3-14-1989

Natural Language Processing Applied to a Data Base of Engineering Design Documents

Larry E. Wood
Randy L. Orr
Alan R. Parkinson

Follow this and additional works at: https://scholarsarchive.byu.edu/dlls

BYU ScholarsArchive Citation
Available at: https://scholarsarchive.byu.edu/dlls/vol15/iss1/17

This Article is brought to you for free and open access by the All Journals at BYU ScholarsArchive. It has been accepted for inclusion in Deseret Language and Linguistic Society Symposium by an authorized editor of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen amatangelo@byu.edu.
Abstract

The paper describes an application of Roger Schank's approach to natural language processing by computer, called Conceptual Analysis, which focuses on the underlying conceptual representation of an event. By definition, an event involves an action, an actor, an object, and a direction to which the action is oriented, plus other aspects that might be relevant (e.g., time, place, etc.). The particular problem to which Conceptual Analysis has been applied is a database of engineering design documents. A prototype system was developed to analyze functional descriptions of design documents, allowing it to recognize that components of different designs perform similar functions.

Introduction

This research describes a prototype intelligent system for the storage and retrieval of function-oriented descriptions of engineering design documents. While functional concepts have potential utility to the design process, they pose some difficulty for storage and retrieval systems because a functional description requires a phrase or sentence and the same concept can be described in a variety of ways. For example, consider the phrases "increase speed" and "amplify rate of motion." These phrases are composed of different words but they imply the same underlying concept of "varying" the magnitude of an attribute (velocity) of an object.

What is needed to store and retrieve a design function description is a natural text processing scheme that can "understand" the underlying concept that a specific function represents, so that designs performing similar functions can be recognized. Such a scheme, referred to as Conceptual Analysis (Cullingford, 1986) was used to develop a prototype system for storage and retrieval of functional descriptions of design documents. The system is called FORRDD for Function-Oriented Representation and Retrieval of Design Documents.

Conceptual Analysis

Conceptual Analysis is based on Conceptual Dependency theory developed by Roger Schank (e.g., Schank, 1973), a method of natural language processing based primarily on semantics rather than syntax. It's foundation is the notion of an underlying conceptual representation. Schank proposed three types of elemental concepts: nominals, actions and modifiers. Such things as a man, a duck, a book and a pen are nominals. An action is what a an animate object does to another object. Some examples are
"John hit Bill", where "hit" is the action or "Mary took the book", where "took" is the action. **Modifiers** are such things as "red" in the phrase "the red ball" or "slowly" in the sentence "He approached slowly".

In order to capture the underlying conceptual representation of an event, Conceptual Dependency focuses on the action being performed. Schank claims that all words describing actions can be mapped onto a relatively few primitives. Each primitive has its own concept frame containing "slots" that can only be filled by specific nominals or modifiers. It assumes that a statement refers to some type of an event having the following aspects (at least):

- an ACTOR
- an ACTION performed by that actor
- an OBJECT that the action is performed upon
- a DIRECTION in which that action is oriented

By definition, an event contains all of the above elements and if not explicitly mentioned in the sentence, a hearer/reader is caused to infer actors, objects etc. One of the key functions of the system is slot filling. Initially empty slots are filled by items obtained from an analysis of the sentence or from "world knowledge" previously entered into a knowledge base.

One of the primitives proposed by Schank, called ATRANS, refers to actions related to transfer of possession. It encompasses such verbs as "give", "take", "buy", "sell", "rob", and "borrow". An example of a sentence involving the primitive ATRANS would be "John gave Mary a book." A conceptual analysis program based on Conceptual Dependency analyzes such a sentence from left to right, searching for elements such as the primitive action, its actor, an object transferred, and the direction of transfer. The resulting analysis fills in the initially empty conceptual frame producing a meaning structure such as the following:

```
ATRANS  Actor (Person Name (John))
         Object (book)
         To (Person Name (Mary))
         From (Person Name (John))
```

As indicated earlier, the FORRDD system actually uses a conceptual analysis system based on work by Richard Cullingford (Cullingford, 1986), which is an outgrowth of Conceptual Dependency. In addition to characteristics of Conceptual Dependency discussed above, Cullingford's system imposes a basic classification scheme on common sense knowledge, which he refers to as Eclectic Representation for Knowledge Structures (ERKS).

The ERKS system consists of entities, acts, and relationships which are subdivided into a number of primitive types of conceptual structures. These types are organized into a simple inheritance hierarchy (sometimes called a ISA hierarchy, because items share or inherit attributes from items above them in the hierarchy). Thus, when we say "a person ISA animate object," what we mean is that certain things true of animate objects are true of people.

152
Imposing such an organization of primitive classes provides a standard, useful means of controlling the inference processes that are concerned with reasoning about primitive classes. Problem-solving processes trying to determine, for example, whether a book is a suitable "object" of a giving action, can test if "book" is a member of the class of movable or transferrable objects. Once primitive classes have been developed for a particular domain of analysis, then words that will appear in the sentences and/or phrases to be analyzed must be defined for the system. The word definitions contain two types of information: (1) associations between the word and specific conceptual structures and (2) expectations or predictions about the other concepts in the context surrounding the word as it is used in a sentence.

**Functional Design Concepts**

As was discussed earlier, an important aspect of Conceptual Analysis is the development of a set of primitive concepts which capture the underlying meaning of a variety of similar language descriptions. For storage and retrieval of design documents, the FORRDD system was inspired by the hypothesis that functional characteristics or concepts would be useful because they are independent of particular implementations (see Pahl & Beitz, 1984). Thus, in a company that manufactures both hand tools and small appliances, a designer might not recognize close similarities in terms of physical characteristics. However, there likely could be components in a small appliance that perform functions similar to those in a hand tool, and these would be made available to a designer through a retrieval system based on functional concepts.

Pahl and Beitz (1984) propose that the overall function of a design expresses the relationship between system inputs and system outputs, which can be reduced to three forms of matter in the case of electro-mechanical designs: **energy**, **material** and **signals**. The functional relationships between inputs and outputs can then be conceptualized as the conversion in various ways of these three forms of matter.

Another important aspect of the work of Pahl and Beitz (1984) is their notion of establishing function structures during the design process. This consists of dividing the overall function of a design into sub-functions recursively until a point is reached where a sub-function cannot be sub-divided further and still remain generally applicable. Pahl and Beitz refer to functions at this level as **generally valid functions**. They propose the following set of generally valid functions: **CHANGE**, **VARY**, **CONNECT**, **CHANNEL** and **STORE**.

**CHANGE** involves the transformation of one type of matter to another. **VARY** applies to changes in the magnitude of a particular type of matter. **CONNECT** refers to conversions resulting in differing numbers of inputs and outputs. **CHANNEL** designates changes in location or place. **STORE** refers to time-dependent functions. The mapping of design functions to generally valid functions will be illustrated with some examples from an electric can opener. The overall function "Remove the lid from a can" could map to a CONNECT with one input (the can) and two outputs (the can and the lid). The function of a gearbox assembly to
"Transfer torque from a worm gear to a drive shaft," could map to a CHANNEL. Finally, an electric motor with a function to "Transform electrical energy into mechanical energy" could map to a CHANGE.

**Conceptual Analysis of Functional Design Concepts**

As was discussed earlier, an important aspect of Conceptual Analysis is the development of a set of primitive concepts which capture the underlying meaning of a variety of similar language descriptions. In the FORRDD system, the five generally valid functions proposed by Pahl and Beitz (1984) were used to fill that role. First a conceptual structure was developed for each generally valid function, establishing the various slots that would be required to accomplish its task. For example, because CHANNEL pertains to changing the location of some type of matter, it was necessary to provide slots for the item to be relocated (Channellee), the input location (InLoc), and the output location (OutLoc). An additional slot (Actor) was also added to accommodate the name of an assembly that performed the function in question. This results in a conceptual structure such as the following:

```
(Channel Actor ( )
   Channellee ( )
   InLoc ( )
   OutLoc ( )
```

Following the development of conceptual structures, a domain of designs was chosen for developing a prototype system. In this case the domain of small electrical appliances was chosen, consisting of the following items: a can opener, a hand mixer, a knife sharpener, a toaster, and a coffee maker. For each of these, sub-assemblies were described including such components as gear boxes, electric motors, shafts, gears, filters, heating elements, etc.

A set of functional descriptions of the items, assemblies, sub-assemblies, and components in the domain were then produced, thereby establishing the entities which the system would have to account for and the vocabulary that would be required to analyze the descriptions. Next, an ERKS hierarchy of the types of objects (e.g., gears), and entities (e.g., types of matter, energy, and material) was constructed. Finally, definitions for all the words contained in the set of functional descriptions was developed. The verbs are listed in table 1 and the nouns and adjectives are shown in table 2. A sample description, "A drive shaft transfers torque from a worm gear to a drive gear", will be used to illustrate this process in a simplified form. First a definition was derived for "drive shaft" by associating it
Table 1 -- Verbs (indented words are synonyms) currently defined in the FORRDD system

<table>
<thead>
<tr>
<th>transfer</th>
<th>support</th>
<th>turn</th>
<th>transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>transpose</td>
<td>connect</td>
<td>move</td>
<td>regulate</td>
</tr>
<tr>
<td>direct</td>
<td>mount</td>
<td>lower</td>
<td>control</td>
</tr>
<tr>
<td>shift</td>
<td>mix</td>
<td>lift</td>
<td>vary</td>
</tr>
<tr>
<td>convey</td>
<td>pierce</td>
<td>raise</td>
<td>decrease</td>
</tr>
<tr>
<td>transmit</td>
<td>sheer</td>
<td>remove</td>
<td>reduce</td>
</tr>
<tr>
<td>channel</td>
<td>penetrate</td>
<td>release</td>
<td>increase</td>
</tr>
<tr>
<td>conduct</td>
<td>cut</td>
<td>store</td>
<td>amplify</td>
</tr>
<tr>
<td>guide</td>
<td>hold</td>
<td>generate</td>
<td></td>
</tr>
<tr>
<td>deliver</td>
<td>clamp</td>
<td>create</td>
<td></td>
</tr>
<tr>
<td>receive</td>
<td>secure</td>
<td>indicate</td>
<td></td>
</tr>
<tr>
<td>locate</td>
<td>rotate</td>
<td>convert</td>
<td></td>
</tr>
</tbody>
</table>

with a simple conceptual structure that specified it to be an assembly which is a type of material, which is a type of matter, which is a type of physical object. When the system encounters the term "drive shaft" the following conceptual structure is simply added to a concept list: ( Assembly Type (Driveshaft)).

The word "transfer" was mapped to the primitive CHANNELL as described above. When "transfer" is read by the system, the first thing that happens is that an empty transfer concept is added to the list as follows:

(Channel Actor
Channellee
InLoc
OutLoc )

As might be expected, the action words "do the most work" and therefore have the most complex definitions. In addition to associating "transfer" with a CHANNELL structure and adding it to the concept list, its definition also must activate expectations (or requests as they are called) for concepts that can fill the four slots in the CHANNEL structure. The request to fill the Actor slot looks for something already on the concept list that could be an assembly, will find the concept created for "drive shaft", and will fill the Actor slot with it. The other requests must wait for additional concepts to be created as the remaining words are processed.

The definitions for "torque", "worm gear", and "drive gear" are similar to the one for "drive shaft" in that each is associated with a simple conceptual structure. "Torque" is associated with a type of motion (rotation) which is a type of energy which is a type of matter. The request to fill the Channellee slot searches for a type of energy, and uses the concept added by "torque" to do so. "worm gear" and "drive gear" are both types of assemblies as was "drive shaft". The requests for the InLoc and OutLoc slots search for types of assemblies following the
Table 2. Nouns and Adjectives (indented words indicate sub-types) currently defined in the FORRDD system

<table>
<thead>
<tr>
<th>signal</th>
<th>rotational</th>
<th>motor</th>
<th>filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>fluid</td>
<td>torque</td>
<td>filter</td>
<td>housing</td>
</tr>
<tr>
<td>water</td>
<td>translation</td>
<td>worm gear</td>
<td>motor</td>
</tr>
<tr>
<td>coffee</td>
<td>linear</td>
<td>worm gear</td>
<td>flow</td>
</tr>
<tr>
<td>matter</td>
<td>oscillation</td>
<td>shaft</td>
<td>regulator</td>
</tr>
<tr>
<td>input</td>
<td>counter-rotation</td>
<td>drive gear</td>
<td>temperature</td>
</tr>
<tr>
<td>output</td>
<td>rotation</td>
<td>supporter</td>
<td>sensor</td>
</tr>
<tr>
<td>energy</td>
<td>mechanical</td>
<td>shaft</td>
<td>sub-assembly</td>
</tr>
</tbody>
</table>

power friction spring unit
force material drive shaft rpm
electricity can bushing mph
current lid knife ft/s
electrical food knife blade lb
potential lubrication holder hz
kinetic appliance beater inch oz
heat can opener heat element ft lb
temperature knife assembly location
hydro sharpener gearbox direction
flowrate coffee maker can opener orthogonal
motion mixer cutter plane
speed toaster electric knife
rotation-circular component toaster

prepositions "from" and "to", respectively. Thus when the two prepositional phrases are encountered, the concepts are added to the concept list and promptly used to fill the InLoc and OutLoc slots. Thus, the analysis results in the following meaning structure:

(Channel Actor (Component Type (Driveshaft))
 Channellee (Motion Type (Rotation))
 InLoc (Component Type (Wormgear))
 OutLoc (Component Type (Drivegear)))

As an indication of the versatility of the Conceptual Analysis system, the definitions and ERKS types discussed above would be adequate for the following collection of phrases and sentences:

(a) A drive shaft transfers torque from a worm gear to a drive gear.
(b) A drive shaft transfers torque
(c) A drive shaft transfers.
(d) Torque is transferred by a drive shaft to a drive gear from a worm gear
(e) Torque is transferred from a worm gear to a drive gear
(f) Torque is transferred.
(g) Transfer.
Once all the ERKS hierarchy and all the necessary word definitions were
developed, the storage and retrieval system was developed. That aspect of the
system will now be described.

**Storage and Retrieval System Description**

Storage of functional design concepts involves two basic procedures. The first is
the creation of a full text file of a design document, including specifications and any
information about assemblies and sub-assemblies that might be desired. The
second is the entering of functional descriptions of the assemblies and sub-
assemblies described in the design documentation into a database of descriptions
and linking them to the design document file. In the current implementation, the
document file can be created by typing the information directly the system or by a
file transfer from another system.

Entering functional descriptions of assemblies into the database is accomplished
by following system prompts. First, the system prompts for the name of the
document file to which the functional descriptions apply. This would typically be
the name of the object (appliance, sub-assembly, or component) which the
document describes. Next, the system prompts for the functional description of an
assembly such as the example discussed earlier, “The drive shaft transfers torque
from the worm gear to the drive gear.” The system then analyzes the description,
produces a meaning structure of the type discussed earlier, adds the name of the
file to a "PartOf" slot in the structure, and adds the structure to the database of
meaning structures. Additional descriptions are added as needed in this manner
until all assemblies and sub-assemblies in the document have been entered.

Retrieval of document information is then accomplished in a similar manner to
entry, except that the user is prompted for a general functional description not
including the name of an assembly, e.g., “Transfers torque from the worm gear to
the drive gear.” The system analyzes the description and searches the database
for one that matches. If a match occurs, the system uses the assembly name
(stored in the Actor slot) and the file name (stored in the PartOf slot) and displays a
message such as, “That function is performed by a drive shaft which is part of a
Can Opener.” The user can then review the relevant file containing the design
document.

Currently, the matching algorithm will produce a match to a meaning structure
(based on the same primitive function) that is at the same level of specificity or
more specific than the query,. In the example given, the user could omit either
"from the worm gear", "to the drive gear", or both. The less specific the query, the
more likely it will result in a match, as with most database queries. Results of
queries varying from general to specific are shown in Table 3.
Table 3 -- Retrieval Examples

1. A query consisting only of the word "transfer", will result in 25 items such as: (a) housings for each of the appliances supporting sub-assemblies (transfer of supporting energy), (b) gears transferring mechanical energy from a source to a destination, (c) a flow regulator transferring signals to regulate the flow, and (d) heating element transferring heat to water or to bread.

2. A query consisting of the phrase "transfer energy", results in 15 items, now restricted to housings, gears, drives shafts, etc.

3. A query consisting of the phrase "transfer torque" (a particular type of energy), results in 8 items, now restricted to drive shafts and gears.

4. If you ask for things that "transfer torque of 45 in. oz.", results in 1 item, a drive shaft that is part of a can opener.

Conclusions

The objective of this research was to develop a prototype system that applies natural language processing to design classification for the purpose of storing and retrieving designs by their functional concepts. The prototype has thus far demonstrated the feasibility of a retrieval system based on functional design descriptions. However, it has yet to be demonstrated that such descriptions will be useful to designers. According to French (1988), functional design has not been studied extensively, and there is not yet much evidence that engineers think explicitly in terms of function during the design process.

While the development of FORRDD is centered around functional design concepts, there are certainly other design concepts which might be useful for retrieval purposes (e.g., physical specifications, maintainability, reliability). Further work in this area is needed. Indeed, Wasserman and Lebowitz (1983) have reported a system, called RESEARCHER, which uses conceptual dependency to analyze patent descriptions in terms of physical characteristics and relationships.

In addition to those discussed, there are several considerations that simply need evaluation by testing the system and the principles behind it on designers in a realistic setting. We are exploring various means by which such an evaluation might be accomplished.
References


Notes

1. The research described here is based on work performed by the second as part of the requirements of an M.S. degree in engineering technology.