Parsing Natural Language into Content for Storage and Retrieval in a Content-Addressable Memory

Roland Hausser

Universität Erlangen-Nürnberg
Abteilung Computerlinguistik (CLUE)
rrh@linguistik.uni-erlangen.de

Abstract. This paper explores the possibility of applying Database Semantic (DBS) to textual databases and the WWW. The DBS model of natural language communication is designed as an artificial cognitive agent with a hearer mode, a think mode, and a speaker mode. For the application at hand, the hearer mode is used for (i) parsing language data into sets of proplets, defined as non-recursive feature structures, which are stored in a content-addressable memory called Word Bank, and (ii) for parsing the user query into a DBS schema employed for retrieval. The think mode is used to expand the primary data activated by the query schema to a wider range of relevant secondary and tertiary data. The speaker mode is used to realize the data retrieved in the natural language of the query.

Key words: NLP, DBS, Word Bank, Content-Addressable Memory

1 Four Levels of Abstraction for Storing Language Data

A written text may be electronically stored at different levels of abstraction. At the lowest level, the pages may be scanned into the computer as bitmaps. This preserves the appearance of the page (which may be important, as in a medieval manuscript), but does not allow any text processing.

At the second level, the bitmap representation of the letters is transferred automatically into a digital representation (e.g., ASCII or Unicode) by means of an OCR software. The result allows text processing, such as automatic search based on letter sequences, simultaneous substitution of letter sequences, the movement of text parts, etc.

At the third level, the digital letter sequences are enriched with a markup, for example in XML (preferably in stand-off), which characterizes chapter and/or section headings, the paragraph structure, name and address of the author, bibliography, etc., depending on the kind of text, e.g., newspaper article, novel, play, dictionary, etc. As a result, the text may be printed in different styles while maintaining the encoded text structure. Furthermore, the markup may be extended to a semantic characterization of content, for example the text's domain, thus supporting retrieval.

At the fourth level of abstraction, the text is represented as content. The content is automatically derived from the letter sequence by means of a syntactic-semantic parser. The resulting output depends on the underlying linguistic theory.

2 The Retrieval Problem

Today's search engines build their indices on the basis of significant letter sequences (words) occurring in the document texts. Though automatic and highly effective, such a second level approach has the drawback that the indexing based on significant word distributions is not as precise as required by some applications.
Consider a car dealer interested in the question of whether a university professor would be likely to drive a BMW. A search with Google (2009-12-06CET12:45:23) using the query *professor drives BMW* returns 454 000 sites, beginning as follows:

> Green Car Congress: **BMW** Developing Steam Assist **Drive** Based on ...
> —**Professor** Burkhard Göschel, BMW Board of Management. BMW designed the components of this drive system to fit in existing model series. ...

This and the next few hundred documents retrieved are not what the user had in mind.
Leaving aside how to formulate a better query, one approach to improve recall and precision is a third level XML markup conforming to RDF [2], intended for the Semantic Web. This requires skilled work from the persons posting their documents. The alternative proposed here is an automatic fourth level derivation of content based on syntactic-semantic parsing. This method improves recall and precision by coding and utilizing the grammatical relations in the sentence sequence.

3 Data Structure of Proplets

In DBS, propositional content is coded as an order-free set of flat (non-recursive) feature structures[1] called *proplets*, serving as the abstract data type. As a simple example, consider the following representation of Julia knows John. as a content:

3.1 SET OF PROPLETS CODING CONTENT

noun: Julia	verb: know	noun: John
cat: nm	cat: decl	cat: nm
fnc: know	arg: Julia John	fnc: know
prn: 625	prn: 625	prn: 625

In a proplet, the lexical and the compositional aspects of meaning are systematically distinguished. The lexical aspect is represented by the core value, e.g., know, of the core attribute specifying the part of speech, e.g., verb. The compositional aspect is represented by the continuation attribute(s), e.g., arg, and its continuation value(s), e.g., Julia John, which code the compositional semantic relations between proplets, namely functor-argument and coordination structure, intra- as well as extrapropositionally. The order-free proplets of a proposition are held together by a common prn (for proposition number) value, here 625.

[1] This is in contrast to the feature structures of HPSG or LFG, which are recursive in order to model phrase structure trees, based on the principle of substitution. Cf. [6], 3.4.5.

4 Coordinate-Addressable vs. Content-Addressable Memory

The representation of content as a set of proplets raises the question of how they should be stored. The most basic choice is between a *coordinate*-addressable and a *content*-addressable memory (cf. [3] for an overview). Though peppered with patents, the content approach is much less widely used than the coordinate approach, and employed in applications for the super-fast retrieval of fixed content.

A coordinate-addressable memory resembles a modern public library in which books can be stored wherever there is space (random access) and retrieved using a separate index (inverted file) relating a primary key (e.g., author, title, year) to its location of storage (e.g., 1365). A content-addressable memory, in contrast, is like a private library in which books with certain properties are grouped together on certain shelves, ready to be browsed without the help of a separate index. For example, at Oxford University the 2 500 volumes of Sir Thomas Bodley's library from the year 1598 are still organized according to the century and the country of their origin.

As an initial reaction to a content-addressable approach, main stream database scientists usually point out that it can be simulated by the coordinate-addressable approach, using well-established relational databases. The issue here, however, is whether the formal intuitions of the content-addressable approach can be refined naturally into an efficient retrieval method with good recall and precision.

5 Structure of a Word Bank

In DBS, the storage and retrieval of a content like 3.1 uses the letter sequence of the core values for completely determining the proplets' location:

5.1 WORD BANK STORING CONTENT 3.1

	member proplets	*position for new member proplets*	*owner proplets*
...	noun: John cat: nm fnc: ... prn: 610	noun: John cat: nm fnc: know prn: 625	core: John
...	noun: Julia cat: nm fnc: ... prn: 605	noun: Julia cat: nm fnc: know prn: 625	core: Julia
...	verb: know cat: decl arg: ... prn: 608	verb: know cat: decl arg: Julia John prn: 625	core: know

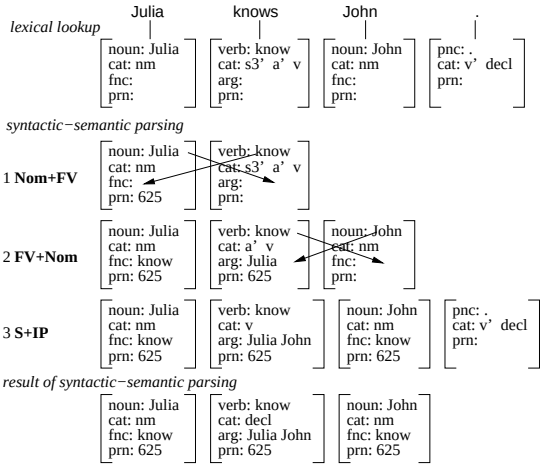
This conceptual schema resembles a classic network database with owner records and member records. It is just that the records are represented equivalently as proplets.

A sequence of member proplets followed by an owner proplet is called a *token line*. The proplets in a token line must all have the same core value and are in the temporal order of their arrival (reflected by the value of a proplet's prn attribute).
The sorting of proplets into a Word Bank is simple and mechanical. It is content-addressable in that no separate index (inverted file) is required. Instead, any incoming proplet is always stored in the penultimate position of the corresponding token line. Like the XML markup of the Semantic Web, such a DBS content representation may be added to the documents as standoff markup in accordance with the TEI guidelines.
A Word Bank is scalable (a property absent or problematic in some other content-addressable systems): the cost of insertion is constant, independent of the size of the stored data, and the cost of retrieving a specified proplet grows only logarithmically with the data size (external access) or is constant (internal access). External access to a proplet requires (i) its core and (ii) its prn value, e.g., know 625.[2] Most retrieval operations, however, require internal access, as in the navigation from one proplet to the next (e.g., 6.2). Because content is fixed, internal access may be based on pointers, resulting in a major speed advantage over the coordinate-addressable approach.

6 Cycle of Natural Language Communication

Having outlined the specifics of the content-addressable DBS memory, let us review the cognitive operations which are based on it. We begin with the cycle of natural language communication [1], consisting of the hearer mode, the think mode, and the speaker mode. Consider the following hearer mode derivation:

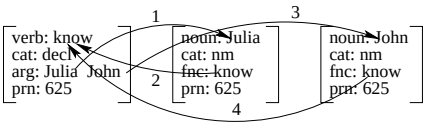
6.1 DBS HEARER-MODE DERIVATION OF Julia knows John.



[2] The token line for any core value is found by using a trie structure [4]. Finding a proplet within a token line may be based on binary search (*O(log(n))*) or interpolation (*O(log(log (n)))*), where *n* is the length of the token line. The search uses the prn value of the address in relation to the strictly linear increasing prn values of the token line. Thus, there is no need for a hash function (which is unusual compared to most other content-addressable approaches).

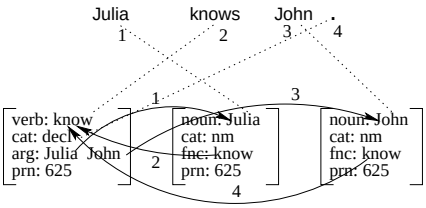
The grammatical analysis is surface compositional in that each word form is analyzed as a lexical proplet (lexical lookup). The derivation is time-linear, as shown by the stair-like addition of one lexical proplet in each new line. Each line represents a derivation step, based on the application of the specified LA-hear grammar rule, e.g., **Nom+FV**. The rules establish semantic relations by copying values (as indicated by the diagonal arrows).
The result of the derivation is an order-free set of proplets, ready to be stored in the agent's content addressable memory (as shown in 5.1). Based on the semantic relations between the stored proplets, the second step in the cycle of natural language communication is a navigation which activates content selectively in the think mode:

6.2 DBS THINK MODE NAVIGATION



Using the arg, fnc, and prn values, the navigation proceeds from the verb to the subject noun (1), back to the verb (2), to the object noun (3), and back to the verb (4).
Such a think mode navigation provides the *what to say* for language production from stored content, while the third step in the cycle of communication, i.e., the speaker mode, provides the *how to say it* in the natural language of choice:

6.3 DBS SPEAKER MODE REALIZATION



The surfaces are realized from the *goal proplet* of each navigation step, using mainly the core value. In [6], the DBS cycle of communication has been worked out in detail for more than 100 English constructions of intra- and extrapropositional functor-argument and coordination structures as well as coreference.[3]

7 Retrieving Answers to Questions

So far, the database schema of a Word Bank, i.e., ordered token lines listing connected proplets (cf. 5.1), has been shown to be suitable for (i) storage in the hearer mode (cf. 6.1) and (ii) visiting successor proplets (cf. 6.2) in the most basic kind of the think mode, with the speaker mode riding piggyback (cf. 6.3). Next we turn to another operation enabled by this database schema, namely (iii) retrieving answers to questions. This

[3] For a concise summary see [5].

operation is based on moving a query pattern along a token line until matching between the query pattern and a member proplet is successful.
Consider an agent thinking about girls. This means activating the corresponding token line, as in the following example:

7.1 EXAMPLE OF A TOKEN LINE

			member proplets	*owner proplet*
noun: girl fnc: walk mdr: young prn: 10	noun: girl fnc: sleep mdr: blond prn: 12	noun: girl fnc: eat mdr: small prn: 15	noun: girl fnc: read mdr: smart prn: 19	core: *girl*

As indicated by the fnc and mdr values of the connected proplets (member proplets), the agent happened to observe or hear about a young girl walking, a blonde girl sleeping, a small girl eating, and a smart girl reading.
For retrieval, the content proplets of a token line may be checked by using a pattern proplet as the query; a pattern proplet has one or more variables as values. Consider the following example, in which a pattern proplet representing the query Which girl walked? is applied systematically to the content proplets in the token line 7.1:

7.2 APPLYING A QUERY PATTERN

			pattern proplet	noun:*girl* fnc: walk mdr: σ prn: K
			matching?	
noun: girl fnc: walk mdr: young prn: 10	noun: girl fnc: sleep mdr: blonde prn: 12	noun: girl fnc: eat mdr: footnotesize prn: 15	noun: girl fnc: read mdr: smart prn: 19	core: *girl*

The indicated attempt at matching fails because the fnc values of the pattern proplet (i.e., walk) and of the content proplet (i.e., read) are incompatible. The same holds after moving the pattern proplet one content proplet to the left. Only after reaching the leftmost content proplet is the matching successful. Now the variable σ is bound to the value young and the variable K to the value 10. Accordingly, the answer provided to the question Which girl walked? is The young girl (walked) (cf. [6], Sect. 5.1).

8 Pattern/Proplet Matching

A set of connected pattern proplets is called a DBS schema. Matching between a schema and a content is based on the matching between the schema's order-free set of pattern proplets and the content's order-free set of content proplets. Matching between an individual pattern proplet and a corresponding content proplet is based in turn on their non-recursive (flat) feature structures. Consider the following example:

8.1 APPLYING A SCHEMA TO A CONTENT

schema level	noun: α cat: nm fnc: know prn: K	verb: know cat: decl arg: α β prn: K	noun: β cat: nm fnc: know prn: K	*where* α ε *{Julia, Suzy, ...}* *and* β ε *{John, Mary, Bill, ...}*
	matching and binding			
content level	noun: Julia cat: nm fnc: know prn: 625	verb: know cat: decl arg: Julia John prn: 625	noun: John cat: nm fnc: know prn: 625	

For example, in the first pair of a pattern proplet and a content proplet, matching is successful (i) because they share the same set of attributes, and (ii) because the value *Julia* satisfies the restriction on the variable α. The simplicity of pattern/proplet matching is supplemented by the efficiency of finding a proplet or a set of proplets in the content-addressable memory of a Word Bank (cf. Sect. 5). For example, the yield of the schema in 8.1 is determined exactly by (i) accessing the token line of *know* (cf. 5.1), and (ii) by using the arg and the prn values of each proplet found there to access all and only propositions in which someone knows someone – resulting in practically[4] perfect recall and precision at very high speed.

9 Discussion

This brief outline of storing and retrieving level four language data in DBS raises the following questions. First, is it practically feasible to automatically parse large amounts of natural language into representations of content like 3.1? In this respect, DBS is in the same boat as competing approaches to representing content, such as truth-conditional semantics, phrase-structure analysis, and their combination. In response we point to the continuous expansion of functional completeness and data coverage in DBS, applied to a wide range of grammatical constructions and to some very different natural languages, such as Chinese, Russian, and Tagalog in addition to English, German, Bulgarian, etc.
Second, how does DBS compare with the other systems in terms of storage and retrieval? In truth-conditional semantics, content 3.1 would be represented as follows:

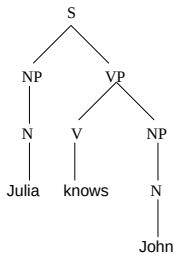
9.1 CONTENT 3.1 AS LOGICAL FORMULA

$$know'_{(e\backslash(e/t))}\ (Julia'_{(e)},\ John'_{(e)})$$

This Montague-style formula characterizes functor-argument structure by means of complex categories subscripted to the items know', Julia' and John'. In contrast to 3.1, which codes content as an (order-free) set of proplets at the word level, formula 9.1 applies to the sentence (proposition) level. This is because order within the formula cannot be changed without destroying either wellformedness or the original meaning. Consequently, 9.1 must be stored as a whole, which raises the question of what the primary key should be. For this, the sentence level has no obvious answer.
Similarly for content represented as a phrase or a dependency structure:

[4] Recall and precision are defined in terms of subjective user satisfaction. Cf. [8].

9.2 CONTENT 3.1 AS PHRASE STRUCTURE



This two-dimensional representation defined in terms of the dominance and precedence of nodes represents the whole sentence structure as one unit – again raising the paradoxical question of what the primary key should be, for example for storage and retrieval in a tree-bank.

Conclusion

The DBS approach to practical applications of natural language processing is based on solving the most important theoretical question first. This is the question of how the mechanism of natural language communication works. It is answered in DBS by modeling the cycle of natural language communication in the form of an artificial agent with interfaces for recognition and action, and a hearer, a think, and speaker mode.
The application discussed in this paper is the storage and retrieval of language data in a textual database. For this the three most relevant properties of the overall system are (i) the efficiency of retrieval based on the pointers in a content-addressable memory, (ii) the semantic relations between proplets, implemented as the pointers on which most of the retrieval is based, and (iii) the ease of turning content into DBS schemata which provide for effective database querying based on pattern matching.

References

1. Hausser, R.: Database Semantics for natural language. In: Artificial Intelligence, vol. 130, issue. 1, pp. 27–74. Elsevier (2001)
2. Berners-Lee, T., J. Hendler, & O. Lassila: The Semantic Web. In: Scientific American, vol. 284, issue. 5, pp. 34-43 (2001)
3. Chisvin, L., and R. J. Duckworth. Content-Addressable and Associative Memory. In: Yovits, M.C. (ed.) Advances in Computer Science, pp. 159–235. Academic Press (1992)
4. Fredkin, E.: Trie Memory. In: Communication of the ACM, vol. 3, issue. 9, pp. 490–499 (1960)
5. Hausser, R.: Modeling Natural Language Communication in Database Semantics. In: Proceedings of the APCCM, vol. 96, pp. 17–26. ACS, CIPRIT, Wellington, New Zealand (2009)
6. Hausser, R.: A Computational Model of Natural Language Communication. Springer, Berlin, Heidelberg, New York (2006)
7. Quillian, M.: Semantic memory. In: M. Minsky (ed.), Semantic Information Processing, pp. 227–270. MIT Press, Cambridge, MA (1968)
8. Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley Longman Publishing Co., Inc., Boston, MA (1989)