How to Create Topic Maps

Ronny Kerk, Stefan Groschupf
media style GmbH, Halle, Germany
email: ronny.kerk@gmx.de, sg@media-style.com
2003-10-23

Abstract

This paper is a work report of an intershipt at media style and describes the search for a feasible approach for Topic Maps creating. After having a look on Topic Map authoring by humans, two strategies for automatic Topic Map generation are discussed in the main part of this report. One is to use semantic metadata of the corpus the Topic Map should be created of and the second is to use named entity extraction for getting the topics and further using a part-of-spech-tagger for getting their associations. We argue, that the latter is a promising approach for large text bases with unstructured documents.
Though the main concepts of Topic Maps will be mentioned, it is not our purpose to fully explain the technology of Topic Maps. There are other papers which do this [1, 2].

Contents

Introduction

The Topic Map concept provides a possibility to represent knowledge about a specific domain, based on a corpus of - mostly unstructured - documents. You can compare a Topic Map roughly with the index at the end of a book. There are a lot of keywords (*topics*) which refer to one or more pages, paragraphs or figures in the book (*occurrences*).
A Topic Map is more than this. Additionally the *types* of topics (e.g. person) and occurrences (e.g. html-site) can be defined, but the richest feature for knowledge-representation is the possibility to describe relations and classifications (*associations* in Topic Maps) between two or more topics.
Each association can own a type (e.g. is component of), too, and has several *members* (concrete topics) including their *role* playing in the association.
There is also a pool of pre-defined topics with a special meaning called *published subject indicators*. For instance there are association-types (each type is a topic, too) representing class-instance-relations.

There are mostly two widely discussed application areas for Topic Maps. The first is their usage to navigate through the documents a Topic Map is based on and the second is the usage for representing the knowledge of the documents for further use. Both use cases do not exclude each other.

So it is a great advantage for a user who has to browse through a lot of documents - for example the output of a search-engine - when he can retrieve information about what is connected to the searched subject and how.

Our task was to find out how to create a Topic Map out of a large text-corpus of unstructured documents and save the contained knowledge about its subjects. An important aspect of this task was that any discovered or own approach should be usable in practice and not only serve as a scientific theory.
Furthermore it was not our purpose to reinvent the wheel. So if there was an open-source software which provides what we were searching for - or parts of it - it was considered for reuse.
The programming language of choice was Java because of its portability to different operating systems. Furthermore it is powerful and easy to understand.

The remainder of this paper is structured as follows. First we briefly introduce the Topic Map standard XTM. Then we discuss the usage of TM4J, a Java API that encapsulates the handling of XML Topic Maps, in our task to automate the generation of Topic Maps. After that an outline of manual Topic Map generation is presented. In chapter 4 we discuss the two methods to automatically generate Topic maps based on meta data or via named entity extraction. We conclude this paper with a summary of our work and directions for future research.

1. The XTM-Standard

It is always a good decision to use standards if there exist some. For Topic Maps there is an ISO-standard [3] and an applicability-standard based on XML which is called XTM (XML Topic Maps) [4]. It specifies the usable items and the structure of XTM-files which represent Topic Maps. In addition to topics, associations and occurrences it is possible to define scopes for certain items of a Topic Map. Scopes are often used for multilingual Topic Maps.
Along with the subject indicators mentioned in the introduction and resource referencers which refer to an addressable subject (e.g. if a certain website is the topic), scopes enable merging of several Topic Maps.
Without these features problems occur while merging. Especially if there are two topics which have the same title (base name) but different meanings within their scope.

Our self-created Topic Map should be created in the XTM-format or at least be transformable into it.

2. Topic Maps for Java

TM4J (Topic Maps for Java) [5] is an open-source software package supporting the XTM-standard. The set of Java-APIs is enabling import, export and manipulation of Topic Maps. With TM4J it is possible to create, alter or delete just single elements of a Topic Map as well as to merge Topic Maps, get statistical information about them or extract chosen fragments.
The processing of Topic Maps takes place in main memory and the persistent data can alternatively be saved in an object-oriented or relational database. An XTM-file with thousands of topics would be too large to be processed at an acceptable speed, so the XTM-format is more suitable for transporting and exchanging Topic Maps.

TM4J is easy to use and provides all the operations we needed in our project.

3. Topic Map Authoring

For extracting relevant topics and their associations it seems necessary to understand the content of the documents. It is a difficult job to decide whether a word (often a noun) or a word group is a topic or not. The Topic Map concept allows to make any word a topic.
Two readers (e.g. two knowledge engineers) will most likely develop two different Topic Maps just of the same corpus, dependent on their domain-specific knowledge and their background.
Nevertheless a handmade Topic Map should reach a much higher quality than a machine-made.

As described in [6] and [7], a feasible way for Topic Map authoring is to go through the corpus first and define a template for possible topic types, occurrence types and associations with their possible member-role-structures.

With help of this template the Topic Map creators have to extract the topics and associations by reading the documents using their knowledge about the certain domain.
The assignment of the occurrences of topics can be done while topic-extracting or later in an automatic phase. A parser could look for topic-names occurring more frequent in a document than expected of documents of the domain. Of course this doesn't work if the document is not machine-readable (e.g. a figure).
However, the questions are how much time and manpower does it need to get a high-quality Topic Map out of a large corpus, and is manually updating the Topic Map as quick as new documents are attached to the corpus.

Writing a template-creator-tool and implementing computer dialogs which support a human Topic Map creator while extracting items and arranging them with the template could have been the next steps of our work. The dialogs could give any suggestions for potential topics and associations for instance.
According to our given large text-corpus, we didn’t follow this way and concentrated on finding out if there is a fully automatic solution for Topic Map generation.

4. Automatic Topic Map Generation

4.1 Using available Metadata

A way for Topic Map creating with less effort is using of available semantic metadata of the documents. For XML-documents with a previously known structure (e.g. via a DTD or schema) it is possible to transform them via XSLT (XSL Transformations) [8] directly into XTM-files as suggested in [9]. The precondition in this case is – next to the required metadata – a defined Topic Map template, as described above, with all the types that can occur. With help of this template and the knowledge about the metadata an XSLT has to be developed and the automatic transformation-phase can start.

There is also an open-source software for supporting such transformations using metadata with the name MDF (A Metadata Processing Framework) [10]. It was written in Java by Kal Ahmed, the main developer of TM4J. The modular design makes MDF flexible for usage with different metadata structures. The XTM-standard and some other formats are supported as output.

Nevertheless, for our task this was no way simply because we had no structured documents.

4.2 Using Named Entity Extraction with GATE

A possibility for automatic topic extraction is to use a named entity extraction tool.
A named entity consists of a type (e.g. person) and a concrete string (e.g. Roger Wilco). This is exactly our understanding of a topic in a Topic Map. At the University of Sheffield a programme called GATE (General Architecture for Text Engineering) [11] has been developed which provides - among other things - named entity extraction. GATE is an open-source software written in Java, so we decided to extend it for our purpose. The software consists of several modules, which can be invoked via a GUI or directly out of a Java programme.
There is a gazetteer-module looking for all the words on the gazetteer-list if they occur in any document of the corpus and annotating the matches with a type (e.g. male first-name).
With our own gazetteer-lists we extended GATE's standard entity types (e.g. persons, locations and organizations) by our domain-dependent types (e.g. product names).
It is a simple method of GATE to go through word-lists and although this is not the fastest way it worked quite good for our purpose.

The annotations of the gazetteer-module will be used in transducer-modules, which processes self-written JAPE-grammars (Java Annotation Patterns Engine) [12] to match words and patterns in a document and make annotations to the matches. These annotations can be reused for a new matching with another grammar.

With a first self-written grammar we adapted GATE to our purpose.
The standard named entity transducer that comes with GATE doesn't take notice of the words next to a match, even if the match is only part of a compound noun or another group of words belonging together (e.g. Roger Wilco's starship).
Our grammar tries to match complete compound nouns and annotates them as potential topics which inherit the type of a named entity if possible. For example if the right side of a match is typed by GATE's named entity transducer as "vehicle", the whole noun group inherits the type "vehicle".

Another standard module of GATE is a part-of-speech-tagger which tags each word with its grammatical type and tense.
We used this to annotate the verbs occurring directly between two of our matched topics with a second JAPE-grammar. These matches are our automatically extracted associations in the Topic Map. The verb represents the association type (e.g. marry) and the topics before and after the verb are the association members.

We tried to improve the association extraction using GATE's coreferencer-module but this has proven too slow to be feasible. A coreference chain defines which pronoun belongs to which noun. So we could find associations not only directly between two topics but also between topics and pronouns referring to topics (e.g. she is married with him). The coreference also helps with different terms referring to the same item (e.g. Mr. R. Wilco and Roger Wilco are identical).

Before writing the association type into our Topic Map, we built the infinitive of the verb with help of the annotated tense by the part-of-speech-tagger. So one and the same association will not be saved several times only because of its different syntax.
Our first idea was using a stemmer for this task (e.g. Lovins [13] or Porter [14]), but this idea was discarded because of the insufficient quality of a stem for this purpose.
We could have used a synonym dictionary, too, but you have to be careful with deleting characteristics. The same applies to the topics, which are nouns in our case and can occur in singular and plural.

The assignment of the occurrences of topics can be realised while topic extraction phase or in the end by parsing the documents and analysing the frequency of occurring topic names.

5. Conclusions and Perspective

We saw that the Topic Map creating strategy is heavily dependent on the input corpus and the intended use of the Topic Map. If the aim is a high-quality Topic Map with strict structure and content (e.g. as navigation index of a manual) there will be the need for manually Topic Map authoring.
Using available metadata of a corpus can also lead to a strict Topic Map but is only usable for highly structured documents with a known semantic structure.
In our case with plenty of unstructured documents we were forced to choose another strategy. We found out that named entity extraction is suitable for getting topics out of documents. The constraint is that no decision is made whether a named entity should become a topic or not. It is possible that not every named entity is a topic in any case and there may exist topics which are not marked as named entities. Though the main challenge for automatic Topic Map generation is the extraction of associations between topics. Therefore we used the verbs occurring directly between two topics within one and the same sentence.

With improving text-recognition and intelligent linguistic methods the extraction of topics and associations will be raised. Especially if relations between several topics, occurring not closely to each other in the text, could be recognized in future.

References

[1] A. Altenburger: Authoring XTM Topic Maps, Part I, http://topicmaps.it.bond.edu.au/docs/6?style=printable, 2000

[2] H. H. Rath: The Topic Maps Handbook, White Paper, http://www.empolis.com/download/docs/whitepapers/empolistopicmapswhitepaper_eng.pdf, 2003

[3] ISO/IEC 13250: Topic Maps – Information Technology, Document Description and Processing Languages, http://www.y12.doe.gov/sgml/sc34/document/0129.pdf, 1999

[4] TopicMaps.Org: XML Topic Maps (XTM) 1.0, http://www.topicmaps.org/xtm/1.0/, 2001

[5] K. Ahmed et al.: TM4J – Topic Maps for Java, Version 0.8.2, http://sourceforge.net/projects/tm4j, 2003

[6] T. Bandholtz: A Taxi in Knowledge Land - A Use Case that Combines Topic Maps and Web Services in a Public Portal, http://www.idealliance.org/papers/xmle02/dx_xmle02/papers/03-05-03/03-05-03.html, last access: 2003-09-29

[7] S. Pepper, L. M. Garshol: The XML Papers – Lessons on Applying Topic Maps, http://www.ontopia.net/topicmaps/materials/xmlconf.html, last access: 2003-09-29

[8] W3C: XSL Transformations (XSLT), version 1.0, http://www.w3.org/TR/xslt, 1999

[9] J. Reynolds, W. E. Kimber: Topic Map Authoring With Reusable Ontologies and Automated Knowledge Mining, http://www.idealliance.org/papers/xml02/dx_xml02/papers/04-03-02/04-03-02.pdf, last access: 2003-09-30

[10] K. Ahmed: MDF – A Metadata Processing Framework, version 0.3, http://www.techquila.com/mdf.html, last access: 2003-09-30

[11] H. Cunningham et al.: GATE – A General Architecture for Text Engineering, Version 2.2, http://gate.ac.uk, 2003

[12] H. Cunningham et al.: Developing Language Processing Components with GATE (a User Guide), For GATE version 2.1, http://gate.ac.uk/sale/tao/tao.pdf, pp 86-98, 2003

[13] J. P. Lovins: Mechanical Translation and Computational Linguistics, Vol. 11, pp 22-31, 1968, after E. Frank: Lovins-Stemmer-Java, version 1.0, http://sourceforge.net/projects/stemmers, 2001

[14] M. F. Porter: An Algorithm for Suffix Stripping, Vol. 14, no. 3, pp 130-137, 1980, code at http://www.tartarus.org/~martin/PorterStemmer, last access: 2003-09-29