concepts and Techniques

In this section, we provide some details about basic concepts and techniques that will be used in this chapter. First, in Paragraph 2.2.1 some basic concepts about terms are given. Paragraph 2.2.2 then briefly outlines three categories of automatic term extraction. The last paragraph of this section, Paragraph 2.2.3, presents the basic performance evaluation measures used in term extraction.

2.2.1 Terms

Following Sager (1990), we define a term as a word or a phrase that denotes a concept in a certain subject field (e.g. *neural network* and *fuzzy system* are terms that denote concepts in the computational intelligence domain). Several terms may denote the same concept. In that case they are called synonyms (e.g. the term *neural network* and the term *neural net* are synonyms). Non-terms, i.e. non-terminological words or phrases, differ from terms in a couple of aspects. These differences can be used to distinguish between terms and non-terms.

Terms vs Non-Terms

Terms differ from non-terms in their structure. When we speak about the structure of terms and non-terms, we are thinking of the lexical syntactical categories of the individual words of which they are made up (e.g. the

structure of the term *neural network* is adjective-noun and the structure of the non-term *is crucial* is verb-adjective). The structure of terms is far less varied than the structure of non-terms. The structure of a term is typically a single noun (e.g. *algorithm*) or a noun phrase consisting of a noun-noun sequence (e.g. *pattern recognition*), an adjective-noun sequence (e.g. *fuzzy system*), or a noun followed by a prepositional phrase (e.g. *curse of dimensionality*).

Terms differ also from non-terms in their variation in expression. When we speak about the variation of terms and non-terms, we are thinking of the number of ways in which they can be expressed linguistically (e.g. the term *structure of the network* is a variation of the term *network structure*). The number of variations of terms is low, while the number of variations of non-terms is high. One of the consequences is that in technical texts exact repetitions of terms occur more frequently than exact repetitions of non-terms. Exact repetitions of non-terms occur primarily in relatively large texts and sometimes as an accidental effect in smaller texts (Justeson and Katz, 1995).

Single-Word Terms vs Multi-Word Terms

Two types of terms can be distinguished: single-word terms and multi-word terms. A single-word term is a term consisting of a single word, whereas a multi-word term consists of more than one word, i.e. a phrase.

Single-word terms are generally polysemous, i.e. they have multiple meanings. Multi-word terms are far less polysemous than single-word-terms. The extraction of single-word terms calls for word-sense disambiguation and context analysis (Jacquemin, 2001). Using only structure and repetition, we expect that the extraction of single-word terms is far more prone to errors than the extraction of multi-word terms. This is also the reason why in many term extraction studies the extraction of single-word terms is excluded. We will not exclude the extraction of single-word terms because we believe they may represent key concepts for a certain domain.

2.2.2 Term Extraction Techniques

The problem of extracting terms from text documents is called term extraction (e.g. Jacquemin, 2001). In fact, a term is a special type of collocation, i.e. a string of two or more words that frequently co-occur in natural language. Closely related to term extraction is therefore the extraction of collocations (Manning and Schütze, 1999, Chapter 5). A number of techniques have been developed for automatic term extraction. These techniques can be divided in three categories: linguistic techniques, statistical techniques, and hybrid techniques.

Linguistic techniques make use of the syntactic structure of terms. The structure of a term is typically a single noun or a noun phrase consisting of a noun-noun sequence, an adjective-noun sequence, or a noun followed by a prepositional phrase. To identify such sequences, part-of-speech tagging and parsing techniques are used (e.g. Jackson and Moulinier, 2002). Linguistic techniques are mainly used for extracting multi-word terms.

Statistical techniques make use of statistical measures. The most common and simple measure used is the frequency of occurrence of words. Words that appear more than a given number of times are considered as single-word terms. To determine whether a phrase is a multi-word term, and not just an accidental co-occurrence, statistical measures can be used which compute the collocational strength of a phrase (Manning and Schütze, 1999, Chapter 5). Measures that are used for this purpose are, e.g. mutual information (Church and Hanks, 1990), log-likelihood ratio (Dunning, 1993), relative frequency ratio (Damerau, 1993), t-test, and chi-square test. These measures aim to indicate whether a combination of words occurs more often than might be expected on the basis of the frequency of the single words. When this is the case, the combination of words is likely to be a multi-word term.

Hybrid techniques make use of both linguistic and statistical information. The combination of linguistic and statistical information generally leads to better results, especially in the case of small corpora or very specialized domains. Most hybrid techniques start to identify potential candidate terms using syntactical patterns. After the potential candidate terms have been

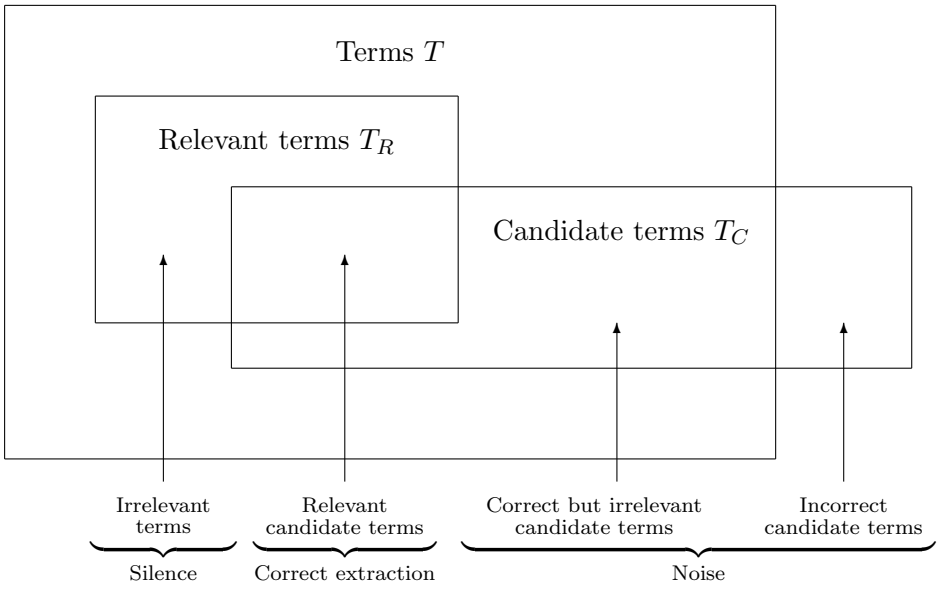

Figure 2.1: Evaluation of our term extraction system.

identified, statistical information is used to decide whether they can be considered as candidate terms.

2.2.3 Performance Evaluation

The goal of our term extraction system is to extract all the terms that are relevant for a certain domain and only them. But in general, the candidate terms $T_C = \{c_1, c_2, \ldots\}$ that are actually extracted by the system only partially overlap with the relevant terms $T_R = \{r_1, r_2, \ldots\}$. This results in a certain noise and silence. The noise consists of candidate terms which are not correct and of candidate terms which are irrelevant. The silence consists of relevant terms which are not extracted. This is shown graphically in Figure 2.1.

It should be mentioned that in other term extraction studies the noise is sometimes defined differently, namely as the candidate terms which are not correct. In these studies it is not important whether an candidate term is relevant. The goal of these term extraction studies is only to extract correct terms. This differs from our goal in which an candidate term has to be relevant too.

Quality measures are used to measure the performance of a term extraction system. These quality measures are adopted information retrieval. The most widely used measures in information retrieval are recall and precision (e.g. Salton, 1989). In our term extraction context, the recall R is defined as the proportion of relevant terms that are extracted

$$R = \begin{cases} \frac{|T_C \bigcap T_R|}{|T_R|} & \text{if } |T_R| \neq 0, \text{ else} \\ 0, \end{cases} \tag{2.1}$$

where $|T_C \bigcap T_R|$ is the number of relevant candidate terms, $|T_R|$ is the number of relevant terms, and $|T_C|$ is the number of candidate terms. The precision P is defined as the proportion of candidate terms that are relevant

$$P = \begin{cases} \frac{|T_C \bigcap T_R|}{|T_C|} & \text{if } |T_C| \neq 0, \text{ else} \\ 0. \end{cases} \tag{2.2}$$

It can be seen that recall is high when silence is low and precision is high when noise is low.

Improving recall and improving precision are two opposite goals. Efforts to increase one often result in decreasing the other. Trying to increase recall typically introduces more terms that are irrelevant and thereby reduces precision. Trying to increase precision typically reduces recall by removing some terms that are relevant. In other words, there is a trade-off between recall and precision. Because of this trade-off it is common to present precision results at different levels of recall in a graph. Such a precision-recall graph typically has a concave shape. Figure 2.2 shows an example of a precision-recall graph.

2.3 Our Approach

As we discussed in Paragraph 2.2.1, terms mainly differ from non-terms in their structure and in the number of repetitions in technical texts. We will use these differences between terms and non-terms in our approach to extract terms from a domain-specific technical corpus, i.e. a collection of technical text documents that is representative for a certain domain.

Our approach first extracts potential candidate terms from texts on the basis of their structure. For this, we apply a linguistic technique that uses

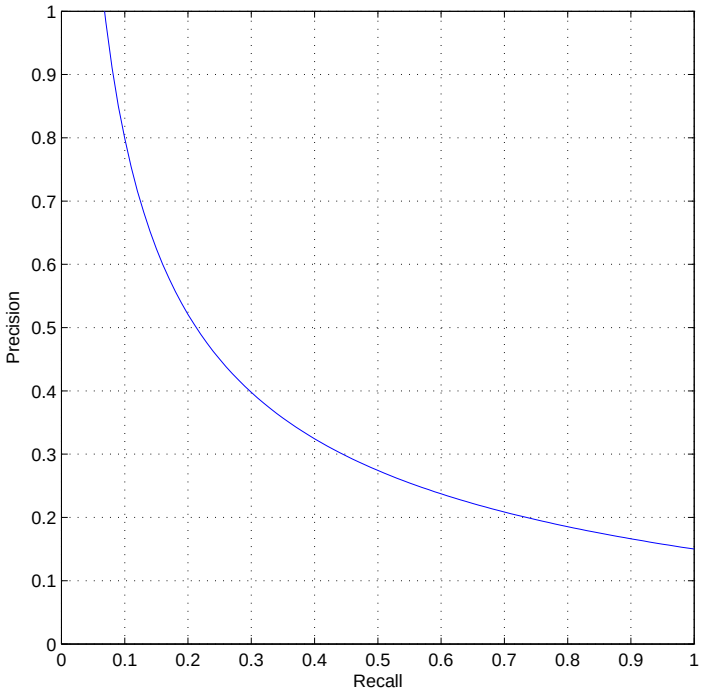

Figure 2.2: A typical precision-recall graph.

a regular expression to select words or phrases that are likely to be terms. This technique is based on the research of Justeson and Katz (1995). Our approach then decides whether a potential candidate term can be considered as a candidate term on the basis of its number of repetitions in the corpus. We will compare two statistical techniques that can be used in this step. The first, and the simplest, statistical technique considers a potential candidate term as a candidate term when it occurs more than a given number of times in the corpus. The second statistical technique decides for each potential candidate term how specific that term is for the corpus. This involves a comparison with another corpus. The more specific a potential candidate term is for the corpus, the more the term is considered as domain relevant. This technique therefore considers the most specific potential candidate terms as candidate terms.

In the next section, a term extraction system that is based on our approach is presented.

2.4 Architecture of Our Term Extraction System

In this section, we present a term extraction system that is based on the approach discussed in the previous section. The input to the term extraction system is a domain-specific technical corpus from which terms have to be extracted. The output of the system consists of a list of candidate terms that have been extracted by the system. The candidate terms in this list are likely to be domain relevant. To know if this is indeed the case, the list has to be validated by a domain expert. This validation step is not part of the actual term extraction system.

Figure 2.3 provides and overview of the architecture of the term extraction system. The system consists of three components: a corpus annotator, a linguistic filter, and a statistical filter. In the next paragraphs, these components are described in detail.

2.4.1 Corpus Annotator

The corpus annotator assigns to every word in a corpus a part-of-speech tag and a lemma. The part-of-speech tag of a word is the lexical syntactical category associated with that word (e.g. the part-of-speech tag of the word *algorithm* is a noun). The lemma of a word is its base or uninflected form (e.g. the lemma of the word *systems* is *system*). To assign to every word in a corpus a part-of-speech tag and a lemma, the corpus annotator uses some components of MontyLingua (Liu, 2004). MontyLingua is a commonsense-enriched, end-to-end natural language understander for English text. The components that are used from MontyLingua are:

Tokenizer The tokenizer component tokenizes an English text with sensitivity to abbreviations and resolves contractions (e.g. *you're* becomes *you are*).

Part-of-speech tagger The part-of-speech tagger component assigns to every word in an English text the most likely part-of-speech tag. The tagger uses the Penn Treebank tagset (Marcus et al., 1993). It is based on Brill (1994) and has been enriched with common sense knowledge about the every day world to improve accuracy up to 97%.

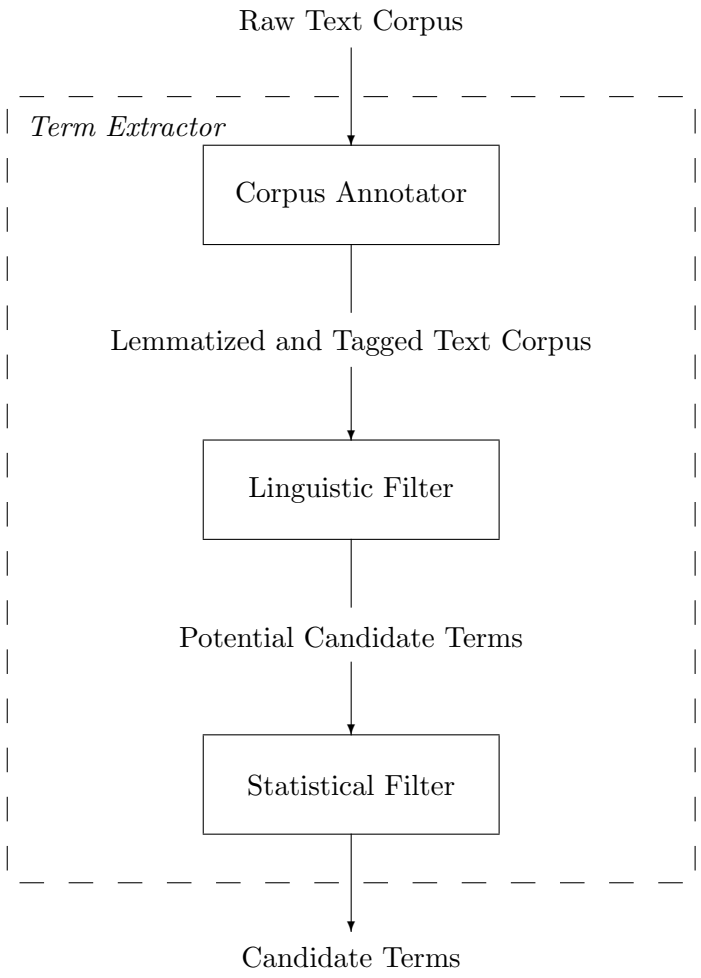

Figure 2.3: Architecture of the term extraction system.

Lemmatizer The lemmatizer component strips inflectional morphology, i.e. it changes nouns to singular form (e.g. *geese* becomes *goose*) and changes verbs to infinitive form (e.g. *were* becomes *be* and *had* becomes *have*).

In the architecture of the term extraction system (see Figure 2.3), the corpus annotator is followed by a linguistic filter. The linguistic filter is described next.

Syntactical structure	**Example**
N	algorithm
AN	fuzzy system
NN	pattern recognition
AAN	artificial neural network
ANN	radial basis function
NNN	support vector machine
NPN	curse of dimensionality

Table 2.1: Some syntactic structures.

2.4.2 Linguistic Filter

The linguistic filter selects from an annotated corpus words or sequences of words that are likely to be terms. This is accomplished by scanning the annotated corpus for all sequences of words that match certain syntactic structures. These structures can be described using the following regular expression E that is based on the part-of-speech tags of the words

$$\mathrm{E} = (\mathrm{A}|\mathrm{N})^*(\mathrm{NP})^?(\mathrm{A}|\mathrm{N})^*\mathrm{N} \tag{2.3}$$

In this regular expression, A is an adjective, N is a noun, P is a preposition, XY denotes element X followed by element Y, (X|Y) denotes either element X or element Y, $\mathrm{X}^?$ denotes one occurrence of element X or nothing at all, and X^* denotes zero or more occurrences of element X. Some of the syntactic structures which are covered by the regular expression are shown in Table 2.1. In this table each syntactic structure is accompanied with an example.

The regular expression in (2.3) has been taken from the work of Justeson and Katz (1995). The linguistic filtering procedure of Justeson and Katz makes use of a very similar regular expression. The only difference is that their regular expression only selects potential candidate terms that consist of more than one word (i.e. noun phrases), whereas our regular expression selects single-word candidate terms (i.e. nouns) too. Note further that in this chapter we take another approach to evaluate the effectiveness of the regular expression.

To improve the performance of the linguistic filter, we only allow the preposition *of*. It turns out that this gives better results in terms of recall and precision. Besides this, we make use of a stop word list. Phrases that are made up of one or more of the words on the stop word list are rejected as potential candidate term. The stop word list that we use is shown in Table 2.2. As one can see, the greater part of the stop word list consists of adjectives. The rest of the words in the list are nouns. The stop word list was obtained by analyzing the output of the term extraction system without the use of a stop word list. Words that were part of irrelevant terms and that were not part of relevant terms were put on the list.

absence	increase	presence
alternative	interesting	present
amount	less	previous
application	little	procedure
approach	main	property
article	many	real
author	methodology	result
basic	moderate	role
case	more	same
certain	much	setting
complex	natural	several
current	new	simple
decrease	newer	simpler
different	newest	simplest
extensive	next	special
few	old	specific
finding	older	study
general	oldest	such
hand	original	true
idea	other	use
illustrative	paper	way
important	practical	work

Table 2.2: Stop word list.

In the architecture of the term extraction system (see Figure 2.3), the linguistic filter is followed by a statistical filter. The statistical filter is described next.

2.4.3 Statistical Filter

The goal of the statistical filter is to reduce the amount of noise in the output of the linguistic filter. To do this, the statistical filter selects terms from the output of the linguistic filter based on some statistical criterion.

Two types of statistical filters will be used in the experiments: a filter which is based on frequency counts and a filter which is based on a population proportion test. In the next paragraphs the two filters are described in detail.

Frequency Counts

The working of a statistical filter which is based on frequency counts is quite simple and common. To reduce the amount of noise in the list of potential candidate terms, a statistical filter rejects potential candidate terms that occur less than a certain number of times in the corpus. In other words, potential candidate terms with a frequency lower than a certain threshold value are rejected as candidate term.

The probability that a combination of three or more words occurs by accident is lower than the probability that a combination of two words occurs by accident. As a consequence, for 3-word terms, 4-word terms, etc. a lower threshold value may be appropriate than for 2-word terms. Different threshold values may therefore be used for different term lengths.

Population Proportion Test

The procedure of a statistical filter which is based on a population proportion test is somewhat more complicated. The procedure not only involves the consideration of the frequencies of the potential candidate terms in the corpus from which terms have to be extracted (the analysis corpus), but also the consideration of the frequencies of the same terms in a reference corpus. The relative frequencies of the terms (i.e. the frequencies relative to the total number of terms in a corpus) as observed in both corpora are

compared with each other. The more significant the difference between the relative frequency of a potential candidate term in the analysis corpus and the relative frequency of the same term in the reference corpus, the more likely it is that the term is specific for one of the corpora.

By performing a population proportion test (e.g. Newbold, 1995), it can be decided how likely it is that a particular term is specific for the analysis corpus. A population proportion test involves the calculation of a test-score for each potential candidate term that has been extracted from the analysis corpus using the linguistic filter. The test-score of an extracted term is calculated as follows

$$\text{test-score} = \frac{\hat{p}_{AC} - \hat{p}_{RC}}{\sqrt{\hat{p}_0(1-\hat{p}_0)(\frac{n_{AC}+n_{RC}}{n_{AC}n_{RC}})}}, \tag{2.4}$$

where n_{AC} is the number of term occurrences observed in the analysis corpus, n_{RC} is the number of term occurrences observed in the reference corpus, $\hat{p}_{AC}$ is the proportion of term occurrences observed in the analysis corpus, and $\hat{p}_{RC}$ is the proportion of term occurrences observed in the reference corpus. The estimated common proportion $\hat{p_0}$ is given by

$$\hat{p}_0 = \frac{n_{AC}\hat{p}_{AC} + n_{RC}\hat{p}_{RC}}{n_{AC} + n_{RC}}. \tag{2.5}$$

The higher the test-score of a potential candidate term, the more likely it is that the term is relevant for the domain of the analysis corpus. So to reduce the amount of noise in the list of potential candidate terms, terms that have a test-score less than a certain threshold value are rejected. Just as was the case with the other statistical filter, for each term length a different threshold value may be used.

As stated before, the corpus from which the terms have to be extracted is a domain-specific technical corpus. The reference corpus is also a technical corpus but from another domain. By choosing a reference corpus which is also technical we hope to filter out general technical terms which are not domain relevant (i.e. terms without a special meaning). Examples of such terms are *problem*, *method*, *technique*, *experiment*, etc. Such general technical terms supposedly occur a lot in both the analysis corpus and the reference corpus. As a consequence, the test-scores of these terms are low

and the terms are therefore rejected as irrelevant. One might notice that the choice of a reference corpus may have an important influence on the effectiveness of the statistical filter.

2.5 Experiments

In this section, we describe the experiments we have carried out using the term extraction system presented the previous section. The corpora that were used in the experiments are described in Appendix A. The results of the experiments are reported in the next section.

2.5.1 Experiment 1: Extraction of Candidate Terms From the Computational Intelligence Corpus

The purpose of the first experiment was to extract a list of candidate terms for three subfields of the computational intelligence field. The extraction of the candidate terms was done by applying an implementation of the term extraction system presented in Section 2.4 to each of the computational intelligence subcorpora (see Section A.2). The implementation made use of a statistical filter which is based on frequency counts.

The procedure of extracting a list of candidate terms for each of the three subfields of the computational intelligence field went as follows. First, each computational intelligence subcorpus was lemmatized and tagged using the annotator component of the term extraction system. Next, from each lemmatized and tagged subcorpus, 1-word, 2-word, and 3-word potential candidate terms were extracted using the linguistic filter. For each computational intelligence subfield, this resulted in three list of potential candidate terms (one for each term length). Finally, to reduce the amount of noise in each of the potential candidate term lists, some terms were rejected using a statistical filter which is based on frequency counts. For this filtering, different frequency threshold values were used to obtain different levels of recall.

2.5.2 Experiment 2: Comparison of Statistical Filters

The purpose of the second experiment was to compare the performance of a statistical filter which is based on frequency counts with the performance of a statistical filter which is based on a population proportion test.

In order to compare the performance of the two types of statistical filters, two different implementations of the term extraction system presented in Section 2.4 were applied to each of the computational intelligence subcorpora (see Section A.2). One of these implementations made use of a statistical filter which is based on frequency counts. This implementation was the same as the one used in the first experiment. The other implementation made use of a statistical filter which is based on a population proportion test. With this implementation two tests were performed. In the first one the finance corpus (see Section A.3) served as reference corpus, in the other one the statistics corpus (see Section A.4) served as reference corpus.

The difference in performance between the two types of statistical filters is probably best seen in the results of the extraction of single-word terms. The aim of the statistical filter which is based on a population proportion test is to filter out general terms like, e.g., *problem* and *method*. Usually, general terms exist of a single word. This is the reason why in this experiment we focused on the extraction of 1-word candidate terms.

2.6 Results and Discussion

In this section, we present and discuss the results of the experiments described in the previous section. First, we describe the procedure that was followed in order to validate the results. Then, we present and discuss the results of this validation.

2.6.1 Validation Procedure

The results of the experiments are lists of candidate terms for each of the three subfields of the computational intelligence field. In order to evaluate the quality of these lists of candidate terms, the lists were validated manually by a human domain expert. This expert had good knowledge of each of the three subfields of the computational intelligence field. For each candidate

term the expert decided whether the term is relevant for the corresponding computational intelligence subfield.

The validation procedure relies on human judgement. As a consequence, we cannot ensure that the obtained results are free of subjectivity. Different domain experts will assumably consider different terms to be relevant for a certain field and the same holds for one domain expert over a longer period of time. It might be interesting to take into account this influence of human judgement in the validation procedure, but doing so is beyond the scope of this thesis.

After completing the validation procedure for the two different experiments, for each candidate term it was known whether the term was judged as relevant. This made it possible to express the results of the experiments using the quality measures of recall and precision (see Paragraph 2.2.3). Recall is defined as the ratio between the number of relevant candidate terms and the total number of relevant terms (extracted or not). The total number of relevant terms for the computational intelligence subfields was not known. We estimated the total number of relevant terms as the number of potential candidate terms (i.e. the terms selected by the linguistic filter) that were judged as relevant by the domain expert. This is an estimate because there may exist relevant terms that were not selected as potential candidate term by the linguistic filter. Consequently, the measured recall is not equal to the actual recall but to an estimate of the actual recall. Precision is defined as the ratio between the number of relevant candidate terms and the total number of candidate terms.

In the next paragraphs, the results of the experiments are presented and discussed.

2.6.2 Experiment 1: Extraction of Candidate Terms From the Computational Intelligence Corpus

In Table 2.3, 2.4, and 2.5, the results of the first experiment are shown. In the tables, the most frequent 1-word, 2-word, and 3-word candidate terms extracted from the three computational intelligence subcorpora are presented. For each candidate term the human domain expert decided whether the term was relevant to the corresponding subfield. The result of this validation is

reported in the last column of the tables.

Inspecting the tables results in a number of notable observations. These observations are discussed next:

1. A relatively large number of the most frequent 1-word candidate terms was not considered as relevant by the domain expert (see Table 2.3). For the different computational intelligence subfields, it concerns mostly the same terms. These terms are *problem*, *algorithm*, *method*, *model*, *performance*, and *simulation*. On their own these are correct terms, but because these terms are so general in meaning they were judged by the domain expert as irrelevant for the computational intelligence subfields. This phenomenon does not arise, or to a far lesser degree, for the most frequent 2-word and 3-word candidate terms (see Table 2.4 and 2.5).

2. The number of occurrences of the most frequent 1-word and 2-word candidate terms (see Table 2.3 and 2.4) was much higher than the number of occurrences of the most frequent 3-word candidate terms (see Table 2.5). This corresponds with the observations of Justeson and Katz (1995).

3. The number of occurrences of the most frequent candidate terms for the neural network subfield was much higher than the number of occurrences of the most frequent candidate terms for the fuzzy system and evolutionary computation subfield. This difference in the number of occurrences of the candidate terms between the different subfields can be explained by the fact that the computational intelligence corpus contained much more abstracts related to neural networks than abstracts related to fuzzy systems and evolutionary computation.

4. Many of the most frequent 3-word candidate terms for the evolutionary computation subfield were judged by the domain expert as irrelevant (see Table 2.5). The small number of abstracts related to this subfield is assumed to be the explanation. Because of the few abstracts, the number of occurrences of the extracted candidate terms were very

Subfield	Candidate terms	Number of occurrences	Relevant for subfield
Evolutionary computation	problem	113	no
	algorithm	106	no
	method	71	no
	ga	57	yes
	solution	44	yes
	ea	42	yes
	performance	39	no
	parameter	38	yes
	model	37	no
	population	35	yes
Fuzzy systems	method	205	no
	algorithm	182	no
	system	171	yes
	model	137	no
	rule	117	yes
	controller	108	yes
	performance	107	no
	problem	97	no
	simulation	75	no
	design	69	no
Neural networks	network	1379	yes
	algorithm	1195	no
	model	878	no
	method	851	no
	problem	583	no
	system	478	no
	performance	423	no
	simulation	406	no
	neuron	390	yes
	data	363	yes

Table 2.3: Most frequent 1-word candidate terms extracted from the computational intelligence subcorpora.

Subfield	Candidate terms	Number of occurrences	Relevant for subfield
Evolutionary computation	genetic algorithm	85	yes
	evolutionary algorithm	45	yes
	optimization problem	18	yes
	search space	18	yes
	evolution strategy	17	yes
	genetic programming	16	yes
	genetic operator	15	yes
	pareto front	13	yes
	local search	12	yes
	benchmark problem	11	no
Fuzzy systems	fuzzy system	83	yes
	fuzzy model	80	yes
	membership function	66	yes
	fuzzy rule	60	yes
	fuzzy controller	58	yes
	fuzzy set	46	yes
	neural network	42	yes
	fuzzy logic	41	yes
	nonlinear system	29	yes
	genetic algorithm	28	yes
Neural networks	neural network	921	yes
	computer simulation	117	no
	multilayer perceptron	91	yes
	training data	68	yes
	activation function	67	yes
	cost function	60	yes
	associative memory	59	yes
	input pattern	57	yes
	local minimum	54	yes
	sufficient condition	54	no

Table 2.4: Most frequent 2-word candidate terms extracted from the computational intelligence subcorpora.

Subfield	Candidate terms	Number of occurrences	Relevant for subfield
Evolutionary computation	number of generations	7	no
	multiobjective evolutionary algorithm	6	yes
	network random key	5	no
	optimal wheel slip	5	no
	d ca prng	4	no
	learning classifier system	4	yes
	class of problems	3	no
	classical numerical method	3	no
	evolutionary computation technique	3	no
	multiobjective optimization problem	3	yes
Fuzzy systems	fuzzy logic controller	26	yes
	linear matrix inequality	18	yes
	fuzzy rule base	16	yes
	fuzzy logic system	14	yes
	fuzzy neural network	14	yes
	fuzzy inference system	13	yes
	loop fuzzy system	11	no
	number of rules	11	no
	fuzzy control system	10	yes
	fuzzy pd controller	9	no
Neural networks	recurrent neural network	101	yes
	neural network model	97	yes
	artificial neural network	87	yes
	radial basis function	62	yes
	feedforward neural network	53	yes
	basin of attraction	41	yes
	number of neurons	41	no
	support vector machine	34	yes
	principal component analysis	33	yes
	large scale integration	32	no

Table 2.5: Most frequent 3-word candidate terms extracted from the computational intelligence subcorpora.

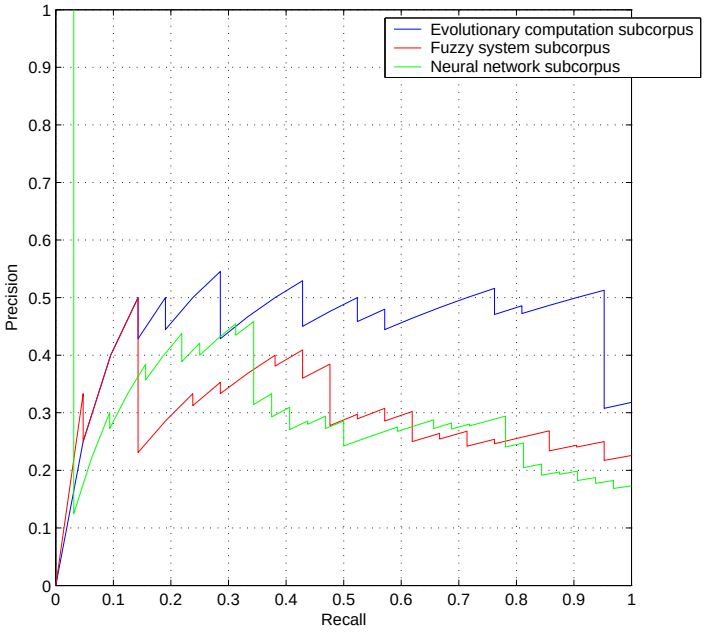

Figure 2.4: Precision-recall graph for the extraction of 1-word candidate terms from the computational intelligence subcorpora.

low. Consequently, the likelihood that candidate terms were correct and relevant was low too.

Using recall and precision as quality measures, we can evaluate the performance of the extraction of candidate terms from the computational intelligence subcorpora.

Recall and Precision

In Figure 2.4, 2.5, and 2.6, the results of the experiment are shown in terms of recall and precision. In the figures, the relation between recall and precision for the extraction of 1-word, 2-word, and 3-word candidate terms from the different computational intelligence subcorpora is shown. The blue line represents the extraction of candidate terms from the evolutionary computation subcorpus, the red line represents the extraction of candidate terms from the fuzzy system subcorpus, and the green line represents the extraction of candidate terms from the neural network subcorpus.

Inspecting the figures results in a number of notable observations. These

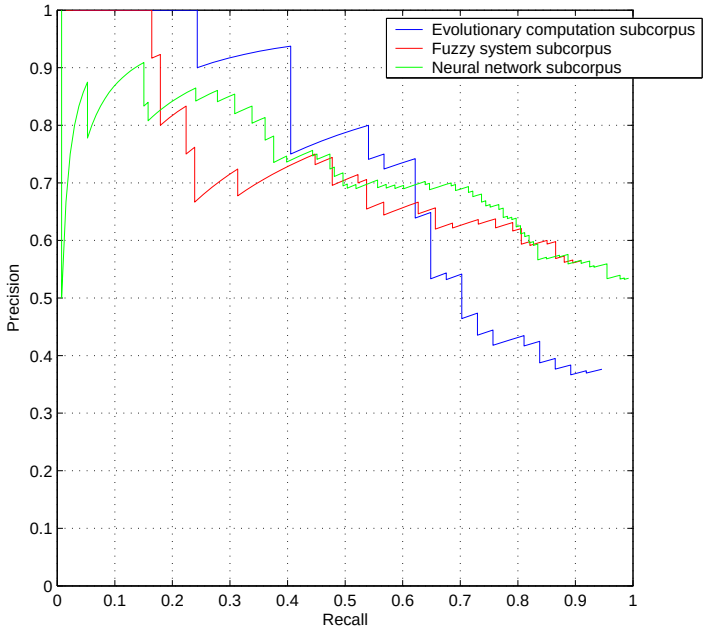

Figure 2.5: Precision-recall graph for the extraction of 2-word candidate terms from the computational intelligence subcorpora.

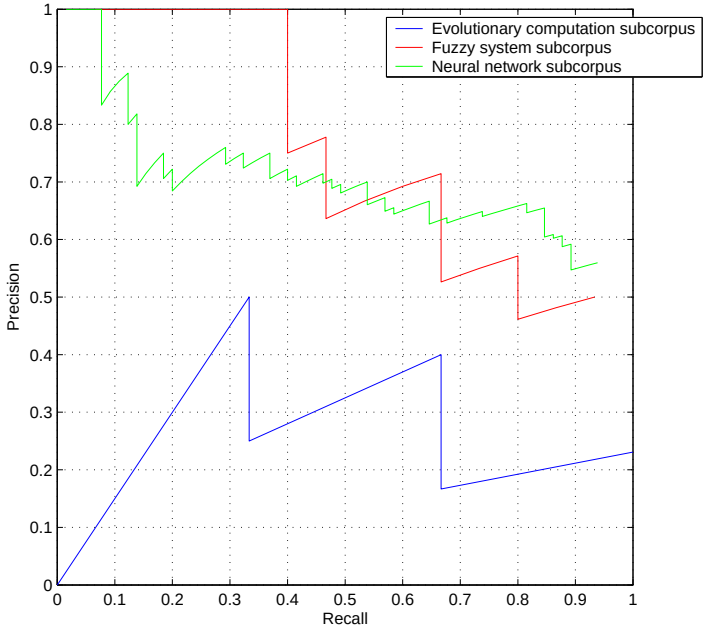

Figure 2.6: Precision-recall graph for the extraction of 3-word candidate terms from the computational intelligence subcorpora.

observations are discussed next:

1. In the figures, the step size of the precision-recall line for the neural network subfield is much smaller than the step size of the precision-recall line for the fuzzy system subfield and the evolutionary computation subfield. This difference in step size can especially be seen in the precision-recall graph for the extraction of 3-word candidate terms (see Figure 2.6). The difference in step size can be explained by the fact that more different relevant candidate terms were extracted for the neural network subfield than for the fuzzy system subfield and the evolutionary computation subfield. The precision-recall trade-off can therefore be shown much more precisely for the neural network subfield than for the other two subfields.

2. The performance of the extraction of 1-word candidate terms from the different computational intelligence subcorpora (see Figure 2.4) is a little bit disappointing. The precision at low recall rates was very poor. This is a disappointing result that was caused by the fact that general terms which were irrelevant for the computational intelligence subfields (e.g. *problem* and *algorithm*) belonged to the most frequent candidate terms. While at higher recall rates the precision did not decreases very much, the combined precision and recall is still disappointing.

3. The performance of the extraction of 2-word candidate terms (see Figure 2.5) was much better than the extraction of 1-word candidate terms (see Figure 2.4). Especially at low recall rates, the precision was much higher. Furthermore, it can be seen that as recall increases, the precision of the extraction from the evolutionary computation subcorpus (blue line) decreased faster than the precision of the extraction from the fuzzy system subcorpus (red line) and the neural network subcorpus (green line).

4. Except for the evolutionary computation subcorpus (blue line), the performance of the extraction of 3-word candidate terms (see Figure 2.6) corresponded quite well to the performance of the extraction of 2-word candidate terms (see Figure 2.5). The precision of the ex-

traction from the evolutionary computation subcorpus was low at all recall rates because of the before mentioned fact that this subcorpus contained a small number of abstracts. Because of the small number of abstracts, the number of occurrences of the extracted candidate terms was very low. Consequently, it was difficult to determine whether a term was correct and relevant.

2.6.3 Experiment 2: Comparison of Statistical Filters

Using recall and precision as quality measures, we can compare the performance of the two different statistical filters.

Recall and Precision

In Figure 2.7, 2.8, and 2.9 the results of the experiment are shown in terms of recall and precision. In the figures, the relation between recall and precision for the extraction of 1-word candidate terms from the evolutionary computation subcorpus, the fuzzy system subcorpus, and the neural network subcorpus is shown. The blue line represents the extraction of candidate terms using a statistical filter which is based on frequency counts, the red line represents the extraction of candidate terms using a statistical filter which is based on a population proportion test with a finance corpus (see Section A.3) as reference corpus, and the green line represents the extraction of candidate terms using a statistical filter which is based on a population test with a statistics corpus (see Section A.4) as reference corpus.

By inspecting the figures, no large differences in the performance between the two statistical filters can be observed. At certain recall rates, the precision of the extraction with a statistical filter which is based on frequency counts is a little bit better than the precision of the extraction with a statistical filter which is based on a population test with the finance or statistics corpus. At other recall rates, just the opposite is the case.

2.7 Conclusions and Future Research

Term extraction is the computer-assisted process of extracting terms from text documents. In this chapter, we have presented a term extraction sys-

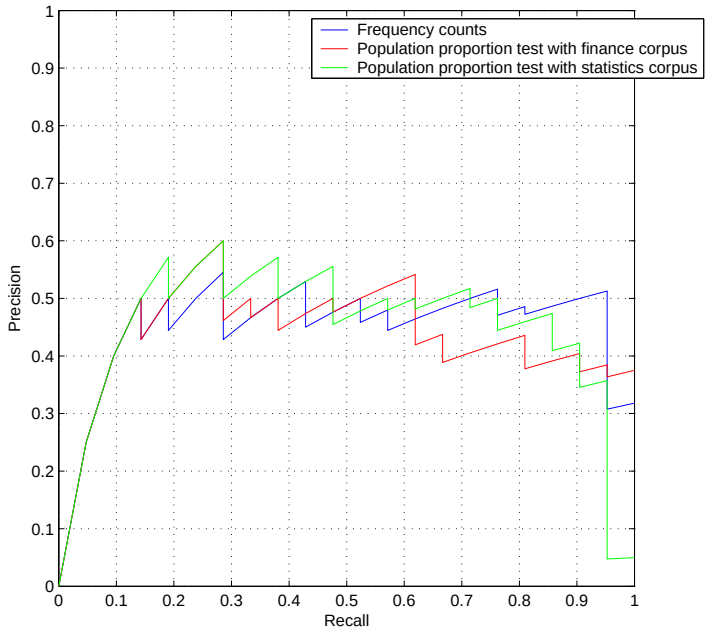

Figure 2.7: Precision-recall graph for the extraction of 1-word candidate terms from the evolutionary computation subcorpus using different statistical filters.

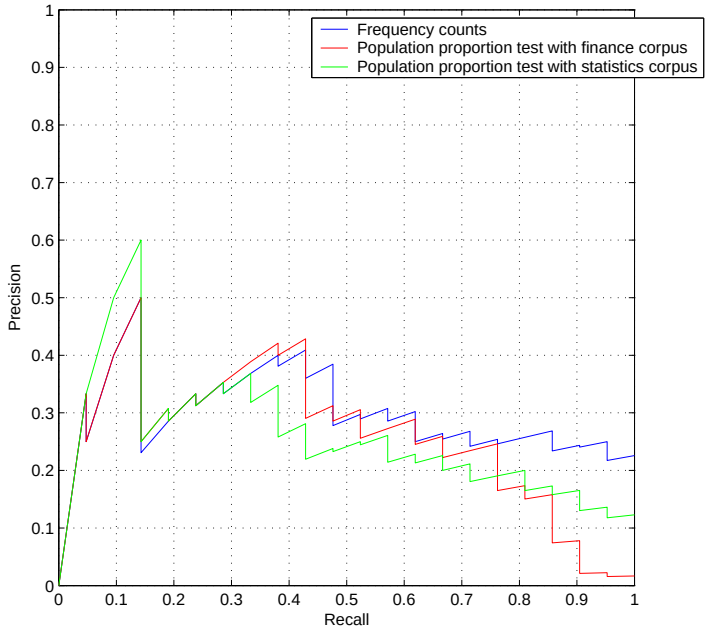

Figure 2.8: Precision-recall graph for the extraction of 1-word candidate terms from the fuzzy system subcorpus using different statistical filters.

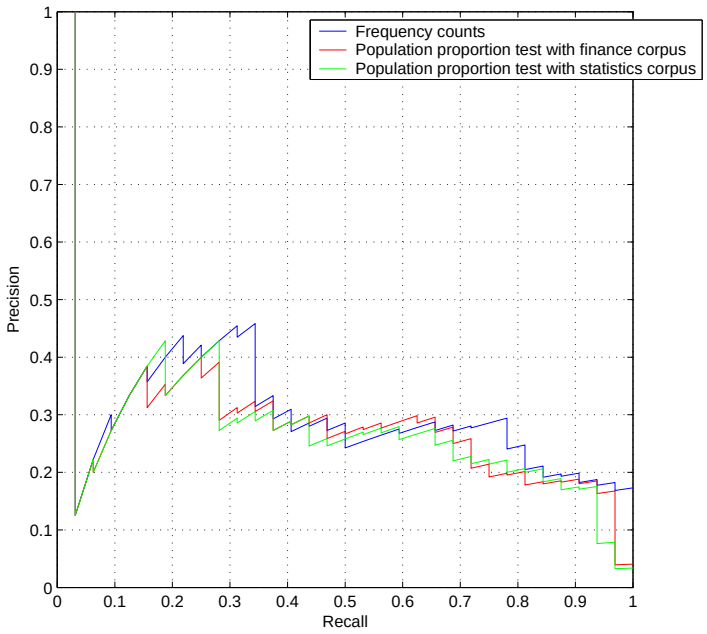

Figure 2.9: Precision-recall graph for the extraction of 1-word candidate terms from the neural network subcorpus using different statistical filters.

tem that aims to extract domain relevant terms from text documents. The system makes use of both a linguistic filter and a statistical filter. First, the linguistic filter was used to select from text documents words and phrases that were likely to be terms. The statistical filter was then used to decide whether the selected words and phrases could be considered as candidate terms. We have described experiments in which our term extraction system had been applied to abstracts from journals that are representative for the computational intelligence field. In one experiment, candidate terms were extracted for each of the subfields of the computational intelligence field. In this experiment, it has turned out that the quality of the extracted multi-word candidate terms is higher than that of the extracted single-word candidate terms. Furthermore, it has turned out that as more abstracts are used as input to the term extraction system, the quality of the extracted candidate terms improves. It is therefore important to have a sufficient number of abstracts available. In another experiment, two different implementations of our term extraction system were compared with each other. The two implementations made use of different statistical filters. In this

experiment, it has turned out that the quality of candidate terms extracted using the two implementations does not differ significantly in terms of recall and precision.

The most important issue for future research is the improvement of the extraction of single-word candidate terms. The quality of the extracted single-word terms is disappointing due to the extraction of general terms that are not domain relevant. A simple option to filter out such general terms is to precisely find out which terms are general in meaning and simply reject them during extraction. Other issues for future research are the investigation of a more advanced statistical filter, the application of our term extraction system to other corpora, and the comparison of the performance of our term extraction system with the performance of existing systems.

Chapter 3

Visualization of Concept Associations

3.1 Introduction

Knowledge domain visualization is concerned with the creation of maps that depict the structure and evolution of a scientific field (Börner et al., 2003). These domain maps are typically constructed on the basis of a corpus of scientific texts. In a domain map, items are located in such a way that the distance between two items reflects their similarity. The stronger the similarity between two items, the smaller the distance between the items. The type of item in a domain map depends on the question that one wants to answer. The most common types of items are scientific journals, scientific articles, authors, and descriptive words or terms. Each type of item can be used to visualize a different aspect of a scientific field.

The process of knowledge domain visualization can be divided into two steps:

1. the calculation of the similarities between items, and
2. the positioning of items in a two- or three-dimensional space based on the similarities.

The second step is usually performed using multidimensional scaling (MDS) (e.g. Mardia et al., 1979). In the first step, one typically calculates for each combination of two items the co-occurrence frequency in a corpus of scientific texts. The co-occurrence frequencies are stored in a co-occurrence

matrix. This matrix is converted into a similarity matrix. In the literature, two approaches are described that can be used for this conversion. One approach is to normalize the frequencies in the co-occurrence matrix using, for example, the Dice, Jaccard (e.g. Peters and van Raan, 1993; Kopcsa and Schiebel, 1998), or cosine measure. Another approach is to calculate the Pearson correlation coefficients between the rows (or columns) of the co-occurrence matrix and to store these coefficients in the similarity matrix (e.g. McCain, 1990; Ding et al., 2001). In the first approach, two items are considered similar if they co-occur frequently. This approach only takes into account what we call the direct similarity between items. In the second approach, two items are considered similar if they have similar co-occurrence profiles, i.e. if they co-occur with the same items. This means that two items that do not co-occur with each other are still considered similar if their co-occurrence profiles are similar. Instead of the direct similarity, the second approach takes into account what we call the indirect similarity between items.

This chapter focuses on the construction of maps that visualize the associations between concepts in a scientific field. Following van den Berg and Schuemie (1999); van der Eijk et al. (2004), we refer to these maps as associative concept spaces. In an associative concept space (ACS), concepts that are strongly associated are located close to each other. An ACS can be used to obtain an overview of a scientific field and, more specifically, of a field's important concepts and their mutual associations. Another application of an ACS, which we do not consider in this chapter, is to support the discovery of unknown associations between concepts (van der Eijk et al., 2004). When constructing an ACS, not only direct associations between concepts should be taken into account but also indirect associations, for example, the association between the concepts c_1 and c_2 because they are both (directly) associated with concept c_3. Indirect associations are important because we believe that they result in a better visualization and because they may indicate associations that are still unknown. An ACS may be constructed using MDS in combination with the Pearson correlation coefficient, as described above. The other approach described above, in which MDS is applied to a normalized co-occurrence matrix, cannot be followed if indirect associations

have to be taken into account.

The purpose of this chapter is to propose a new algorithm for constructing an ACS. This algorithm takes both direct and indirect associations between concepts into account. It can be seen as an alternative to the combination of MDS and the Pearson correlation coefficient. In the experiments that we describe in this chapter, the proposed algorithm and MDS are both used for constructing an ACS of the computational intelligence field. It turns out that the associations between concepts in this field are better reflected in the ACS constructed using the proposed algorithm than in the ACS constructed using MDS.

The chapter is organized as follows. Related research is discussed in Section 3.2. In Section 3.3, the new algorithm for visualizing concept associations is presented. The setup and the results of the experiments are described in Section 3.4 and 3.5, respectively. Finally, in Section 3.6, conclusions and future research are discussed.

3.2 Related Research

Co-word analysis (e.g. Peters and van Raan, 1993; Kopcsa and Schiebel, 1998; Ding et al., 2001) is a technique for knowledge domain visualization that is closely related to the research presented in this chapter. It is concerned with the construction of maps of keywords on the basis of co-occurrences in a text corpus. The maps, which are referred to as co-word maps, can be used to visualize the structure of a scientific field and to reveal new developments in a field. The construction of a co-word map is similar to the general process of knowledge domain visualization, which was discussed in the previous section. First, a co-occurrence matrix is calculated and converted into a similarity matrix. Then, the similarity matrix is mapped to a co-word map, usually by applying MDS (e.g. Mardia et al., 1979). Cluster analysis is also frequently used in the last step.

As an alternative to MDS and cluster analysis, the algorithm introduced by Kopcsa and Schiebel (1998) may be used to construct a co-word map. This algorithm is inspired by mass point mechanics. Each keyword is seen as a mass point, and the similarity between two keywords is seen as the

elasticity between the corresponding mass points. A mass point is affected by two types of forces, one is caused by the elasticities with other mass points and the other is caused by frictional resistance. The idea of the algorithm is that keywords can be assigned to appropriate locations in a co-word map by finding a solution in which the forces on each mass point are in equilibrium. It is claimed by Kopcsa and Schiebel (1998) that the proposed algorithm constructs better co-word maps than MDS and cluster analysis. It should be noted that the algorithm only considers direct similarities between keywords.

In Schuemie (1998); van den Berg and Schuemie (1999); van der Eijk (2001); van der Eijk et al. (2004), so-called Hebbian learning algorithms are used to construct an ACS. These algorithms first assign concepts to randomly chosen locations in a concept space and then iteratively move concepts through the concept space according to a learning rule and a forgetting rule. The learning rule is responsible for moving concepts that are associated towards each other. The forgetting rule, on the other hand, is responsible for moving concepts away from each other to prevent concepts from being positioned at the same location in the concept space. The idea of the Hebbian learning algorithms is that after a sufficient number of iterations the distances between the concepts in a concept space should reflect the strengths of the associations between the concepts. However, in van den Berg et al. (2004) we argue that the Hebbian learning algorithms have some unfavorable properties because of which concepts may not be positioned at appropriate locations in a concept space.

In van den Berg et al. (2004), we also present an alternative algorithm for constructing an ACS. This algorithm can be seen as a precursor to the algorithm that we propose in this chapter. The most important modification that we have made to the algorithm presented in van den Berg et al. (2004) is a refinement of the objective function.

3.3 A Novel Algorithm for Constructing Associative Concept Spaces

In this section, a new algorithm for constructing an ACS is proposed. The algorithm needs a concept association matrix as input. This is a matrix that

contains the strengths of the associations between concepts. The proposed algorithm uses the association strengths to position concepts at appropriate locations in an ACS.

The properties of a concept association matrix and the calculation of such a matrix on the basis of co-occurrences in a text corpus are discussed in Paragraph 3.3.1. The algorithm for constructing an ACS is presented in Paragraph 3.3.2.

3.3.1 Concept Association Matrices

Let $c_1, \ldots, c_n$ denote the concepts of interest, where n indicates the number of concepts. A concept association matrix $\mathbf{A}$ is an $n \times n$ matrix that contains for each combination of two concepts the strength of their association. Element a_{ij} of $\mathbf{A}$ is referred to as the association strength between the concepts c_i and c_j. The association strengths a_{ij} must satisfy the following conditions

$$a_{ij} \geq 0, \qquad \text{for } i, j = 1, \ldots, n \tag{3.1}$$

$$a_{ii} = 0, \qquad \text{for } i = 1, \ldots, n \tag{3.2}$$

$$a_{ij} = a_{ji}, \qquad \text{for } i, j = 1, \ldots, n \tag{3.3}$$

$$\sum_{j=1}^{n} a_{ij} > 0, \qquad \text{for } i = 1, \ldots, n. \tag{3.4}$$

The association strengths in a concept association matrix can be determined in different ways. One approach is to calculate them on the basis of co-occurrences in a text corpus. In this approach, concepts are first identified in the corpus using a thesaurus. For each combination of two concepts, the association strength is then calculated as the co-occurrence frequency of the concepts in the corpus. This approach is taken in the experiments that are described in Section 3.4 and 3.5. Notice that in this approach an association matrix is identical to a co-occurrence matrix.

3.3.2 Associative Concept Space Algorithm

We propose an algorithm that minimizes an objective function in order to assign concepts to appropriate locations in an ACS. We denote the location

of concept c_i by the vector $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})^T$, where the parameter d indicates the number of dimensions of the concept space.

The algorithm assigns to each concept c_i a weight w_i. A concept's weight is calculated as follows

$$w_i = \left(\sum_{j=1}^{n} a_{ij} \right)^{\alpha} \tag{3.5}$$

where the value of the parameter $\alpha \in [0, 1]$ has to be specified by the user. Using (3.5), the stronger a concept's associations with other concepts, the higher the concept's weight.

The underlying idea of the algorithm is that each concept should be positioned as close as possible to its ideal location. The ideal location $\mathbf{x}_i^*$ of concept c_i is defined as

$$\mathbf{x}_i^* = \frac{\sum_{j=1}^{n} a_{ij} \mathbf{x}_j}{\sum_{j=1}^{n} a_{ij}}. \tag{3.6}$$

Or, in words, a concept's ideal location is the weighted average of the locations of the concepts with which it is associated, where the association strengths a_{ij} are used as weights. It follows from (3.6) that the only way to position each concept at its ideal location is to assign all concepts to the same location. This, of course, does not result in a useful concept space. The algorithm therefore attempts not only to position concepts as close as possible to their ideal location but also to prevent concepts from being located too close to each other. To achieve this, the algorithm minimizes the following objective function

$$E = \sum_{i=1}^{n} \left(\bar{w}_i \|\mathbf{x}_i - \mathbf{x}_i^*\|^2 + \beta \sum_{\substack{j=1 \\ j \neq i}}^{n} e^{-\|\mathbf{x}_i - \mathbf{x}_j\|} \right) \tag{3.7}$$

where $\bar{w}_i = n w_i / \sum_{j=1}^{n} w_j$ and where $\| \cdot \|$ denotes the Euclidean norm. The value of the parameter $\beta > 0$ has to be specified by the user. In (3.7), the first term within the parentheses is responsible for positioning concepts as close as possible to their ideal location. This term pays more attention to concepts with higher weights. The second term within the parentheses is responsible for preventing concepts from being located too close to each other. It should be noted that the objective function in (3.7) was chosen

after some experimenting. A number of variants of (3.7) were also tested but gave less satisfactory results.

Starting from a random initialization of concept locations, a gradient descent algorithm may be used to find a (local) minimum of the objective function in (3.7). Concepts are then iteratively moved through the concept space in the opposite direction of the gradient of the objective function. The gradient of the objective function with respect to concept location $\mathbf{x}_i$ is given by

$$\nabla_{\mathbf{x}_i} E = 2 \left(\bar{w}_i (\mathbf{x}_i - \mathbf{x}_i^*) - \sum_{j=1}^{n} \frac{a_{ji} \bar{w}_j (\mathbf{x}_j - \mathbf{x}_j^*)}{\sum_{k=1}^{n} a_{jk}} + \beta \sum_{\substack{j=1 \\ j \neq i}}^{n} e^{-\|\mathbf{x}_j - \mathbf{x}_i\|} \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} \right). \tag{3.8}$$

3.4 Experiments

In this section, we discuss the experiments that were performed. The results of the experiments are reported in the next section.

The purpose of the experiments was to compare the results of the algorithm introduced in Section 3.3 with the results of MDS. Both algorithms were applied to the same concept association matrix $\mathbf{A}$. This matrix contained the strengths of the concept associations in the corpus of computational intelligence abstracts described in Appendix A. Element a_{ij} of $\mathbf{A}$ was calculated as the number of abstracts in which concept c_i and concept c_j co-occurred ($a_{ij} = 0$ for $i = j$). Concepts were identified in an abstract using the thesaurus of the computational intelligence field described in Appendix B.

In the first experiment, the association matrix was mapped to a two-dimensional concept space using the algorithm discussed in Section 3.3. The parameters α and β were set to 1/3 and 0.001, respectively. By trial-and-error experimenting, it had been found that these values give good results. The initial locations of the concepts were drawn from a uniform distribution

on $[0, 1]^2$. A gradient descent algorithm was used to minimize the objective function. The number of iterations and the learning rate of the gradient descent algorithm were set to fixed values of, respectively, 500 and 0.1. It was found that after 500 iterations the gradient descent algorithm had always converged to a local minimum of the objective function. The gradient descent algorithm was run 10 times using different initial concept locations. The concept space with the lowest value of the objective function was taken as the final result of the experiment.

In the second experiment, the association matrix was mapped to a two-dimensional concept space using MDS (e.g. Mardia et al., 1979). The association matrix was first converted into a correlation matrix $\mathbf{R}$. Element r_{ij} of $\mathbf{R}$ was calculated as the Pearson correlation coefficient between the ith and the jth row (or, equivalently, column) of the association matrix. The elements on the main diagonal of the association matrix were treated as missing values. The approach of converting an association matrix into a correlation matrix is also taken in Ding et al. (2001), where two-dimensional maps of keywords are constructed based on co-word analysis. A more detailed discussion of the calculation of a correlation matrix is provided in McCain (1990), which is concerned with author co-citation analysis. In the experiment, we applied MDS to the correlation matrix by using the PROXSCAL algorithm in the statistical software package SPSS. The elements of the correlation matrix were treated as ordinal data (non-metric MDS). The default parameter settings of the PROXSCAL algorithm were used.

3.5 Results and Discussion

In Figure 3.1, an overview of the ACS constructed using the algorithm discussed in Section 3.3 is shown. An overview of the ACS constructed using MDS is shown in Figure 3.2. In this section, we will simply refer to these concept spaces as ACS1 and ACS2, respectively. In Figure 3.1 and 3.2, each dot denotes a concept from the field of computational intelligence. The color of a dot depends on the thesaurus from which the corresponding concept was taken. Blue dots refer to concepts from the evolutionary computation thesaurus, red dots refer to concepts from the fuzzy system thesaurus, and

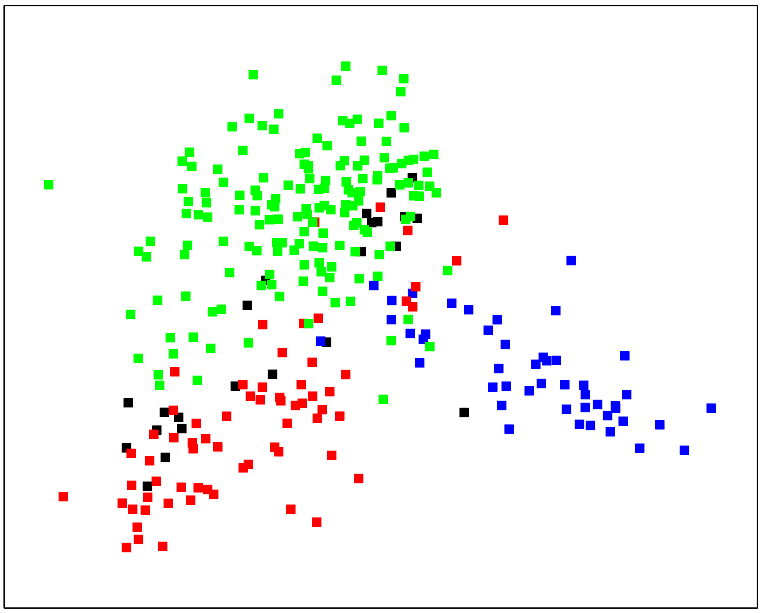

Figure 3.1: The ACS constructed using the algorithm discussed in Section 3.3 (ACS1).

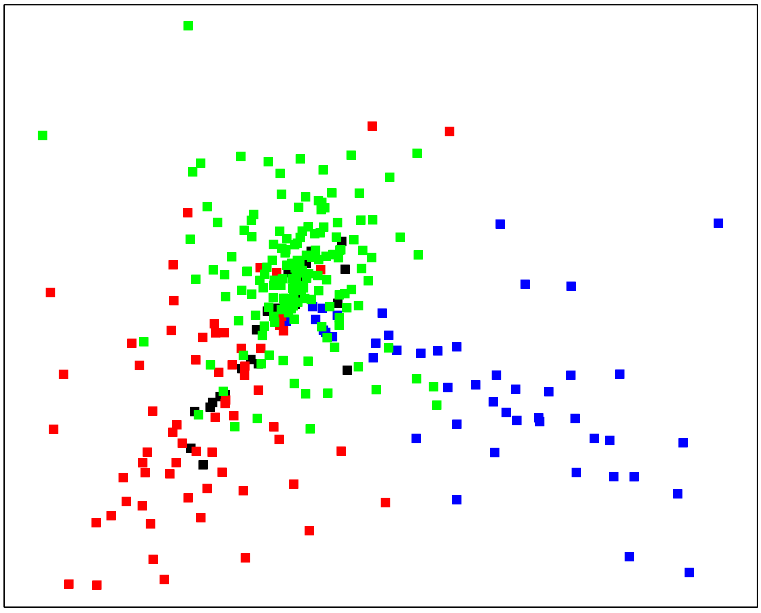

Figure 3.2: The ACS constructed using MDS (ACS2).

green dots refer to concepts from the neural network thesaurus. Furthermore, black dots refer to general concepts that were found in more than one thesaurus.

As might be expected, both in ACS1 and in ACS2 three clusters can be identified that roughly correspond with the three subfields of the computational intelligence field. We think, however, that in ACS1 the clustering of the concepts is much better than in ACS2. This is based on two observations. The first observation is that in Figure 3.1 the different colors are better separated than in Figure 3.2. This indicates that in ACS1 there is a better separation of concepts from different subfields than in ACS2. The second observation is that in Figure 3.1 the blue dots (the concepts from the evolutionary computation subfield) are less scattered through the concept space than in Figure 3.2. The same holds for the red dots (the concepts from the fuzzy system subfield).

For a detailed comparison of ACS1 and ACS2, the viewer software that we have made available online at `http://www.few.eur.nl/few/people/nvaneck/` can be used. The software has scroll, zoom, and search functionality to support a comprehensive examination of a concept space. In Figure 3.3, the interface of the software is shown. Using the software, we found the following notable differences between ACS1 and ACS2:

1. ACS1 contains a cluster of about 15 concepts (e.g. *associative memory*, *Hopfield network*, and *storage capacity*) that are all related to a specific type of neural network called Hopfield network. A similar cluster cannot be found in ACS2. In ACS2, there is a small cluster that contains some concepts related to Hopfield networks (e.g. *Hopfield network* and *energy function*). However, most concepts related to Hopfield networks (e.g. *associative memory*, *storage capacity*, and *Boltzmann machine*) are not located in this cluster. These concepts are scattered between other neural network concepts, which are not directly related to Hopfield networks.

2. In the evolutionary computation cluster in ACS1, strongly related concepts are located closer to each other than in the evolutionary computation cluster in ACS2. Some examples are the concepts *crossover*

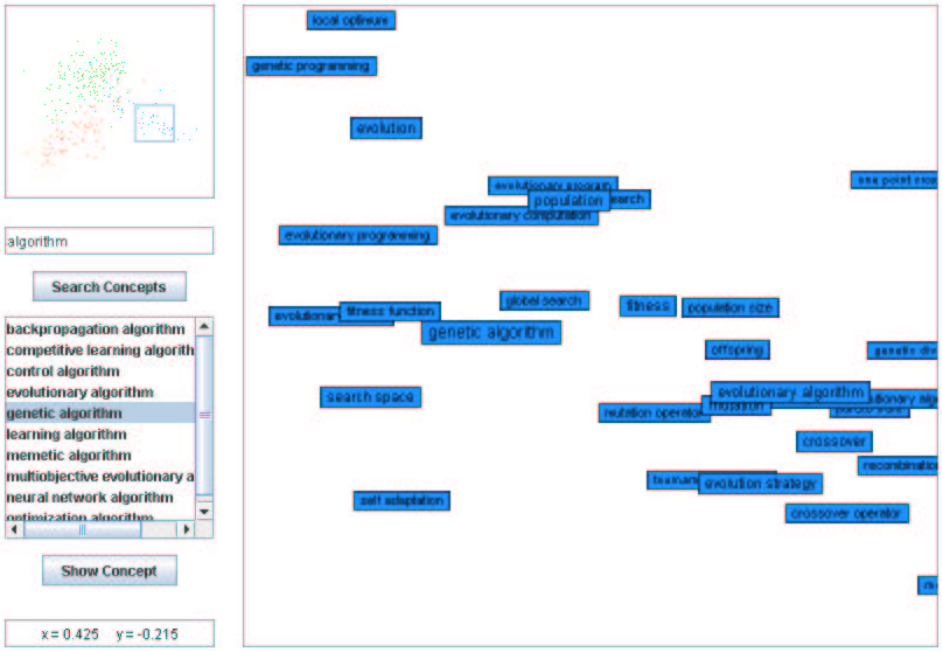

Figure 3.3: Interface of the ACS viewer software.

and *crossover operator*, the concepts *recombination* and *recombination operator*, the concepts *evolutionary algorithm* and *multiobjective evolutionary algorithm*, and the concepts *optimization*, *optimization algorithm*, and *optimal solution*. Furthermore, in ACS1 *genetic algorithm*, which is a central concept in the evolutionary computation subfield, is located in the middle of the evolutionary computation cluster. In ACS2, on the other hand, *genetic algorithm* is located close to the boundary between the evolutionary computation cluster and the neural network cluster.

3. Similarly to *genetic algorithm*, *fuzzy system*, which is, of course, a central concept in the fuzzy system subfield, is located differently in ACS1 and ACS2. In ACS1, *fuzzy system* has a central location between a number of strongly related concepts. In ACS2, on the other hand, *fuzzy system* is located close to the boundary between the fuzzy system cluster and the neural network cluster. Therefore, the distance between *fuzzy system* and, for example, *Takagi-Sugeno system*, which

is a specific type of fuzzy system, is much larger in ACS2 than in ACS1.

Based on both the high-level comparison of ACS1 and ACS2 using Figure 3.1 and 3.2 and the detailed comparison using the viewer software, we believe that associations between concepts from the field of computational intelligence are better reflected in ACS1 than in ACS2.

3.6 Conclusions and Future Research

An ACS is a map that visualizes the associations between concepts in a scientific field. In this chapter, we have proposed a new algorithm for constructing an ACS. This algorithm can be seen as an alternative to the combination of MDS and the Pearson correlation coefficient. In the literature on knowledge domain visualization, the combination of MDS and the Pearson correlation coefficient is frequently applied to problems that are similar to the construction of an ACS (e.g. Ding et al., 2001). We have described experiments in which the proposed algorithm and MDS had both been used for constructing an ACS of the computational intelligence field. It has turned out that the associations between concepts in this field are better reflected in the concept space that had been constructed using the proposed algorithm than in the concept space that had been constructed using MDS. In the experiments, the proposed algorithm had therefore performed better than MDS. However, the results of the experiments are not entirely conclusive, since only a limited evaluation of the concept spaces had been performed and only a single scientific field had been considered.

The most important issue for future research is a more comprehensive evaluation of the algorithm presented in this chapter. Other issues for future research are the choice of appropriate values for the parameters α and β, the investigation of variants of the objective function in (3.7), and the application of the algorithm to other problems in the field of knowledge domain visualization, like the construction of maps of authors based on co-citation analysis (McCain, 1990).

Chapter 4

Conclusions and Future Research

The objective of our research project was to develop computer based tools that assist scientists in effectively extracting and reviewing information from collections of text documents from a certain scientific field. To achieve this objective, we have focused on two research questions.

The first research question dealt with how terms that are relevant for a certain scientific field can be extracted from a collection of scientific text documents. We have discussed that extraction of terms can be done manually by human experts. However, extraction of terms by human experts is very expensive and time consuming. Term extraction by human experts also suffers from problems of bias and lack of coverage and consistency. We have therefore examined automatic term extraction, i.e. the computer assisted process of extracting terms. This process is not fully automatic, since it still requires human experts to validate whether the extracted terms are indeed correct and relevant. We have presented a term extraction system that aims to extract domain relevant terms from text documents. We have performed experiments in which this term extraction system was applied to abstracts from journals that are representative for the computational intelligence field. It has turned out that the quality of extracted multi-word candidate terms is higher than that of extracted single-word candidate terms. The extraction of single-word candidate terms needs substantial improvement. It has also turned out that as more abstracts are used as input to the term extraction system, the quality of the extracted candidate terms improves. This, of

course, is not a very surprising result.

The second research question dealt with how the associations between terms from a certain scientific field can be visualized. To answer this question, we have examined the construction of associative concept spaces. An associative concept space is a map that visualizes the associations between concepts in a scientific field. It can be used to obtain an overview a scientific field and to support the discovery of unknown associations between concepts. We have proposed a new algorithm for constructing an associative concept space. This algorithm can be seen as an alternative to multidimensional scaling, which is typically used in the literature on knowledge domain visualization. We have performed experiments in which the proposed algorithm and multidimensional scaling were both used to construct an associative concept space of the computational intelligence field. The results of the experiments suggest that the associations between concepts are better reflected in a concept space constructed using the proposed algorithm than in a concept space that is constructed using multidimensional scaling.

It is interesting to consider the results of our research from the perspective of the text data mining approach to knowledge discovery, which we discussed in Section 1.2. From this perspective, the extraction of domain relevant terms can be considered as a text preprocessing operation (step one of the text data mining approach to knowledge discovery). Note, however, that fully automatic term extraction is not feasible using the methods that are available today, including the method discussed in this thesis. Moreover, we do not expect fully automatic term extraction to become feasible in the near future. We think that, at least for some time, the first step of the text data mining approach to knowledge discovery will remain dependent on some human intervention.

The construction of an associative concept space can be considered as an information extraction and analysis operation (step three of the text data mining approach to knowledge discovery). The visualization of a concept space can be considered as a knowledge presentation operation (step four of the text data mining approach to knowledge discovery). So far, associative concept spaces have only been used as a tool for assisting people in discovering new information or, more precisely, in discovering unknown associations

between concepts. We expect that in the near future the process of discovering unknown associations can be further automated. Some efforts in this direction have already been made (e.g. van der Eijk et al., 2004).

Further steps towards automated knowledge discovery will be made in our future research. One of the first steps will be the extraction of the type of a relation between concepts. A promising method for extracting relation types from text documents is the method proposed by Hearst (1992, 1998). The idea of the method is to find lexico-syntactic patterns which identify a certain relation type. After such patterns have been found, they can be used to search for occurrences of the corresponding relation type in a collection of text documents. Hearst (1992, 1998) applied this idea to search for hyponyms (*is-a* relations). Berland and Charniak (1999) applied the idea to search for myronyms (*part-of* relations). As far as we know, the method of Hearst (1992, 1998) has never been used to extract other relation types than hyponymy and myronymy. In our future research, we plan to use this method to extract other relation types as well. We think about relation types like antonymy, entailment, and cause.

Appendix A

Corpora

A.1 Introduction

In this appendix we describe three different corpora that were used in the experiments in this thesis. Each corpus covers a certain scientific field. The fields that are covered are: computational intelligence, finance, and statistics.

Each corpus is made up of English-written abstracts that were obtained from the Science Citation Index Expanded (SCIE) or the Social Sciences Citation Index (SSCI). The abstracts were taken from relevant journals in the corresponding field. Table A.1 summarizes the size of the different corpora. In the next sections we will describe the different corpora in more detail.

A.2 Computational Intelligence Corpus

The computational intelligence field (e.g. Jang et al., 1997), which can be seen as a part of the larger artificial intelligence field, deals with topics like

Corpus	**Number of abstracts**	**Number of words**
Computational intelligence	3,834	559,089
Finance	1,896	198,266
Statistics	2,675	378,139

Table A.1: Size of the corpora.

Subcorpus	**Number of abstracts**	**Number of words**
Evolutionary computation	231	36,233
Fuzzy systems	568	82,619
Neural networks	3,035	440,237
	3,834	559,089

Table A.2: Characteristics of the subcorpora of the computational intelligence corpus

evolutionary computation, fuzzy systems, and neural networks. Each of these three topics constitutes a subfield within the field of computational intelligence, and each topic also has its own scientific journals. For each of the three subfields we have made a subcorpus with the same name.

The evolutionary computation subcorpus contained abstracts from the following journals:

- Evolutionary Computation (2001–2003)
- IEEE Transactions on Evolutionary Computation (1999–2003)

The fuzzy systems subcorpus contained abstracts from the following journal:

- IEEE Transactions on Fuzzy Systems (1994–2003)

The neural networks subcorpus contained abstracts from the following journals:

- IEEE Transactions on Neural Networks (1991–2003)
- Neural Networks (1991–2003)

For each journal, the years within parentheses indicate the period for which abstracts were available in the SCIE or SSCI. In the second and the third column of Table A.2, the number of abstracts and the number of words in the abstracts is reported for each of the three subcorpora of the computational intelligence corpus. In total the computational intelligence corpus contained about 3,800 abstracts and about 560,000 words. Notice further that the computational intelligence corpus contained much more abstracts on topics

related to neural networks than abstracts on topics related to evolutionary computation or fuzzy systems.

A.3 Finance Corpus

The finance corpus contained abstracts from the following journals:

- Journal of Finance (1994–2003)
- Journal of Financial and Quantitative Analysis (1994–2003)
- Journal of Financial Economics (1994–2003)
- Review of Financial Studies (1994–2003)

In total this corpus contained about $1,900$ abstracts and about $200,000$ words.

A.4 Statistics Corpus

The statistics corpus contained abstracts from the following journals:

- Annals of Statistics (1994–2003)
- Journal of the American Statistical Association (1994–2003)
- Journal of the Royal Statistical Society, Series B (1994–2003)

In total this corpus contained about $2,700$ abstracts and about $380,000$ words.

Appendix B

Thesaurus

B.1 Introduction

In this appendix we present a simple thesaurus of the computational intelligence field. The thesaurus was used in the experiments in this thesis for identifying concepts in abstracts. The way in which we constructed the thesaurus is described in Section B.2. In Section B.3, the way in which the thesaurus is displayed is discussed. The relationships that are used in the thesaurus are explained in Section B.4. Finally, in Section B.5, the thesaurus itself is presented.

B.2 Construction

To construct the thesaurus of the computational intelligence field, we first constructed three separate thesauri, one for each of the three subfields of the computational intelligence field. We then merged these thesauri into a single thesaurus. For the construction of the thesauri of the subfields, we used the term extraction system that is presented in Section 2.4. For each of the three subfields of the computational intelligence field, we generated a list of candidate terms by applying the term extraction system to the corresponding subcorpus of the computational intelligence corpus (see Section A.2). A human domain expert then manually validated each list of candidate terms. For each candidate term the domain expert decided whether the term was relevant to the subfield. When the domain expert considered a candidate term relevant, the expert also identified its synonyms. In this way, three

Thesaurus	**Number of terms**	**Number of concepts**
Evolutionary computation	61	51
Fuzzy systems	108	89
Neural networks	241	184
	410	324

Table B.1: Characteristics of the thesauri of the computational intelligence subfields

thesauri were constructed, one for each subfield of the computational intelligence field. The number of terms and the number of concepts in each thesaurus is reported in the second and third column of Table B.1. By merging the three thesauri, we obtained a single thesaurus of the computational intelligence field. This thesaurus contained 378 terms that referred to 294 concepts. Notice that the number of terms and the number of concepts in the computational intelligence thesaurus was less than the total number of terms and the total number of concepts in the thesauri of the subfields (see Table B.1). The difference was caused by general terms and general concepts that were included in the thesaurus of more than one subfield (e.g. *data*, *input*, and *output*).

B.3 Display

The thesaurus is presented as an alphabetically sorted list of terms. All terms are displayed in lower case letters. If available, of each preferred term its synonyms and its broader and narrower terms are given. Non-preferred terms are also displayed, accompanied with a reference to their preferred term.

B.4 Relationships

In the thesaurus several relationships are used. The explanations of these relationships are given below.

Broader Term (BT) A broader term refers to a term which is conceptu-

ally broader in meaning than the current term.

Example:

adaptive fuzzy controller

BT: fuzzy controller

Narrower Term (NT) A narrower term refers to a term which is conceptually narrower in meaning than the current term.

Example:

activation function

NT: nonlinear activation function

Used for (UF) Indicates a synonymical term (a non-preferred term) of the current term.

Example:

neural network

UF: ann

UF: artificial neural network

UF: network

UF: neural net

UF: neural network model

UF: nn

Use (USE) Indicates the preferred term that should be used in the place of the current term.

Example:

neural net

USE: neural network

B.5 Computational Intelligence Thesaurus

A

activation function

NT: nonlinear activation function

adaptive fuzzy controller

BT: fuzzy controller

adaptive fuzzy system

BT: fuzzy system

adaptive resonance theory

agent

ann

USE: neural network

approximation error

architecture

USE: network architecture

art network

artificial neural network

USE: neural network

associative memory

NT: bidirectional associative memory

associative memory model

B

backpropagation

USE: backpropagation algorithm

backpropagation algorithm

BT: learning algorithm

UF: backpropagation

UF: bp

UF: bp algorithm

UF: error backpropagation

UF: error backpropagation algorithm

basin of attraction

basis function

bidirectional associative memory

BT: associative memory

boltzmann machine

bp

USE: backpropagation algorithm

bp algorithm

USE: backpropagation algorithm

C

cascade correlation

cellular neural network

BT: neural network

cerebellar model articulation controller

chaotic dynamics

chaotic neural network

BT: neural network

chaotic system

BT: system

classification

UF: pattern classification

classification accuracy

classification problem

UF: classification task

UF: pattern classification problem

classification task

USE: classification problem

classifier

NT: nearest neighbor classifier

NT: neural network classifier

classifier system

NT: learning classifier system

closed loop fuzzy control system

closed loop fuzzy system

BT: closed loop system

closed loop system

NT: closed loop fuzzy system

cluster

combinatorial optimization problem

competitive learning

USE: competitive learning algorithm

competitive learning algorithm

BT: learning algorithm

UF: competitive learning

connection matrix

connection weight

USE: weight

connectionist model

continuous time hopfield neural network
BT: hopfield network

continuous time neural network
BT: neural network

control
NT: fuzzy control
NT: sliding mode control

control algorithm

control design

control input
BT: input

control law

control performance

control problem

control rule
BT: rule

control scheme

control system
BT: system

controller
NT: fuzzy controller
NT: neural controller
NT: neural network controller
NT: pid controller
NT: sliding mode controller
NT: supervisory controller

controller design

convergence

cost function
BT: objective function

crossover

crossover operator
BT: operator

curse of dimensionality

D

data
NT: input data
NT: input output data
NT: training data
UF: data set

data point

data set
USE: data

decision boundary

decision tree

defuzzification method

delta rule

design method

discrete time fuzzy system
BT: fuzzy system

discrete time recurrent neural network
BT: recurrent neural network

domain of attraction

dual heuristic programming

dynamic neural network

BT: neural network

dynamic system

USE: dynamical system

dynamical system

NT: nonlinear dynamic system

UF: dynamic system

E

ea

USE: evolutionary algorithm

energy function

equilibrium point

error

NT: generalization error

error backpropagation

USE: backpropagation algorithm

error backpropagation algorithm

USE: backpropagation algorithm

error function

UF: error surface

error rate

error surface

USE: error function

evolution

evolution strategy

BT: evolutionary algorithm

evolutionary algorithm
BT: optimization algorithm
NT: evolution strategy
NT: evolutionary programming
NT: multiobjective evolutionary algorithm
UF: ea

evolutionary computation

evolutionary program

evolutionary programming
BT: evolutionary algorithm

evolutionary search

expectation maximization

expert system
BT: system

external disturbance

F

fault tolerance

feature

feature extraction

feature map

feature space
USE: input space

feature vector
BT: input pattern

feedforward artificial neural network
USE: feedforward neural network

feedforward network
USE: feedforward neural network

feedforward neural network
BT: neural network
UF: feedforward artificial neural network
UF: feedforward network

field programmable gate array

finite state automaton

finite state machine

fisher information matrix

fitness

fitness function

fl
USE: fuzzy logic

flc
USE: fuzzy logic controller

fnn
USE: fuzzy neural network

function approximation

fuzzy art

fuzzy artmap

fuzzy automaton

fuzzy c means
BT: fuzzy clustering

fuzzy classifier
BT: fuzzy system

fuzzy clustering

NT: fuzzy c means

UF: fuzzy clustering algorithm

fuzzy clustering algorithm

USE: fuzzy clustering

fuzzy control

BT: control

UF: fuzzy logic control

fuzzy control system

USE: fuzzy controller

fuzzy controller

BT: controller

NT: adaptive fuzzy controller

UF: fuzzy control system

UF: fuzzy logic control system

UF: fuzzy logic controller

fuzzy if then rule

USE: fuzzy rule

fuzzy inference

USE: inference

fuzzy inference system

USE: fuzzy system

fuzzy integral

fuzzy logic

UF: fl

fuzzy logic control

USE: fuzzy control

fuzzy logic control system

USE: fuzzy controller

fuzzy logic controller
UF: flc

fuzzy logic system
USE: fuzzy system

fuzzy measure

fuzzy membership function
USE: membership function

fuzzy model
USE: fuzzy system

fuzzy modeling

fuzzy neural network
BT: neural network
UF: fnn

fuzzy number

fuzzy observer

fuzzy relation

fuzzy rule
BT: rule
UF: fuzzy if then rule

fuzzy rule base
BT: rule base

fuzzy set

fuzzy system
BT: system
NT: adaptive fuzzy system
NT: discrete time fuzzy system
NT: fuzzy classifier

NT: neuro fuzzy system

NT: takagi sugeno system

UF: fuzzy inference system

UF: fuzzy logic system

UF: fuzzy model

G

ga

USE: genetic algorithm

gas

USE: genetic algorithm

gaussian mixture model

generalization

generalization error

BT: error

UF: generalization performance

generalization performance

USE: generalization error

generation

genetic algorithm

UF: ga

UF: gas

genetic diversity

genetic operator

USE: operator

genetic programming

UF: gp

global asymptotic stability

global exponential stability

global search

gp
USE: genetic programming

gradient descent
UF: gradient descent method
UF: steepest descent method

gradient descent method
USE: gradient descent

H

hardware implementation

hopfield model

hopfield network
BT: neural network
NT: continuous time hopfield neural network
UF: hopfield neural network
UF: hopfield type neural network

hopfield neural network
USE: hopfield network

hopfield type neural network
USE: hopfield network

I

identification
NT: parameter identification
NT: structure identification

independent component analysis

individual

inference
UF: fuzzy inference

inference engine

inference rule
USE: rule

input
NT: control input

input data
BT: data

input layer
BT: layer

input output data
BT: data

input pattern
NT: feature vector
NT: training pattern
UF: input vector
UF: pattern

input signal
BT: signal

input space
UF: feature space

input variable

input vector
USE: input pattern

internal representation

J

–

K

k nearest neighbor

USE: nearest neighbor

kalman filter

knowledge base

NT: rule base

L

layer

NT: input layer

NT: output layer

learning algorithm

NT: backpropagation algorithm

NT: competitive learning algorithm

NT: neural network algorithm

UF: training algorithm

UF: training scheme

learning classifier system

BT: classifier system

least squares

linear discriminant analysis

linear matrix inequality

UF: lmi

linear programming

linear programming problem

linear system

linguistic information

linguistic model

linguistic term

lmi
USE: linear matrix inequality

local minimum
BT: local optimum

local optimum
NT: local minimum

local search

locally excitatory globally inhibitory oscillator network

long term dependency problem

lyapunov function

M

mean squared error

measurement noise

membership function
UF: fuzzy membership function

memetic algorithm

memory capacity
USE: storage capacity

memory pattern

mixture of experts

mlp
USE: multi layer perceptron

model selection

model selection criterion

modeling error

moea
USE: multiobjective evolutionary algorithm

multi layer perceptron
UF: mlp

multilayer feedforward network
BT: neural network

multilayer feedforward neural network
BT: neural network

multilayer neural network
BT: neural network

multilayer perceptron
BT: neural network
UF: multi layer perceptron

multiobjective ea
USE: multiobjective evolutionary algorithm

multiobjective evolutionary algorithm
BT: evolutionary algorithm
UF: moea
UF: multiobjective ea

multiobjective optimization problem
BT: optimization problem

mutation
UF: mutation operation

mutation operation
USE: mutation

mutation operator
BT: operator

mutual information

N

nearest neighbor
UF: k nearest neighbor

nearest neighbor classifier
BT: classifier

neighborhood function

network
USE: neural network

network architecture
UF: architecture
UF: neural architecture
UF: neural network architecture

network dynamics

network model
USE: neural network model

network output

network structure

network topology
UF: topology

network weight
USE: weight

neural architecture

USE: network architecture

neural classifier

USE: neural network classifier

neural computation

neural controller

BT: controller

neural model

USE: neural network model

neural net

USE: neural network

neural network

NT: cellular neural network
NT: chaotic neural network
NT: continuous time neural network
NT: dynamic neural network
NT: feedforward neural network
NT: fuzzy neural network
NT: hopfield network
NT: multilayer feedforward network
NT: multilayer feedforward neural network
NT: multilayer neural network
NT: multilayer perceptron
NT: nonlinear neural network
NT: probabilistic neural network
NT: radial basis function network
NT: recurrent neural network
UF: ann
UF: artificial neural network
UF: network
UF: neural net

UF: neural network model

UF: nn

neural network algorithm

BT: learning algorithm

neural network architecture

USE: network architecture

neural network classifier

BT: classifier

UF: neural classifier

neural network controller

BT: controller

UF: nn controller

neural network model

UF: network model

UF: neural model

neural system

neuro fuzzy system

BT: fuzzy system

neuron

USE: unit

neuron model

nn

USE: neural network

nn controller

USE: neural network controller

node

USE: unit

noise

nonlinear activation function

BT: activation function

nonlinear dynamic system

BT: dynamical system

nonlinear neural network

BT: neural network

nonlinear system

BT: system

O

objective function

NT: cost function

offspring

one point crossover

operator

NT: crossover operator

NT: mutation operator

UF: genetic operator

UF: search operator

optimal solution

BT: solution

optimization

optimization algorithm

NT: evolutionary algorithm

optimization problem

NT: multiobjective optimization problem

orthogonal least squares

output

output error

output layer
BT: layer

output neuron
USE: output unit

output node
USE: output unit

output space

output unit
BT: unit
UF: output neuron
UF: output node

P

parameter

parameter identification
BT: identification

pareto front

pattern
USE: input pattern

pattern classification
USE: classification

pattern classification problem
USE: classification problem

pattern recognition

pattern recognition problem

pca
USE: principal component analysis

pid controller
BT: controller

population

population size

prediction

principal component

principal component analysis
UF: pca

probabilistic neural network
BT: neural network

prototype

Q

quadratic programming problem

R

radial basis function
UF: rbf

radial basis function network
BT: neural network
UF: radial basis function neural network
UF: rbf network
UF: rbf neural network

radial basis function neural network
USE: radial basis function network

rbf
USE: radial basis function

rbf network

USE: radial basis function network

rbf neural network

USE: radial basis function network

receptive field

recognition rate

recombination

recombination operator

recurrent network

USE: recurrent neural network

recurrent neural network

BT: neural network

NT: discrete time recurrent neural network

UF: recurrent network

recursive least squares

rule

NT: control rule

NT: fuzzy rule

UF: inference rule

rule base

BT: knowledge base NT: fuzzy rule base

S

search operator

USE: operator

search space

UF: solution space

selection

self adaptation

self organization

self organizing map
UF: som

short term memory

sigma pi unit
BT: unit

sigmoid function

signal
NT: input signal

signal processing

single layer perceptron

singular value decomposition

sliding mode control
BT: control

sliding mode controller
BT: controller

solution
NT: optimal solution

solution space
USE: search space

som
USE: self organizing map

stability

stability analysis

stability condition

stable equilibrium point

stable state

steepest descent method
USE: gradient descent

storage capacity
UF: memory capacity

structure identification
BT: identification

supervisory controller
BT: controller

support vector machine
UF: svm

svm
USE: support vector machine

synapse

synaptic weight

system
NT: chaotic system
NT: control system
NT: expert system
NT: fuzzy system
NT: nonlinear system

T

t norm

t s fuzzy model
USE: takagi sugeno system

takagi sugeno fuzzy model
USE: takagi sugeno system

takagi sugeno system
BT: fuzzy system
UF: t s fuzzy model
UF: takagi sugeno fuzzy model

target function

time delay

time series

time series prediction

topology
USE: network topology

tournament selection

training
UF: training process

training algorithm
USE: learning algorithm

training data
BT: data
UF: training set

training error

training example

training pattern
BT: input pattern
UF: training vector

training process
USE: training

training sample

training scheme

USE: learning algorithm

training set

USE: training data

training vector

USE: training pattern

transfer function

two point crossover

U

uncertainty

unit

NT: output unit

NT: sigma pi unit

UF: neuron

UF: node

universal approximator

unsupervised competitive learning rule

V

vapnik chervonenki

vector quantization

very large scale integration

W

weight

UF: connection weight

UF: network weight

weight matrix

weight space

X

–

Y

–

Z

–

Appendix C

Definitions of Used Terms

This appendix gives definitions of all the terms relevant to this thesis.

Annotation Annotation is the process of adding (linguistic) information to text. This could be, e.g., part-of-speech tags.

Associative concept space An associative concept space is a map that visualizes the associations between concepts in a scientific field.

Broader term A broader term is a term which is conceptually broader in meaning than the original term. E.g. *fuzzy system* is a broader term than *Takagi-Sugeno system* because a Takagi-Sugeno system is a specific type of fuzzy system.

Candidate term A candidate term is a term produced by term extraction that has not been manually validated.

Cluster analysis Cluster analysis is the process of partitioning items into meaningful groups or clusters so that items within a cluster have similar characteristics and are dissimilar to items in other clusters.

Co-occurrence matrix A co-occurrence matrix is a square and (usually) symmetric matrix in which the entries are co-occurrence frequencies between items. The most common types of items are scientific journals, scientific articles, authors, and descriptive words, terms, or concepts.

Co-word analysis Co-word analysis is a technique for knowledge domain visualization that is concerned with the construction of maps of keywords on the basis of co-occurrences in a corpus of text documents.

Collocation A collocation is a recurrent combination of words that co-occur more frequently in natural language than it would be expected just by chance.

Concept A concept is a unit of thought. It may be possible to express a concept using a number of different terms. These terms are called synonyms of each other. Of all the terms that refer to a concept, one is chosen to label the concept. This term is called the preferred term. The other terms that refer to the concept are called non-preferred terms.

Concept association matrix A concept association matrix is a square and symmetric matrix in which the entries represent the strengths of the associations between concepts.

Corpus See *document collection*.

Document A document is an information item. This could be, e.g., a paragraph, a section, a chapter, a book, an article, a Web page, an email message, a newsgroup posting, etc.

Document collection A document collection is a large set of documents.

Knowledge discovery Knowledge discovery is the process of finding novel, interesting, and useful patterns in data.

Knowledge domain visualization Knowledge domain visualization is a visual exploratory approach to study a domain of knowledge and its development.

Lemma A lemma is the base or uninflected form of a word. E.g. the lemma of the word *networks* is *network*.

Lemmatization Lemmatization is the process or result of dividing text into lemmas.

Multi-word term A multi-word term is a term consisting of more than one word. E.g. *neural network* or *fuzzy system*.

Multidimensional scaling Multidimensional scaling is the process of representing dissimilarities between items as distances in a low dimensional, e.g. two-dimensional, Euclidian space. It is somewhat similar to cluster analysis but returns points in space rather than distinct groupings.

Narrower term A narrower term is a term which is conceptually narrower in meaning than the original term. E.g. *Hopfield network* is a narrower term than *neural network* because a Hopfield network is a specific type of neural network.

Natural language Natural language is the term used for human language, as opposed to artificial language, which is the term used for, e.g., computer programming language and formal logic.

Non-preferred term A non-preferred term is a term that has the same meaning as a preferred term but is not used for indexing.

Part-of-speech tag A part-of-speech tag of a word is the lexical syntactical category associated with that word. E.g. the part-of-speech tag of the word *algorithm* is noun.

Part-of-speech tagging Part-of-speech tagging is the process of assigning part-of-speech tags to words in a text.

Precision In a term extraction context, the precision is the proportion of extracted terms that are relevant for the subject domain.

Preferred term A preferred term is a term used consistently for indexing to represent a given concept.

Recall In a term extraction context, the recall is the proportion of relevant terms that are extracted.

Similarity matrix A similarity matrix is a square and (usually) symmetric matrix in which the entries are similarities between items.

Single-word term A single-word term is a term consisting of one word. E.g. *network* or *controller*.

Synonym A synonym is a term whose meanings is considered to be the same as the meaning of another term. E.g. *neural net* and *neural network*. Abbreviations and their full forms may be treated as synonyms. E.g. *NN* and *neural network*.

Tag A tag is a label associated with a word providing information about the word.

Tagging Tagging is the process of assigning tags to text. See *annotation*.

Term A term is a word or a phrase that denotes a concept in a specific subject field. E.g. *neural network* or *fuzzy system* are terms that denote concepts in the computational intelligence domain. The structure of a term is typically a single noun or a noun phrase.

Term extraction Term extraction is the computer-assisted process of extracting a list of candidate terms from text documents. The resulting list of candidates must be verified by a human terminologist or translator.

Note: Often this process is also called *term recognition*. However, term extraction and term recognition are two different tasks.

Term recognition Term recognition is the process of automatically looking up in a term base all terms that occur in a text document.

Note: To be distinguished from *term extraction*.

Text data mining Text data mining is the process of discovering heretofore-unknown information from collections of text documents.

Text mining See *text data mining*.

Thesaurus A thesaurus is a list of important terms used in a given domain of knowledge with for each term in the list a set of hierarchically and synonymically related terms.

Token A token is an individual word.

Tokenization The process or result of dividing a text into tokens.

Bibliography

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, Harlow, UK, 1999.

M. Berland and E. Charniak. Finding parts in very large corpora. In *Proceedings of the the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pages 57–64, College Park, Maryland, USA, June 1999.

K. Börner, C. Chen, and K. W. Boyack. Visualizing knowledge domains. *Annual Review of Information Science and Technology*, 37:179–255, 2003.

E. Brill. Some advances in transformation-based part of speech tagging. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 722–727, Seattle, Washington, USA, August 1994.

S. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, San Francisco, California, USA, 1999.

K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

F. J. Damerau. Generating and evaluating domain-oriented multi-word terms from texts. *Information Processing and Management*, 29(4):433–447, 1993.

Y. Ding, G. G. Chowdhury, and S. Foo. Bibliometric cartography of information retrieval research by using co-word analysis. *Information Processing and Management*, 37:817–842, 2001.

T. E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/The MIT Press, Cambridge, Massachusetts, USA, 1996.

M. Hearst. Untangling text data mining. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics (ACL'99)*, 1999.

M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539–545, Nantes, France, July 1992.

M. A. Hearst. Automated discovery of WordNet relations. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 131–151. MIT Press, Cambridge, Massachusetts, USA, 1998.

P. Jackson and I. Moulinier. *Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization*. John Benjamins Publishing Company, Amsterdam, The Netherlands, 2002.

C. Jacquemin. *Spotting and Discovering Terms through Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, USA, 2001.

J.-S R. Jang, C.-T. Sun, and Mizutani E. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall, Upper Saddle River, New Jersey, USA, 1997.

J. S. Justeson and S. M. Katz. Technical terminology: Some linguistic properties and an algorithm for identification of terms in text. *Natural Language Engineering*, 1(1):9–27, 1995.

A. Kopcsa and E. Schiebel. Science and technology mapping: A new iteration model for representing multidimensional relationships. *Journal of the American Society for Information Science*, 49:7–17, 1998.

H. Liu. MontyLingua: An end-to-end natural language processor with common sense, 2004. Available at: `http://web.media.mit.edu/~hugo/montylingua`.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, USA, 1999.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, London, UK, 1979.

K. W. McCain. Mapping authors in intellectual space: A technical overview. *Journal of the American Society for Information Science*, 41:433–443, 1990.

J. Meij and A. van den Bosch. Text mining techniques. In J. Meij, editor, *Dealing With the Data Flood: Mining Data, Text and Multimedia*, pages 746–753. STT/Bewton, Den Hague, The Netherlands, 2002.

P. Newbold. *Statistics for Business and Economics*. Prentice-Hall, Englewood Cliffs, New Jersey, USA, 4th edition, 1995.

H. P. F. Peters and A. F. J. van Raan. Co-word based science maps of chemical engineering. Part I: Representations by direct multidimensional scaling. *Research Policy*, 22:23–45, 1993.

D. L. Sackett, R. B. Haynes, G. H. Guyatt, and P. Tugwell. *Clinical Epidemiology: A Basic Science for Clinical Medicine*. Little, Brown and Company, Boston, Massachusetts, USA, second edition, 1991.

J. C. Sager. *A Practical Course in Terminology Processing*. John Benjamins, Amsterdam, The Netherlands, 1990.

G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, Massachusetts, USA, 1989.

M. Schuemie. Associatieve conceptuele ruimte, een vorm van kennisrepresentatie ten behoeve van informatie-zoeksystemen. Master's thesis, Erasmus University Rotterdam, 1998. In Dutch.

J. van den Berg and M. Schuemie. Information retrieval systems using an associative conceptual space. In M. Verleysen, editor, *Proceedings of the 7th European Symposium on Artificial Neural Networks (ESANN-99)*, pages 351–356, Bruges, Belgium, April 1999.

J. van den Berg, N. J. van Eck, L. Waltman, and U. Kaymak. A VICORE architecture for intelligent knowledge management. In D. Malerba and M. May, editors, *Proceedings of the KDNet Symposium on Knowledge-Based Services for the Public Sector (MOD-04)*, pages 63–74, Bonn, Germany, June 2004.

C. C. van der Eijk. Knowledge discovery in scientific literature. Master's thesis, Erasmus University Rotterdam, 2001.

C. C. van der Eijk, E. M. van Mulligen, J. A. Kors, B. Mons, and J. van den Berg. Constructing an associative concept space for literature-based discovery. *Journal of the American Society for Information Science and Technology*, 55:436–444, 2004.

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, UK, second edition, 1979.

J. Wyatt. Use and sources of medical knowledge. *The Lancet*, 338:1368–1373, 1991.