Automatic Tools for Multichannel Publishing

Asta Bäck [1], Helene Juhola[1], Ville Ollikainen [1] & Jarno Tenni [1]

Keywords: Automation, publishing processes, XML, tagging, SOM, abstracting

Abstract: The publishing sector has changed essentially during the last five years. Earlier, the publishing process was in many cases unique, i.e. a certain content was prepared for a certain publication. In this kind of a process there was not so much room or need for tools that would automate the content manipulation and compilation processes.

This has changed. To serve the different channels and to publish personalized content, much more work needs to be done with the same content. Content preparation and compilation should be automated, where possible.

This paper discusses two implementations where automation and computer support are created for two content processing tasks. The first case concerns the XML tagging of a news article. The software searches for proper names and acronyms in the text and shows them to the user for classification. The classified words are tagged according to the classification. These tags create a new level of information within the document, and they can be used to improve the searces of the documents or to modify their rendition.

The second case shows an application of Kohonen's one-dimensional self-organising map to sort articles in a good sequential order. A logical sequential

[1] VTT Information Technology, P.O.Box 1204, FIN-02044 VTT, Finland. email: firstname.lastname@vtt.fi; fax +358 9 455 2839.

order is the best solution with devices like eBooks and MP3 players. Some test results are included to demonstrate the performance level of the application.

This paper also discusses automatic abstracting. A respective application is currently under development at VTT Information Technology. Automatic abstracting of text documents is very useful in various media applications. Abstracts may be used as part of the metadata, or they can be delivered to the readers as such, for example, to mobile terminals with restricted transfer and display capabilities.

It is difficult to fully automate content-related tasks, but the implementations give encouraging results. With automation and computer assistance, content manipulation tasks can be speeded up and the prerequisites can be created for increased publishing through multiple channels and in a personalized way.

Introduction

The publishing sector has changed essentially during the last five to ten years. Earlier, the publishing process was in many cases unique, i.e. a certain content was prepared for a certain publication and the same content was used in very few, if any, other contexts. Now that electronic publishing has become more common in an increasing number of publishing processes, the same content should be publishable through multiple channels. The publisher, however, can seldom afford to make separate contents for every channel. Therefore, channel-neutral formats, such as XML should be employed and all those content manipulation tasks which are suited for automation should be automated. Computer assistance should be used where full automation cannot be achieved. Automation is especially important if the users are offered personalized versions or editions, because these services are not feasible in any other way.

Automation and computer assistance can be used with many different tasks in a publishing process. This paper reports on three cases in which tasks relating to content preparation, content compilation and content creation are either automated or computer-aided.

Automatic/ computer-aided XML-tagging

Need for XML tagging

XML plays an important role in multiple channel publishing, because it is based on the concept of separating the content and the presentation of the document. When the structure and the content of a document are described by tagging the document content according to a DTD (Document Type Definition), the tags can be used to select or to pinpoint elements with a certain content, or to create different renditions of the document by means of stylesheets. The stylesheet defines the rendition based on the elements, their attributes, and element hierarchy. [Bäck, 1999]

The inconvenience of XML tagging is the extra effort it requires, even when a specialized structure editor is used. To reduce the requirements for the authors, texts may be converted from various formats into XML by using styles or other presentation-related information to indicate which tags should be used in the XML version of the document. These conversions are necessary, also when older texts and documents have to be reused. This solution, however, only creates the basic structure of the document, and does not give any indication of the document content.

Computer-aided tagging of proper names

VTT Information Technology has introduced XML into the publishing process of its monthly newsletter, GT-Bulletin. In the articles of this newsletter, proper nouns and acronyms provide valuable information of the content and the topic of the article. It would, however, be very time-consuming to tag these terms manually. Therefore we have developed a system which helps with the tagging task.

The articles are written in Finnish, and perhaps the most distinctive feature of our language it that there are numerous inflected forms of the words. In our application, we chose to create a separate metadata document of each article, and the proper names and acronyms are an essential part of the metadata document. In the metadata document, the words are included in their basic form to make the search easier. The proper names are also tagged in the article, but the words are tagged in the respective case.

An application for the semiautomatic collection of metadata

Our tagging application is an Internet application where an HTML browser is used as the user interface. The documents to be processed may originally be on a

local computer or on a remote computer as long as the user has the necessary access rights. Actual document processing takes place in a server unit where the selected file is copied.

In the document the application looks up proper names and acronyms for classification. Since the application must master the inflection of the Finnish language, a special software TextMorfo by the Finnish company Kielikone Oy, was included as one of the system components. It can parse Finnish sentences and find the basic forms of the words. In our application, we run the software so that it brings up for classification all proper nouns, abbreviations and the words it classifies as codes. The codes are either acronyms, misspelt Finnish words, jargon (e.g. English words recently adopted into the Finnish language), or foreign words.

When the application finds a word that belongs to one of the selected groups, it shows it to the user for classification. If the user chooses to classify the word, it is included in the metadata document tagged with the appropriate tag. The word is also tagged accordingly in the article. We use the following categories to classify the words:

- personal name
- location
- company name
- acronym
- product name
- event name
- project name
- url

When the user classifies a word, the classification is stored in a database. This database is used to make a suggestion for classification, or, if a special mode is selected, to automatically classify the words based on the previously made classifications, if such a classification can be found for the word. The user is free to accept, change or discard the classification suggestions made by the system. Since the articles in our application only deal with printing and publishing relating topics, we assume that a large number of the classifications can fairly soon be made automatically.

Our application goes through the articles one by one, and after every article has been processed, a list of the words selected into the metadata document is shown to the user for final approval. At this point, it is still possible to change the classification, add new words into the metadata document or remove words therefrom.

Table 1 shows some figures which indicate the performance of the application. Five random articles of the GT-bulletin were taken as examples. In these articles, a classifiable word was used, on an average, almost twice which means that the classifier's work load is reduced considerably, when every word is proposed for classification only once per article. If, however, a term occurs in several senses within an article (e.g. as a company name and as a location name), the second instance of the term will end up with a wrong classification. Now our database only stores the latest classification of each word, but if it stored all the classifications of one word, we would gradually learn which words have several meanings and we could always require manual classification for them.

The percentage of rejected words was fairly high in the five documents, varying from 12 % to 27 %. By rejected words we mean words which were proposed by the system but rejected by the user when the program was run in the mode that proposes each term only once per article. The majority of the rejected words were words in a foreign language or recently adopted words which the software did not recognize. If we adjusted the system so that it only shows the users those words which it recognizes as proper nouns, we would raise the acceptance level of the proposed words. This means, however, that relevant words are missed. As the table shows, with this configuration the system misses only very few relevant words. Another point is that by showing the user all the codes the system reveals misspelt words and the user can correct them by using the same user interface. For each implementation, the users should try to find the right balance between the number of misses and the cost of manually rejecting incorrect proposals.

Table 1. The performance of the XML tag enhancer application measured as the number of incorrect and missed words.

# of words in the article	# of sugges-tions when every proper noun is shown only once/ article	# of times these proper nouns are used in the document	# of rejected words	Percentage rejected words	# of missed proper nouns
1 025	33	1.9	4	12 %	0
758	26	1.7	6	23 %	0
1 021	88	1.7	24	27 %	1
1 720	97	1.8	26	27 %	1
510	55	1.9	15	27 %	1 (the same word as in the case 3)

The handling of names that consist of several words is another aspect of the system that should be improved. Now the user must combine the words by hand, which, of course, should be an exception and not a rule.

The system collects information of the classifications gradually as articles are processed, which is slow and laborious. This step could be accelerated by importing names and locations with their proper classifications.

Compiling sequential presentations from a news database

Where are sequential presentations needed?

In recent years, a number of new information carriers, like electronic books and MP3 players, have come onto the market. Other technologies that have appeared in recent years include re-usable electronic paper and electrically erasable ink. These have the common feature that browsing forwards and backwards through the material is easy, but movement, for instance, using Internet-style hyperlinks is at least difficult, if not impossible. These platforms can be regarded as supporting a *sequential* form of presentation.

Simultaneously, an increasing amount of material is being produced electronically, as individual pieces of information in a database. Usually, publications are compiled from these pieces, by the person compiling a publication setting the pieces in the order that suits her or him best. This procedure – manual work – can certainly also be used to produce a sequential form of presentation.

However, as publications become increasing personalized, the profitability of manual operation becomes questionable. In personalized publication, material is produced for end users, according to their personal interest profiles. The smaller the target group of a compilation, the greater the cost of manual ordering.

End users can also collect material directly from several different sources – this may take place because of copyright considerations – with compilation taking place on the end user's hardware, so that it is not sensible to commission someone else to carry out the sequential ordering. Figure 1 shows this situation.

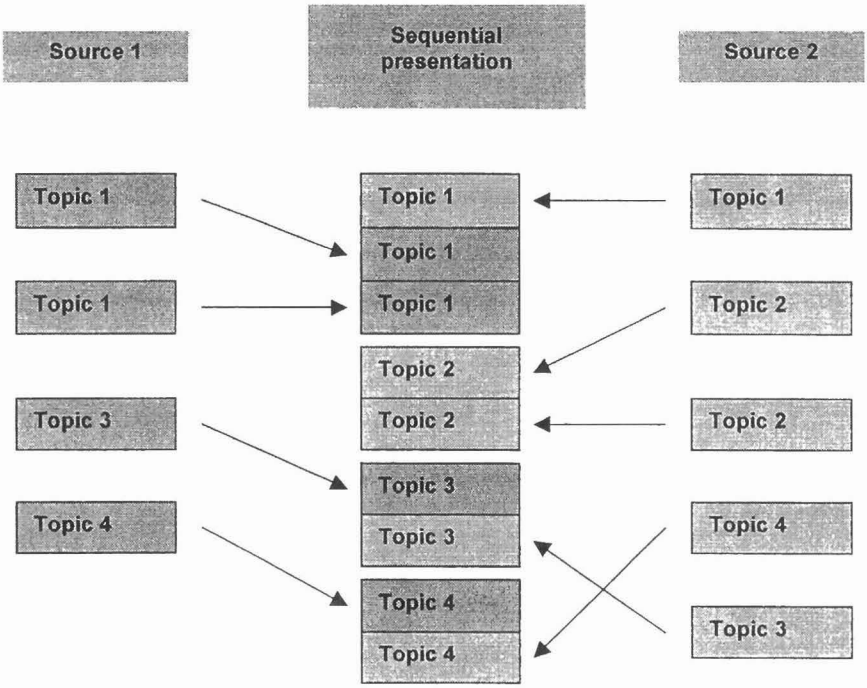

Figure 1. A sequential presentation compiled from different sources.

In 1999, VTT studied the automatic creation of a smooth sequential order in a group of separate news articles. The material used was the news database of more than 20 000 news articles and TV news topics created in VTT's IMU project. The material was produced mainly for an electronic book and a speech synthesizer, with the aim of building a narrative proceeding smoothly from one topic to the next. The material was ordered using neural computing, specifically Kohonen's self-organizing map (SOM).

The self-organizing map, SOM

Kohonen's self-organizing map, (SOM), is one widely known method used in neural computing.

Neural computing uses a group of simple linked computing elements. These computing elements are called neurons, and, nearly without exception, the

neurons responsible for calculations have several inputs (input branches) and one output (output branch). There is a great variety of topologies and teaching methods [Haykin, 1999].

Kohonen's self-organizing map is a neural network simulation comprising mutually competing neurons, in which the same information is fed to the input of each neuron (Fig. 2). Each input of each neuron has specific weight coefficients $\mathbf{w}_i$, which form neuron-specific weight vectors $\mathbf{w}$. The output of a neuron is the square of the Euclidean distance between the input and the weight vector $||\mathbf{x}-\mathbf{w}||^2 = \Sigma(\mathbf{x}_i\mathbf{w}_i)^2$, which depicts how much a neuron's weight vector differs from the input.

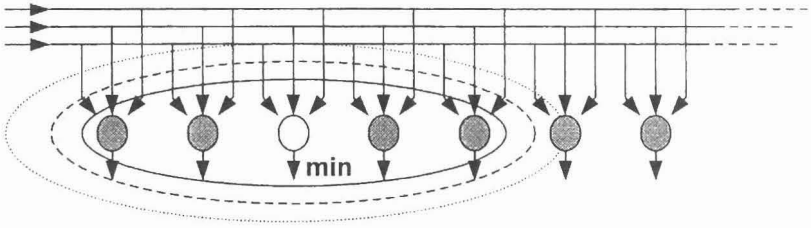

Figure 2. A 1-dimensional self-organizing map, SOM.

In the study, the neurons' weight coefficients w_i were mainly initialized using random numbers, the input used being feature vectors, which depicted the occurrence of various nouns in the articles, and which were formed from the articles.

The map organizes in such a way that the neurons with input corresponding to each article have outputs that are unavoidably of different magnitudes. The neuron with the smallest output value is termed the winner neuron. The weight coefficients of the winner neuron and those in its neighborhood are adjusted towards the input, making them increasingly sensitive to articles similar to the input in question. The entire group of articles is processed numerous times, with an increasingly narrow group of neighbors being sensitized on each new teaching round. Finally, only winner neurons are sensitized.

As a result of the organization of the map, the weight coefficients of neurons close to each other finally resemble each other, which leads up to a situation, where the winner neurons triggered by articles resembling each other are located close to each other as well.

The self-organizing map was implemented using the SOMPAK software developed by the Helsinki University of Technology, which is freely available for research purposes.

The sequential ordering process

The ordering method can be divided into four parts: the extraction, from the material, of the key data required for ordering, the preprocessing of the key data, the sequential ordering of the material on the basis of the preliminarily processed data, and the compilation of a presentation from the sequential material (Fig. 3).

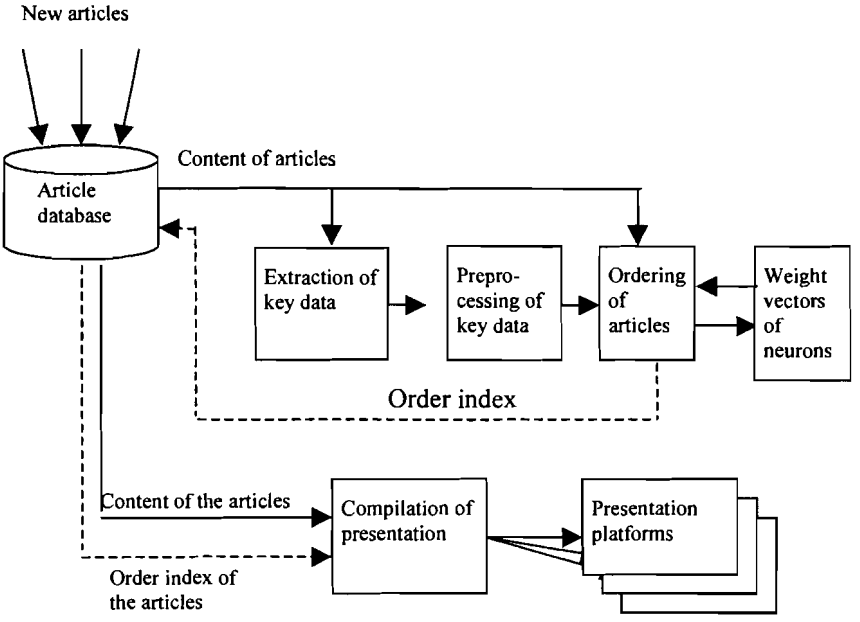

Figure 3. Sequential ordering process.

Ordering the articles was performed as a batch run, which started a few minutes after new articles had arrived in the IMU database. Typically, the one hundred most recent articles were ordered, irrespective of how many articles had arrived in the database. Generally, about twenty articles arrived at a time.

To collect the key data, key words, from which the feature vectors of the articles were formed, were extracted from contents of the articles. Following the presentable results obtained from the IMU project [Glödstaf, 1999], nouns appearing in the articles were used as the key words. The nouns were extracted from the sentences by using a morphological analysis program (Kielikone Oy's Morfo) for Finnish language vocables.

From the point of view of the operability of neural computing, it is particularly important that the information to be analyzed has been appropriately preprocessed. The objective is to extract the most essential data from all the available information. Unnecessary information increases noise and weakens the final result. Insufficient information, on the other hand, does not exploit the full content of the material. When preprocessing the key data, all the key words of the articles to be organized are mutually compared and all words appearing in only a single article are filtered out. This is done, as there is no benefit, in terms of the mutual comparison of articles, from a word that appears only in a single article. On the other hand, excessive elimination is not required, as one special feature of Kohonen's self-organizing map is that there should preferably be too much, rather than too little input data. A meaningful order can only arise among articles than have a sufficient number of key words in common.

Table 2 shows an example of the feature vectors of nine articles. If the key word appears in the article, the feature vector shows the value 1.0 it, otherwise its value is 0.0. In practice, one hundred articles retrieved from the IMU database will provide 1000 – 1200 key words.

Table 2. Example of feature vectors.

Key words	*art1*	*art2*	*art3*	*art4*	*art5*	*art6*	*art7*	*art8*	*art9*
President	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
Kosovo	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
Häkkinen	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0
Time	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0
Independence day	1.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0
Celebrations	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0
Yeltsin	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0

Feature vector of article 4

The articles are ordered by neural computing, using a Kohonen's self-organizing map (SOM) comprising one hundred neurons. The significant order index for each article in the database is the number of the winner neuron, when the feature vector of the relevant article is entered in the map.

The presentation is compiled from the database using a separate program, which collects the articles from the IMU database in the sequence according to the order index and composes the material in a form suitable for each presentation platform.

In the project, compilation alternatives for the material were made for three different presentation platforms:

- Web browser (html)
- Rocket Book electronic book
- Windows Media Player (wav)

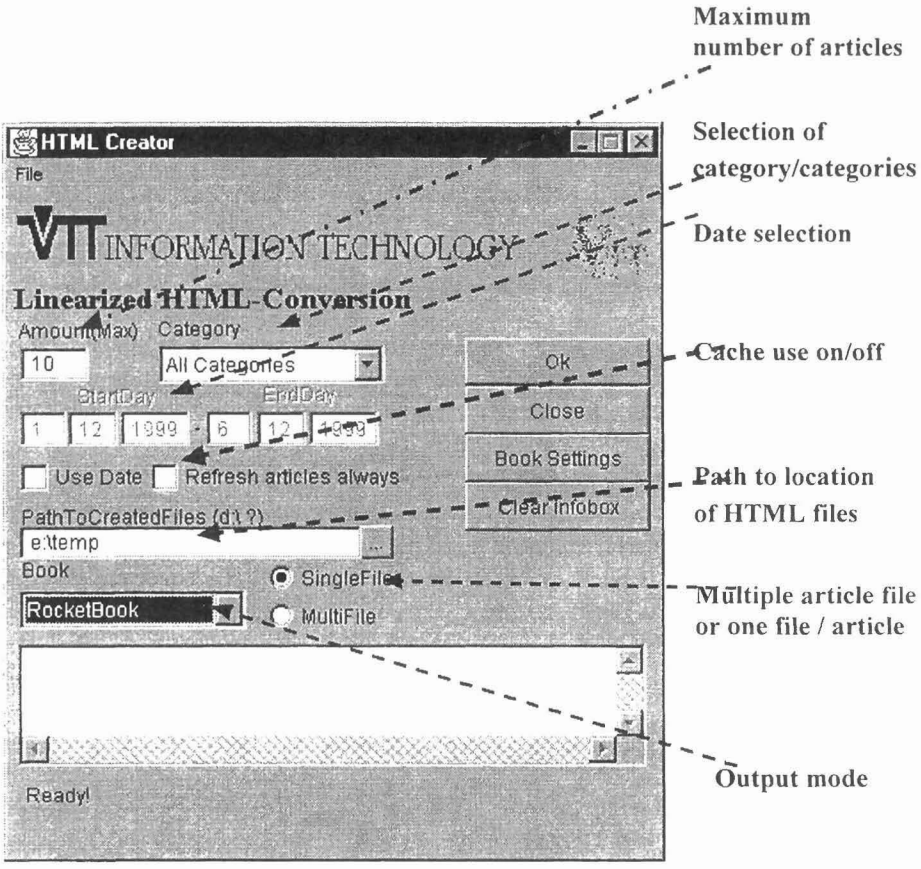

Figure 4. User interface of the compilation program.

Wav files were created for a speech synthesizer (Timehouse Oy's Mikropuhe) and can be re-coded for an MP3 player or written onto an audio CD. The presentation platform is selected from a pull-down menu in the compilation program, which can also be used to set the number of the latest articles, and from what period of time, to take into the compilation. Figure 4 shows the compilation program user interface.

Experiences in practice

Although it seems to work fairly well, article ordering using an SOM is not perfect. A SOM will put certain articles in quite obviously the wrong place, often in the middle of an otherwise apparently coherent totality. Such frogs that have hopped into the middle of totalities are estimated as accounting for about 10 – 20% of all articles.

There is generally no absolutely preferable order for the articles. A test was made to determine the functionality of the order created.

Twenty-one testees participated in the test. Each testee was given eleven newspaper articles retrieved from the IMU database, printed on paper and mixed, and which were to be placed in order. All extraneous information, such as dates, times, and database indices was removed from the printed articles. Instructions were given to create an order that would allow reading to progress as naturally as possible from the first to the last article. The SOM was treated as if it were the 22nd testee.

In the test, the testees' article orders were compared to each other in pairs, their similarity being measured by three characteristics. Three characteristic-specific symmetrical tables, with a row and column corresponding to each testee, were formed from the pairs of testees. The intersection point of the row and the column indicated how closely the orders produced by the testees in question resembled each other, when measured by the relevant characteristics. A testee-specific value was calculated by rows (or by columns), showing how closely the testee's order, according to the relevant characteristic, resembled the orders produced by other testees.

Table 3 summarizes the results, by characteristic. Besides the mean and deviation measured, the mean and deviation for an entirely random order are calculated. The first characteristic mainly emphasizes the macro-level order and the third the micro-level sequence. Details of the method of calculating the characteristics will be published later this year at neural networking conferences.

Table 3. Summary of test results.

	Preference	Mean	Deviation [s]	Mean of random results	Deviation of random results [σ]	SOM
Characteristic A	Smaller is better	2.08	0.14	2.93	2.22	2.03
Characteristic B	Smaller is better	0.56	0.04	1.532	0.765	0.53
Characteristic C	Greater is better	3.56	0.9	0.263	0.541	3.38

Measured by the first two characteristics, the SOM produced a better result than an average testee and, measured by the third, a nearly average result, so that the system can be regarded as functional. The reason why the SOM did not attain the testee average at the micro-level lies in noise – in random words that begin to have an effect, when differences between the articles decrease. Major themes, however, can be distinguished among the noise.

Initially, it is intended to apply the system and its components in VTT Information Technology's projects, while seeking for external uses.

Automatic abstracting

Metadata gives one view of the document. Another way to describe an article is to condense its content into an abstract. The manual creation of an abstract is a time consuming task, because the article must first be read and analysed and only after that can the abstract be written. Abstracts are also very subjective, because people have different views of what is important in a document, and they also have different styles to formulate that information.

Abstracting, even though it sounds implicitly clear, is not too clear in itself. In fact, abstraction is only a subtype of a summary. There are two types of summaries: extraction and abstraction. Ideally, the computer should work in the same way as humans do, i. e. read the text, retrieve information and then produce a new shorter text that contains the information in a compact form, like in the following definition

"*Abstracting = a reductive transformation of source text to summary text through content reduction by selection and/or generalization on what is important in the source.*"

by [Karen & al., 1999] emphazing the need to understand the text to be abstracted.

Abstracting is a difficult task also for the computer, because it requires an understanding of the context, which again requires an understanding of the world. Surely that is too difficult for a computer. But at some level abstracting can be done automatically, especially if the abstraction is tailored and restricted to a certain document type or domain.

So far, the best practical results have typically been achieved in two cases: text extraction and fact extraction. Text extraction identifies sentences which provide important information of the document, and then concatenates these sentences to create a new document. This system can be used with any document type. Fact extraction means retrieval of facts, i.e. it fills a template (e.g. an action description), and tries to find the missing content from the document. Finally a a new document is generated based on the template. These tasks are easier for the computer than actual abstracting, because in the first case the computer does not have to understand the context, and in the second case the computer does not have to generate new sentences because it use predefined sentence templates.

There are already industrial solutions on the market, but the full potential of automatic abstracting has not been reached yet. The existing systems mostly apply the text extraction method to summarize the text for people to read, but in the future we shall probably have applications which combine text (or fact) abstraction for further processing by a computer (text mining).

Probably the best-known systems in the market are AutoSummarize [Gore, 1997] in the Microsoft Word and AltaVista Discovery which summarizes web pages [Anon. 2000].

Abstracting methods

Abstracting can be based on various methods. There are systems which use statistical information, corpus, knowledge or templates, or a combination of these. Statistical and corpus-based systems count word/term statistics to find the relevant items whereas knowledge and template-based systems focus on some topic(s) and use a priori knowledge of the kind of information that is being looked for.

In our current METABS project, we shall develope a method for abstracting. We begin with text extraction but, based on our experiences and needs, the system may be developed towards text summarization system.

The text extraction system will be a combination of methods. This means that, to evaluate which sentences are most the important, the system will use formatting information, XML tags as well as statistical and lexical information.

Formatting information can be used, because the information in the document is typically repeated at several levels. Headings often give the most general and compact version of the important information in a document, and they can be used as they are. They may serve as a clue, showing what we should look for in the paragraphs. Formatting information also gives different weights to the paragraphs, i.e. the most important information may be found at the beginning and at the end of the paragraphs. Also the depth of the paragraph matters because, as we go deeper into a document, the information gets more and more detailed. To produce an abstract more weight should be given to the information at the higher levels.

The information in XML tags may also be used for guidance in text extraction. They help us to find concepts which may be used in a statistical lexicon.

Statistical and lexical analyses are the basic means of determining the importance of a sentence for abstracting. The system first calculates the statistical importance of the existing words in an examplary document base. This helps us to find the words which occur too often and make no difference, as well as the words which exist often enough amd make a real difference.

Some implicit lexical clue words are also be used in abstracting. Phrases like "In conclusion" or "to summarize" used by the authors to summarize or conclude their own text, may be very helpful.

To process a document which should be abstracted, each sentence is given a score based on the three things, i.e. formatting information, XML tags and the words/terms which occur therein. The extract is generated based on these values. Later onwe shall explore the possibility of combining sentences to produce real abstracts.

Conclusions and discussion

Multiple media publishing and the increasing demand for tailoring and personalization of publications set new requirements for publishing processes as a whole. Publisher cannot afford to have as much manual work put to different publications as before. Instead, automatic methods and systems should be applied to retrieval, selection, preparation, compilation and presentation of content to different information carriers and media platforms. This paper presents the latest activities of VTT Information Technology in the field of content process automation.

Storing content in a structured and channel-independent format is one of the prerequisites to multiple channel publishing. Another required feature is the

existence of metadata. Manual tagging and metadata creation are, however, time consuming and boring tasks. Our application makes it possible to tag proper nouns and acronyms in news articles partly automatically and partly semi-automatically. The resulting tags can also be collected into a separate matadata document. Further development is needed to increase the level of automation. More knowledge of language, terms and expressions should be included in the tagging application to reduce the number of manual corrections needed.

Traditional information carriers like paper are based on sequential presentation of information. New, electronic platforms like eBooks, MP3 players, electronic papers and carriers based on electronically erasable ink have the the same sequential mode of information presentation. Content which is read with these new devices is in many cases preferred to be personalized (the selection of the content is made based on user preferences or profiles) and collected from multiple sources. This cannot be profitably achieved with manual processes. Compilation of articles must be automated. The system presented here uses Kohonen's self-organizing map (SOM) to put a selection of articles into a logical, sequential order. The presentation was produced primarily for an electronic book and a speech synthesizer. The system was tested by comparing the automatically created order of articles with the order made by a reference group of 22 persons. The results show that the system functions quite well.

Both of these implementations show that content manipulation tasks can be automated. The results of these automatic or semi-automatic processes can be described as useable or good, but not equal to what people at their best could achieve. But even as such, these types of application speed up the publication processes and help in creating prerequisites for serving multiple channels and user groups.

Automatic abstracting of text documents is very useful with various media applications. This is the case especially with mobile terminal with restricted transfer and display capabilities but also in case where abstracts are used as a part of the metadata. Full potential of automatic abstracting is not yet utilized. The development work has been started at VTT to create an application where an automatic method for creating abstracts will be created. The development has been started with text extraction but will be continued towards text summarization system.

The solutions described above show that there is a lot of need and potential to automate content processing tasks and publication processes. These solutions have dealt with text. The challenges and opportunities expand a lot when images and other data formats are included. We VTT Information Technology have already gained experience in automatic indexing of video and will continue in that field as well.

Literature Cited

Anon.
2000 Search Everywere. http://discovery.altavista.digital.com/pages/2.shtml, http://discovery.altavista.digital.com/pages/2d.shtml

Bäck, A. al
1999 Publishing to XML and XSL W3C Recommendations. Technical Association of Graphic Arts. TAGA'99. Vancouver, CA, 2 - 5 May 1999 . Technical Association for Graphic Arts TAGA. (1999), pp. 92 - 106.

Glödstaf, H. (ed.)
1999 Integroitu julkaiseminen, tekniikka ja käyttökokemukset. Digitaalisen median raportti 2/99. Tekes, Helsinki 1999.

Gore, K.
1997 Cogito Auto Sum. What less can we say? Computers have the answer. http://Slate.msn.com/features/cogitoautosum/cogitoautosum.asp

Haykin, S.
1999 Neural Networks a Comprehensive Foundation Second Edition. Prentice Hall 1999.

Karen, Sparck, Jones
1999 Automatic summarizing: factors and directions. In Advances in Automatic text Summarization, Mani, J., Maybury, M. (ed.), The MIT Press, Cambridge, Massachusetts, London, England, 1999.