Acquiring and Representing Background Knowledge for a Natural Language Processing System

Andrei Mikheev and Marc Moens

HCRC Language Technology Group
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, Scotland, UK
E-mail: Andrei.Mikheev@ed.ac.uk

Abstract

In this paper we describe a methodology for acquiring, structuring and encoding background knowledge for natural language processing. The methodology has been used for the development of a knowledge base for a system which processes patient discharge summaries in a restricted medical domain. On the basis of some example texts and system output, we will illustrate how acquired and represented background knowledge is used in processing to yield the desired interpretations.

1 Introduction

We are currently involved in the development of a system for the processing of patient discharge summaries (PDSs)—letters sent by a hospital to a patient's own doctor. The PDSs we are working on are all in the area of heart disorders. Here is a typical example:

Dear Dr Grippo,
This 62 year old patient was admitted for a spontaneous and recurrent angina pectoris. In January 1989, he developed an antero-septal myocardial infarction diagnosed by exercise stress test. The MI *was treated with a standard angioplasty of a proximal* LAD *stenosis. The patient has been asymptomatic since.*
During a hospitalisation in Saint Epistemi, an exercise testing was performed. The coronary arteriography shows two severe stenoses at the origin of the dominant right coronary artery and of one marginal artery. The left ventricle function is impaired with an anterior akinesia.
With the family history of coronary disease and some elevation of lipids, he should keep to a prudent diet.

Because of the nature of the system for which we are building the natural language front end, rather deep processing of these texts is needed, involving the resolution of various referential expressions, metonymy, implicit assumptions, etc.

For example, in the first sentence the system has to work out that it is the 62-year old patient who has the disease called angina pectoris. Compare the structurally equivalent *This patient was admitted for a treadmill test*, where the *for*-adverbial expresses the purpose of the hospital admission, rather than a disease. In the second sentence, the system needs to infer that the exercise stress test was carried out on the patient mentioned in the first sentence, that the result of the stress test is a diagnosis of myocardial infarction, and that this is a condition of the aforementioned patient. In the third sentence, the system has to infer that it was on this 62-year old patient who underwent an angioplasty, that the angioplasty was performed in or soon after January 1989, that the angioplasty was performed on an artery known as a proximal left anterior descending vessel (LAD), and that the result of the angioplasty was that the symptoms of the myocardial infarction disappeared.

Apart from considerable natural language expertise, this level of understanding also requires the system to be endowed with a lot of specialized domain knowledge as well as general knowledge. The process of acquiring this background knowledge and representing it is a time-consuming one about which little is known and for which hardly any tools or techniques have been developed. There is a considerable literature on the acquisition of knowledge from experts for the construction of expert systems, but the process of acquiring information for the construction of knowledge-based natural language processing systems has remained relatively unexplored.

Jerry Hobbs is a notable exception: in various papers, he has outlined a methodology for acquiring and structuring knowledge for use in message understanding tasks. In Hobbs (1986), he outlined a 3-step methodology, exemplified in Bech (1989) and extended in Hobbs and Navarretta (1993). Our goal was to examine the extent to which this methodology could be automated, and to develop computational tools to assist linguists and knowledge engineers in applying this methodology when building knowledge bases for natural language processing applications.

But an examination of Hobbs' work revealed that a number of the steps in his methodology were difficult to automate. Some of its recommendations seemed too general or too implicit. This is not surprising: Hobbs, as an expert on knowledge acquisition and knowledge structuring, was trying to make explicit the methodology he adopts. But as we know from studies on knowledge extraction from experts, what is obvious to experts very

often is not all that obvious to others, and methodological steps which appear obvious and explicit to experts often hide many implicit assumptions.

In an attempt to make this methodology more explicit, and building on an analysis of Hobbs' work, we have developed a new methodology and are implementing a set of tools for acquiring and structuring background knowledge for natural language processing applications. The main differences with Hobbs' methodology are that we decided that the knowledge acquisition methodology should be concept-oriented, i.e. guided by the structure of the domain, rather than by the lexical semantics of the words occurring in the text. Through practical applications of our methodology we also realised that the knowledge acquisition work has to take as its starting point large collections of texts rather than just prototypical fragments. This in turn requires robust corpus handling tools. And finally, we decided that declarative knowledge representation is not sufficient for many text understanding tasks, and that a representation that mixes different languages allows for greater expressiveness with increased processing efficiency.

In this paper we will outline the knowledge acquisition methodology we developed, concentrating on the resulting representations and how these are used in text interpretation.

2 The methodology

The main idea behind the proposed methodology is to allow the knowledge engineer or computational linguist to provide a conceptualization of expressions in the sublanguage in terms of the conceptual structure of the underlying domain. The domain terminology can then be mapped virtually unambiguously into the conceptual types. These types create the context in which other words can be defined as well. This in essence removes the distinction between semantic representation and world knowledge.

The main principles of the proposed methodology can be summarized as follows:

Concept-oriented approach: Our primary concern is to arrive at a conceptualization of the domain rather than a general semantics of the words in the studied corpus. Words are mapped into conceptual structures and can be organized into groups of local synonymy.

From a knowledge engineering point of view, there is a crucial difference with more standard approaches: instead of determining the possible meanings for a word, we start from a structured concept-oriented representation of the domain and determine which words can express which concepts. There is no difference between the two approaches during the processing of the text. But during the knowledge engineering phase this concept-oriented approach provides much more guidance for the knowledge engineer or computational linguist in defining the meanings of words.

Sublanguage approach: The corpus is decomposed according to the subparts of the conceptual domain they talk about; each of the corresponding subparts of the corpus can be characterized by its own sublanguage. For each sublanguage specific categories are defined. Domain-specific pragmatic knowledge and specific features of a corresponding sublanguage narrow the search space during text processing.

Structured knowledge representation: An object-oriented analysis is used for the characterization of concepts. It relies on the following assumptions (more details on these assumptions can be found in Bech *et al* 1993):

- use different conceptualizations for different areas of reasoning. For example, the conceptualization of scales is better done in a procedural model than in a declarative one.
- distinguish different semantic categories (entities, properties, roles, values, relations, etc.) and use different techniques for their conceptualization.
- distinguish extensional and intensional concepts: *e-concepts* do not have definitions and are recognized by their denoting sets; *i-concepts* have definitions and are recognized as specializations of the e-concepts.

The investment needed to perform the acquisition and structuring of domain knowledge in this way pays off at the level of processing, allowing for more robustness, due to a reduction of ambiguities, and the possibility of stating pragmatically inspired recovery procedures to handle various types of syntactically ill-formed input.

The first step in the methodology is to provide a structural description of the underlying conceptual domain of the texts (section 4). The next step is to arrange these conceptual structures into temporal-causal schemata with pre-packaged inferences which are performed on the basis of an analysis of questions solicited from the intended end-users of the system (section 5). The final step of the proposed methodology is a characterization of so-called commonsense words and phrases in the established domain framework (section 6). The net result of these steps is a characterization of not only lexical entries but also cognitive schemata of the domain.

But before we embark on a more detailed description of the results achieved with each of these steps, we first turn to a more general matter.

3 Knowledge representation framework

It is commonly accepted (e.g. Lenat & Brown 1984) that the choice of representation formalism is of critical importance for getting useful results in the knowledge discovery process. So before embarking on a description of our methodology and its application for building a real system we first outline the knowledge representation framework we use. This framework consists of the choice of the formalism (section 3.1) and domain-independent ontological postulates (section 3.2).

3.1 Representation formalism

The knowledge formalism used in our implementation is a conceptual graphs notation akin to Sowa's Conceptual Graphs (Sowa 1984). This does not mean that we want to argue that the knowledge engineering process can only be carried out by means of this formalism. But conceptual graphs do provide a useful notation formalism for the conceptual modeling aspect of knowledge engineering. It has also been argued to be a more perspicuous notation formalism for domain experts who are not trained logicians. This is important when a computational linguist is building a knowledge base for a particular natural language processing application and the contents of this knowledge base needs to be checked by domain experts.

A conceptual graph consists of concepts linked to each other by means of conceptual relations. A concept node is represented in square brackets: [CONCEPT]. A relation node is represented in round brackets: →(relation)→. A concept node includes two fields: a type field and a referential field. They are separated by means of a colon. The following examples clarify this usage:

[PERSON]—this is a concept of type person. By default it is assigned a singular distributed denotation, corresponding to its usage in the indefinite "a person".

[PERSON: x]—the variable x in the referential field fixes the distributed denotation, and it is now possible to make references to "a particular person x".

[PERSON: *x]—this is a reference to an already introduced concept [PERSON: x].

[PERSON: #]—"#" indicates a discourse link. It says that this is a person who has already been introduced in the discourse but whose reference is not known yet. This corresponds to anaphors (*he, she,...*) or definite expressions ("the person").

[PERSON: {*}]—stands for plural denotation, "persons".

In addition to this basic vocabulary, we introduced new syntactic constructions to distinguish concepts and words: every concept has the "@" prefix; words are always in quotes.

3.2 Ontological postulates

A knowledge representation machinery like the one introduced in the previous section allows for knowledge to be represented, but it doesn't provide guidelines on how to represent things in the "right" way and in a consistent manner. To provide the starting point for knowledge engineers or computational linguists using our toolset we have established the following ontological postulates:

1. Everything that can be quantified over, measured or referred back to is granted the concept status.
2. There are two different concept categories: discrete and quantitative. While discrete concepts are instantiated into individuals, quantitative concepts are instantiated into points in a dimension or into intervals.
3. There are three main types of concepts: *entities*, *properties* and *roles* (cf. Guarino 1991).

Entities are semantically rigid (they preserve their reference in every situative framework) and essentially independent (they can be referred to directly). Entities have the following subcategorization:

entity →
- **stative**
  - → **object** - *discrete individual*
  - → **group** - *aggregation of objects*
  - → **substance** - *mass*
- → **durational**
  - → **event** - *discrete individual*
  - → **process** - *aggregation of events*
  - → **state** - *mass*
- → **quantitative**
  - → **point** - *discrete individual*
  - → **path** - *aggregation of points*
  - → **dimension** - *mass*

Properties are concepts which can be instantiated as values from a corresponding set of values. They are semantically rigid and founded on entities (i.e. recognized only in the presence of some entity). As with entities we can distinguish two kinds of properties: discrete and quantitative (scalar). Discrete properties are instantiated into members of their value set. Scalar properties are instantiated into AMOUNTS.

Roles are types which are semantically non-rigid. They are founded on rigid types. For example, the role type DOCTOR is founded on the rigid type HUMAN and exists only in the framework of professional relations.

Finally, *relations* correspond to the organisational properties of concepts and map concepts into concepts. Unlike concepts relations are characterized by their inferential properties. The most important of these relations are:

is-a: a relation of subsumption. We distinguish subsumption of an instance (discrete or quantitative) by its type and subsumption of a type by its supertype (including subsumption of a role-type by its host type and a value by its property).

char: relates an entity type with a compatible property. Consider:

[@disease:]→(char)→[@disease-degree:∀]

This says that the concept @disease has the property @disease-degree.

has: relation of structural inclusion. There can be many different types of inclusion: the most popular ones are *has-component*, *has-part* and *has-area* for physical objects, *has-subevent*, *has-θ-role* for eventualities and *in*, *at* etc. for quantitative entities (time, space).

Objects are traditionally represented as a collection of compatible properties and components. Properties are represented as value sets (or scales) and are linked with types they can be applied to. For example, the earlier mentioned property @disease-degree can be defined as having @low as a possible value, expressed in this domain by means of words like "some" and "mild":

[@disease-degree:∀] =
@low = {"some", "mild"};

@mod = { "moderate", "moderately severe"};
@severe = {"quite severe", "severe"};
@high = { "high grade", "critical", "tight", "heavy"}

Eventualities are represented as a structure of their thematic roles, locative, temporal and causative relations. In our analysis of eventualities we follow Parsons (1990) in distinguishing two kinds of eventualities: events, which culminate in time, and states, which hold. The following conceptual structure can be inherited by all @durational-hold category subclasses:

```
[@durational-hold:∀]
  →(hold)→[@time-interval]
            →(int-from)→[@time-point:tf]
            →(int-to)→[@time-point:tt]
  →(start)→[@time-point:*tf]
  →(cul)→[@time-point:*tt]
```

This says that concepts of type @durational-hold start at a time *tf* and culminate at a time *tt* and hold over the interval in between these two time points.

Apart from representing stative features of an eventuality it is important to provide it with its causative-inchoative conceptualization if possible. For example, in some domains it may be useful to represent the fact that an event of "killing x" can cause an event of "x dying", which results in a consequent state of "x being dead":

```
[@event: e <> TV] - transitive verb, e.g. "to kill"
  →(theme)→[entity:x]
  →(cul)→[@time-point:t]
  →(cause)→[@event:e1 <> IV] - intransitive verb, e.g. "to die"
            →(theme)→[entity:*x]
            →(cul)→[@time-point:t1]
            →(become)→[@state:s <> Adj] - adj., e.g.: "dead"
                      →(exprs)→[entity:*x]
                      →(hold)→[@time-interval:ti]
```

It is worth mentioning that when an event is referred to by a noun or gerund it is translated in exactly the same way as when it is referred to by a verb.

4 Step 1: Domain Structuring

The first step in the proposed methodology is the structuring of the underlying domain. The aim of this step is to establish domain categories and nomenclature and to provide a mapping of the sublanguage terminology into these concepts; we have already given some examples of this in the previous section—e.g. the different degrees a disease can have, and the way English words in this corpus map onto these different degrees. The methodology for acquiring this knowledge is described in more detail in section 4.1. The other important goals of this step are to specify structural relations among these categories, to provide definitions for i-concepts, and to determine rules for type composition. This is described in section 4.2. We then describe some of the tools under development to assist in this step of the methodology (section 4.3), and we give some examples of the text interpretation that can alread be achieved with the knowledge structures that result from this step in the methodology (section 4.4).

4.1 Domain categories and nomenclature

The first task in the Domain Structuring phase is to establish basic domain types and their subtypes—the nomenclature. In a restricted domain corpus these types and nomenclature are referred to by a terminology which can cover up to 70% of the lexicon.

This task consists of two activities: extracting the domain terminology (basic domain vocabulary) and arranging this vocabulary into a subsumption hierarchy with basic domain categories at the top. For example:

```
"heart disease"
        →  "infarction".
                  →  "myocardial infarction",
        →  "arteritis"                "MI"
```

The domain vocabulary is an extensive list of lexical items occurring in the corpus and corresponding to domain specific notions. The main characteristic of this vocabulary is its orientation on the sublanguage of the domain. A lexical entry can be a word (*"admit"*), a phrase ({*cardiac catheterisation*}) or a pattern ({*date*~{{*day*~}{*month*~}{*year*~}}}). During this stage of the knowledge acquisition process it is essential to extract terms which correspond to domain specific objects, eventualities and their roles.

Each lexical entry in the domain vocabulary should be supplied with its part of speech and frequency of appearance in the corpus. Morphologically related entries should be grouped together since they will share the same conceptualization. For example, lexical entries *"to admit - Transitive Verb"* and *"admission - Noun"* correspond to the same conceptualization.

Lexical entries are then organised into hierarchically ordered conceptual types. The main difference between a lexical entry and a concept is that concepts have a single fixed meaning. A lexical entry can correspond to several concepts (polysemy) and a concept can be mapped onto several lexical entries (synonymy).

If in a corpus two or more lexical entries are used as synonyms the conceptualization of one of them should be taken as the main one and the others will correspond to this concept. For example, if in a medical domain the words "discharge" and "release" both correspond to the meaning *to let a patient out of a hospital*, they should correspond to the same concept.

For the domain dependent part of the vocabulary a correspondence between lexical entries and concepts is usually straightforward. Thus virtually every lexical entry from this part of the vocabulary is granted concept status. Lexical entries from the other parts of the vocabulary may not have that straightforward a mapping and are conceptualized at later stages of the KB development (especially step 3).

An important task at this stage is to provide a structuring among concepts. This structuring need not be total: instead one can create many conceptual type lattices even in one microdomain. Hierarchical inclusion of subdomains also contributes to the structuring of conceptual types—several types from one microdomain can be subsumed by a single type from its parent domain.

In our PDS corpus which is in the area of heart disorders we have distinguished the following basic typology:

- the heart and its components:
- heart disfunctions
- medical procedures: medical tests and medical treatments.
- miscellaneous role types: patient, doctor, general procedures (admit, discharge), etc.

4.2 Compositional structuring

The main aim at this stage is to characterize a compositional structure of the basic domain types elaborated at the previous stage. This compositional structure relies on two major relations of aggregation: a relation of structural inclusion (*has*) and a relation of characterization (*char*).

Compositional descriptions can be used for many purposes: as canonical structures for assembling surface lexical structures into conceptual types, as semantic constraints for checking the validity of propositions, as the basis for a compositional semantics of surface lexical structures, etc.

For example, for concepts of the category OBJECT there is a componential inclusion (part-whole). For durational concepts there is an aggregation of their thematic role fillers and presupposed relations between them. Consider:

```
[@disease:∀]
  →(is-a)→[@durational-hold]
  →(expr)→[@person:y]
  →(loc)→[@body-COMP]←(has-comp)←[@person:*y]
  →(char)→[@disease-degree:∀]
```

This says that a disease not only has the property of occurring in various degrees (as discussed above), but also that it holds over a period of time, that its experiencer (in this domain) is a person, and that diseases are "located in" a particular body part of that person.

Each entity type should be related with a set of compatible properties and these properties can be subcategorized into *intrinsic* and *extrinsic* if needed. Properties should be associated with their measures and constraints on values. Then one can determine number and value restrictions on structural attributes.

Here is an example of a compositional description of the type BODY-VESSEL which is one of the central concepts in our medical domain.

```
[@body-vessel:∀]
  →(ling)→{"vessel"}
  →(is-a)→[@physical-object]
  ←(has-comp)←[@human-body:∀]
  →(vessel-fluid)→[@body-fluid]
  →(has-comp)→[@bl-ves-branch:{*}∀]→(ling)→{"branch"}
  →(has-area)→[@bl-ves-origin:∀]→(ling)→{"origin"}
  →(has-area)→[@bl-ves-stem:∀]→(ling)→{"stem"}
  →(has-comp)→[@bl-ves-valve:{*}∀]→(ling)→{"valve"}
  →(has-comp)→[@bl-ves-fork:∀]→(ling)→{"fork"}
  →(has-area)→[@bl-ves-segment:{*}∀]→(ling)→{"segment"}
                →(rel-pos)→[@bl-ves-rel-pos:∀] =
                        {@nr-bv-pos = {"proximal" };
                         @far-bv-pos ={"distal"} }
  →(dir)→[@bl-ves-direction:∀] =
                {@up-vsl-dir ={"ascending"};
                 @down-vsl-dir = {"descending"} }
```

To create a compositional description the following strategy can be applied:

1. Choose a prototypical text fragment.
2. Determine the central concepts of the fragment.
3. Look for all occurrences of the corresponding type (including all of its subtypes) in the corpus to check the contexts in which it is used. First look for occurrences of the type in the analyzed fragment and then in other fragments. When necessary, look at previous or following sentences to resolve anaphoric and other references.
4. Apply a type oriented analysis with a predefined type schematization.
5. Impose semantic constraints on structures:
   a) in each citation determine fillers for the structures;
   b) find general characterizations for the fillers;
   c) if there are several cases for one filler, try to reduce them to one—i.e. resolve metonymy.
6. Characterize morphological and grammatical features for the fillers.
7. Determine which are allowable modifiers.

One of the first issues in the execution of this task is to determine central concepts of a fragment. Main recommendations are to start with concepts which correspond to basic domain types or their nomenclature: since verbs have the richest valency structures it is recommended to start with eventualities.

4.3 Elicitation tools

For some domains a structural representation of the domain vocabulary can be found in specialized on-line thesauri and dictionaries. For the work reported here, we have been using material made available as part of the Unified Medical Language System and the Dorland Medical Illustrated Dictionary. However one cannot avoid extensive elicitation of domain knowledge from an underlying text corpus. In the project reported here we have been using a special corpus-engineering tool for Distributional Analysis.

Distributional Analysis (Hirshman 86) is a technique used for the acquisition of semantic features of the domain (so called domain schemata) from an underlying text corpus. It is based on the identification of the sublanguage specific co-occurrence properties of the words in the syntactic relations in which they occur in the texts. These co-occurrence properties indicate important semantic characteristics of the domain: classes of objects and their hierarchical inclusion, properties of these classes, relations among them, lexico-semantic patterns for referring to certain conceptual propositions, etc.

The automatic identification of the co-occurrence patterns in a large corpus requires several modules:

- a word tagger and a specialized partial robust parser. These components support the extraction of phrases with predefined syntactic structures from the corpus and allow the system to determine the equivalence of different paraphrases of the same phrase.
- a clustering module. It generates statistically justified clusters of words which are the starting point in the creation of the conceptual type lattice for the KB.
- a thesaurus-like subsumption checking module. This module allows the knowledge engineer to attach existing thesauri to the system thus providing the system with a certain amount of knowledge beforehand.
- a module for the identification of lexical-semantic patterns. These patterns describe legitimate combinations of words and classes and are generated by generalization of word co-occurrence patterns. Lexical-semantic patterns are the starting point in the specification of semantic constraints on type combinations in the knowledge base.
- the user interface with different modes of presentation of the digested information. It assists in the conceptualization of the extracted data presenting different properties of the digested information in different form to the user.

Another useful elicitation tool helps us to identify domain terminology by processing the corpus. This tool extracts compound noun phrases with several restrictions on their contents (cf. Huizhong 1986) and count their frequency of occurrence in the corpus. This preliminary term bank is then post-processed manually.

4.4 Processing example: processing compound terms

After equipping the knowledge base with a structural representation of domain objects, events, etc., it is possible to disambiguate terms in the underlying corpus. Consider phrases like:

myocardium infarction, antero-septal myocardium infarction etc.

Using the KB entries it is possible to map these phrases into their conceptual contents. The interpreting algorithm follows the following steps:

- *take an input phrase.* For example: "antero-septal myocardial infarction".
- *take the head word of the noun phrase and activate all conceptual structures it is related to.* In our example the head word is "infarction" and it is mapped into the conceptual type @infarct which is a subtype of the @disease type:

  [@infarct:∀]
  →(is-a)→[@disease]→(kind)→[@dis-TYPE:"infarct"]
  →(loc)→[@muscle]

  Note that if there had been a multiple mapping of the word "infarction" all corresponding conceptual structures would be activated and in the process of interpretation of other items in the noun phrase or sentence uncompatible ones would be rejected. For example, let's imagine that there is the following entry in the KB:

  [@dummy:∀]
  →(ling)→{"infarct", "infarction"}
  →(is-a)→[@table]

  This structure says that there is a type @dummy which is called "infarction" or "infarct" and is a specialization of the type @table. So the word "infarction" would activate this structure as well as the @infarct.
- the next word in the phrase is "myocardial" which is unambiguously mapped into @myocardium:

  [@myocardium:∀]
  →(is-a)→[@muscle]←(has-comp)←[@heart-wall:∀]

  Both @infarct and @dummy structures try to absorb the @myocardium structure and result in the following:

  [@MI:∀] = (λ x)[@infarct:*x]→(loc)→[@myocardium]
  →(ling)→{"myocardial infarction"}

  The @dummy structure failed to absorb the @myocardium structure and was therefore rejected from the rest of the interpretation process.
- the next word is "antero-septal". It doesn't have a direct mapping into conceptual structures, so first the word "septal" is unambiguously mapped into:

  [@intr-sept:∀]
  →(ling)→{"interventricular septum"}
  →(is-a)→[@muscle]←(has-comp)←[@myocardium:∀]

  and then the word "antero" is mapped into @frnt-ha. The composition of the phrase "antero-septal" is:

  [@ant-sept:∀]=(λx)[@intr-sept:*x]→(loc)→[@frnt-ha]
  →(ling)→{"antero-septal"}

- now the two conceptual structures are unified and a conceptual structure for the phrase "antero-septal myocardium infarction" is:

  [@ant-sept-inf] = (λ x) [@infarct:*x]→(loc)→[@intr-sept]→(loc)→[@frnt-ha]

5 Step 2: Text Schematization

After the KB is equipped with a structural representation of the main domain types and their nomenclature, larger pieces of text can be characterized in terms of conceptual schemata. These schemata are usually based on temporal and causal relations among main events, plan-related goals of main actants and pre-packaged inferences.

An important property of temporal-causal schemata is the inheritance of thematic role fillers among subevents. Since these script-like structures encode information which is often not stated explicitly in the text, they are a very powerful knowledge source for resolving implicitly assumed information.

There are certain steps that can be followed in the creation of the temporal-causal structures:

1. determine general events and their subevents:
2. determine temporal precedence among the subevents and assign necessity and plausability to temporally ordered pairs of subevents;
3. determine causal connections between the temporally ordered pairs;
4. determine thematic role fillers and their cross-correspondence and inheritance among subevents.

There are no real limits on the possible elaborations of this sort of knowledge, as they can include plans, goals, effects, etc. But in many cases this additional knowledge is not needed.

The PDSs in the corpus we have been working with are all 15 to 20 sentences long. Their structure can be described as follows:

1. Header -
   a) Patient attributes: name, address, sex, date of birth, etc.
   b) Context: hearer = doctor, speaker = consultant, theme = patient, instrument = PDS.
2. Social greeting of the speaker to the hearer: "Dear Dr. Grippo"
3. Details of the admission procedure: data, place, mannerof admission (referral, ambulance,...), diagnosis, history of previous treatments...
4. Medical procedures performed in the hospital.
5. Details of the Patient discharge: when, medications on discharge, suggestions on further treatment.

Each lexical entry can be marked according to the structural parts of the text it is found in with its frequency of occurrence. For example:

admit, admission, admitted: 1 -500, 2 -279.

{*cardiac catheterisation*}: 2 -386

{*left ventricular function*}: 2 -12,

{*date*~{{*day*~}{*month*~}{*year*~}}}: 1 -500, 2 -500, 5 -500.

{*amount*~{{*quantity*~}{*unit*~}{*measure*~}}}: 5 -420.

A PDS can be described as a report about hospitalization. This in its turn can be decomposed into different stages: admission, testing, treatment, discharge etc. An important feature of a structure like that is the inheritance of fillers for thematic roles among subevents.

```
[@hospitalization:∀]
  →(agnt)→[@doctor:d]
  →(ptnt)→[@patient:p]
  →(to)→[@hospital:h]
  →(reason)→[@proposition:
              [@disease]→(exps)→[@person:*p] ]
  →(reason)→[@proposition:
              [@medical-treatment]→(exps)→[@person:*p] ]
  →(hold)→[@time-interval]
             →(from)→[@time-pnt:t1]
             →(to)→[@time-pnt:t5]
  →(subevent)→[@admit:e1]
             →(agnt)→[@doctor:*d]
             →(ptnt)→[@person:*p]
             →(loc)→[@hospital:*h]
             →(cul)→[@time-pnt:*t1]
  →(subevent)→[@med-testing:e2]
             →(agnt)→[@doctor:*d]
             →(ptnt)→[@person:*p]
             →(loc)→[@hospital:*h]
             →(cul)→[@time-pnt:t2]→(>)→[*t1]
  →(subevent)→[@med-treatm:e3]
             →(agnt)→[@doctor:*d]
             →(ptnt)→[@person:*p]
             →(loc)→[@hospital:*h]
             →(cul)→[@time-pnt:t3]→(>)→[*t2]
  →(subevent)→[@drug-prescr:e4]
             →(agnt)→[@doctor:*d]
             →(ptnt)→[@person:*p]
             →(loc)→[@hospital:*h]
             →(cul)→[@time-pnt:t4]→(>)→[*t3]
  →(subevent)→[@discharge:e5]
             →(agnt)→[@doctor:*d]
             →(ptnt)→[@person:*p]
             →(from)→[@hospital:*h]
             →(cul)→[@time-pnt:*t5]→(>)→[*t4]
```

This script-like structure represents information which is often not explicitly stated in the text, e.g. that a medical test or a treatment is carried out in a hospital and on the patient. This implicit information can be inherited by all members of the hospitalization structure and is added to the emerging text interpretation.

It is important to note that each of the events in the script has its own conceptualization provided during the first step in the knowledge acquisition process and the script only states interdependencies between events.

Further elaborations can be provided to the initial script. For example:

```
[medical-testing:e2]
  →(ptnt)→[@patient: *p]
  →(agnt)→[@doctor: *d]
  →(cul)→[@time-pnt:t2]
  →(cause)→[@to-be-known]
             →(exprs)→[@doctor:*d]
             →(theme)→[@disease]→(exprs)→[@patient:*p]
```

This says that a medical test procedure performed by a doctor on a patient allows this doctor to get to know about a disease of this patient.

6 Step 3: Characterization of domain independent words

The results of steps 1 and 2 are a characterization of the domain of interest and established meaning structures for the text in question. Commonsense words in domain oriented texts often serve as linguistic devices and indicate a realization of the domain structures: they can assign modality (will, be capable of, need, seem), culmination (occur, take place, manifest, fact), result, etc.

Generally these commonsense words are very ambiguous but in a context of domain dependent patterns they can be mapped less ambiguously into conceptual contents.

Commonsense Theories

There are several conceptual theories which are important for almost any domain. And the most frequently needed one is a conceptualization of time. As many others this one is based on a conceptualization of scales with underlying interval logic.

In our conceptualization of scales there is a type @amount. The structure of a quantitative amount is: **number*unit+number*subunit+....** The structure for a qualitative amount is: **interval-name+subinterval-name+....** Each instance of amount has its name which is composed from its value: **"3*kg.+27*gr"**, **"green+light+very"** etc.

The temporal domain is characterized as a set of consistent scales with a main scale whose units are years, subunits months and subsubunits days. So, for example a phrase like "15 January 1991" is represented as:

[@time-pnt:"1991*year+January+15*day"]

Culminative verbs

A large group of commonsense words and phrases asserts a culmination of the related event. Verbs like: "occur", "happen", "take place", etc. take their related events in the nominalized form as a subject. The transformational rule

Culminator(e) & Subject(e,e1) & Cul(e) & Event(e1) : e1 →e.

determines that a related event overrides the culminator and inherits all its thematic role fillers. So, for example, for a sentence like *"Heart failure occurred on 15th of January"* the surface structure

Occur(e) & Theme(e,e1) & Failure(e1) & Theme(e1,h) & Heart(h) & Cul(e,"Jan+15*day")

is transformed into

Failure(e) & Theme(e,h) & Heart(h) & Cul(e,"Jan+15*day")

and then into a conceptual structure with inheritance of implicit roles.

Other verbs of this class, for example, "undergo" or "perform", take the related event as an object and assign the passive reading.

The past tense of the culminator usually asserts that the event really did happen. We mark such culmination with the stress-mark: Cul'(e). However, some culminators even in the past tense don't assign the real culmination, for example, the verb "to schedule".

Verbs like "to have","to be under" also indicate that the states they talk about (to have a disease, for example), also actually occurred:

Have(e) & Object(e,e1) & Hold(e) & Durational-hold(e1) : e1 →e

So, in the sentence *"Patient had angina"* the structure:

Have(e) & Expr(e,p) & Patient(p) & Theme(e,e1) & Hold(e) & Angina(e1)

will be transformed into:

Angina(e) & Expr(e,p) & Patient(p) & Hold(e)

Resultators and causators

Another group of commonsense words ("result in", "cause") actualize a result of an event and the causation of the resulting state.

Local mapping of commonsense words

Unlike in the previously described groups many commonsense words take their particular meaning only in a context of domain categories. For example, in the pattern:

Show(e) & Subj(e,x) & Test(x) & Obj(e,y) & Disease(y)

the word "show" is not mapped onto a particular concept but rather the whole pattern is mapped onto the structure:

[medical-testing:e2]
→(ptnt)→[@patient: *p]
→(agnt)→[@doctor: *d]
→(cul)→[@time-pnt:t2]
→(cause)→[@to-be-known]
→(exprs)→[@doctor:*d]
→(theme)→[@disease]→(exprs)→[@patient:*p]

The generalized lexical pattern which corresponds to the above named structure is:

Reveal(ed) & Agnt(ed,e1) & Medical-testing(e1) & Theme(ed,e12) & Disease(e12)

Reveal(e) = {"show", "reveal", "demonstrate", "indicate", "confirmed", "suggested", "diagnosed"}

Such patterns are a powerful source for resolving metonymy because any two of the constituents imply the existence of the third one. For example in the sentence "chest showed angina" the pattern "@test {show} @disease" is activated by the "{show} @disease" constituents and since the word "chest" fails to be of the type @test the actual relation is reconstructed from the KB entry for the type @med-test created in the first phase:

[@med-testing]→(loc)→[@body-COMP]

Another example is the pattern

Develop(e) & Agnt(e,p) & Patient(p) & Theme(e,d) & Disease(d)

in a sentence *"He developed MI in 1984."*

This pattern unambiguously characterizes usage of the commonsense word "develop":

[@disease]
→(expr)→[@patient]
→(hold)→[@time-interval:ti]
→(int-from)→[@time-point:tf]

To develop a disease means the disease holds.

7 Processing example: text

The process of text interpretation proceeds by first mapping the most domain specific lexical items into their domain categories, since this is supposed to be a near unambiguous process. This was illustrated in section 4.4. Then, as the domain context becomes established, other lexical items are mapped into their categories.

In this processing work, we rely on a a robust parser which yields some syntactic bracketing of the input strings. We don't describe that output here, but instead concentrate on the interaction with the knowledge base described above.

sentence 1:

This 62 year old patient was admitted for a spontaneous and recurrent angina pectoris.

The longest domain dependent string is "angina pectoris" which is mapped into the type @angina-pectoris. This type is subsumed by its domain category @disease.

The type @disease absorbs the modifiers "spontaneous" and "recurrent" and the type @angina-pectoris determines that the body component of the disease was a heart.

```
[@disease:d]
  →(is-a)→[@durational-hold]
  →(expr)→[@person:y]
  →(loc)→[@body-COMP]←(has-comp)←[@person:*y]
  →(char)→[@dis-degree:∀]
  →(char)→[@dis-ext:∀]
  →(char)→[@dis-stability]→(val)→[@unst] // spontaneous
  →(char)→[@dis-periodicy]→(val)→[@freq] //recurrent


[@angina-pectoris]
  →(is-a)→[@disease]
  →(loc)→[@heart]
```

The time expression "62 year" can only be conjoined with the structure of the type @person:

```
[@person]→(char)→[@age]→(val)→[@time-interval:"62*year"]
```

or in a more compact form:

```
[@person]→(char)→[@age:@"62*year"] .
```

In this case the word "old" is redundant and is omitted by the system.

The word "to admit" in the general language is ambiguous, but in this context the VP **Admit(e) & Patient(e,p) & Person(p)** is not, and is mapped onto the following entry in the KB:

```
[@admit:∀]
  →(is-a)→[@health-care-procedure]
  →(is-a)→[@motion]
  →(ling)→{"admit","transfer"}
  →(agnt)→[@doctor]
  →(ptnt)→[@person:p]
  →(to)→[@hospital]
  →(from)→[@hospital]
  →(reason)→[@proposition:
                [@disease]→(exps)→[@person:*p] ]
  →(reason)→[@proposition:
                [@medical-treatment]→(exps)→[@person:*p]]
```

This structure not only absorbs structures for @disease and @person but also reconstructs implicit information that the same person who was admitted suffered from the "recurrent and spontaneous angina pectoris".

The final conceptualization of sentence 1 is as follows:

```
[@admit:∀]
  →(ptnt)→[@person:p]→(char)→[@age:@"62*year"]
  →(reason)→[@proposition:
                [@angina-pectoris:d]
                  →(exps)→[@person:*p]
                  →(loc)→[@heart]←(has-comp)←[*p]
                  →(char)→[@dis-stability]→(val)→[@unst]
                  →(char)→[@dis-periodicy]→(val)→[@freq]]
```

Then the whole structure is placed into a corresponding place in the hospitalization script.

Sentence 2:

In January 1989 he developed an antero-septal myocardial infarction diagnosed by exercise stress test.

Two strings, "an antero-septal myocardial infarction" and "exercise stress test" are mapped into conceptual structures.

Then the structure

Develop(e) & Agnt(e,p) & Person#,male(p) & Theme(e,d) & Ant-sept-mi(e) & In(e,"1989+Jan")

is transformed in accordance with the transformational rule for

@person {develop} @disease

(cf. above) into:

```
[@ant-sept-mi]
  →(exprs)→[@patient:#]→(char)→[@sex:"male"]
  →(hold)→[@time-interval]
              →(from)→[@time-point:"1989+Jan"]
```

The pronominal anaphor "he" is resolved by looking for an earlier occurrence of instances of the type @patient. There is such an occurrence in the previous sentence.

Next the remaining surface lexical structure of the sentence corresponds to the pattern

Reveal(e) & Agnt(e,e1) & Med-test(e1) & Theme(e,e2) & Disease(e2)

which is translated (cf. above) into the following conceptual structure:

```
[medical-testing:]
  →(ptnt)→[@patient:p]
  →(agnt)→[@doctor:d]
  →(cul)→[@time-point:t1]
  →(cause)→[@to-be-known]
                →(exprs)→[@doctor:*d]
                →(theme)→[@disease]→(exprs)→[@patient:*p]
```

This structure is compatible with the previous one and the resulting structure is as follows:

```
[exerc-str-test:]
  →(ptnt)→[@patient:*p] //resolved to the sentence 1
  →(agnt)→[@doctor:d]
  →(cul)→[@time-point:t1]
  →(cause)→[@to-be-known]
              →(exprs)→[@doctor:*d]
              →(theme)→[@ant-sept-mi:e]
                    →(exprs)→[*p]→(char)→[@sex:"male"]
                    →(hold)→[@time-interval]
                    →(start)→[@time-point:"1989+Jan"]
```

As a result, we have reconstructed the implicit assumption that it was the patient who underwent the exercise stress test, and we have resolved the anaphoric reference.

This conceptual structure is then related to the hospitalization script and the system reconstructs that the doctor who was the agent of the patient's admission is the same as the one who performed the medical testing.

8 Conclusion

We have developed a methodology for building knowledge bases for natural language processing applications. It uses a representation framework which mixes different languages to allow for greater expressiveness with

increased processing efficiency. The knowledge acquisition methodology is concept-oriented, i.e. guided by the structure of the domain, rather than by the lexical semantics of the words occurring in the text. And the knowledge acquisition work takes as its starting point large collections of texts rather than just prototypical fragments. This in turn requires robust corpus handling tools.

Our approach works for sublanguage handling. This means that each natural language task in a different domain requires the creation of a new knowledge base. Efficient tools that can assist in this process will make the creation of new knowledge bases less labour intensive. But it is still a major task.

Knowledge engineering for expert systems is faced with the same problem. One approach taken there is to try and identify "generic task models" (Bylander and Chandrasekaran 1987) to guide the knowledge acquisition process for new domains. A new knowledge base is then defined as a specialization of the generic model.

In Mikheev and Moens (1994) we have proposed to adopt a similar approach to knowledge engineering for natural language systems. As suggested in Marsh (1986), there is a great similarity between reports on engine failures and patient discharge summaries. In terms of the generic task models, "dealing with equipment failure" can be described as a generic task with specializations "engine failure" and "cardiac failure" and these in turn can have further specializations. A proper conceptualization of these generic tasks could contribute to the construction of knowledge bases for many fields and subfields: instead of porting from the engine failure domain to the heart failure domain, one would instead be instantiating a generic task on equipment failure to the cardiac domain.

9 Acknowledgements

This work was supported in part by the EC-funded ET-12 project "Methodologies for Constructing Knowledge Bases for Natural Language Processing Systems" and by the EC-funded AIM Project A2023 "An Access System for Medical Records using Natural Language".

References

Bech, A., A. Mikheev, M. Moens and C. Navarretta (1993) Typology for information contents. ET-12 Project Report 2, Center for Sprokteknologi, University of Copenhagen.

Bech, A. (1989) The design and application of a domain specific knowledgebase in the TACITUS text understanding system. In J. Pind and E. Rognvaldsson, eds., *Papers from the Seventh Scandinavian Conference of Computational Linguistics*, pp. 114–126.

Brachman, R., D. McGuinness, P. Patel-Schneider, L. A. Resnick and A. Borgida (1991) Living with classic: When and how to use a KL-ONE–like language. In J. Sowa, ed., *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pp. 401–456. Los Altos, Ca.: Morgan Kaufmann.

Bylander, T. and B. Chandrasekaran (1987) Generic tasks for knowledge-based reasoning: The "right" level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies* **26**, 231–243.

Finin, T. (1986) Constraining the interpretation of nominal compounds in a limited context. In R. Grishman and R. Kittredge, eds., *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, pp. 163–173. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Guarino, N. (1991) A non-standard approach to terminological knowledge: the ITL system. In B. Nebel, C. Peltason and K. von Luck, eds., *International Workshop on Terminological Logics*, pp. 47–50, IBM Germany Research Center.

Hirschman, L. (1986) Discovering sublanguage structures. In R. Grishman and R. Kittredge, eds., *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, pp. 211–234. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Hobbs, J. (1986) Sublanguage and knowledge. In R. Grishman and R. Kittredge, eds., *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, pp. 53–68. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Hobbs, J. and C. Navarretta (1993) Methodology for knowledge acquisition. Unpublished ms., January 1993.

Huizhong, Y. (1986) A new technique for identifying scientific/technical terms and describing science texts: An interim report. *Literary and Linguistic Computing* **1**(2), 93–103.

Lenat, D. B. and J. S. Brown (1984) Why AM and EURISCO appear to work. *Artificial Intelligence* **23**(3), 269–294.

Marsh, E. (1986) General semantic patterns in different sublanguages. In R. Grishman and R. Kittredge, eds., *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, pp. 103–127. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Mikheev, A. and M. Moens (1994) KADS methodology for knowledge-based language processing systems. In B. R. Gaines and M. Musen, eds., *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pp. 5-1–5-17.

Parsons, T. (1990) *Events in the Semantics of English.* Cambridge, Mass.: MIT Press.

Sowa, J. F. (1984) *Conceptual Structures: Information Processing in Mind and Machine.* Reading, Mass.: Addison-Wesley.