Natural Language Engineering: page 1 of 32. © 2006 Cambridge University Press
doi:10.1017/S135132490600427X

Natural language processing in CLIME, *a multilingual legal advisory system*[†]

ROGER EVANS
University of Brighton, Brighton BN2 4GJ, UK
e-mail: R.P.Evans@brighton.ac.uk

PAUL PIWEK
Centre for Research in Computing, The Open University,
Milton Keynes MK7 6AA, UK
e-mail: P.Piwek@open.ac.uk

LYNNE CAHILL
Department of Linguistics and English Language, University of Sussex,
Falmer, Brighton BN1 9QN, UK
e-mail: lynneca@sussex.ac.uk

NEIL TIPPER
Alcatel Austria Aktiengesellschaft, Scheydgasse 41, 1210 Wien, Austria
e-mail: Neil.Tipper@alcatel.at

(*Received 16 October 2002; revised 26 January 2006*)

Abstract

This paper describes CLIME, a web-based legal advisory system with a multilingual natural language interface. CLIME is a 'proof-of-concept' system which answers queries relating to ship-building and ship-operating regulations. Its core knowledge source is a set of such regulations encoded as a conceptual domain model and a set of formalised legal inference rules. The system supports retrieval of regulations via the conceptual model, and assessment of the legality of a situation or activity on a ship according to the legal inference rules. The focus of this paper is on the natural language aspects of the system, which help the user to construct semantically complex queries using WYSIWYM technology, allow the system to produce extended and cohesive responses and explanations, and support the whole interaction through a hybrid synchronous/asynchronous dialogue structure. Multilinguality (English and French) is viewed simply as interface localisation: the core representations are language-neutral, and the system can present extended or local interactions in either language at any time. The development of CLIME featured a high degree of client involvement, and the specification, implementation and evaluation of natural language components in this context are also discussed.

[†] The research reported in this paper was supported by the European Commission ESPRIT funding programme, project number EP 25.414. The partners in the project were British Maritime Technology Ltd., Bureau Veritas, TXT E-Solutions SPA, the University of Amsterdam and the University of Brighton.

1 Introduction

With the advent of wide-scale publication of and access to electronically stored information, a class of applications which has become increasingly prominent is *question-answering systems*. A question-answering system can be loosely characterised as an application which takes as input a question – essentially a search request over a body of knowledge – and returns an answer – the result of the search – and some amount (possibly none) of justification for the answer. Using this characterisation, we can identify a number of key discriminators for such systems:[1]

- What is the form and scope of a question?
- What pre-processing on the body of knowledge is required to support the search?
- What are the properties of the search algorithm (complexity, inferential requirements, recall and precision, etc.)?
- What is the form and scope of an answer?
- What is the form and scope of the justification?

Although these discriminators are to some extent independent of each other, it is common to classify systems on a linear scale, with 'shallow' systems, offering simple answers to simple questions, at one end, and 'deep' systems, in which semantically rich questions may have complex answers requiring 'AI-complete' processing, at the other (cf. Chaudri & Fikes (1999), where various systems are presented as shallow, deep or somewhere in between). In general, shallow systems only allow the user to ask questions which approximate the user's real query, and the answers are correspondingly of limited precision, often requiring further manual refinement of the search result. Deep systems, on the other hand, run quite quickly into serious theoretical issues of knowledge representation, question interpretation etc., as well as practical problems of complexity, all of which are still largely unsolved.

Using natural language technology introduces a further dimension of variation: the question may be expressed linguistically, ranging from keyword specification to free text input; the body of knowledge may be preprocessed linguistically to facilitate the search (lemmatised, divided into sentences, marked up), or to present the answer; the answer may be text that is generated automatically; the justification may be expressed linguistically as an explanation; the entire interaction may take the form of a linguistically structured dialogue. The use of such technology needs to be carefully matched to the abilities of the system: natural language support that is either too restrictive or too expressive leads to systems that are often frustrating and counter-productive to use.

Many of the most prominent systems, notably the popular web search engines, inhabit the shallow end of the range and use minimal natural language support. Typically the question is just a set of key words or phrases and a limited set of combination functions over them; the body of knowledge is a collection of web pages, pre-processed by indexing key phrases – an often very large but not

[1] See also Hirschman & Gaizauskas (2001) and Burger, Cardie, Chaudri *et al.* (2002) for related analyses.

intrinsically complex task; the algorithm is essentially indexed look-up and relevance ranking; the answer consists of displaying initial fragments of the pages found; the justification consists of highlighting the words in the displayed fragment that matched the question. Much recent research attention, supported in particular by the TREC question-answering initiative (Voorhees 2001, 2004), has been given to extending this technology to provide more focused and accurate answers to 'naturally occurring' questions. Nevertheless this work remains quite firmly tied to a specific application context – in Hirschman & Gaizauskas's paper "Natural language question answering: the view from here" (Hirschman & Gaizauskas 2001), *here* is quite clearly *retrieval from free text corpora*, and Burger *et al.* (2002), although outlining a more general programme, is primarily concerned with *answer extraction* from text resources.

In this paper we describe a system, clime (Computerised Legal Information Management and Explanation), which offers question-answering functionality with somewhat deeper semantics, and uses natural language technology to facilitate complex query construction, answer generation and dialogue management, in order to provide a sophisticated yet still practical legal advisory system. clime provides conceptual search over, and legal assessment based on, a set of regulations relating to some application domain,[2] that are encoded as a conceptual domain model plus a set of formalised legal inference rules. In terms of the discriminators introduced above, clime offers two successively 'deeper' question-answering models, each with a different trade-off between development time, query execution speed and comprehensiveness of response:

Conceptual retrieval

the *question* is a set of domain concepts, provided explicitly or derived from a 'case description' (see below); the rules are *pre-processed* by indexing them relative to the conceptual domain model; the *search algorithm* expands the concept set using ontological links (such as `subtype-of`, `has-part`, `described-by`, `connected-to`), and uses the expanded set to index rules; the *answer* is a list of rules referencing concepts related to the query, ranked according to relevance; the *justification* explains (on request) how a concept mentioned in a rule is related to the query.

Normative assessment

the *question* is a 'case description', a relational representation of a state of affairs in the application domain; the rules are *pre-processed* by encoding them as formal inference rules; the *search algorithm* determines how the inference rules apply to the case and concludes whether the case described by the question is allowed, disallowed or not decided by the rules; the *answer* consists of just this conclusion; the (often quite lengthy) *justification* explains how the conclusion was reached with reference to the rules.

The focus of this paper is on the natural language interface aspects of clime – the legal encoding and inferencing aspects of the system are described in Winkels,

[2] In this paper we restrict attention primarily to the maritime domain, in an instantiation of clime called mile; the project also explored the domain of environmental health regulations, and some evaluation results for this system are also provided.

Breuker, Boer & Bosscher (1999) and Winkels, Boer and Hoekstra (2002). This is a sharper distinction than is generally made in discussing many TREC-style systems, which are primarily document indexing systems and use natural language techniques primarily to build their document indexes (henceforth, we shall call this general class of applications TREC QA systems). Evaluation of such systems, and indeed the whole evaluation methodology that has been developed around the TREC QA track, is mostly focused on the effectiveness of this indexing task (what levels of recall and precision it supports etc.). In CLIME, the inferential component is a logical inference engine, which does not directly involve NLP technology, either in its construction or use. The natural language components provide the interfaces to the system, allowing the user to construct the complex queries the system is capable of addressing, and delivering complex answers and justifications in a human-readable form. These NLP components are largely deterministic and predictable, and more appropriately evaluated as components of an interface, rather than empirical NLP systems. We consider that for more advanced question-answering applications it will be increasingly important to make this clear separation between the language technology of the interfaces and the knowledge/inference technology of the underlying functionality of the system (which may or may not involve NLP-based processing), as the requirements for both become independently more demanding. References to the actual inference processes in this paper will therefore be limited to what is required to describe the interface aspects.[3]

CLIME makes extensive use of natural language technology in its underlying representations and interfaces as follows:

- questions are constructed by manipulating a natural language representation of an underlying formal query, using a WYSIWYM (Power, Scott & Evans, 1998) interface;
- natural language answers and explanations are generated dynamically from legal inference engine output;
- the lexical resources used by the natural language interfaces are derived automatically from the encoding of the domain regulations as formal rules (Cahill, 2000), plus a small number of hand-coded items;
- Questions and answers are managed as part of a hybrid synchronous/asynchronous dialogue (Piwek, Evans & Power, 1999; Piwek, Evans, Cahill & Tipper, 2000) extending over the whole interaction;
- the whole system operates multilingually: internal representations are language neutral; questions and answers (new and previously submitted) are available in all languages supported by the system (currently English and French).

The paper is organised as follows. Section 2 provides a general description of the CLIME system as a whole. We discuss the application scenario of the system, the requirements of the system as determined by the application and prospective users,

[3] This is not to say that these boundaries are always easily or clearly identified – see Evans, Piwek & Cahill (2002) for further discussion of this issue in the context of the CLIME system.

the overall system architecture designed to meet those requirements, and finally we give a brief description of the top-level user interface to the system. Section 3 describes the natural language components of the system in more detail, covering conceptual resources, query formulation, answer generation, dialogue issues and underlying linguistic resources. In sections 4 and 5, we discuss our experience of system development and evaluation in a largely client-centred development process.

2 System overview

2.1 The CLIME *scenario*

The fundamental idea of CLIME is quite simple. Many fields of activity are governed by rules, laws or regulations of some sort. These typically exist in book form, or are available on-line as HTML or XML, but ordinary practitioners in the field often find them difficult to access effectively. In many fields there are expert practitioners whose primary role is to advise on these regulations. CLIME aims to provide a middle level of expertise, by supporting automatic concept-based searching and legal inferencing over such regulations, via a natural language, multilingual, web-based interface.

The expert domain of the CLIME prototype is maritime regulations – specifically regulations relating to ship-building and modification, and marine pollution. The CLIME prototype covers annexes I and II of MARPOL, an international treaty on marine pollution which regulates the safe operation of ships (MARPOL, 2002), and about 15% – approximately 4500 rules – of the Bureau Veritas in-house maritime classification rules (Bureau Veritas, 1997).[4] Both of these sources are substantial, multi-volumed sets of regulations, expressed in relatively legalistic terms. Ship surveyors, architects and engineers need to have a working knowledge of them to carry out their duties, but do not, in general, have a comprehensive expert knowledge of them.

The typical application scenario for CLIME is as follows. Suppose a surveyor is surveying a 2000 tonne cargo ship and discovers that ballast water is being stored in a fuel oil tank. He/she needs to know whether this breaks any regulations (in this case MARPOL regulations, since it may mean that ballast water polluted with fuel oil could be pumped out into the sea). He/she connects to the CLIME web server from his/her laptop and constructs a query relating to the regulations in his/her preferred language. CLIME supports two ways of accessing regulations: ‘conceptual retrieval’ and ‘normative assessment’.

The *conceptual retrieval* function returns a list of regulations which are *conceptually related* to the query. The simplest form of a query consists of a list of concepts followed by the user’s question, e.g.[5]:

> new oil tanker, gross tonnage, 2000 tonnes, fuel oil tank, ballast water.
>
> Show me regulations relating to these concepts.

[4] These 4500 rules were selected by domain experts as a relatively self-contained subset on the specific topic of oil tankers.

[5] As we discuss in more detail below, the user enters queries using a specialised interface, not by entering free text. The text shown here is the representation of the query that the user sees and manipulates, but is generated by the system. In particular, although in some places the phrasing may seem unnatural, the intent is to provide clear, unambiguous feedback of the query that has been constructed.

The query interface also supports the formulation of a query as a ‘case description’ from which the concepts are extracted:

> I have a new oil tanker. It has a gross tonnage of 2000 tonnes. It is fitted with a fuel oil tank. Ballast water is stored in the fuel oil tank.
>
> Show me regulations relating to the concepts in this description.

The answer (in both cases) is a list of regulations most closely associated with the concepts provided:[6]

> The following rules are relevant to your situation:
>
> *MARPOL-AI-P2-14-01*: ballast water, fuel oil tank, gross tonnage, new ship, oil tanker
>
> *MARPOL-AI-P2-14-02*: ballast water, fuel oil, fuel oil tank
>
> . . .
>
> *MARPOL-AI-P2-10-03-01*: dilution, distance from nearest land, gross tonnage, oil tanker, ship

Regulations are selected if they contain the concepts mentioned in the query, or other concepts related to them in the underlying ontology. They are ranked according to their relevance to the query – regulations featuring a larger number of concepts from the query, and particularly uncommon concepts, are considered most relevant. Further information and explanation can be obtained in two ways. Firstly, the user can click on a specific regulation name to view the full text of the corresponding regulation in a separate window:

> *MARPOL-AI-P2-14-01*:
>
> Except as provided in paragraph (2) of this Regulation, in new ships of 4,000 tons gross tonnage and above other than oil tankers, and in new oil tankers of 150 tons gross tonnage and above, no ballast water shall be carried in any oil fuel tank.

Secondly, concept names are also mouse sensitive, and when the user clicks on one, an explanation of why the concept was included in the answer is provided. For example, if the user clicks on dilution in the answer text given above, the following information is provided:

> The concept dilution is related as follows to the concept ballast water: Ballast water is water, water is a liquid substance and dilution is a measurable property of a liquid substance.

The **normative assessment** function attempts to determine whether a given situation is or is not *acceptable* according to the regulations.[7] The initial part of the query is a case description of the same sort used in the previous example. However, in a normative assessment enquiry the question itself is different:

[6] Here and below, regulations are referenced by section identifiers: *MARPOL-AI-P2-14-01* refers to the MARPOL convention, Annex I, part 2, clause 14, paragraph 01.

[7] CLIME is intended to serve an advisory role only, the ultimate judgement remains the responsibility of the surveyor.

> I have a new oil tanker. It has a gross tonnage of 2000 tonnes.
> It is fitted with a fuel oil tank. Ballast water is stored in
> the fuel oil tank.
>
> Tell me whether the regulations allow, disallow or are silent
> about the situation described.

The system responds with an answer which cross-references the regulations themselves, provides an explanation of its reasoning, and advises whether slight variations in the situation, for example the presence of certain equipment on board, would change the conclusion.

> This situation is disallowed. This is because:
>
> *MARPOL-AI-P2-14-01*: the fuel oil tank is part of the new oil
> tanker, the ballast water is located in the fuel oil tank and
> the gross tonnage of the new oil tanker is equal to
> or more than 150 tonnes.
>
> However: If an oily water separating equipment is part of
> the new fuel oil tanker, then the situation is allowed
> (*MARPOL-AI-P2-16-03*).

Here the first paragraph cites a MARPOL regulation and then explains why the regulation applies to the case described. The second paragraph offers a possibly relevant variation to the case. As before, regulation names are hyperlinks which display the corresponding regulation fragment when selected.

The intention, then, is that the system can be used either for simple rule retrieval from a list of concepts (a more sophisticated variant of 'keyword' lookup) or for the construction of a case description from which both rule retrieval and full assessment are possible. Additionally, as discussed further below, the entire interface can operate in either English or French.

2.2 Application requirements

The specification of a system such as CLIME is a delicate balance between technology-push and market-pull. As technologists, we may believe that we can draw together elements from legal reasoning, explanation and natural language interfaces into an application which *should* be a useful step forward for people accessing regulations. But if the proposal is sufficiently innovative that it does not relate very directly to existing practice, it can be difficult to elicit user requirements effectively: if the user has little experience of what is being offered, they find it difficult to say what they would like it to do.

This was the situation we faced in this case. The proposed system would offer more than simple text retrieval over the rules and not as much as a detailed telephone call to an expert, but it could be accessed more quickly (and cheaply). It was difficult to judge how useful such a facility would be. Nevertheless we set about obtaining user requirements for the system, but inevitably these tended to relate more to ergonomic and pragmatic issues than the system's intended core functionality. Two of the project partners had a direct interest in the application scenario for the maritime version of CLIME: Bureau Veritas, one of the largest ship classification and insurance agencies, who have a large body of proprietary regulations, and employ thousands of surveyors, and British Maritime Technology Ltd., one of the world's leading maritime and engineering consultancies. Through consultation with practising surveyors in these companies, the following requirements were established.

1. The user should be able to formulate semantically complex queries.
2. Given the legal nature of the application, the interpretation of the user's queries should have a high level of accuracy.
3. The system should be securely accessible from anywhere in the world.
4. When the system is computing an answer to the user's query, the user should be able to direct his/her attention to other tasks (including the formulation of further queries) and be able to modify and resubmit queries which were posed earlier.
5. The system is to be used in a world-wide operating company, which means that it should be adaptable to the language of the local users.
6. System responses should contain, or make available on demand, enough explanatory text to ensure the user correctly understands and has confidence in the answer given.
7. Given the ever changing world of maritime regulations, mechanisms should be in place for the maintenance of the system.

CLIME aims to meet these requirements in the following way (the relevant requirements are indicated in parentheses). The service is provided using a web-based client-server model: the user accesses the system by connecting to (and authenticating with) the CLIME server website (3). The system's user interface is then downloaded as a JAVA applet (3). Centralising the service also centralises the maintenance of the legal knowledge base, so that any updates to it are immediately accessible to all users (7). The top-level interaction dialogue is analogous to an email reading application: the user can construct queries, and submit them to the server, but does not wait for a response – the system informs the user when a response becomes available; meanwhile the user can construct (and submit) other queries, browse through previous queries, organise queries into subfolders, etc. (4). The actual construction of queries uses a WYSIWYM ('What You See Is What You Meant') interface (Power et al., 1998), which employs natural language generation (NLG) technology to support the user in reliably constructing semantically complex queries (1, 2). WYSIWYM also has the added benefit of supporting multilinguality in a powerful and natural way, using linguistic information derived from domain knowledge (5, 7). Finally, natural language generation is also used to construct the answer. This is also multilingual, sharing resources with the input NLG component; furthermore the answer is generated as part of the same discourse as the question, so that the overall result is coherent and fluent (5, 6).

2.3 System architecture

The CLIME run-time architecture is shown in Figure 1. The modules in the left-hand column are web-delivered JAVA applets, the other modules make up the CLIME

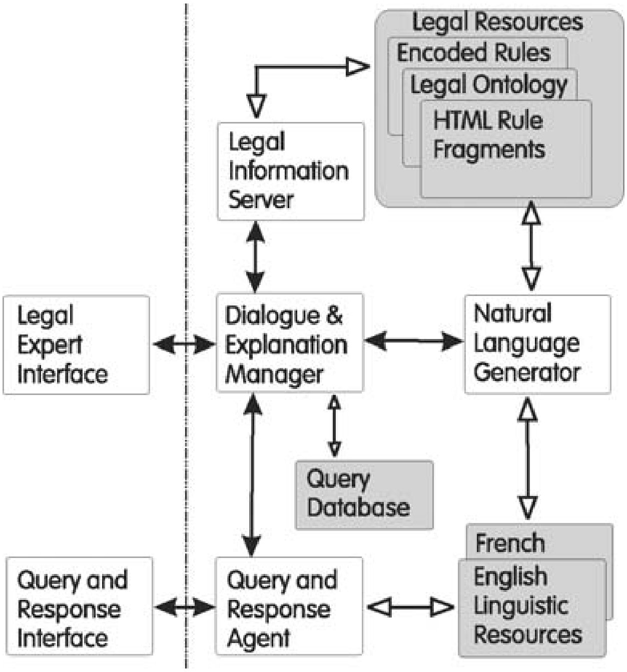
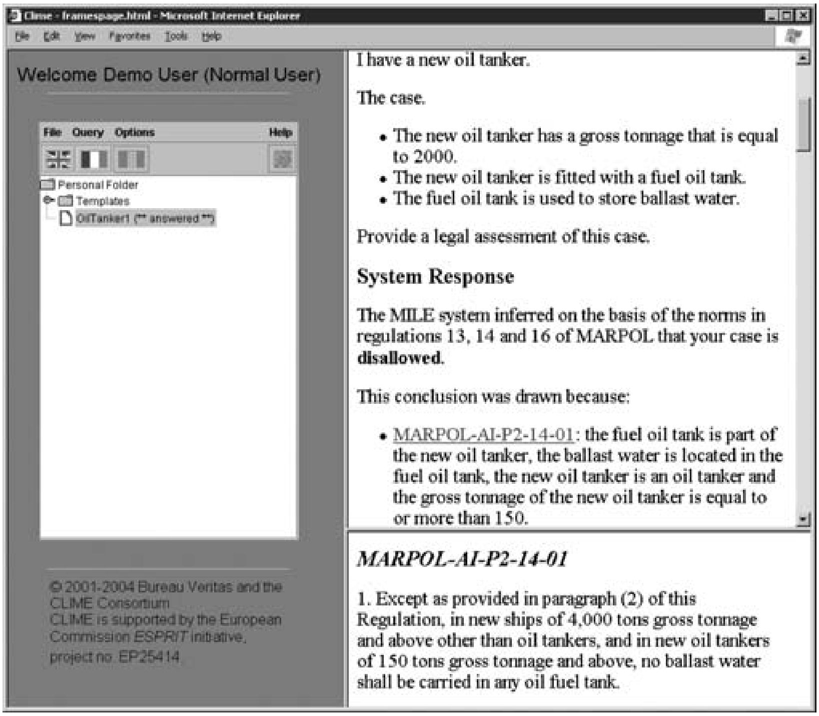


Fig. 1. The CLIME architecture.

server. This diagram does not show the additional off-line modules of the CLIME architecture, notably the *Legal Encoding Tools* used to encode regulations as formal rules.

The Query and Response Interface (QRI) is a JAVA applet providing query construction (using WYSIWYM) and management (browsing, filing, submitting). As an applet, it is relatively lightweight, and relies on a server-side module, the Query and Response Agent (QRA) for the heavier processing, notably WYSIWYM natural language feedback generation. Thus the QRI is really just a client-side presentation manager for the QRA. The Dialogue and Explanation Manager (DEM) provides the persistent database storage of queries and answers, manages the interactions with the user, and between the server modules, and provides explanation functionality. The Legal Information Server (LIS) is the engine that actually provides answers to questions, by reference to its knowledge base of formally encoded legal regulations which are linked to their source documents. The Natural Language Generator (NLG) is responsible for turning the LIS answers into natural language, potentially including explanations in a readable form. Finally the Legal Expert Interface (LEI) is a secondary web-based interface to the system, allowing a legal expert to manually browse and insert answers into the system database if the system is unable to provide the answer itself.

The overall operation of the system is as follows: the user manipulates the interface provided by the QRI, supported by the QRA, to construct a query. When the query is complete, the user submits it to the server – it is passed to the DEM, which stores it in its persistent query database, and then passes it to the LIS for legal processing. The LIS returns a response to the DEM, which processes the response to present most



Clime - framespage.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
File Query Options Help
Personal Folder
Templates
OilTanker1 (** answered **)

Welcome Demo User (Normal User)

© 2001-2004 Bureau Veritas and the CLIME Consortium
CLIME is supported by the European Commission *ESPRIT* initiative, project no. EP25414

I have a new oil tanker.

The case.

- The new oil tanker has a gross tonnage that is equal to 2000.
- The new oil tanker is fitted with a fuel oil tank.
- The fuel oil tank is used to store ballast water.

Provide a legal assessment of this case.

System Response

The MILE system inferred on the basis of the norms in regulations 13, 14 and 16 of MARPOL that your case is **disallowed.**

This conclusion was drawn because:

- MARPOL-AI-P2-14-01: the fuel oil tank is part of the new oil tanker, the ballast water is located in the fuel oil tank, the new oil tanker is an oil tanker and the gross tonnage of the new oil tanker is equal to or more than 150.

MARPOL-AI-P2-14-01

1. Except as provided in paragraph (2) of this Regulation, in new ships of 4,000 tons gross tonnage and above other than oil tankers, and in new oil tankers of 150 tons gross tonnage and above, no ballast water shall be carried in any oil fuel tank.

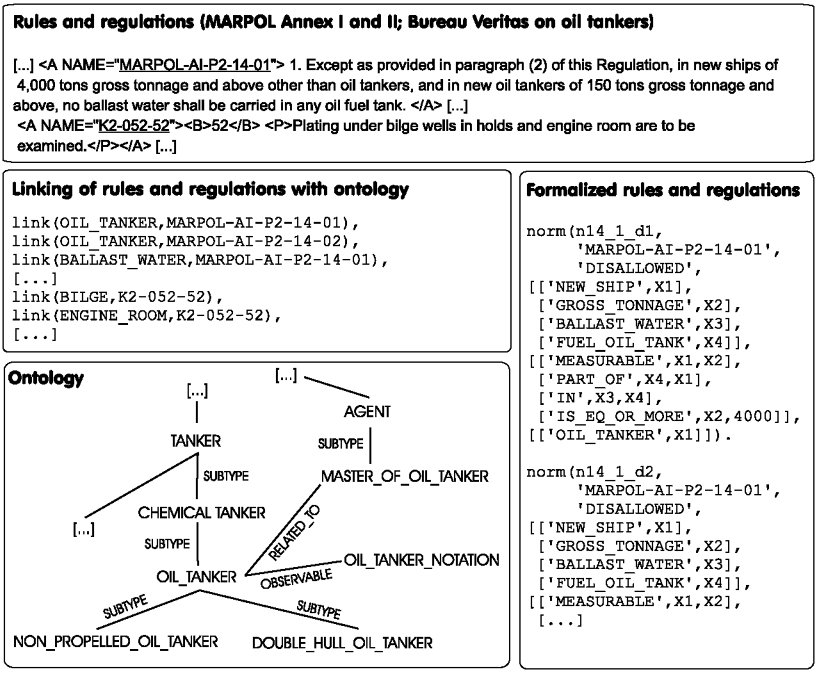
Fig. 2. The CLIME user interface.

relevant information first and incorporate explanatory material, and then passes it to the NLG. The NLG generates text, which it returns to the DEM. The DEM then notifies the QRA, and hence the QRI, that an answer is available, and the user can access it whenever they wish. If the LIS is unable to deliver an answer, the query is automatically emailed to a human expert. This expert then connects to the system using the LEI to insert a response into the database manually[8].

2.4 The CLIME *user interface*

The top-level user interface to CLIME is shown in Figure 2. This interface provides management and browsing functionality for existing queries stored in the central database, plus the ability to open the query construction interface to create and submit new queries. It is loosely modelled on conventional email-reading tools. The left-hand pane provides standard menus plus a tree-view which allows the user to navigate existing queries and folders (in the figure, the user has one subfolder, 'Templates', and one query, 'OilTanker1', which has been answered). The upper right-hand pane displays selected queries, together with their answers (once available) as HTML. Hyperlinks in the HTML answers can be used to display additional information (fragments of the regulation texts, extracted as part of the legal encoding process, and system-generated explanations) in the lower right-hand pane (in the figure, a fragment of the Marine Pollution regulations is shown).

[8] This aspect of the architecture is motivated primarily by deployment considerations, and was not explored in detail in the project.



Rules and regulations (MARPOL Annex I and II; Bureau Veritas on oil tankers)

[...] <A NAME="MARPOL-AI-P2-14-01"> 1. Except as provided in paragraph (2) of this Regulation, in new ships of 4,000 tons gross tonnage and above other than oil tankers, and in new oil tankers of 150 tons gross tonnage and above, no ballast water shall be carried in any oil fuel tank. </A> [...]
<A NAME="K2-052-52"><B>52</B> <P>Plating under bilge wells in holds and engine room are to be examined.</P></A> [...]

Linking of rules and regulations with ontology

```
link(OIL_TANKER,MARPOL-AI-P2-14-01),
link(OIL_TANKER,MARPOL-AI-P2-14-02),
link(BALLAST_WATER,MARPOL-AI-P2-14-01),
[...]
link(BILGE,K2-052-52),
link(ENGINE_ROOM,K2-052-52),
[...]
```

Formalized rules and regulations

```
norm(n14_1_d1,
     'MARPOL-AI-P2-14-01',
     'DISALLOWED',
[['NEW_SHIP',X1],
 ['GROSS_TONNAGE',X2],
 ['BALLAST_WATER',X3],
 ['FUEL_OIL_TANK',X4]],
[['MEASURABLE',X1,X2],
 ['PART_OF',X4,X1],
 ['IN',X3,X4],
 ['IS_EQ_OR_MORE',X2,4000]],
[['OIL_TANKER',X1]]).

norm(n14_1_d2,
     'MARPOL-AI-P2-14-01',
     'DISALLOWED',
[['NEW_SHIP',X1],
 ['GROSS_TONNAGE',X2],
 ['BALLAST_WATER',X3],
 ['FUEL_OIL_TANK',X4]],
[['MEASURABLE',X1,X2],
 [...]
```

Fig. 3. Conceptual encoding of shipping regulations.

The implementation of this interface as a web-delivered application is technically straightforward: the left-hand pane is an embedded JAVA applet which uses standard (SWING) interface elements and language localisation functions, and controls the other two panes which are simple HTML documents. The more interesting parts of the interface are the query construction interface (a free-standing applet window containing the WYSIWYM interface applet), and the server-side answer generation module, which creates the HTML to be displayed. These are described in the following sections.

3 Natural language processing in CLIME

3.1 Conceptual resources

The natural language processing (NLP) components are built on a conceptual foundation derived from the body of regulations for the domain. The regulations were preprocessed to provide (a) an ontology for the domain of shipping regulations, using familiar ontological relations such as `subtype-of`, `has-part`, `described-by`, `connected-to`; (b) mappings from ontology concepts to the regulations (fragments of regulation source text) that refer to them; and (c) encoding of the regulations as formal inference rules which can be used to 'apply' the regulations to particular cases. Figure 3 illustrates these components for a sample regulation. The encoding process is described in detail in Boer, Hoekstra and Winkels (2001). The final ontology contains 3377 concepts, 11897 relations (including supertype and subtype), with 8289 reference links between concepts and regulations. As well as using the acquired concepts, the inference systems is able to reason about numbers in a limited

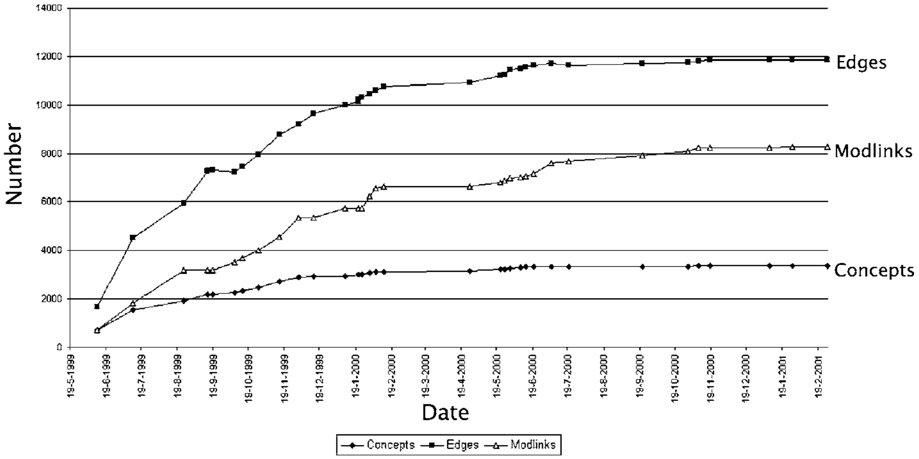


Fig. 4. Profile of ontology development (from Boer *et al*, 2001).

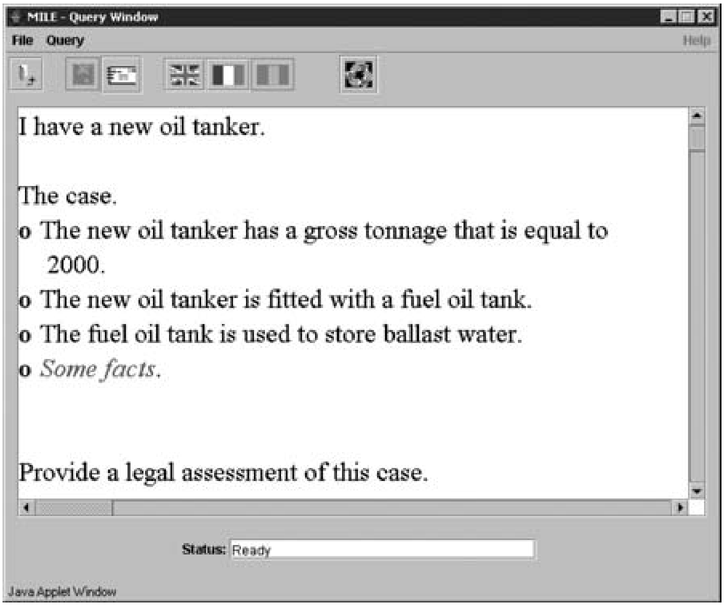
way, using predicates such as `is-eq-or-more`. Although only 15% of the Bureau Veritas rules were modelled, the number of concepts stabilised early in the process (see Figure 4). If we believe that many of the most frequently occurring concepts are common to all the regulations, this suggests that the ontology actually provides a good basis for covering a much greater proportion of the rules.

These conceptual resources support the two inferential processes the system can undertake: conceptual retrieval searches the ontology for concepts related to specified query concepts (and returns those concepts, plus links to the corresponding regulations), while normative assessment applies the legal inference rules to a case description, to determine what the regulations say about the case.

3.2 Constructing queries

The principal input task for the CLIME user is the construction of an input query for the Legal Information Server (LIS) module (leaving aside the relatively straightforward query-management actions discussed above). The two kinds of query, conceptual retrieval and normative assessment, are both entered using the same interface, which constructs a linguistically-oriented internal representation of the query. After this, processing of the two types of query splits: the internal representation is mapped into two different types of query to the LIS, which are processed as legal queries in different ways and generate different kinds of answer. In this section we describe the initial common task of query construction; the following two sections describe the subsequent processing for each query type.

CLIME uses a WYSIWYM interface for the linguistic construction of queries (Power *et al.*, 1998). WYSIWYM first appeared in the literature as a means of producing documents in several languages in parallel. The underlying approach was to produce a document by creating a language-neutral representation of the document's content and then using NLG technology to generate the document in several languages. The key novelty of WYSIWYM was that the NLG technology is not only used to generate



I have a new oil tanker.

The case.

- The new oil tanker has a gross tonnage that is equal to 2000.
- The new oil tanker is fitted with a fuel oil tank.
- The fuel oil tank is used to store ballast water.
- *Some facts.*

Provide a legal assessment of this case.

Status: Ready

Fig. 5. The ‘new oil tanker’ case description query. The use of italics in the last bullet point indicates a clickable span of text where further modification (addition of more information) is possible.

the final document but *also* as an interface for constructing the language-neutral representation itself.

In CLIME we have taken this idea one stage further: we use the same NLG techniques as an interface to create the language-neutral representation, but our main goal is not generation into other languages (although the approach does still support this, so that the CLIME query interface localises to other languages ‘for free’). Instead, we use WYSIWYM to support the precise specification of the internal semantic representation of the displayed text, which can then be readily transformed into the form required for the LIS. The main interface task for the system is the construction of a ‘case’, that is, a description of a state of affairs on a ship which the user would like further information about. Figure 5 shows the query interface presentation of the case description corresponding to the ‘new oil tanker’ example introduced above. The user can select spans of text and further develop or modify them, but the editing options are strictly controlled by an underlying semantic model. Each change made is presented back to the user as a new description text, with new selectable spans to continue the creation of a complete description. This allows the user to control very directly the representation which the system has of the query, ensures the user and the system have the same interpretation of the query (e.g., the user directly controls coreference, thus avoiding ambiguous references; see Appendix 1 for an example), and prevents the user from entering queries for which the system cannot build a representation. Here there is a marked difference with systems based on free text input. Such systems typically rely on shallow analysis of the query due to the fact that reliable deep semantic analysis of free text is not yet possible. As a consequence, such

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{assessment\_enquiry}} \\
\text{INTRO} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{ship\_intro}} \\ \text{ARG1} & \left[\textit{new\_oil\_tanker}\right]\boxed{1} \end{array}\right] \\
\text{PROP} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{case\_proposal}} \\
\text{SIT} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{set(state)}} \\
\text{FIRST} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{state\_fact}} \\
\text{ATTR} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{gross\_tonnage}} \\
\text{ARG1} & \boxed{1} \\
\text{ARG2} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{is\_equal}} \\ \text{ATTR} & 2000 \end{array}\right] \end{array}\right] \end{array}\right] \\
\text{REST} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{set(state)}} \\
\text{FIRST} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{state\_fact}} \\
\text{ATTR} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{fitted\_with}} \\
\text{ARG1} & \boxed{1} \\
\text{ARG2} & \left[\textit{fuel\_oil\_tank}\right]\boxed{2} \end{array}\right] \end{array}\right] \\
\text{REST} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{set(state)}} \\
\text{FIRST} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{state\_fact}} \\
\text{ATTR} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{space\_contents}} \\
\text{ARG1} & \boxed{2} \\
\text{ARG2} & \left[\textit{ballast\_water}\right] \end{array}\right] \end{array}\right] \\
\text{REST} & [\,] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \\
\text{QUEST} & \left[\textit{acceptable}\right]
\end{array}\right]$$

Fig. 6. Attribute-value matrix encoding of the query in Figure 5. Items in italics are types, and boxed numbers represent re-entrancy in a conventional fashion.

systems have no access to a deep underlying semantic representation (specifying, amongst other things, whether a fact is negated or not and which expressions in the query are about the same object, i.e, coreference).

The underlying semantic representation corresponds to a typed directed acyclic graph. The example in Figure 5 is equivalent to the attribute-value matrix encoding of such a graph shown in Figure 6. This internal representation can be used to generate both normative assessment and conceptual retrieval queries, as discussed in the following sections. The system also provides a simpler interface for conceptual retrieval queries (which are in essence just lists of concepts), which still uses the WYSIWYM architecture (using very simple generation), but avoids the need to construct a complete case description.

A detailed example of the use of WYSIWYM to construct this query can be found in Appendix 1. Finally here, we note briefly the main extensions to WYSIWYM embodied in CLIME:

- the application of WYSIWYM to the construction of formal queries;
- support for navigation through larger ontologies (over 3000 concepts);
- support for plurals and groups of objects (Piwek, 2000);
- web delivery using a JAVA applet.

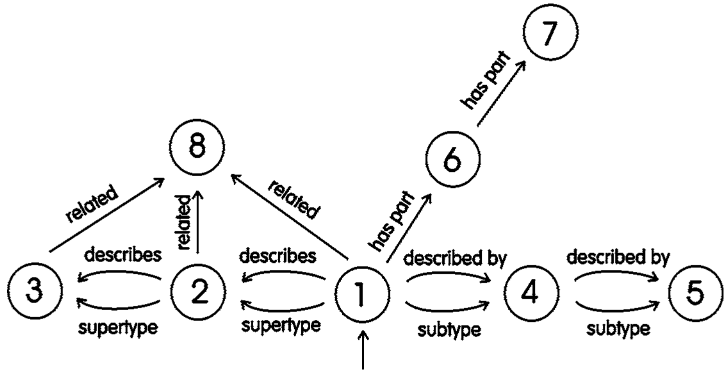


Fig. 7. An example ontology search automaton for Conceptual Retrieval.

3.3 Conceptual retrieval queries

Creating the legal information server input

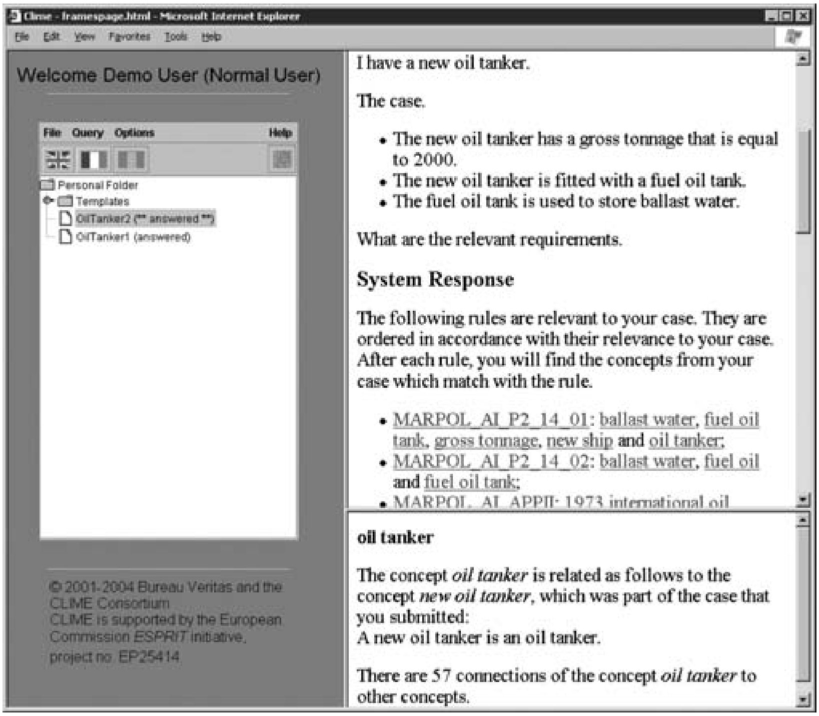
A conceptual retrieval query to the LIS is a set of initial nodes in the ontology, plus a search pattern for searching the ontology from these nodes. Search patterns are represented as finite state automata, as shown, for example, in Figure 7. In the CLIME demonstrator, we use a fixed automaton (slightly more complex than the one shown in Figure 7); so to construct a query the user needs only to provide a list of initial concepts. This list can be either specified explicitly or extracted from a case description – for the example in Figure 5, the list would be `new-oil-tanker, gross-tonnage, fuel-oil-tank, ballast-water`.

Processing the legal query

Given a list of concepts and a search automaton, the LIS runs the automaton over the ontology, starting at each concept in the query in turn, and collects up all the concepts visited in the process. Thus for the automaton in Figure 7, it would return the concept, any supertypes or subtypes to a depth of two, any concepts describing or described by this concept to a depth of two, subparts to a depth of two, and any concepts related to the concept, its supertypes or concepts it describes.

The information returned by the LIS is as follows:

- the expanded set of concepts, found by following ontological links according to the search automaton;
- for each concept, the legal rules which refer to it;
- for each concept, a positive integer which represents the *centrality* of the concept – the number of links to the concept in the ontology (a measure of the concept's 'ordinariness');
- for each of the referenced rules, the number of concepts in the rule that were in the original query (a measure of the rule's relevance).


Clime - framespage.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
File Query Options Help
Personal Folder
Templates
OilTanker2 (** answered **)
OilTanker1 (answered)

Welcome Demo User (Normal User)

© 2001-2004 Bureau Veritas and the CLIME Consortium
CLIME is supported by the European Commission *ESPRIT* initiative, project no. EP25414

I have a new oil tanker.

The case.

- The new oil tanker has a gross tonnage that is equal to 2000.
- The new oil tanker is fitted with a fuel oil tank.
- The fuel oil tank is used to store ballast water.

What are the relevant requirements.

System Response

The following rules are relevant to your case. They are ordered in accordance with their relevance to your case. After each rule, you will find the concepts from your case which match with the rule.

- MARPOL_AI_P2_14_01: ballast water, fuel oil tank, gross tonnage, new ship and oil tanker;
- MARPOL_AI_P2_14_02: ballast water, fuel oil and fuel oil tank;
- MARPOL_AI_APPII: 1973 international oil

oil tanker

The concept *oil tanker* is related as follows to the concept *new oil tanker*, which was part of the case that you submitted:
A new oil tanker is an oil tanker.

There are 57 connections of the concept *oil tanker* to other concepts.

Fig. 8. A conceptual retrieval answer. Clicking on a rule name displays the corresponding fragment of the source regulations (MARPOL); clicking on a concept, as shown here, displays a 'trace' sentence which explains how this concept is related to the original query.

Generating the response

This information undergoes the following transformations to produce textual answers:

1. The rules are ranked according to relevance to the user's query. Ranking depends on how many query concepts the rule refers to, and how central those concepts are (more central concepts are more common and so less relevant).
2. The top 30 rules in the ranked order are selected and the rest discarded.
3. For each concept in both the expanded set and these remaining 30 rules, a trace of how the concept is linked to concepts in the query is calculated, e.g. if the query contains `water` and the expanded set contains `liquid`, the fact that `water` is a subtype of `liquid` would be added.
4. Sentences describing each trace relation are generated – these serve as explanations of why the concept is included in the answer (see figure 8 for an example).
5. The answer is composed using a simple template-based approach: the query is repeated, followed by a canned text introduction to the answer, then each rule name is listed, together with the concepts it contains. Rule names are hypertext links to the corresponding rule texts (from the original regulations), concept names are hypertext links to the corresponding explanatory trace relation sentences.

3.4 Normative assessment queries

Creating the legal information server input

For a normative assessment query, the LIS requires a predicate-logic representation of the case description such as the following (for the example in figure 5):

```
[[new-oil-tanker, i5], [gross-tonnage, i11],
 [ballast-water, i22], [fuel-oil-tank, i17]]

[[measurable, i5, i11], [is-eq, i11, 2000],
 [has-part, i5, i17], [in, i22, i17]]

[]
```

This representation has three components: a list of type declarations for variables (`i5, i11` etc.), a set of positive assertions and a set of negative assertions (empty in this case). The transformation into this form is straightforward but not a completely trivial rewrite. In general, the predicates correspond to semantic types and the identifiers correspond to argument subgraphs. However some semantic structure is present only for presentational/rhetorical purposes and is ignored (for example, the distinguished *ship_introduction role*, corresponding with the template 'I have *an S*', where *S* is a type of ship), while in other places the structure is simplified (for example, the semantic type *fitted_with* maps to the ontological relation `has-part`), or elaborated (for example, the semantic relation *gross_tonnage* becomes an instance of the ontological relation `measurable` whose argument is an instance of the ontological type `gross-tonnage`).

Processing the legal query

A normative assessment query is processed by attempting to match the case description with the encoded inference rules, then ranking the rules that match in order of priority and reading off the result from the highest priority match. Rules have two components: a *generic case*, which is a schema to match a case description, and a *normative status*, one of `allowed`, `disallowed` or `silent`, which specifies the result of the rule. (`silent` means the rule does not draw any definite conclusion, typically because it includes a phrase like 'Normally,...'.) The following rule matches the case description introduced above:

```
Generic Case:     [[new-ship, S], [gross-tonnage, T],
                   [ballast-water, W],[fuel-oil-tank, F]]

                  [[measurable, S, T], [has-part, S, F],
                   [in, W, F], [is-eq-or-more, T, 150],
                   [oil-tanker, S]]

                  []

Normative Status: disallowed
```

The generic case describes a class of individual cases through the use of free variables (represented here with capitals). An individual case is said to match a generic case if the individual case, together with a background theory (e.g., the ontology and arithmetic for reasoning about `is-eq-or-more`) entails a substitution instance of the generic case (i.e., the generic case with all its free variables replaced by constants).[9]

This rule states that a new ship which is an oil tanker, with a gross tonnage greater than or equal to 150 tonnes, which has ballast water in its fuel oil tank is a disallowed situation. It matches the specific case through ontological matching of the `[new-oil-tanker, i5]` instance with both `[new-ship, S]` and `[oil-tanker, S]`, and arithmetic matching of `[is-eq, i11, 2000]` with `[is-eq-or-more, T, 150]`.

The LIS attempts to apply all its rules to the case supplied in this way and returns the following results:

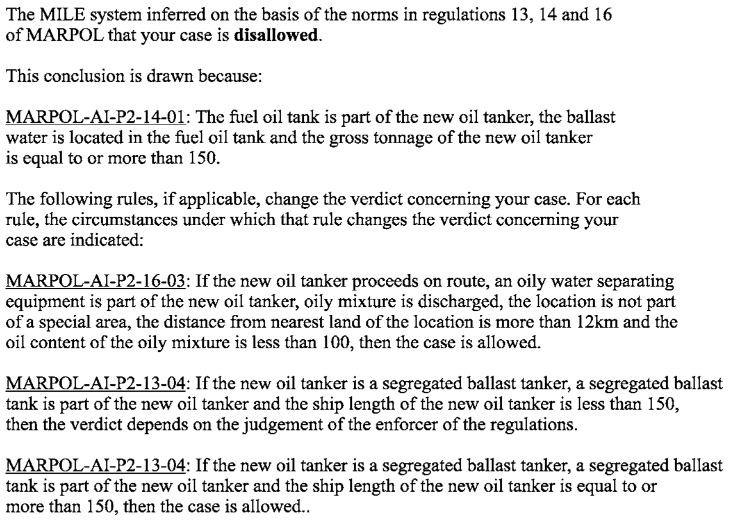
- a list of inference rule applications which apply to the case, each consisting of (a) the rule name (b) the facts in the situation which triggered the rule and (c) the status which the rule assigns to the case – allowed, disallowed or silent;
- a partial ordering over the rule applications according to legal precedence – more specific, or more recent, rules take precedence;
- a list of ‘continuations’: instances of rule applications that would apply, and would change the overall result, if a small change was made to the case description.

Generating the response

An example of a typical normative assessment answer can be found in Figure 9. The mapping from the LIS output to HTML is achieved in the following stages:

1. Using the partial order for legal precedence, the system determines whether the situation in question is allowed or not (this is the actual answer to the query) and which rules support, contradict or say nothing about this conclusion.
2. A subset of rule applications is selected to be expressed, namely those which contribute to the conclusion (the highest precedence rules), and those directly overruled by them.
3. Each of these selected rule applications and each continuation is transformed into a textual form describing the circumstance which caused the rule to be applicable. This is the most significant ‘real’ NLG the system undertakes, with simple aggregation of predicates with the same subject and referring expression generation.
4. Finally, the whole answer document is pieced together using high level templates and transformed into HTML. The final answer consists of the conclusion, the rules supporting the conclusion, the rules against but overruled

[9] See Valente (1995) for further technical details.



The MILE system inferred on the basis of the norms in regulations 13, 14 and 16 of MARPOL that your case is **disallowed**.

This conclusion is drawn because:

MARPOL-AI-P2-14-01: The fuel oil tank is part of the new oil tanker, the ballast water is located in the fuel oil tank and the gross tonnage of the new oil tanker is equal to or more than 150.

The following rules, if applicable, change the verdict concerning your case. For each rule, the circumstances under which that rule changes the verdict concerning your case are indicated:

MARPOL-AI-P2-16-03: If the new oil tanker proceeds on route, an oily water separating equipment is part of the new oil tanker, oily mixture is discharged, the location is not part of a special area, the distance from nearest land of the location is more than 12km and the oil content of the oily mixture is less than 100, then the case is allowed.

MARPOL-AI-P2-13-04: If the new oil tanker is a segregated ballast tanker, a segregated ballast tank is part of the new oil tanker and the ship length of the new oil tanker is less than 150, then the verdict depends on the judgement of the enforcer of the regulations.

MARPOL-AI-P2-13-04: If the new oil tanker is a segregated ballast tanker, a segregated ballast tank is part of the new oil tanker and the ship length of the new oil tanker is equal to or more than 150, then the case is allowed..

Fig. 9. An automatically generated normative assessment answer.

by those supporting, any rules which apply but draw no conclusion, and any continuations (of the form "If ... were also the case, the conclusion would have been ...").

The mapping from the input generated by the LIS to the answer HTML conforms roughly to the pipeline paradigm in NLG as described in, for instance, Reiter & Dale (2000). We have a *document planner* (steps 1 and 2 above) performing content determination and document structuring; a *micro planner* doing both text specification and linguistic realisation (step 3) and a *surface realiser* which renders the text in HTML format (step 4). Here, we shall focus on the text specification and linguistic realisation step in the CLIME system, where individual rule applications are realised.

In a first pass, the LIS representation of the case is mapped to a semantic representation whose predicate argument structure closely mimics the predicate argument structure of the natural language sentences to which it will eventually be mapped:

```
[[is-eq-or-more, [gross-tonnage, [i5]], 150, pos],
 [has-part,[i5], [i17], pos],
 [in, [i22], [i17], pos],
 [oil-tanker, [i5], pos]]
```

The group of conceptual facts and type declarations (`[gross-tonnage, i11]`, `[measurable, i5, i11]` and `[is-eq-or-more, i11, 150]`) now corresponds with one semantic fact (`[is-eq-or-more, [gross-tonnage, [i5]], 150, pos]`). Furthermore, each fact now carries the information whether it is positive or negative (i.e., negated). Next, the semantic relations in the semantic facts are linguistically realised:

```
[[the, gross tonnage, of, np([i5]), is equal to or more than, 150],
 [np([i17]), is part of, np([i5])],
 [np([i22]), is located in, np([i17])],
 [np([i5]), is, an, oil tanker]]
```

At this point a simple aggregation algorithm is applied which collects facts which share a subject (e.g., `[np([i1]),R1],[np([i1]),R2]` and `[np([i1]),R3]` into one fact (i.e., `[np([i1]), R1, R2 and R3]`). Thus, for instance, 'The ballast tank is part of the new oil tanker.' and 'The ballast tank is not a segregated ballast tank' becomes 'The ballast tank is part of the new oil tanker and is not a segregated ballast tank'.[10]

Finally the referring expression generation module replaces expressions of the form `np([Index`$_1$`,...Index`$_n$`,])` with indefinite or definite noun phrases depending on whether we are dealing with a first or a repeated reference to an index, respectively. Referring expression generation is applied to the indices in the order in which they are referred to in the text. During this process a store is maintained which records which indices have been mentioned in order to support the choice between definite and indefinite noun phrases. The store is initialised with the indices which were mentioned in the query. For this purpose, the formal representation of the query is part of the input to the answer generation. Thus, if an oil tanker has been introduced in the query, it will be referred to in the answer with the definite noun phrase 'the oil tanker'. The store also maintains information on how recently an index has been referred to. This information can be used to choose a pronominalisation of a definite noun phrase. The descriptive material of the noun phrase is obtained from the type declarations which are a part of the rule applications. The model can be adapted to cater for more sophisticated models of salience in reference generation as proposed by, for instance, Krahmer & Theune (2001).

3.5 The CLIME *dialogue model*

The dialogue model which underlies the CLIME system was devised to address the requirements of knowledge-intensive advisory dialogue. Such dialogues are essentially hybrid: they involve both synchronous interaction typical of face-to-face conversations and asynchronous interaction as found in written or email communication. The principal steps of an advisory dialogue are asynchronous, as the advising party will often not be able to produce an answer instantaneously – because they need to consult background knowledge, or their expertise is needed elsewhere. The peripheral steps, namely the specification of the question and clarification of the answer, are ideally synchronous. In CLIME a similar situation arises. The processing time responding to a query is potentially too long for convenient synchronous dialogue (i.e., one minute or more) and the system may need to consult a human expert by email. So asynchronous communication is acceptable, indeed stipulated as a system requirement. However, during query construction, interaction with answers (clarification and explanation), and (in a multilingual context) language changing operations, the system must provide a fast, synchronous response.

As discussed in section 2, the asynchronous aspects of the CLIME dialogue are modelled at the user-interface level rather like a conventional email system. The

[10] The placement of aggregation after linguistic realisation allows for predicate aggregation but is not well suited for argument aggregation, because the latter can affect morphological agreement, which may have already been realised. This suggests that earlier aggregation (as recommended by, for example, Reape & Mellish, 1999) might be better.

user constructs queries ‘off-line’ and submits them but does not wait for an answer to be returned. Subsequently the system notifies the user if any new responses (to this or other queries) are available, and the user picks up the answer at his/her own convenience. The system response includes both the query and the answer – in effect the whole dialogue history (cf. Piwek *et al.*, 1999) – so that the user has an appropriate context for interpreting the answer and constructing follow-up queries.

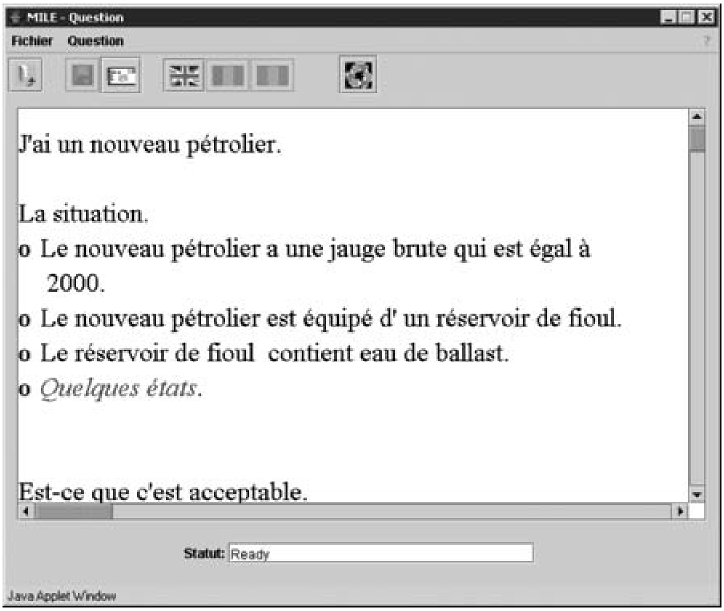
Synchronous responses, however, are technically more problematic, due to the relatively heavyweight natural language and legal reasoning technology that is deployed. At the implementation level, there are actually three levels of dialogue response, corresponding to depth of processing within the architecture:

1. Immediate response to user actions (such as popping up a menu) are handled directly by the QRI, an applet running in the user’s own web browser;
2. Synchronous dialogues as described above ideally involve the QRI interacting with the QRA and possibly the DEM (for browsing the query database) over the web link.
3. Asynchronous dialogues ideally occur when the DEM stores a query in its database, returns a synchronous acknowledgement, and then invokes the LIS and/or NLG ‘off-line’ to respond to a query.

But this model potentially breaks for some types of system interactions, because the required functionality is in the wrong part of the system. The main instances of this, and our technical solutions to them, are as follows:

- During query construction, the QRI needs to pop-up menus which are context-specific, defined by the QRA, and potentially large (and so slow to download to the browser). To minimise delays, the system uses a menu representation in which variability is carefully factored, and non-variable components are cached in the QRI whenever possible.
- Some explanation responses need to access the LIS/NLG components of the system synchronously. We overcome this by taking advantage of the asynchronous expectation of the initial response – we precompute explanation responses along with the original answer (even though this takes a little longer), so that they are already embedded in the HTML representation delivered to the QRI.
- Changing language requires the NLG module to compute the response in the new requested language. Again, we overcome this by precomputing the response in both supported languages at the outset.

These solutions are pragmatic rather than the most flexible or efficient: indeed for simple queries the response time may well be dominated by such ‘second-guessing’ of the user’s future requirements which might all be wasted. However, a system which undertakes non-trivial natural language processing needs to consider response time issues seriously, and making use of ‘slack-time’ in the overall dialogue to precompute potentially useful data may be a useful practical approach.



J'ai un nouveau pétrolier.

La situation.

o Le nouveau pétrolier a une jauge brute qui est égal à 2000.

o Le nouveau pétrolier est équipé d' un réservoir de fioul.

o Le réservoir de fioul contient eau de ballast.

o *Quelques états.*

Est-ce que c'est acceptable.

Statut: Ready

Fig. 10. A CLIME query in French

3.6 Multilinguality and linguistic resources

Multilinguality in the CLIME system is viewed as interface localisation: the internal representations of queries and answers are language-neutral, and they are simply rendered in a language-specific way according to the current language setting of the interface. This allows great flexibility in the use of different languages when interacting with the system: a question can be posed in one language, and the answer viewed in another, previous questions and answers can be browsed in any supported language, not just the one they were posed in, in fact the user can change language half way through creating a query if desired (for example, if he/she decides to involve a colleague in the interaction).[11] Figure 10 shows the French version of the example query from Figure 5.

To achieve this functionality, language-specific resources are accessed by three modules of the system: the QRI, the QRA and the NLG. As a standard JAVA SWING applet, the QRI uses SWING resource bundles to localise around 70 fixed displayed strings (labels on buttons, menus etc.) to the current language. The contents of the query and answer windows are also language-specific, but their localisation is managed by the QRA and NLG modules respectively; the QRI simply receives and displays content data. The QRA is responsible for the WYSIWYM query window. It uses English and French unification-based grammars to generate text representations of the current (language-neutral) query state, including the menu options associated

[11] This model is not fully implemented in the demonstrator system: the answers to normative assessment queries are only implemented in English. The displayed fragments of the source regulations are also only in English – a French version of the regulations does exist, but was not available in suitably indexed form.

with the WYSIWYM anchors. The grammars are mainly hand-crafted, but augmented by domain-specific lexicons derived semi-automatically from the domain ontology (see below). The QRA regenerates the entire query text every time the query is changed, or whenever the current language changes. The NLG module generates responses using a combination of hand-crafted templates and grammars, plus the same domain-specific lexicons used by the QRA. As noted in the previous section, for pragmatic reasons the NLG generates both English and French versions of responses in advance, so language localisation is simply a matter of choosing which to display.

The domain-specific lexicon is derived from the domain ontology described in section 3.1. The lexicon, containing around 3100 lexical items, was compiled from an HTML representation of this ontology. It includes concept names in English (which may be phrasal – "oil tanker", "1973 international oil pollution prevention certificate") and subclass relationships ("oil tanker" is-a "tanker" is-a "ship"), which are needed to drive the WYSIWYM interface: an anchor representing an instance of "ship" can be expanded only to subclasses of "ship". From the concept names, French translations (including gender information) were added manually by domain experts, since very few of the terms occur in a standard machine-readable dictionary. The entries were then organised into an inheritance hierarchy, linking automatically generated entries into a hand-crafted abstract class hierarchy to distribute additional syntactic information (major category, syntactic form, etc.). This inheritance hierarchy supported both languages, with English acting as a default where no French form was available (or desired, as in the case of many standard domain-specific abbreviations).

Subclass information from the domain ontology was also added to the lexical hierarchy, and used as the basis for assigning mass/count features to concepts. However, this approach was not very successful – although semantic class considerations are clearly helpful for deciding mass/count distinctions, there are many lexically conditioned exceptions to be accounted for manually.

The final lexicon was then used to generate the on-line lexicon formats for the QRA and NLG generators. Further details of the entire lexicon derivation process can be found in Cahill (2000).

4 Application development

In this section we summarise the overall application development and how it influenced the design and implementation of the natural language subsystems.

The initial system concept was an example of 'informed' technology-push: software engineers with knowledge of the maritime industry believed that (a) a legal advisory system of this sort was technically possible and (b) the maritime domain offered a plausible application scenario – the proposed system would offer more than simple text retrieval from the regulations, and less than a detailed telephone call to an expert, but could provide a very effective intermediate service. Detailed specification of the system, however, ran into classic technology-push problems. High level requirements (such as "the system should support semantically complex queries, be accessible from anywhere in the world, be multilingual, be asynchronous and provide explanations") could be established through consultation with potential users, and have largely been met in the overall system architecture. However it was difficult to develop more detailed specifications because the system was not an obvious development of anything the users already used, except interaction with human experts.

This was a particular issue for the natural language components. Even the standard starting point of a set of example queries and responses to target was difficult to elicit at an appropriate level of expertise – we started with about eight possible (fictional) queries and over the entire project we were able to collect a corpus of just 120 queries (mainly from experts filling in pro-formas relating to actual questions), and even these varied widely in form, complexity and topic. This made it difficult to build early demonstrators, especially of the input interface which relies on having a precise model of, and grammar for, queries. Similarly, eliciting appropriate feasible requirements for explanation support was problematic, since the users' only experience of explanation was with a human advisor. Not all requirements suffered these problems, of course. In particular, very useful analysis of the problems of concept navigation took place, resulting in a clear preference for simple alphabetical listing with a substring search facility, rather than the advanced ontological tree-search tool the project had originally envisaged.

With the initial specification established as far as it could be, implementation progressed on a cyclical basis, delivering successive demonstrators and receiving informal feedback and formal evaluation to feed into subsequent development (details of this process are given in Piwek (2002)). There were about four to five iterations of this cycle, but three main phases of development can be identified:

1. the development of a normative assessment pilot based on the initial requirements;
2. the addition of conceptual retrieval support, primarily to quickly increase coverage to demonstrate the effectiveness of the overall model;
3. development of full normative assessment and consolidation into a final integrated whole.

What is interesting about this sequence is the extent to which it is driven by user perceptions. The initial specification for the project involved only normative assessment queries, but the early demonstrators were hampered by a lack of coverage of the legal domain (because encoding regulations into inference rules, while undertaken in parallel to building the demonstrator, was a much slower process) and the lack of a clear model of possible queries. This led to mixed user reactions when the system was subjected to evaluation (see section 5). The introduction of conceptual retrieval was seen by the project's user partners as a solution to these problems: coverage could be extended more quickly (because legal encoding just for conceptual retrieval is much easier), and the specification of queries became much simpler. And because the entire enterprise was in novel territory for the users, the difference in *utility* between the two query types was not perceived as a significant issue – they could see apparent added-value in conceptual retrieval, but could not immediately appreciate the additional potential of normative assessment.

The third phase of the project began when the difference between conceptual retrieval and normative assessment was finally appreciated. The conceptual retrieval demonstrators, though successful, were not adding as much value as anticipated (scarcely more, to the untrained eye, than ordinary information retrieval over the original regulations), and effort was redirected back to legal encoding for normative assessment, though only for a small fragment of the regulations.

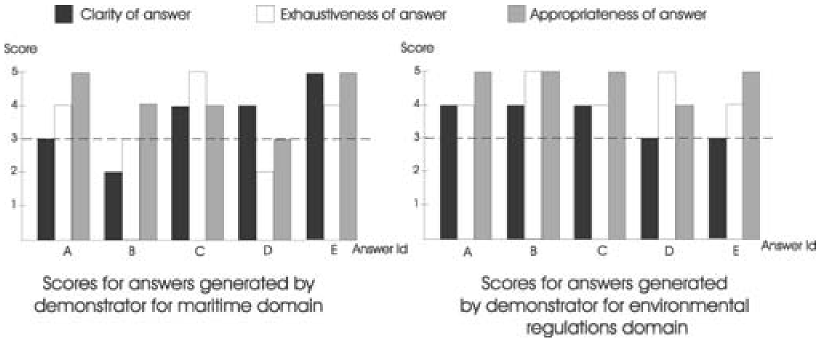
The impact of this vacillation on the development of the NL components was felt in three areas. The decision to adopt WYSIWYM required a clear grammar for queries and not having one early in the project led to obvious evaluation difficulties. The introduction of new functional requirements undermined some of the rationale for the whole architecture (conceptual retrieval requests do not really require advanced NL input interfaces, nor asynchronous dialogue models). Supporting answer generation for two completely different answer types, with different complexities and explanation requirements, made the development of good generic solutions almost impossible. The result, of course, is the somewhat fragmented system described in this paper. Interestingly, however, there was not really a problem in an area one might expect: lexical coverage. As discussed above, the bulk of the lexical information in the system is derived automatically from the legal encoding resources, so it was relatively straightforward for lexical coverage to keep pace with encoding (although convincing users that a problem was one of legal rather than lexical coverage was sometimes another matter).

5 Evaluation

The overall project structure and execution meant that it was not possible to undertake as thorough evaluations of the system as we would have wished. In particular, from the perspective of this paper it would have been interesting to evaluate the linguistic components independently of their embedding within the complete system, but the development and non-academic pressures of the project precluded this. Nevertheless a number of small-scale evaluation studies of different aspects of the system were carried out at various points in its development. In this section we summarise the main conclusions – further details are provided in Piwek (2002). First, we describe two evaluations of the system from a user-interface perspective.

An early prototype of the complete end-to-end system was the subject of a qualitative evaluation of basic usability by the software development partner in the project, which was reported in Bertin & Bagnato (1999)[12]. The gist of their report was that the query interface is easy to learn and use for anybody already familiar with the MS Windows desktop. They suggest that the reason for this is that the WYSIWYM query formulation interface offers the same direct manipulation possibilities as the common windows desktop.

[12] Two of the project-internal reports are restricted circulation (Bertin & Bagnato, 1999; Bertin, Bagnato & Lorenzon, 2001) due to the commercial nature of the project consortium. Piwek (2002) draws together all the evaluation results from these documents and other sources in a single publically accessible report.



Scores for answers generated by demonstrator for maritime domain

Scores for answers generated by demonstrator for environmental regulations domain

Fig. 11. Results of the conceptual retrieval answer evaluation exercise by teams of domain experts from Bertin, Bagnato & Lorenzon (2001).

A more mature version of the system was evaluated by the user partner, using a more quantitative questionnaire-based approach, focusing on each of the two main system interfaces – the browser window and the query window – separately. Each interface was scored on a range of dimensions (such as screen layout, terminology used, predictability of response, speed of response, system messages, ease of use, number of steps required, logical organisation). Overall the browser window scored well except on predictability and speed of response. These were in part caused by coding inefficiencies and bugs which caused the system to stall or crash from time to time, and which were resolved in subsequent prototypes. The query window scored well on ease of use, clarity of system messages, number of steps required and logical organisation, but less well on speed and ability to construct queries – primarily due to lack of ontological coverage (and also the users' lack of knowledge of the ontological coverage).

Evaluation of the system from a functional point of view was also undertaken, but only for conceptual retrieval queries – the normative assessment capabilities of the system were only developed to a proof-of-concept implementation level which was not suitable for detailed functional evaluation. Conceptual retrieval functionality was in fact evaluated in two domains: in addition to the original maritime domain (in English), a second domain of environmental health regulations (in Italian) was also coded for conceptual retrieval.[13]

For each domain, domain experts formulated five queries, each consisting of a single concept to retrieve regulations for, with varying degrees of complexity (measured by the number of rules and other concepts the query concept was related to). The system output for these queries was carefully studied by five experts who agreed a score on a scale from 1 to 5 (1 = very low, 2 = low, 3 = acceptable, 4 = good, 5 = very good) regarding the clarity, exhaustiveness and appropriateness of the answers. The results can be found in Figure 11, where query 'A' is the

[13] The effort involved in porting the CLIME platform to this second domain consisted of 1 person year and resulted in a system for environmental regulations with an ontology containing 543 concepts, 497 references and 1218 edges.

Date	TREC requirement	CLIME functionality
Year 1	Answer scattered across two or more documents and may need to be fused into one response Answer no longer guaranteed to be present	CLIME does not use document text directly in responses, but does combine different aspects of the response (answer, justification etc.) into a single coherent text. Question construction is controlled so that 'out of coverage' issues do not arise, but CLIME does cope with the case where the no regulations apply to a situation, offering 'near miss' responses (continuations).
Year 2	Questions are posed within a context	A major feature of CLIME is the construction of context, typically a ship description, for a query. Primitive dialogue support, in the form of shared common ground between question and answer, is also provided, and in principle supports detailed follow-up questions.
Year 3	Use text generation for answer, with explanation and justification	This is one of the key natural language components of CLIME.
Year 4	Requests for summaries	CLIME does not directly process documents, so summarisation is not appropriate. However responses involve a high degree of information organisation and aggregation, in much the same way summaries do.
Year 5	Expert-level questions	CLIME presupposes a medium-to-high level of domain expertise in the user, and the query and response interfaces are designed accordingly.

Fig. 12. Comparison of CLIME with TREC QA roadmap.

most complex (for example `existing_ship`), and query 'E' is the least complex (for example `auxiliary_engine_propulsion`) in each case. Note that all the scores of the environmental regulations demonstrator are equal to or above score 3 (= acceptable). Regarding the maritime demonstrator, there are only two cases where the score is below acceptable.

6 Conclusions

When one builds a question answering/dialogue system it is tempting to take ordinary face to face conversations as the reference model. Although for many applications this might be appropriate (e.g., train, theatre, hotel or flight information), there are also important applications involving, for instance, knowledge intensive advisory dialogue, where asynchronous or hybrid synchronous/asynchronous dialogue models are more appropriate. Between people, synchronous communication is often face to face or by phone, while asynchronous communication is often achieved using email. In person/machine communication, the phone may be replaced by a web interface, but the email analogy remains a viable model for asynchronous dialogue. The system we have described here uses this combination of web and email-like dialogue to support a complex, hybrid, knowledge-intensive question-answering application. Use of NLG technology allows the system to maintain a degree of coherence across the dialogue – from synchronously constructed query, through asynchronously generated answers and then synchronously delivered explanations.

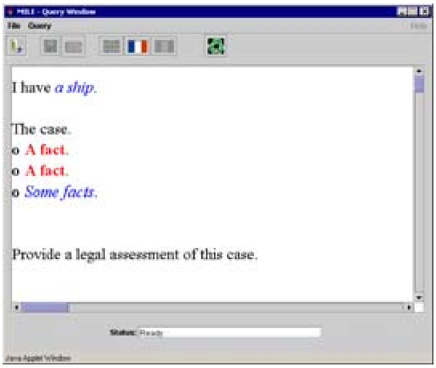
CLIME also demonstrates that it is possible to provide relatively 'deep' question-answering functionality in some specialised, but still practically useful domains. A key component of the approach is the ability to produce controlled yet complex queries using WYSIWYM technology, delivered over the web. An additional benefit of this architecture is that it supports language localisation of the entire application interface, including the user's queries.

The CLIME system was developed explicitly as a practical system. This meant that end users were involved from the stage of requirements formulation, through development and validation to evaluation. Although this was at times difficult, and led to a system that was functionally more diverse than originally anticipated, it ensured that CLIME remained practically relevant, as well as demonstrating the adaptability of the technology deployed. Overall, the system has demonstrated an effective approach to the organisation of knowledge-intensive advisory dialogues, and contributed an interesting and worthwhile demonstrator application in an important commercial domain.

Finally, Burger *et al.* (2002) outline a five year programme for TREC evaluation of QA systems. As we discussed in the introduction, CLIME does not share all the same prior assumptions as TREC QA, but it is interesting to compare the achievements of CLIME on the same scale as the TREC QA initiative, especially bearing in mind that the technical development reported here is contemporaneous with the roadmap document. Table 12 lists each of the new requirements proposed by Burger *et al.*, together with the extent to which CLIME addresses them. This comparison demonstrates the complementary nature of the work reported here compared with TREC QA – although the core inference engine has little NLP in common with TREC QA systems, its interface modules manifest many of the features projected in the TREC QA future.

Appendix 1: Using WYSIWYM to construct queries

In this appendix we briefly describe the initial steps in the construction of the query shown in Figure 5 using CLIME's WYSIWYM interface. For further information on WYSIWYM interfaces in general, see `http://www.itri.brighton.ac.uk/wysiwym` and Power *et al.* (1998).

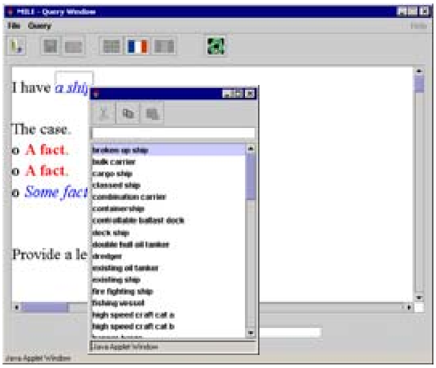


I have *a ship*.

The case.
o **A fact**.
o **A fact**.
o *Some facts*.

Provide a legal assessment of this case.

Creation of a query usually starts from a template such as the one shown here. The main window provides a folder of such templates for different query types. It is possible to create a query from scratch, but the templates have some of the common initial steps already instantiated. It is also possible to take an existing query and modify it to make a new query.

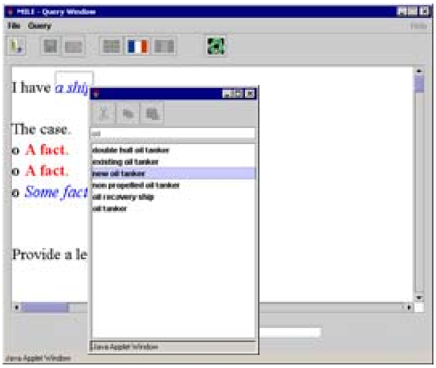


I have *a shi*

The case.
o **A fact**.
o **A fact**.
o *Some fact*

Provide a le

The query window displays the current partial query as text. Because the query is not yet complete, some of the phrases are generic and need to be further specified. These are indicated by red, bold text (for phrases which must be expanded) or blue, italic text (for phrases which can optionally be expanded). By selecting one of these spans with the mouse, the user can pop up a menu of possible expansions. In this picture, the user has selected the span **'a ship'**.

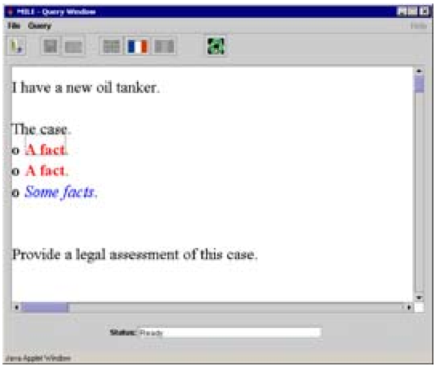


I have *a shi*

The case.
o **A fact**.
o **A fact**.
o *Some fact*

Provide a le

The menu displays all the ship types the system knows about. The user can either scroll down to find the right option, or type a substring of the desired option to restrict the search. Here the user has typed 'oil' to reduce the menu, and then chooses 'new oil tanker' from the list. Menus are dynamically generated according to the semantic type required for the linguistic context. The biggest menu in the demonstrator has about 3000 concepts in it.

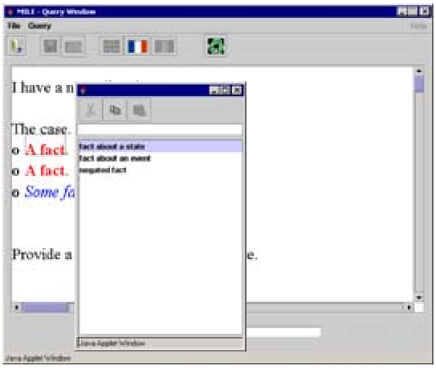


I have a new oil tanker.

The case.
o **A fact**.
o **A fact**.
o *Some facts*.

Provide a legal assessment of this case.

The text is redisplayed with 'a new oil tanker' instead of 'a ship'. The user can now start to provide information about the ship. The template includes two 'a fact' anchors which can each be expanded. Additional facts can be added later by expanding the (optional) 'Some facts' anchor.



I have a n

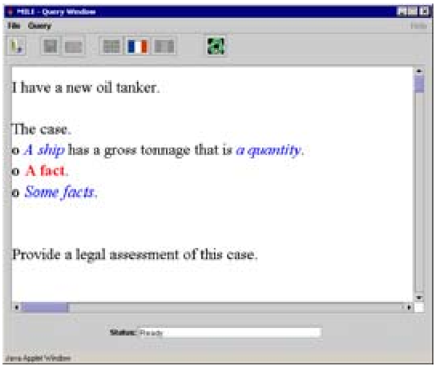
The case.
o A fact.
o A fact.
o Some fa

Provide a e.

fact about a state
fact about an event
negated fact

Selecting the first ‘a fact’ anchor pops up a menu of different fact types, from which the user selects ‘fact about a state’. A further expansion of this item (not shown here) allows the user to choose which type of state fact is required, and in this example ‘gross tonnage’ is chosen.

The next picture shows the resulting text, and alongside it the attribute-value matrix representation of the underlying structure the interface has built so far.



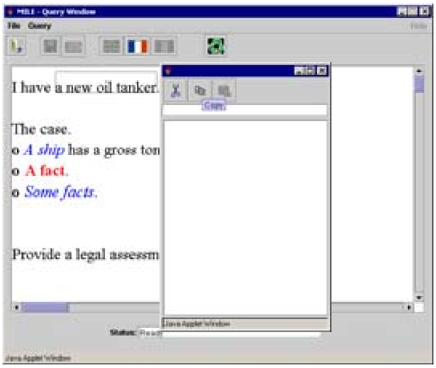
I have a new oil tanker.

The case.
o *A ship* has a gross tonnage that is *a quantity*.
o A fact.
o *Some facts*.

Provide a legal assessment of this case.

Status: Ready

$$\begin{bmatrix} \textit{assessment\_enquiry} & \\ \text{INTRO} & \begin{bmatrix}\textit{ship\_intro} & \\ \text{ARG1} & \begin{bmatrix}\textit{new\_oil\_tanker}\end{bmatrix}\end{bmatrix} \\ \text{PROP} & \begin{bmatrix}\textit{case\_proposal} & \\ \text{SIT} & \begin{bmatrix}\textit{set(state)} & \\ \text{FIRST} & \begin{bmatrix}\textit{state\_fact} & \\ \text{ATTR} & \begin{bmatrix}\textit{gross\_tonnage} & \\ \text{ARG1} & \\ \text{ARG2} & \end{bmatrix}\end{bmatrix} \\ \text{REST} & \begin{bmatrix}\textit{set(state)} & \\ \text{FIRST} & \\ \text{REST} & \end{bmatrix}\end{bmatrix}\end{bmatrix} \\ \text{QUEST} & \begin{bmatrix}\textit{acceptable}\end{bmatrix}\end{bmatrix}$$



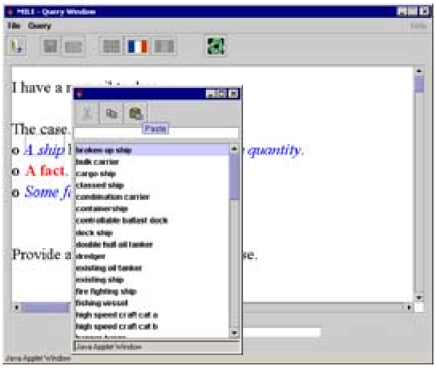
I have a new oil tanker.

The case.
o *A ship* has a gross ton
o A fact.
o *Some facts*.

Provide a legal assessm

Status: Ready

The next step is to establish the coreference between the new oil tanker and the subject of the gross tonnage fact (corresponding to a re-entrancy in the attribute-value matrix representation). This is achieved using familiar copy/paste operations on text spans. In this picture the user has selected the ‘new oil tanker’ span. The main body of the menu is empty (as this item cannot be further expanded), but two of the buttons – ‘Cut’ and ’Copy’ are active. Here, ‘Copy’ is chosen.



I have a r

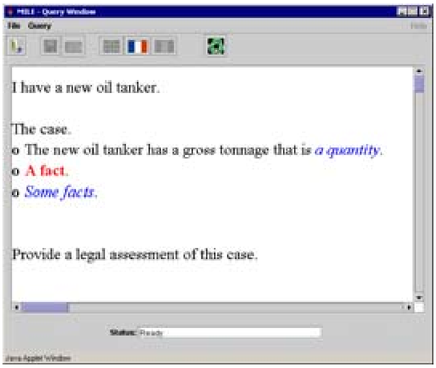
The case.
o *A ship* l *quantity*.
o A fact.
o *Some f*

Provide a e.

broken up ship
bulk carrier
cargo ship
classed ship
combination carrier
container ship
controllable ballast dock
deck ship
double hull oil tanker
dredger
existing oil tanker
existing ship
fire fighting ship
fishing vessel
high speed craft cat a
high speed craft cat b

Next, the ‘A ship’ anchor is selected and a menu of ship expansions pops up. However, unlike in the previous ship menu, the ‘Paste’ button is also active, because the underlying copy-buffer contains an object of the right type (‘ship’). By selecting this ‘Paste’ option, the user establishes the required coreference.

The next picture shows the resulting text and corresponding attribute-value matrix. The use of a definite determiner ensures the user correctly understands the coreference has been established.

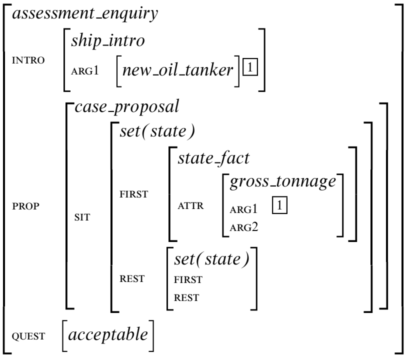


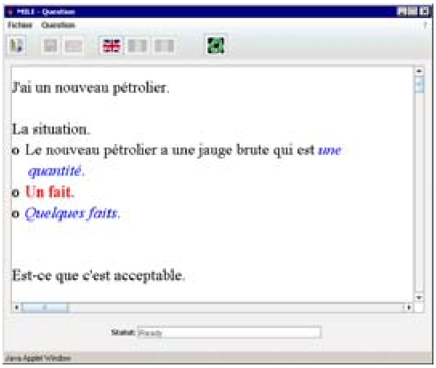
I have a new oil tanker.

The case.

- The new oil tanker has a gross tonnage that is *a quantity*.
- A fact.
- *Some facts*.

Provide a legal assessment of this case.





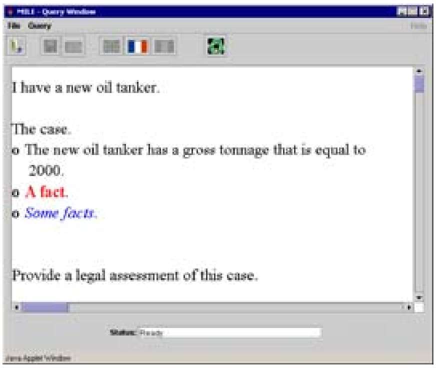
J'ai un nouveau pétrolier.

La situation.

- Le nouveau pétrolier a une jauge brute qui est *une quantité*.
- Un fait.
- *Quelques faits*.

Est-ce que c'est acceptable.

At any point in the editing process (or any other interaction with CLIME), the user can choose to change language. Here, the current state of the query is shown in French, and the interaction could be continued using French menus, (for example to develop ‘une quantité’). This is possible because the underlying representation is language-neutral – the query text and menus are regenerated dynamically in whichever language is currently selected every time the representation changes.



I have a new oil tanker.

The case.

- The new oil tanker has a gross tonnage that is equal to 2000.
- A fact.
- *Some facts*.

Provide a legal assessment of this case.

Reverting to English, the expansion of ‘a quantity’ to ‘equal to 2000’ follows exactly the same basic menu selection approach, resulting in the text shown here. Further development to the full query shown in figure 5 is achieved in exactly the same way.

References

Bertin, A., Bagnato, A. & Lorenzon, D. (2001) Final Evaluation. *CLIME (EP25414) Document*. *WP 5.6* (Circulation restricted to CLIME consortium).

Bertin, A. & Bagnato, A. (1999) Evaluation of the Natural Language Interface. *CLIME (EP25414) Document*. *Task 3.3*. (Circulation restricted to CLIME consortium).

Boer, A., Hoekstra, R. & Winkels, R. (2001) The CLIME Ontology. *Proceedings of the Second International Workshop on Legal Ontologies*, University of Amsterdam, pp. 37–47.

Bureau Veritas (1997) *Rules and Regulations for the Classification of Ships*, Bureau Veritas, Paris, 1997.

Burger, J., Cardie, C., Chaudri, V., Gaizauskas, R., Harabagiu, S., Israel, D., Jacquemin, C., Lin, C.-Y., Maiorano, S., Miller, G., Moldovan, D., Ogden, B., Prager, J., Riloff, E., Singhal, A., Shrihari, R., Strzalkowski, T., Voorhees, E. & Weishedel, R. (2002) Issues, tasks and program structures to roadmap research in question & answering (Q&A), `www-nlpir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc`.

Cahill, L. (2000) Semi-automatic construction of multilingual lexicons. *Machine Translation and Multilingual Applications in the New Millennium (MT2000)*, **15**: 1–15:10.

Chaudri, V. & Fikes, R. (eds.) (1999) Question Answering Systems. *Papers from the 1999 AAAI Fall Symposium*, Menlo Park, California, AAAI Press.

Evans, R., Piwek, P. & Cahill, L. (2002) What is NLG? *Proceedings of the Second International Conference on Natural Language Generation (INLG-02)*, New York, ACL, pp. 144–154.

Hirschman, L. & Gaizauskas, R. (2001) Natural language question answering: the view from here. *Natural Language Engineering* **7**(4): 275–300.

Krahmer, E. & Theune, M. (2001) Efficient Context sensitive generation of referring expressions. In: K. van Deemter and R. Kibble (eds.), *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, CSLI Publications, Stanford, pp. 223–264.

MARPOL (2002) *IMO: MARPOL 73/78*, IMO, London.

Piwek, P. (2000) A Formal Semantics for Generating and Editing Plurals. *Proceedings of COLING 2000*, Saarbruecken, Germany, pp. 607–613.

Piwek, P. (2002) Requirements Definition, Verification, Validation and Evaluation of the CLIME Interface and Natural Language Processing Technology. *ITRI Technical Report ITRI-02-03*, University of Brighton.

Piwek, P., Evans, R. & Power, R. (1999) Editing Speech Acts: A Practical Approach to Human-Machine Dialogue. *Proceedings of AMSTELOGUE '99: Workshop on the Semantics and Pragmatics of Dialogue*. University of Amsterdam.

Piwek, P., Evans, R., Cahill, L. & Tipper, N. (2000) Natural Language Generation in the MILE System. *Proceedings of the IMPACTS in* NLG *Workshop*, Schloss Dagstuhl, Germany, pp. 33–42.

Power, R., Scott, D. & Evans, R. (1998) What You See Is What You Meant: direct knowledge editing with natural language feedback. *ECAI-98. 13th European Conference on AI*, Wiley, Chichester, UK, pp. 677–681.

Reape, M. & Mellish, C. (1999) Just what is aggregation anyway? *Proceedings of the 7th European Workshop on Natural Language Generation*, Toulouse, France, ACL.

Reiter, E. & Dale, R. (2000) *Building Natural Language Generation Systems.* Cambridge University Press.

Valente, A. (1995) *Legal Knowledge Engineering: A modelling approach.* IOS Press.

Voorhees, E. (2001) The TREC question answering track. *Natural Language Engineering* **7**(4): 361–378.

Voorhees, E. (ed.) (2004) *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003).* U.S. National Institute of Standards and Technology (NIST), NIST Special Publication 500-255, Gaithersburg, MD.

Winkels, R. G. F., Breuker, J. A., Boer, A. & Bosscher, D. (1999) Intelligent Information Serving for the Legal Practitioner. *Law and Technology (LawTech-99).* IASTED, ACTA Press, Calgary (CA), pp. 64–70.

Winkels, R. G. F., Boer, A. & Hoekstra, R. (2002) CLIME: Lessons Learned in Legal Information Serving. In: F. van Harmelen (ed.), *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-2002).* IOS Press.