Indexing by Isotopy

Christian Mauceri

ENST Bretagne (e-mail: mauceri@fr.ibm.com)

Abstract. Within the Semantic Web initiative, Topic Maps have enabled a common architecture for indexing applications. In this paper, we present shallow reading methods aimed at semi automatic indexing through the integration of a finite state technology in the framework of Topic Maps.
Keywords: Finite state technology, Indexing, Isotopy, Semantic, Topic Maps.

Introduction

Anybody who has ever looked for precise information in a book knows how valuable an index can be. Indexing is a very ancient activity: in the antique Rome an index was a little slip attached to scrolls on which information about the work was written in order to easily identify its content without having to read it [Wellisch, 1991]. There are several types of indexes with different levels of complexity in the way they are structured. Indexes can be the mere positions of relevant words in a book (the good back of book index) or terms structured in complex thesauri used to describe the subject matter of documents in a library. The term 'index' spread over and may refer to tree structures used to speed up retrieval of records in databases; it is very often this accepted meaning which is used in the electronic document word[1]. Conversely, traditional indexing is too often perceived as a dusty technique because of its cost and slowness. Besides, it varies from an indexer to another and therefore suffers from a reputation of subjectivity. However human indexing is a mind production which, goes far beyond what a program in a machine will ever able to do. Instead of running after hypothetical objectivity, it should be accepted that human indexing is a hermeneutical activity [Mai, 2000]; it supposes an interpretation by a reader of what an author wrote depending on both the specific cultural and social context.
The objective of this paper is to show a way to reconcile automatic and human indexing through the integration of Semantic Web technologies, robust parsing techniques and a shallow reading method. A survey analysis application will be used as reference example in the remaining chapters.
This paper is divided into four sections. The first section presents Topic Maps, a Semantic Web technology used to organize information. The second section presents shallow reading methods aimed at gathering relevant vocabulary on particular semantic objects called isotopies. The third section describes a finite state technology implementation allowing for automatically

[1] Such an index is the list of positions in a document of each word occurring in it.

proposing candidate occurrences of isotopies according to the context they occur in. The fourth section deals with a real case application on survey analysis.

Topic Maps

The Semantic Web initiative shows the growing interest in sharing information between people which can be intelligible by computers. In this framework, Topic Maps is the technology addressing document indexing. Their core concepts are Topics, Associations, and Occurrences (TAO [Pepper, 2000]). Each of these objects can have a type; there are, hence, topic types, association types, occurrence types, all of these types being themselves topics. Topics represent subjects of anything one can imagine. The only constraint on topics is that one topic must refer to one and only one subject. Topic refers to subject by the mean of Universal Resource Identifiers (URI). In the survey analysis application we are interested in, the main topic types are:
Survey, Interview, Section, Question, Comments, Score, Person, Company, Term, Triggers, Contexts, ...
The main association types are:
• **Contains**; a survey **contains** interviews, an interview **contains** sections, a section **contains** questions.
• **Respondent**; persons are **respondent** of an interview.
• **Interviewer**; a person is the **interviewer** in an interview.
• **Rated**; a question is **rated** with a score,
• **Belongs to**; a person **belongs to** a company,
• **Expressed**; a comment is **expressed** about a question,
• **Is indexed by**; a comment **is indexed by** a term,
• **Is triggered**; a term **is triggered** by a trigger,
Once defined, the topic types and the association types can be used to describe surveys. Hence "Satisfaction Survey 2004" can be a topic of type survey, "Interview IBM 1756" can be a topic of type interview and the association "Satisfaction Survey 2004" **contains** "Interview IBM 1756" is an instance of the association type contains. The mechanism that allows for referring to real objects is called 'occurrence.' Similar to associations and topics, occurrences can have type (these types being topics as well). In the example considered here, topic occurrence types can be:
• **Described by**; the survey "Satisfaction Survey 2004" is **described by**:
"http://www.hermeneutician.com/#satisfaction%20survey%20guidelines"
• **Transcript**; the comment "Comment on Overall Section of Interview IBM 201756" has a **transcript** in:
"http://www.hermeneutician.com/Interview%20IBM%201756/#Overall"
• **Value**; the score "Score of Overall Section of Interview IBM 201756" has **value**:

"Satisfied"
• **Email**; the person "Christian Mauceri" has **Email**:
"mauceri@hermeneutician.com" .
The next question is why using topics to index documents? Saying they were designed for this purpose is obviously not the right answer, but their authors had in mind a number of common indexing issues:
• Flexibility; it is very easy to add new topics, associations and occurrences to a Topic Map,
• Ease of use in querying and browsing; Topic Maps are well suited to dynamically generate HTML pages allowing intuitive navigation in the map in order to rapidly find information (have a look at http://www.ontopia.net/operamap/).
• Serialization; Topic Maps can be serialized via an XML format named XTM,
• Standard; Topic Maps are a standard (see [ISO, 2000]),
• Collaboration; Topic Maps can quite easily be merged and exchanged.
What Topic Maps do not address is the way to define indexes and to how find them in unrestricted text. We will focus on this in the remaining chapters.

Shallow Reading

The inherent defects of variation and slowness in human indexing can be partially overcome by indexing a corpus rather than isolated documents. The term corpus (see F. Rastier [Rastier, 2002]) is related to law, religion, and linguistics; tt refers to a set of texts. In recent years, a regain of interest for corpus linguistics has been at the origin of a number of papers. In general, a corpus is merely seen as a set of texts or even sentences no matter how they are related to each other.
Originally, however, this term was set up by disciplines like hermeneutic and philology, where the relations between texts were taken into account. In this tradition, the corpus is a structured set of texts sharing characteristics of genre and discourse (legal discourse, medical discourse, etc.). Besides, a corpus is built for a particular purpose, and corresponds to a need. It is this definition of a corpus that will be used in the remaining paragraphs.
There is not isolated text, as a text respects a set of social norms shared by other texts. From an indexing perspective, it is an important point to take into account, because:
• It allows for a global view of text, and therefore reduces the indexing variation by considering what texts share and what makes them different.
• It reduces semantic variation. Indeed, words and expressions tend to have narrower meaning. Syntactic constructions are more regular and are very often almost frozen.
However, it is very often impossible to read, in the traditional way, an entire corpus or even a representative sample of it. (How does one read all of the

articles of the Wall Street Journal over the past 10 years or thousands of commercial reports in a short period of time?). Nevertheless, the indexer has an a priori knowledge of the content of the corpus that he/she is supposed to process; its literary kind, the writing style, what the texts are talking about. This knowledge is very important and is the basis of the reading method proposed here.

Interpretative Semantics gives a theoretical framework of what reading is and how indexing is connected to it. Interpretative Semantics takes its roots in the Saussurian statement that human languages are made of oppositions. People perceive similarities and differences between linguistic objects, A.J. Greimas, in [Greimas, 1986] gives an example of such an opposition, 'bet' Vs 'pet', (he actually used 'pas' Vs 'bas' in French) in fact the fundamental structure in this opposition is given by 'b' Vs 'p' or 'voiced' Vs 'non-voiced'. At the semantic level, these oppositions allow for defining semantic relations; hence the opposition, 'girl' Vs 'boy', defines the semantic axis 'sex'.

The minimal units of sense used to describe these structural relations within a corpus are called semes (see for instance, [Pincemin, 1999]). Semes are not used to describe isolated words but rather are defined as sets of words related to them. For instance, instead of describing a priori 'chair' as {/furniture/, /for sitting/,etc...}, /furniture/ is described by the set {'chair', 'closet', 'table', 'sofa'}. Semes depend on context, are not universal truth; and are defined by the corpus reading. Restricting semantic relations definition at the corpus level avoids looking for an improbable universal ontology of semes.

In Interpretative Semantics, reading is the result of an interpretation, an operation specifying the meaning of a text. An interpretation can add or remove semes to words, depending on the context. A.J. Greimas gives the example of the sentence "The superintendent barks" to show how a seme can be added by the interpretation process. In this sentence a seme /animal/ is added to the word 'superintendent' because it is the subject of the verb 'bark' obviously bearing the seme /animal/, giving the sentence its pejorative interpretation. This example introduces a central concept in Interpretative Semantic, the notion of isotopy (see for instance, [Sonesson, 2004]); an isotopy is the effect produced by a same seme recurrence in a corpus[2]. An isotopy analysis produces a list of words having some contextual semes in common.

Isotopies are useful for word sense disambiguation. For instance, let's consider the word 'bugs' in the two sentences:

- Bugs were crawling everywhere in the room.
- Bugs were found in the program.

In the first sentence there is an isotopy /animal/ between 'bugs' and 'crawling', in the second sentence there is an isotopy /computer/ between

[2] On isotopy in a corpus A.J. Greimas give in [Greimas, 1986] an interesting example of the isotopy /death/∼/life/ in Bernanos' work

'bugs' and 'programs.'
Isotopies are given a priori and come before the semes definition; they are expected by the reader/indexer, they ensure the corpus coherence.
As isotopies are given a priori and entail semes characterization it is much more productive to gather the corpus vocabulary related to an isotopy rather than to build an a priori hierarchy of semes to describe the entire corpus vocabulary. Isotopy recognition is triggered by keywords depending on the context they appear in. A person reading a list of words easily detects potential triggers, so a very productive approach is to detect these triggers in the corpus vocabulary without having to read the entire corpus, it is what we call a shallow reading of the corpus; gathering isotopy trigger candidates by reading the corpus vocabulary, looking at them in their contexts and expressing rules inhibiting or refining them when they occur in particular contexts.
But why read all the vocabulary? Indeed, as an isotopy is a seme recurrence in the corpus it can be thought that only words occurring more than twice are of interest; it is precisely because a seme occurring very often in the corpus can be borne by words occurring only once it is important to look at rare words which represent the major part of the vocabulary, in average words occurring only once represents more than 50% of the vocabulary. For instance, in an application aimed at detection of commercial reports offending the European Privacy Regulation in a French bank the word 'fingernails' appeared only once in the context "her only project is to paint her fingernails which is a clearly sexist statement. This example is interesting because beyond the argument in favor of considering the scarce words, it shows traditional lexicons cannot help in detecting such cases heavily depending on contexts.
Once the candidates are gathered they must be allocated to the isotopies they are supposed to trigger and checked in context in order to write rules inhibiting or refining the triggers. A trivial example of such rules is given by the occurrence of the trigger 'cost' of the isotopy 'Pricing' in the context: **"Mr. Redford was happy with fisher.com commitment to reach their objectives at all cost"**, obviously the indexer doesn't want to trigger the 'Pricing' isotopy in such a case and would like to write something like "I don't want the word 'cost' to trigger the isotopy 'Pricing' when it is preceded by 'at all."' In summary, shallow reading consists in:
• Locating in the corpus vocabulary words triggering isotopies,
• Building a concordance of these words and their contexts in the corpus,
• Writing rules inhibiting or refining the triggers according to these contexts.
Many of these rules can be expressed by the means of regular expressions and integrated in the Topic Maps frameworks; it is the subject of the next chapters.

Finite State Machines

A finite state machine consists of a set of states, a start state, a final state, an input alphabet, and a transition function. A transition function maps an element of the input alphabet and a current state to another state. At the beginning of a computation the machine's current state is the start state, and changes of state depend on an input string and the transition function.

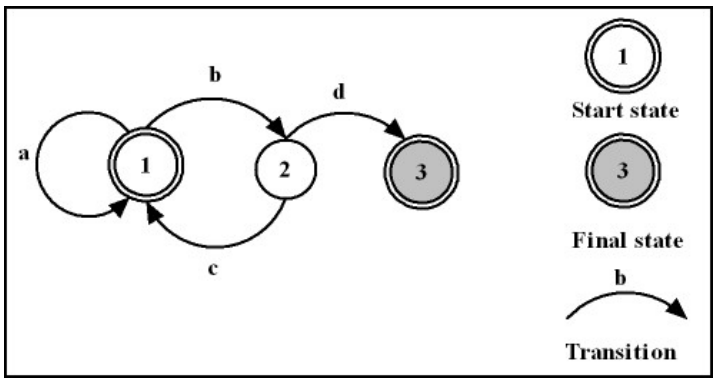

Fig. 1. An automaton example.

The figure above represents an automaton with state set {1, 2, 3}, an input alphabet **{a, b, c, d}**, a start state 1, a finale state 3 and a function transition mapping (1,a) to 1, (1, b) to 2, (2, c) to 1 and (2,d) to 3. This automaton recognizes the strings **{"aaabcbd", "abcbcbcbd", "bcbd", "bd", etc.}**. Finite state machines can be described by regular expressions whose principal operators are union, intersection, complementation, concatenation, and Kleen star on the input alphabet. (See Aho, Sethi and Ullman [Alfred V. Aho, 1986]). For instance, the automaton given in example is described by the regular expression **"a"* "b" "cb"* "d"**; the Kleen star operator means no or many occurrences of the expression it applies to, hence **"a"*** means no or many occurrences of the character **"a"** and **"cb"*** no or many occurrences of the character **"c"** followed by the character **"b"** so **"a"* "b" "cb"* "d"** defines strings beginning by zero or many characters **"a"** followed by the character **"b"** followed by zero or many substrings **"cb"** and ended by the character **"d"**. In the same way the expression **"ab" | "bc"** defines the strings **"ab"** or **"bc"** (| is the union operator). The expression **@* ("ab" | "bc") @*** defines all the strings containing the substrings **"ab"** or **"bc"** (@ means any character). The expression **ˆ (@* ("ab" | "bc") @*)** defines the strings which do not contains the substrings **"ab"** or **"bc"** (ˆ is the complementation operator). The expression **ˆ(@* ("ab" | "bc") @*) & ("a" @* "b" |"b" @* "c")** defines the strings which do not contains the substrings **"ab"** or **"bc"** starting by the character **"a"** and ending by the

character **“b”** or starting by the character **“b”** and ending by the character **“c”** (**&** is the intersection operator).
Transducers, which are finite state machines with output have been widely used in Natural Language Processing (NLP) (see for instance [Abney, 1996], [Grefenstette, 1996], [Hobbs, 1996] or [Roche and all, 1996]). In particular, they have been used in shallow parsing and local grammar implementation. A shallow parser aims to identify phrasal constituents, such as noun phrases, and the functional role of some of the words, such as the main verb, and its direct complements [Abney, 1996]. Local grammars are used to describe local linguistic structures in the form of graphs (See [Silberztein, 1993], [Masson, 1993]).
The rules discussed in the previous chapter are implemented by finite state machines whose transitions are of two types; simple transitions and epsilon transitions. Simple transition maps an element of the input alphabet and a state to another state. Epsilon transitions maps the empty word (commonly called epsilon) and a state to another state. In addition, they produce a meta-character. These meta-characters are used to control further processing on the recognized strings. Typically these machines are union of machines which can be represented by regular expressions of the form:
<lpat1> 0:“(” <cpat1> 0:“)” <rpat1> . . . <lpati> 0:“(” <cpati> 0:“)” <rpati> . . . <lpatn> 0:“(” <cpatn> 0:“)” <rpatn>
0:“command1” . . . 0:“commandi” . . . 0:“commandn”
where epsilon productions **0:“(”** and **0:“)”** are used to mark the beginning and the end of the strings recognized by the surrounded regular expressions **<cpati>**. The epsilon productions **0:“commandi”** specify the processing on the corresponding recognized strings. For instance the expression:
“at” <sep>+ “all” <sep>+
0:“(” “<Pricing>” 0:“)” “cost” 0:“(” “</Pricing>” 0:“)” <sep>+
0:“rep: $1” 0:“rep: $2”
recognizes the string, **“at all <Pricing>cost</Pricing>”** and produces the string **“at all cost”**. In this expression; **<sep>= “ ”|“\t”|“\n”|“\r”;** It means <sep> is a separator (a space, a tabulation, a new line, or a carriage return), and therefore **<sep>+** means one or many separators.
The command **0:“rep: $1”**, means: replace the string matched by the regular expression surrounded by **0:“(”** and **0:“)”** by nothing.
The machine evaluator rewrites all characters not recognized by a machine and applies the specified processing to the recognized substrings. So these machines are transducers; the output of a transducer can be used as input to another one; such an operation is called a cascade. The evaluation is very fast because they are almost deterministic: the only possible backtracks occurring for the epsilon productions **0:“(”**. Despite their simplicity they can be used to capture many linguistic phenomena. It is important to have in mind that other commands than ‘rep’ and ‘tag’ can be easily defined to control their outputs and performing actions on the Topic Maps.

Let's see how to implement the shallow reading method previously described with this mechanism. We suppose the corpus analyzed is organized in a Topic Map having topics of type chapter, section, sentence, or whatsoever whose instances have textual occurrences. In the example of survey analysis we are interested in, these topics are instance of comment type and their occurrences are the transcripts of these comments.
Once the triggers have been collected we build a first automaton <**T1**> which is the union of expressions of the form:
0:"(" "<trigger>" 0:")" 0:"tag: $1 <isotopy>"
In these expressions <**trigger**> is the written form of a trigger and <**isotopy**> the name of the isotopy it triggers, for instance:
0:"(" "cost" 0:")" 0:"tag: $1 Pricing"
This first automaton is applied to the transcripts and produces new transcripts adorned by XML tags corresponding to the argument labels of the tag command. This same command adds dynamic associations of type '**is indexed by**' between the question topic the transcript is an occurrence of and the corresponding isotopy topic. It also adds the position of the trigger in the transcript as an occurrence of the triggering word. For instance, let's suppose the following transcript:
"Mr. Redford was happy with fisher.com commitment to reach their objectives at all cost"
is an occurrence of the question topic 'overall' in the interview "Redford Entertainments 20035," then application of the first automaton will produce:
"<Hum>Mr. Redford</Hum> was <Euph>happy</Euph> with <Comp>fisher.com<Comp> commitment to reach their objectives at all <Pricing>cost</Pricing>"
and adds an association of type '**is indexed by**' between 'overall' and 'Euphoric' just like an occurrence of 'happy' at the position 27 of the transcript, provided that 'happy' has been declared as a trigger of the isotopy 'Euphoric' in the automaton <**T1**>. It is now possible to access through the Topic Maps the texts where the isotopy 'Euphoric' occurs, in particular in the transcript given below.
"Mr. Charles is moderately happy with the service he received"
rewritten as
"<Hum>Mr. Charles</Hum> is moderately <Euph>happy</Euph> with the service he received".
The simplest way to inhibit the trigger 'happy' when it comes after 'moderately' is to add the following rule to the first automaton <**T1**>.
<be> <sep>+ 0:"(" "moderately" <sep>+ "happy" 0:")" <sep>+ 0:"tag: $2 Dysph".
A second automaton <**T2**> allows for the inhibition of certain triggers. This automaton is made of rules like:
"at" <sep>+ "all" <sep>+ 0:"(" "<pricing>" 0:")" "cost"
0:"(" "</pricing>" 0:")" <sep>+ 0:"rep:" 0:"rep:"

which recognizes the string **"at all <pricing>cost</pricing>"** produces the string, **"at all cost"** and remove the association of type '**is indexed by**' between the isotopy topic 'pricing' and the corresponding comment. This second automaton <**T2**> cascaded with the first one <**T1**> allows the implementation of simple positive and negative rules, activating or inhibiting triggers.
The contexts analysis often suggests taking into account recurring linguistics constructions such as:
"Ms. Wilson [**declares, pointed out, thinks ...**] **that ..."**
which can be captured by expressions like:
<**HumanChunk**> <**sep**>+ <**say**> <**sep**>+ **"that" 0:"(" ^(@* "." @*) 0:")" "." 0:"mark: $1 Statement"**,
where <**HumanChunk**> is a regular expression detecting a nominal chunk containing the seme /Human/, <**say**> is a regular expression detecting a verbal chunk containing verbs introducing a statement. Hence the mark comment will surround the string recognized by **^(@* "." @*)** with the tags <**Statement**> and </**Statement**>. Expressions like this, aimed at detecting recurring discourse structures [Marcu, 1999], can be compiled in a third automaton <**T3**> which, when cascaded with the previous automata, allows for stressing on important contexts.

Experimentation

We have used this method to semi-automatically index the customer satisfaction reports of a very big global company. Basically the two generic isotopies we were interested in were:
• The pervasive issues clients were talking about; What were the topics clients were praising or complaining about?
• The feelings of the interviewed customers; Were they happy or not? What were they expecting?
The vocabulary of about 200 reports was read and categorized in relation to the semes:
• /PI/ for pervasive issues (subdivided in more specific semes like /responsiveness/, /costing/, etc...),
• /Euph/ for euphoric,
• /Dysph/ for dysphoric,
• /Exp/ for expectation,
• /H/ for human.
Around 600 words and expressions (out of 20000 words) were selected out of the corpus. We used a <**T0**> automaton to mark grammatical words, modal and auxiliary verbs. The <**T1**> automaton was slightly different than the one we presented in the previous chapter because in a real case application a same word potentially triggers multiple isotopies but overall

the schema was the same as the one sketched in the previous chapter.
A post XSLT [W3C, 2005] processing colored in red chunks containing dysphoric semes, in green those containing euphoric semes. Furthermore this same post processing underscored typical expressions and colored in blue pervasive issues and proper nouns.
The documents have then been indexed manually, focusing on the colored and underlined parts (See B. [Pincemin, 2001]). All these results have been combined in a web application based on XSLT applied to the XML format of the resulting topic map (XTM). The main advantage, from a business point of view, has been that for the first time transcripts of the interviews have been really used by business analysts because of the easiness to access them trough the web Topic Map interface and the pervasive issue indexing sheds a different light on the survey results than the mere scores given by the clients.
Gathering vocabulary and typical contexts took approximately three days and indexing the final 200 reports took an additional day. The main problem we faced was to translate the rules expressed by the business analyst indexer in regular expressions; the regular expressions language being too complex for non-knowledgeable people.

Concluding remarks

Our aim was to evaluate how the notion of isotopy is perceived by business people and how cascades of automaton can be used by them in order to automatically spot them. Even if the reading of a huge vocabulary is a tedious task, the proposed method and the isotopy notion has been quite well accepted. However, the usage of regular expressions and cascades of automaton is completely out of scope, requiring a lengthy learning phase .
The linguistic development environment Intex designed by Max Silberztein [Silberztein, 1993] offers an example of user friendly interface for finite state technology, it is however not designed for indexing purposes and not integrated into a standard indexing framework like Topic Maps, representing finite state machines by the mean of graphs is good and can be used for applications bound to a large public.
A very important issue not discussed in this paper deals with the global analysis of the resulting indexing. Indeed Topic Maps provides great browsing capacity but cannot encompass the global corpus structure. A preliminary work shows that demographic clustering [Johannes Grabmeier, 2002] techniques can be integrated in the Topic Maps framework in order to check the indexing consistency and discrimination power. This point is very important because it extends the structural principle of opposition to the whole corpus; what are the isotopies opposing or gathering texts in a corpus? This complementary technique is the missing retroaction loop in the shallow reading

process.
Future work will focus on graphical user interfaces (GUI) making finite state machines easier to use in the presented framework, and integration of demographic clustering techniques in the shallow reading process.

References

[Abney, 1996]Steven Abney. Partial parsing via finite-state cascades. 1996.
[Alfred V. Aho, 1986]Jeffrey D. Ullman Alfred V. Aho, Ravi Sethi. *Compilers Principles, Techniques and Tools.* Addison Wesley, 1986.
[Grefenstette, 1996]Gregory Grefenstette. Light parsing as finite-state filtering. 1996.
[Greimas, 1986]Algirdas Julien Greimas. *Sémantique Structurale.* PUF, 1986.
[Hobbs, 1996]Jerry Hobbs. Fastus: A cascaded finite-state transducer for extracting information from natural-language text. 1996.
[ISO, 2000]ISO. *ISO/IEC 13250,Information Technology – SGML Applications – Topic Maps.* ISO, Geneva, 2000.
[Johannes Grabmeier, 2002]Andreas Rudolph Johannes Grabmeier. Techniques of cluster algorithms in data mining. 2002.
[Mai, 2000]Jens-Erik Mai. The subject indexing process: an investigation of problems in knowledge representation. 2000.
[Marcu, 1999]Daniel Marcu. Discourse trees are good indicators of importance in text. 1999.
[Masson, 1993]Olivier Masson. Automatic processing of local grammar patterns. 1993.
[Pepper, 2000]Steve Pepper. The tao of topic maps, finding the way in the age of infoglut. 2000.
[Pincemin, 1999]Bénédicte Pincemin. Sémantique interprétative et analyse automatique des textes : que deviennent les sèmes ? 1999.
[Pincemin, 2001]Bénédicte Pincemin. Résoudre la surcharge informationnelle sans la décontextualiser. 2001.
[Rastier, 2002]François Rastier. Enjeux épistémologiques de la linguistique de corpus. 2002.
[Roche and all, 1996]Emmanuel Roche and all. Fastus: A cascaded finite-state transducer for extracting information from natural-language text. 1996.
[Silberztein, 1993]Max Silberztein. Dictionnaires éléctroniques et analyse automatique de textes : le système intex. 1993.
[Sonesson, 2004]Göran Sonesson. Isotopy. *Internet Semiotics Encyclopedia*, 2004.
[W3C, 2005]W3C. *XSL Transformations (XSLT) Version 2.0.* W3C, 2005.
[Wellisch, 1991]H. Wellisch. *Indexing from A to Z.* 1991.