DEXA 95 (Database and Expert System Applications)
Workshop on Digital Libraries, London, UK pp. 357-362

Principles and Tools for Authoring Knowledge-Rich Documents

Robert P. Futrelle and Natalya Fridman

Biological Knowledge Laboratory, College of Computer Science
161 Cullinane Hall, Northeastern University, Boston, MA 02115,
{futrelle,natasha}@ccs.neu.edu
617-373-2076, FAX: 617-373-5121

Abstract. Digital libraries can take advantage of documents that have their content (semantics) explicitly represented as knowledge structures. These knowledge-rich documents can be created by using natural language processing techniques or by acquiring knowledge from the author during the authoring process. We discuss the latter approach, and introduce the notion of Knowledge-Based Authoring Tools (KBATs). It turns out that the primary task for a KBAT is to reduce (apparent) ambiguities in the text to zero, obtaining a unique analysis or, barring that, reduce the ambiguity of a particular piece of text to the point that only one or two simple queries to the author will resolve the issue.

1. Introduction

Digital Libraries of the future will be able to exploit the knowledge contained in documents in a variety of ways, to enhance retrieval, browsing, and summarization of information. The simple 'flat text' of documents does not, by itself, support such activities. The structure and content (meaning) of documents must be discovered by computer analysis or inserted by the author. A lot of work has been done on corpus analysis by computer to support computer analysis [1] . This paper explores what might be done for the author by the development of Knowledge-Based Authoring Tools (KBATs) that will allow authors to contribute their unique knowledge of their document to create a structured, knowledge-rich form of it for use in a Digital Library. There are a number of tools already in existence that are used by authors when they create documents, but they are generally knowledge-poor.

Examples of existing tools include spell-checkers and bibliography systems. Both depend on databases. Bibliography systems take care of cross-referencing and the customized formatting of the bibliography. Spell-checkers have an additional learning capability, because once a user has confirmed that a newly encountered word is to be added to the spelling database (or "custom dictionary"), no correct spelling of that word in the future will raise a flag. We use a broad definition of "structure", so that spelling corrections are said to add structure because they remove erroneous terms, making searches and analyses more accurate. Some structure is required by publisher's standards, and some structure is generated by the author through his or her awareness of the standards and preferences of the readership.

The KBAT we envision operates in the following way. Its goal is to build as much knowledge structure into a document-in-progress as it can. This means determining, as much as possible, the meanings of the terms used and the meanings of sentences and larger structures. Based on this, some type of a limited knowledge representation can be built for the document. The KBAT draws on large, domain-specific resources maintained in the author's machine and in knowledge servers on the net. When the KBAT cannot determine the nature or structure of an item, it can query the author. This has to be done carefully, as we'll explain below. One of the primary purposes of the KBAT is to gather important information from the author during the authoring process that would be very hard to reconstruct from the document later on, without the author's help.

There are two essential topics that we will expound on at some length. The first, *ambiguity*, is critical for understanding the problems faced by the KBAT. The second, *ontologies*, is critical for understanding how the KBAT comes to understand the text contents and build a knowledge representation, or short of that, how it goes about framing questions for the author.

2. Tools — Their Costs and Benefits

We describe some structuring principles and tools that currently exist and compare them with knowledge-based approaches. All these tools add structure or conciseness beyond flat text. Adding structure to a document is not "free", so a cost-benefit analysis must be made.

- **Costs of using an authoring tool:**
  - Mental load on the author in learning and using
  - Time (interruption) for tool use
  - Cost may decrease over time because of familiarity or because the system builds databases
- **Benefits of using an authoring tool:**
  - Authoring may be made easier by tool
  - Reading is more accurate, efficient, because of added structure
  - Automated systems can exploit structure

Document structures — These structures are essentially syntactic with little knowledge content. The tools for generating them are almost wholly dependent on the author for structuring information. Fonts, spacing, centering, and lists, focus the reader's attention and require little effort on the author's part. Titles, abstracts, and standard sections are components required by tradition and style rules. Figures, tables and captions present information that would be difficult or impossible with text. There are numerous cross references within text to bibliographic items; tables of contents show the organization at a different level; indexes can be tedious to construct but very helpful to the reader. Markup and hypertext can burden the author even more but can produce a document suitable for computer-supported navigation.

Databases that assist authoring — Spell-checking databases offer extensive, but not complete help (they will miss errors such as their/there or ton/tin). Authors can customize them to their specialized vocabulary. Authors are willing to use them. Dictionary definitions and thesauri are useful adjuncts. Bibliographic databases are critical for scholarly writing. Style and macro libraries support a consistency of style and formatting according to various standards. The databases can be large and obtained from outside sources, adding a good deal of content with little author input.

Writing style — The controlled vocabulary, nomenclature, and stock phrases of a specific domain makes the job of expression and comprehension much easier. A technical rhetorical style adds to conciseness (reduces ambiguity). This is a natural part of the writing process, but it does not necessarily yield text that is easy for a KBAT to analyze.

KBATs — There are number of challenges that a KBAT must deal with in its goal to produce a knowledge-based description of the contents of a document, or simply put, a knowledge representation. The primary problem is one of ambiguity. Though a well-written text is not ambiguous to a reader that knows the subject domain, there are many parts of the text that are ambiguous to a computer system. This is because of our limited understanding of how to build truly intelligent natural language and knowledge analysis systems. So in parallel with the development of interactive KBATs, it is crucial that we continue to improve our understanding of natural language processing techniques so we can raise the efficiency and accuracy of machine analysis of documents. In the meantime the KBAT will need to rely in part on authors for information, while attempting to do extensive disambiguation on its own.

3. Ambiguity

A computer will have difficulties with the meaning of individual words, e.g., "stock" can be a broth, cattle, or a security. It will also have difficulty with most sentences because it will find analyses that we normally reject without a moment's thought. There are a variety of different types of ambiguities. Thus, "an aluminum wire bin" could be a bin made of aluminum that holds wire or a bin that holds aluminum wire. "They walked around the lab" can mean walking within or outside of the lab. "She put the manual on the shelf in the machine room" could mean that the manual on the shelf was moved to the machine room or the manual was put on the shelf that was in the machine room.

If a natural language processing system (NLP) relies only on syntactic analysis and uses a lexicon (in which each word often has more than one meaning), then even the following simple sentence has serious ambiguity problems:

Sentence: "The net went dead apparently for no reason."

Ambiguity: Does "net" mean a fishing net or communication network?

Ambiguity: Does "went" mean a change of state or "moved"?

Ambiguity: Does "reason" mean a cause or cogitation?

Ambiguity: Does "apparently" say that the going dead was only apparent, and not actual, or does it say the reason was (not) apparent?

The first three ambiguity problems listed are examples of *lexical ambiguity* — words having more than one meaning. The last problem is an example of *structural ambiguity* — it is not clear what constituent "apparently" attaches to.

What is the difference between a human and a simple machine analysis that makes the human so much better at solving the ambiguity problem? Primarily, it is a matter of *knowledge*, knowledge of language and knowledge of the world. If the example sentence above were uttered in a computer context, "net" would be assumed to refer to a communications network. People also know that "net went" is unlikely to be a description of a network moving to another place. Also, the word "dead" is a state, not a place. "Apparently" would be expected before "went" if it described an apparent event; since is appears after "went dead" and adjacent to the prepositional phrase, "for no reason", it is attached to the latter. Since an event is being described, the use of "reason" as a cause is the preferred interpretation. The challenge for a KBAT is to try to reproduce this reasoning using its own resources that describe word meanings and language structure. To the extent it succeeds, it can build a knowledge representation that accurately reflects the structure of the sentence. This in turn can be stored along with the document text in a Digital Library depository. Having such a rich representation will markedly enhance the operation of the Library.

4. Resolving Lexical Ambiguity

There are three types of lexical ambiguity [2]. *Polysemy* is when several senses of a word are related to one another. Thus, "enter" can refer to typing text into a word processor, or entering a room. *Homonymy* is when a word can have a number of unrelated senses. Thus, "times" can refer to multiplication or to temporal events. *Categorial ambiguity* refers to words whose syntactic category may vary, and is orthogonal to the two other classifications, e.g., "crash" can be a verb or a noun.

We will describe one approach to disambiguation, one that can be mechanized and used in a KBAT. We in no way imply that this is easy to do, or that it will be uniformly successful. We are simply describing an approach that has had some success. Disambiguation works by exploiting the semantics of *context*. In contexts such as the following, two polysemous senses of "entered" are being used:

1. "He $\underline{\text{entered}}_1$ the quote into his word processor."
2. "She $\underline{\text{entered}}_2$ the room just after the group meeting started."

For this discussion we will assume that the parts of speech of the other items in the sentences and their relations as arguments or modifiers are known. We further assume that all other words in the sentence except "entered" have been disambiguated.

Disambiguation proceeds by inserting $\underline{\text{entered}}_1$ and $\underline{\text{entered}}_2$ into both sentences and ranking the results for semantic consistency. The consistency is high if the meaning being tested is highly compatible with the controlling constituents in the context. An easy way to see this is to translate $\underline{\text{entered}}_1$ to "typed" and $\underline{\text{entered}}_2$ to "walked" or "walked into",

"He typed the quote into his word processor." (1a)

* "He walked the quote into his word processor." (1b)

"She walked into the room just after the group meeting started." (2a)

* "She typed the room just after the group meeting started." (2b)

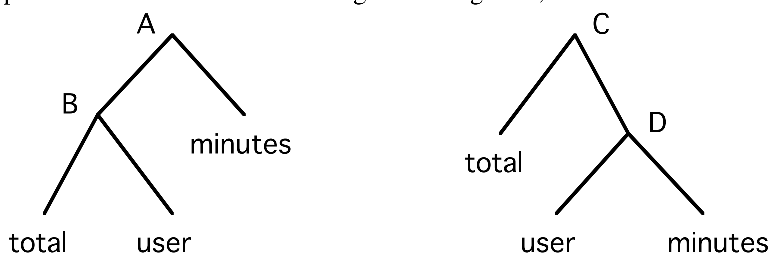
The starred items are the incorrect ones. The problems with 1b are that "quotes" are text or spoken language, not animate and capable of being walked, and that one does not walk "into" a word processor; it is not a building or a room. The problem with 2b is similar. Sentences 1a and 2a are highly consistent so disambiguation is achieved. If the KBAT is not successful in locating the relevant context and identifying its salient properties, it could present the author with the two alternate sentences, e.g., 1a and 1b, and let the author indicate what was intended.

5. Resolving Structural Ambiguity

The structure of a sentence plays an obvious role in determining its meaning. Ambiguities arise when a sentence can be interpreted as having two or more distinct structures. We will start with a common phenomenon in English, noun-noun modifiers. In the phrase, "chocolate cookie lover", the first noun, "chocolate" is most likely a modifier of the adjacent word "cookie". But in the phrase, "chocolate Easter egg", it modifies the third constituent "egg". We will describe an ambiguity resolution strategy that a KBAT could pursue, for the following noun phrase,

"total user minutes"

The two possible structures have "total" modifying "user" or modifying "minutes". The parse trees for the two cases are given in Figure□1,



1a: "total" modifies "user"   1b: "total" modifies "minutes"

Fig. 1. Parse trees for two analyses of the ambiguous noun phrase, "total user minutes". The correct analysis on the right succeeds because the properties of "minutes", as propagated to node D and then C, are consistent with those required by the modifier "total".

The ambiguity is resolved by computing the consistency of the semantic attributes of the modifier "total" with its two possible arguments. This can be done using techniques closely related to those of contemporary unification-based grammars [3,4]. In those approaches, the syntactic consistency of two constituents is computed by unification of the feature structures (the collections of attributes and values) of the children of a node. (For semantics, a more complex computation than unification may be required.) In Fig.□1a, this computation is done at node B, resulting in a low consistency value. This is because in order for "user" to be an argument of "total", "user" would have to be a plural noun which it is not. The computation of consistency in Fig.□1b is more complex. It requires a computation at node C, which in turn requires the determination of the feature structure at node D. The features at node D are dominated by those of "minutes", because "minutes" is the *head* of the noun phrase "user minutes"; the features of "user" at D disappear once the computation at D is done. Consistency is achieved at C. The semantic content of the required feature structures is determined by ontologies, as described in the next section.

Another major type of ambiguity is the prepositional phrase attachment problem (PP-attachment). An example of the standard (left) attachment is "...the book on the table in the room on the second floor of the house". In each case, the prepositional phrase attaches to the noun immediately to its left. But in the sentence, "Our VPE supports the creation of programs by visual means", the prepositional phrase "by visual means" modifies the noun "creation" rather than the noun "programs" immediately to its left. Parse trees for the two candidate structures can be generated as before and their semantic consistencies computed. The result is that both "by" and "means" are highly consistent with the noun "creation", but not with the noun "programs".

Other important ambiguities include structuring conjunctions and resolving references, e.g., pronominal reference. Virtually all structural disambiguation problems can be investigated by the techniques described above. The semantic consistency analysis can be done in parallel with syntactic analysis or afterwards. None of these computations are particularly easy or efficient — they present a serious challenge for NLP methods.

6. Ontologies and Ambiguity

The knowledge needed to resolve ambiguities based on semantics is contained in *ontologies* [5]. An ontology is a collection of interrelated knowledge-description objects and axioms describing how the objects can be meaningfully used in relation to each other. These structures contain the knowledge of a specific domain at the conceptual level and are used by the KBATs to analyze the text. Once the semantics of the sentence is determined, one can represent this knowledge in the ontology by creating instances of the general conceptual structures [5]. This knowledge is subsequently used for the intelligent Information Retrieval in Digital Libraries.

Objects in the ontology are grouped into *classes*. Classes can be defined by *frames* (or any other compatible formalism) and their properties are represented as *slots* in the frame. Slots, in turn, have their own properties, like their domain, cardinality, etc. A frame can be designed to represent a disk drive and have a slot for *capacity* which in turn holds a *bytes* object representing some number of bytes. A specific disk is represented by an *instance* of the disk frame and its capacity by an instance of a bytes slot with an actual value such as 2□GBytes.

Classes are organized in a hierarchy by *inheritance*. Thus, a particular disk drive class may have the following inheritance chain,

peripheral -> storage-unit -> disk -> Acme-90

The properties of the superclass, such as slots and their properties, and axioms associated with the class, are inherited by the subclass. For example, the storage-unit class will introduce *capacity* which will be inherited by *disk*. Multiple inheritance is also possible so that the *Acme-900* class could also inherit from a *SCSI* class. The line between classes and instances is not sharp. We may want *Acme-900* to be an instance of *disk* or we may want to use instances to represent particular Acme-900 disks, with particular serial numbers. Which of the options is adopted depends on the purpose for which the particular ontology is created. There are ontologies created for various clusters of knowledge, such as Time, Space, Quantities, etc. [7]. There is also a large-scale attempt to create the ontology to represent all of the commonsense knowledge: the CYC project [8]. In our case we design ontologies specifically suited for text.

The use of ontologies in disambiguation can be seen for the "total user minutes" example discussed earlier. There are two alternatives. Either "total" modifies "user", or it modifies "minutes" in "user minutes". The *total* class inherits from *sum* which in turn inherits from both *math* and *aggregate.* But the most important attribute of *total* for our purposes is the *subcategorization* slot which defines the restrictions on the arguments acceptable to the word "total", their class and their properties, Fig.☐.

```
Total                              If ?x is Total
Superclasses:                              and (?x.Subcategorization) =
    Sum                                    and ?z is in ?y
Slots:                             then ?z is Scalar-Quantity
    Subcategorization:
        Domain:Collection          If ?x is Total
    Value:                                 and (?x.Subcategorization) =
        Domain:Number              then slot-value (?y.Collection-Type) =
                                                        homogeneous
```

Fig. 2. This shows that *total* is a *sum* and that any argument modified by "total" must be a homogeneous collection of objects that have a scalar quantity associated with them (which can be summed). *minutes* is just such an object, so that *total minutes* is consistent. If we try to apply "total" to the argument "user", an analysis of the *user* object shows that it lacks the collection property required by *total*, so that "total user" is inconsistent.

7. Knowledge-Based Authoring Tools (KBATs)

It should be clear by now that text is rife with potential ambiguities. We will first discuss how interactive disambiguation could be done and then discuss practical difficulties and suggest some solutions.

Word sense ambiguities are the simplest, and not difficult to deal with. Consider two senses of "class", one for a class of students and the other an abstract collection. Rather than present dictionary definitions to the user, it is more efficient for a KBAT to present a constellation of terms (not necessarily synonyms) that has been found by corpus analysis to be highly related. For "class", the user would choose from two sets of terms,

$$\begin{tikzcd}
 & \text{collection, set, type, kind, superclass, subclass} \\
\text{class ??} \arrow[ur] \arrow[dr] & \\
 & \text{students, teach, classroom, course}
\end{tikzcd}$$

Structural ambiguities such as the noun-noun modifier example could be handled by the system's bringing up a dialogue box such as in Fig. 3a. The "OK" button would confirm the machine's decision, or it could be ignored or changed. A similar dialogue could be used to deal with reference ambiguities, as shown in Fig. 3b.

We have to be especially sensitive to the author's need for a free and unobstructed authoring environment. Techniques for capturing knowledge must not interfere with the flow of ideas and the tasks of organizing information that form the basis of the authoring process. But the techniques just described would be very intrusive if not implemented carefully. Imagine having to read, understand and confirm a half-dozen system queries for every sentence you typed. If some of the machine suggestions were wrong, that would be even more time-consuming.

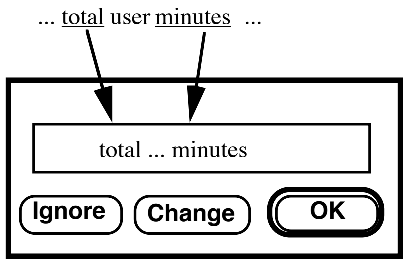


Fig. 3a. Disambiguation dialogue for noun-noun modifier. The system is suggesting that "total" modifies "minutes, not "user".

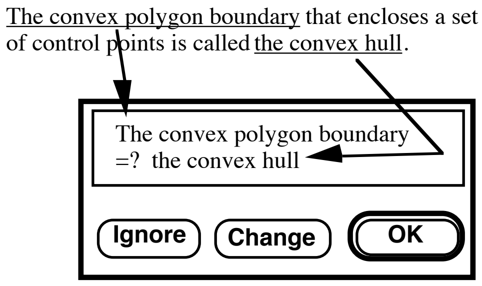

The convex polygon boundary that encloses a set of control points is called the convex hull.
The convex polygon boundary
=? the convex hull
Ignore
Change
OK

Fig. 3b. Disambiguation dialogue for a definite noun phrase reference. The system is suggesting that the hull refers to the earlier mention of a boundary.

There are two ways to escape this problem of extreme user disruption. The first is machine learning, especially generalization, and the second is the pooling of knowledge from many authors. Machine learning in a KBAT is similar in spirit, though more complex, to building a custom dictionary in a spell-checker. If you introduce a new term not in your spell-checker's dictionary, e.g., "nontemplate", then once you have confirmed the spelling, the item will be added to your dictionary and the system will never again complain about a (properly spelled) occurrence of "nontemplate". At the semantic level, when the author tells the system that "total" is an appropriate modifier for "minutes", not only will the system remember this (and not ask again) but it could generalize through the ontology to deduce the same rule for "seconds", "hours" and more.

Even if the system did learn perfectly on each occasion, it is still beyond the patience and energy of a single author to make the thousand disambiguation decisions needed for a single paper. This can be dealt with by having an author only answer a few queries for each paper they write and having the system transmit these decisions to specialized servers that would pool the decisions by tens of thousands or more authors. After the decisions are checked for consistency, they could be codified and sent back to the authors' systems on the network to augment their local semantic knowledge base. The numbers are substantial. In one year, biology authors produce over half-a-million papers. Ten answers from authors for each paper, a modest number, would result in over 5 million items of semantic information. The networked system could be particularly efficient by arranging for a query about a particular word or construct to be presented to no more than a few authors.

8. Conclusion

We have described the basic principles underlying Knowledge-Based Authoring Tools that could be used to build knowledge structures representing the contents of documents. Some of the knowledge is built automatically by the KBAT itself and other knowledge is obtained from the author while the paper is being composed. Other work can be done off-line or after the paper is completed. The primary problem in natural language analysis is disambiguation, and we have discussed the specifics of disambiguation of a variety of constructs.

We are making progress on the implementation of such systems. The problem of capturing knowledge is somewhat easier when graphics is being generated for figures in a paper. Our prototype system, GeneDraw, does this for a class of gene diagrams for biology papers. We have investigated automatic semantic clustering of words [9], and are developing a new technique based on a Balanced Entropy Principle for a variety of NLP knowledge-building tasks [10].

Acknowledgments

Work supported in part by grants from the National Science Foundation, IRI-9117030, and the Department of Energy, DE-FG02-93ER61718.

References

1. Church, K.W. and Mercer, R.L.: Introduction to the Special Issue on Computational Linguistics Using large Corpora. Computational Linguistics. **19** (1993) 1-24.
2. Hirst, G.: Semantic interpretation and the resolution of ambiguity. Cambridge University Press (1987).
3. Shieber, S.M.: An Introduction to Unification-Based Approaches to Grammar. Vol. 4. Center for the Study of Language and Information (1986).
4. Pollard, C. and Sag, I.A.: Information-based Syntax and Semantics. Vol. 1. Center for the Study of Language and Information (1987).
5. Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Knowledge Systems Laboratory, Stanford University, KSL 93-04 (1993).
6. Hafner, C., *et al.*: Creating a Knowledge Base of Biological Research Papers. In 2nd Inter'l Conf. on Intelligent Systems for Molecular Biology, Stanford, CA. AAAI Press (1994) 147-155.
7. Davis, E.: Representations of Commonsense Knowledge. Morgan Kaufman Publishers (1990).
8. Lenat, D.B.: Cyc: Toward programs with common sense. Communications of ACM. **33** (1990) 30-49.
9. Futrelle, R.P. and Gauch, S.: Experiments in syntactic and semantic classification and disambiguation using bootstrapping. In Acquisition of Lexical Knowledge from Text, Columbus, OH. Assoc. Computational Linguistics (1993) 117-127.
10. Futrelle, R.P., Zhang, X., and Sekiya, Y.: Corpus Linguistics for Establishing the Natural Language Content of Digital Library Documents. In Digital Libraries. Current Issues. Adam, N.R., *et al.*, editors. Springer-Verlag (1994) 165-180.