A Large-Scale Resource for Storing and Recognizing Technical Terminology

Henk Harkema, Robert Gaizauskas, Mark Hepple, Neil Davis
Yikun Guo, Angus Roberts, Ian Roberts

Department of Computer Science, University of Sheffield,
Regent Court 211, Portobello Street, Sheffield S1 4DP, UK
biomed@dcs.shef.ac.uk

Abstract

This paper discusses the design and implementation of Termino, a large-scale terminological resource for text processing. Dealing with terminology is a difficult but unavoidable task for natural language processing applications, such as information extraction in technical domains. Complex, heterogeneous information must be stored about large numbers of terms. At the same time term recognition must be performed in realistic times. Termino attempts to reconcile this tension by maintaining a flexible, extensible relational database for storing terminological information and compiling finite state machines from this database to do term recognition.

1. Introduction

Identification and semantic classification of technical terms – single or multiword expressions having some specialized use or meaning in a specific domain – in text is an important step in many natural language processing applications. In the context of information extraction, for example, these terms often point to entities about which information should be extracted. Proper semantic classification of terms also helps in resolving anaphora and extracting relations whose arguments are restricted semantically. In other information-oriented applications, e.g., information retrieval, knowledge of the entities that appear in a text can be used to link a given document to other, related documents, or to make connections to outside knowledge bases containing additional information about the entities mentioned in the document.

In this paper we describe the design and implementation of Termino, a resource to support terminology processing for information extraction, retrieval, and navigation. While Termino has been developed for biomedical applications, its general design allows it to be used for term processing in any domain.

2. Termino

Termino is a large-scale terminological resource for text processing applications. It includes a flexible, extensible relational database which is designed to store large numbers of terms together with complex, heterogeneous information about these terms. Recognition of terms in text is done via finite state machines that are compiled from this terminological database.

The contents of Termino are imported from existing, outside knowledge sources, such as the HUGO Nomenclature database and the UMLS Metathesaurus (Humphreys et al., 1998). Contents can also be induced from text corpora, e.g., MEDLINE citations. Termino thus provides uniform access to terminological information aggregated across many sources, without the need for multiple, source-specific terminological components within a text processing system.

The advantage of "dictionary-based" term recognition with Termino over "inductive" term recognition methods such as, for example, Kim and Tsujii (2002) and Fukuda et al. (1998), is that Termino provides immediate entry points into a variety of outside ontologies and other knowledge sources, making the information in these sources available to processing steps subsequent to term recognition.[1] This is a very attractive feature. For example, using a recognizer compiled to include terms from the HUGO Nomenclature database and the OMIM database (Online Mendelian Inheritance in Man, OMIM (TM), 2000), Termino will return the HUGO and OMIM identifiers for the gene and protein names it recognizes in a text. These identifiers give access to the information stored in these databases about the gene or protein, including alternative names, gene map locus, related disorders, and references to relevant papers.

A general disadvantage of bare dictionary-based approaches to term recognition is their inability to deal with terms that are not in the dictionary. The occurrence in the dictionary of technical terms that happen to be identical to commonly used English words is also problematic. Aware of these limitations, we intend Termino to be the first component, the lexical look-up component, in a multicomponent term processing system. Thus, term look-up as performed by Termino is not the end point of term processing. Look-up might return multiple matching terms for a given string, or for overlapping strings, and subsequent processes may apply to filter these alternatives down to the single option that seems most likely to be correct in the given context. Furthermore, more flexible processes of term recognition might apply over the results of look-up. For example, a term *grammar* can be provided for a given domain, allowing longer terms to be built from shorter terms that have been identified by term look-up. In this paper, however, we will focus on the design and implementation of Termino.

3. Related Work

The UMLS Metathesaurus provides a semantic classification of terms drawn from a wide range of vocabularies in the clinical and biomedical domain. It does so by grouping strings from these vocabularies that are judged to

[1]Obviously, this advantage does not extend to terms that are added to Termino through term induction.

have the same meaning into concepts, and mapping these concepts onto nodes or semantic types in a semantic network. Although it is used in a number of biomedical natural language processing applications, e.g., Rindflesch et al. (2000), Pustejovsky et al. (2002), we have decided not to adopt the UMLS Metathesaurus as the primary terminology resource in our text processing system for a variety of reasons. The first reason is that the Metathesaurus is a closed system: strings are classified in terms of the concepts and the semantic types that are present in the Metathesaurus and the semantic network, whereas we would like to be able to link our terms into multiple ontologies, including in-house ontologies that do not figure in any of the Metathesaurus' source vocabularies and hence are not available through the Metathesaurus. Moreover, we would also like to be able to have access to additional terminological information that is not present in the Metathesaurus, such as, for example, the annotations in the Gene Ontology (The Gene Ontology Consortium, 2001) assigned to a given protein term.[2] Furthermore, as new terms appear constantly in the biomedical field we would like to be able to add these to our terminological resource immediately and not have to wait until they have been included in the UMLS Metathesaurus.

Besides text processing systems that use an existing terminological resource, such as the UMLS Metathesaurus, there are systems that rely on resources that have been specifically built for the application, e.g., Humphreys et al. (2000) and Thomas et al. (2000). With regard to the latter kind of system we note that their terminological resources tend to be limited in the following two respects. First, the structure of these resources is often fixed and in some cases amounts to simple gazetteer lists. Secondly, because of their fixed structure, these resources are usually populated with content from just a few sources, leaving out many other potentially interesting sources of terminological information. Instead, we designed Termino to be an extensible resource that can hold diverse kinds of terminological information from a variety of sources.

4. Architecture

Termino consists of two components: a database holding terminological information and a compiler for generating term recognizers from the database. These two components are discussed in the following two subsections.

4.1. Terminological Database

The terminological database has been designed to meet three requirements. First of all, it is capable of storing large quantities of terms. Any technical domain generates very large numbers of terms. The UMLS Metathesaurus, for example, which includes terms from a wide range of vocabularies in the clinical and biomedical domain, currently contains over 2 million distinct terms. However, as UMLS is just one of many resources whose terms may need to be stored, many millions of terms may need to be stored in total. Secondly, Termino's database is designed to be flexible enough to hold a variety of information about terms, including morpho-syntactic information, such as part of speech and morphological class; semantic information, such as quasi-logical form and links to concepts in ontologies; and provenance information, such as the sources of the information in the database. The design of the database also allows for links connecting synonyms and morphological and orthographic variants to one another, and for connecting abbreviations and acronyms to their full forms. Thirdly, the database is organized in such a way that it allows for fast and efficient recognition of terms in text.

Since the sources feeding into Termino are heterogeneous in both information content and format, Termino's database is "extensional": it stores strings in connection with information about strings.[3] The database is organized as a set of relational tables, each storing a specific type of terminological information. New tables can be added as required. In this way we avoid the strictures of any one fixed representational scheme, thus making the database flexible enough to hold information from disparate sources.

Terminological information about any given string is usually gathered from multiple sources. As information about a string accumulates in the database, we must make sure that co-dependencies between various pieces of information about the string are preserved. This consideration leads to the fundamental element of the terminological database, a *termoid*. A termoid consists of a string together with associated information of various kinds about the string. Information in one termoid holds conjunctively for the termoid's string, while multiple termoids for the same string express disjunctive alternatives.

For instance, taking an example from the UMLS Metathesaurus, we may learn from one source that the string *cold* as an adjective refers to a temperature, whereas another source may tell us that *cold* as a noun refers to a disease. This information is stored in the database as two termoids: abstractly, '*cold*, adjective, temperature' and '*cold*, noun, disease'. A single termoid '*cold*, adjective, noun, temperature, disease' would not capture the co-dependency between the part of speech and the "meaning" of *cold*.[4] This example also illustrates that a string can have more than one termoid. Each termoid, however, pertains to one and only one string.

Figure 1 provides a detailed example of part of the structure of the terminological database. In the table STRINGS every unique string is assigned a string identifier (*str_id*). In the table TERMOID STRINGS each string identifier is associated with one or more termoid identifiers (*trm_id*). These termoid identifiers serve as keys into the tables holding terminological information. Thus, in this particular example, the database includes the information that in the Gene On-

[2]While the terms making up the the tripartite Gene Ontology are present in the UMLS Metathesaurus, assignments of these terms to gene products are not recorded in the Metathesaurus.

[3]There is no explicit attempt to assemble strings and information into terms, as we have no clear definition of the notion "term". However, the various interconnections between strings and information about strings stored in the database bring about a level of organization which goes beyond simple pairings of strings and information.

[4]Note that the UMLS Metathesaurus has no mechanism for retaining this co-dependency between grammatical and semantic information.

STRINGS

string	*str_id*
...	...
neurofibromin	str728
abdomen	str056
mammectomy	str176
...	...

TERMOID STRINGS

trm_id	*str_id*
...	...
trm023	str056
trm656	str056
trm924	str728
trm369	str728
trm278	str176
...	...

PART OF SPEECH

trm_id	*pos*
...	...
trm023	N
...	...

GO ANNOTATIONS

trm_id	*annotation*	*version*
...	...	...
trm924	GO:0004857	9/2003
trm369	GO:0008285	9/2003
...	...	...

UMLS

trm_id	*cui*	*lui*	*sui*	*version*
...	...	...	...	...
trm278	C0024881	L0024669	S0059711	2003AC
trm656	C0000726	L0000726	S0414154	2003AC
...	...	...	...	...

Figure 1: Structure of the terminological database

tology the string *neurofibromin* has been assigned the terms with identifiers GO:0004857 and GO:0008285. Furthermore, in the UMLS Metathesaurus version 2003AC, the string *mammectomy* has been assigned the concept-unique identifier C0024881 (CUI), the lemma-unique identifier L0024669 (LUI), and the string-unique identifier S0059711 (SUI). Connections between termoids such as those arising from synonymy and orthographic variation are recorded in another set of tables. There are also tables containing provenance information for strings and termoids. These tables are not shown in the example.

4.2. Term Recognition

To ensure fast term recognition with Termino's potentially vast terminological database, the system comes with a compiler for generating finite state machines from the strings in the database. Direct look-up of strings in the database is not an option, because it is unknown in advance at which positions in a text terms will start and end. In order to be complete, one would have to look up *all* sequences of words or tokens in the text, which is very inefficient.

Compilation of a finite state recognizer proceeds in the following way. Each string in the database is first broken into tokens, where a token is either a contiguous sequence of alphabetic characters, a contiguous sequence of numeric characters, or a punctuation symbol. Next, starting from a single initial state, a path through the machine is constructed, using the tokens of the string to label transitions. For example, for the string *5-HT3 receptor* the machine will include a path with transitions on *5*, *-*, *HT*, *3*, and *receptor*. New states are only created when necessary. The state reached on the final token of a string will be labeled final and is associated with the identifiers of the termoids for that string. To recognize terms in text, the text is tokenized and the finite state machine is run over the text, starting from the initial state at each token in the text. For each sequence of tokens leading to a final state, the termoid identifiers associated with that state are returned. These identifiers are then used to access the terminological database and retrieve the information contained in the termoids. Where appropriate the machine will produce multiple termoid identifiers for strings. It will also recognize overlapping and embedded strings. The compiler can be parameterized to produce finite state machines that match exact strings only, or abstract away from morphological and orthographical variation.

Figure 2 shows a small terminological database and a finite state recognizer derived from it. Running this recognizer over the phrase ... *thyroid dysfunction, such as Graves' disease* ... produces four annotations: *thyroid* is assigned the termoid identifiers `trm1` and `trm2`; *thyroid dysfunction*, `trm3`; and *Graves' disease*, `trm4`.

The set-up in which term recognizers are compiled from the contents of the terminological database turns Termino into a general terminological resource which is not restricted to any single domain or application. The database can be loaded with terms from multiple domains. Compilation can then be restricted to particular subsets of strings by selecting termoids from the database, for example, based on their source or other characteristics. In this way, one can produce term recognizers that are tailored towards specific domains or specific applications within domains.

5. Implementation

At this moment a first version of Termino has been implemented. It uses a database implemented in MySQL and currently contains almost 300,000 termoids for around 230,000 strings. Content has been imported from various sources. We have loaded into Termino a list of human proteins and their assignments to the Gene Ontology as produced by the European Bioinformatics Institute (http://www.ebi.ac.uk/GOA/), as well as gene names and symbols from the HUGO Nomenclature database, and a set of gene terms from the UMLS Metathesaurus. Furthermore, we have included several gazetteer lists containing terms in the fields of molecular biology and pharmacology that were assembled for previous information extraction projects in our NLP group.

A web services (SOAP) API to the database is under development. We plan to make the resource available to researchers as a web service or in downloadable form.[5] The compiler to construct finite state recognizers

[5]Users may have to sign license agreements with third parties

STRINGS

string	*str_id*
thyroid	str12
thyroid disfunction	str15
Graves' disease	str25

TERMOID STRINGS

trm_id	*str_id*
trm1	str12
trm2	str12
trm3	str15
trm4	str25

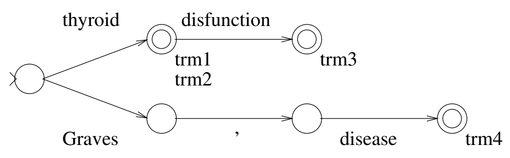

Figure 2: Sample terminological database and finite state term recognizer

from the database is fully implemented, tested and integrated into AMBIT, our natural language processing platform for biomedical text (Gaizauskas et al., 2003).

6. Conclusions & Future Work

Dealing with terminology is an essential step in natural language processing in technical domains. In this paper we have described the design of Termino, a large scale terminology resource for biomedical language processing.

Termino includes a relational database which is designed to store a large number of terms together with complex, heterogeneous information about these terms, such as morpho-syntactic information, links to concepts in ontologies, and other kinds of annotations. The database is also designed to be extensible: it is easy to include terms and links to outside biological databases and ontologies. Term look-up in text is done via finite state machines that are compiled from the contents of the database. This approach allows the database to be very rich without sacrificing speed at look-up time. These three features make Termino a flexible tool for inclusion in a biomedical text processing system.

As mentioned in section 2, Termino is intended to function within a more comprehensive term processing system. In order to establish what further term processing components would work most productively with Termino in a full term processing system, we have started exploring the performance of our current system by applying it to the task of gene and protein name identification in MEDLINE abstracts. As was to be expected, lexical look-up in Termino without further term processing generates a considerable number of false positives and false negatives. False positives mainly arise from terms in Termino that match subterms of compound gene and protein terms occurring in abstracts without the full terms being present in Termino, and from gene and protein terms in Termino which are also common English words. False negatives occur because of terms in text that do not appear in any of the terminological resources loaded into Termino, or result from terms in text that are present in Termino but in a variant form.

These findings can be addressed in various ways. For example, coverage of multi-token terms can be extended by using term grammars, whose rules describe how larger terms can be built from smaller terms. We found that the use of a limited, manually created term grammar for protein names had a significant positive effect on performance for the task described above. We will explore this approach further by considering methods for inducing term grammars from compound terms in term sources such as the UMLS Metathesaurus. The number of false positives can be reduced by adding a component for filtering out incorrectly recognized terms based on their orthographic features – technical terms are often be distinguished from common words by deviant capitalization – or contextual information.

7. References

Fukuda, K., A. Tamura, T. Tsunoda, and T. Takagi, 1998. Toward information extraction: Identifying protein names from biological papers. In *Proc. Pacific Symposium on Biocomputing*.

Gaizauskas, R., M. Hepple, N. Davis, Y. Guo, H. Harkema, A. Roberts, and I. Roberts, 2003. AMBIT: Acquiring medical and biological information from text. In S. Cox (ed.), *Proc. UK e-Science All Hands Meeting 2003, Nottingham, UK*.

Humphreys, K., G. Demetriou, and R. Gaizauskas, 2000. Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structures. In *Proc. Pacific Symposium on Biocomputing*.

Humphreys, L., D.A.B. Lindberg, H.M. Schoolman, and G.O. Barnett, 1998. The Unified Medical Language System: An informatics research collaboration. *Journal of the American Medical Informatics Association*, 1(5).

Kim, J. D. and J. Tsujii, 2002. Corpus-based approach to biological entity recognition. In *Proc. 2nd Meeting Special Interest Group on Text Data Mining, ISMB 2002*.

Online Mendelian Inheritance in Man, OMIM (TM), 2000. McKusick-Nathans Institute for Genetic Medicine, Johns Hopkins University (Baltimore, MD) and National Center for Biotechnology Information, National Library of Medicine (Bethesda, MD). http://www.ncbi.nlm.nih.gov/omim/.

Pustejovsky, J., J. Castaño, R. Saurí, A. Rumshisky, J. Zhang, and W. Luo, 2002. Medstract: Creating large-scale information servers for biomedical libraries. In *Proc. Workshop on NLP in the Biomedical Domain, Association for Computational Linguistics*.

Rindflesch, C.T., J.V. Rajan, and L. Hunter, 2000. Extracting molecular binding relationships from biomedical text. In *Proc. 6th Applied Natural Language Processing Conference, North American chapter of the Association for Computational Linguistics*.

The Gene Ontology Consortium, 2001. Creating the gene ontology resource: Design and implementation. *Genome Research*, 11(8).

Thomas, J., D. Milward, C. Ouzounis, and S. Pulman, 2000. Automatic extraction of protein interactions from scientific abstracts. In *Proc. Pacific Symposium on Biocomputing*.

in order to be able to use restricted resources that have been integrated into Termino.