



# A generic information extraction architecture for financial applications

L.K.A. Wee<sup>a</sup>, L.C. Tong<sup>a</sup>, C.L. Tan<sup>b,\*</sup>

<sup>a</sup>*Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Singapore 119613 Singapore*

<sup>b</sup>*School of Computing, National University of Singapore, Lower Kent Ridge Road, Singapore 119260 Singapore*

## Abstract

The advent of computing has exacerbated the problem of overwhelming information. To manage the deluge of information, information extraction systems can be used to automatically extract relevant information from free-form text for update to databases or for report generation. One of the major challenges to the information extraction is the representation of domain knowledge in the task, that is how to represent the meaning of the input text, the knowledge of the field of application, and the knowledge about the target information to be extracted. We have chosen a directed graph structure, a domain ontology and a frame representation, respectively. We have further developed a generic information extraction (GIE) architecture that combines these knowledge structures for the task of processing. The GIE system is able to extract information from free-form text, further infer and derive new information. It analyzes the input text into a graph structure and subsequently unifies the graph and the ontology for extraction of relevant information, driven by the frame structure during a template filling process. The GIE system has been adopted for use in the message formatting expert system, a large-scale information extraction system for a specific financial application within a major bank in Singapore. © 1999 Elsevier Science Ltd. All rights reserved.

*Keywords:* Generic information extraction architecture; Message formatting expert; Message intermediate representation; Syntax tree structures

## 1. Introduction

The advent of computers and Internet has greatly accelerated the production of information, particularly textual information, beyond the ability of any human being to handle. Vast quantities of text are becoming available in electronic form, ranging from the published documents (e.g. electronic dictionaries, encyclopedias, libraries, CD ROMS, archives, on-line financial reports and newspapers), to the private databases (e.g. marketing information, medical histories and legal records), to memos, reports, emails, faxes, telexes, manuals, newsletters and to the latest Internet web pages. Having the right useful information (or knowledge) at the right time for decision-making is very often the most effective tool that enables the organizations to act fast and effectively. It is vital thus to leverage on advanced information management technology aids for a fast way of sieving through these texts for meaning information. The most common information management strategies are text categorization (Riloff and Lehnert, 1994; Schutze et al., 1995), text summarization (Kupiec et al., 1995), information filtering (Terry and Loeb, 1992; Wee et al., 1997), and

information extraction (Costantino et al., 1997; DARPA, 1991; 1992; 1993; 1995).

Information extraction is the extraction of relevant information from free-form text (or unrestricted text) based on a set of pre-defined template of what information is to be extracted (or what is relevant for extraction). The template specification constrains the relevancy of the information to be extracted, which could be about pre-specified types of events, entities or relationships. For example, in a financial transaction, an information extraction system could extract the transaction type, date, customer, principal, currency, interest rate, and so on. The extracted information is usually formatted as a database record suitable for subsequent processing, which may include database update for on-line access or data trend analysis, abstract, summary or report generation, text categorization, and message routing. The following is a financial message to a bank that requests a specific financial transaction.

```
22 APR 96
Ref FTX0012989

FM ANZ GRINDLAYS BANK SRI LANKA
TO DBS SINGAPORE

RE Your Telex Dated 19.4.96

Your Ref 0011LG012987, AND 0011LG013444
We have advised Arista Pte Ltd. of the earlier transactions
Of two bid bonds that were issued by you. Pls remit USD 10,500
Being our advising charges.
```

\* Corresponding author. Tel.: + 65-874-2900, fax: + 65-779-4580.

E-mail address: tancl@comp.nus.edu.sg (C.L. Tan)

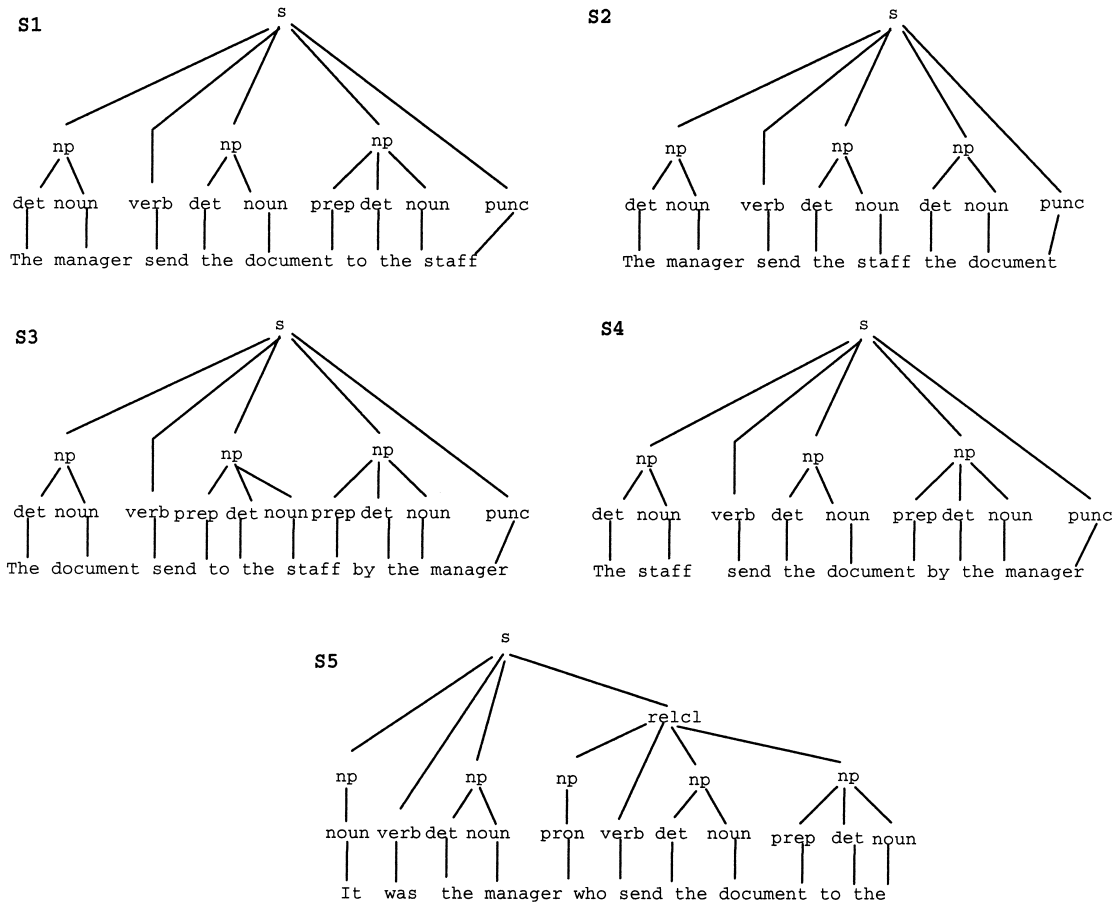


Fig. 1. Variant syntax trees that convey the same meaning.

The corresponding database record is as follows:

Customer:	ANZ GRINDLAYS BANK SRI LANKA
Request Date:	22 04 96
Request Ref:	FTX0012989, 0011LG012987, 0011LG013444
Transaction Type:	Remittance - advising charge
Transaction Date:	23 04 96 09:08
Beneficiary:	ARISTA PTE LTD
Principal Sum:	10,500
Currency:	USD
Interest Rate:	nil

One of the major challenges in information extraction is how to represent the following three different types of domain knowledge needed in the task, namely:

1. the meaning of the input text,
2. the domain knowledge, and
3. the target information that is to be extracted.

Three basic components are thus developed, namely, a message intermediate representation (MIR) scheme, a domain ontology and a frame structure for extracting information from messages (FEIM), to address the aforementioned three issues, respectively. With the three knowledge structures, a natural language processing system is incorporated to realize an integrated generic information

extraction (GIE) architecture. The architecture has been used in implementing several real information extraction systems, particularly in financial applications. In the sections that follow, MIR, the ontology, FEIM and GIE will be presented, respectively, followed by a section that describes the message formatting expert (MFE) system that is now operational in DBS bank in Singapore since 1997.

## 2. Message intermediate representation

Traditionally, input text is represented as a syntax tree where the branching of the tree is governed by the syntax rules of the language grammar. Such syntax representation has an obvious weakness in its syntax-dependency, making it difficult to handle flexibility in allowing different expressions for the same meaning. For instance, the following sentences, S1–S5, have different grammatical constructs and hence have different syntax tree structures as shown in Fig. 1.

- S1-‘The manager sent the document to the staff’.
- S2-‘The manager sent the staff the document’.
- S3-‘The document was sent to the staff by the manager’.
- S4-‘The staff was sent the document by the manager’.

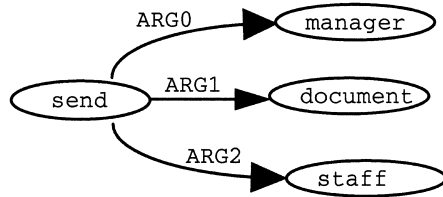


Fig. 2. MIR for variants of “The manager sends document to the staff”.

S5-“It was the manager who sent the document to the staff”.

These sentences are semantically equivalent and state the same facts. The variants of syntax structures made extraction of information difficult. In order to identify the person who sent the document, the variants of the grammatical constructs have to be considered. In some cases, the “manager” appears at the front and sometimes, it appears at the back of the tree structure. To simplify this process of identifying the required concept for information extraction, there needs to be a less syntax-dependent way of representing the input text that is also able to express the meaning of the text. Here, we have chosen semantic graph structures to represent the text meaning. We refer to this semantic graph structure as MIR.

The MIR is a directed acyclic labeled graph with nodes that represent the concepts that occur in the text. Concepts could range from simple keywords, terms, and phrases to more complex patterns, which represent real-world entities that have been discovered during the text analysis. These concepts could be events, activities, transactions, attributes, circumstances, facts, incidents, milestones, occasions, outcome, etc. The nodes of the graphs (referred to as MIR-nodes) are inter-connected through labeled directed links (referred to as MIR-links), that represent the relationships between these concepts. The MIR is a more neutral representation, independent of syntax or linear ordering, more meaningful semantic representation (sometimes referred to as logical form or deep structures) yet preserving syntactic information for reconstruction back to its syntactic tree structure.

The sentences S1–S5 have different syntax tree structures but will have the same MIR structure as shown in Fig. 2. The main concepts that appear in the sentences are “manager”, “send”, “document”, and “staff” and they are represented as nodes of the graph structure. The relationships between these concepts are “ARG0”, “ARG1”, and “ARG2”. With the MIR, identifying the person who sent the document becomes much easier with a single representation.

There are four main classes of MIR-nodes which are classified according to its part-of-speech category-noun, verb, adjective and adverb. For each class of MIR-node, there is set of predefined set of semantic-relationship MIR-links. The existence of an attribute may exclude the

existence of another (mutual exclusivity) or a mandatory existence of another. Table 1 provides details of the four different MIR-classes with their pre-defined sets of semantic-relationship MIR-links.

The designed and implemented MIR is very similar to the PEGASUS semantic representation (Jensen et al., 1993) which also provides a definitive move from syntax to semantics, based on the definition of case frames or thematic roles (or predicate-argument relations). The PEGASUS structure is also a labeled, directed graph. The MIR and PEGASUS differ in argument assignments from the identification of meaningful relationships between head words of the phrases and their modifiers or adjuncts. Hence, both representations also differ in the concepts and the set of relations. PEGASUS is similar to the MIR in adopting the notion of “deep” cases or functional roles. For example, deep subject of ARG0 for MIR and DSUB for PEGASUS; ARG1 for MIR and DOBJ for PEGASUS.

The MIR is also similar to Schank’s conceptual dependency (CD) notion (Allen, 1987). However, there are subtle differences. The basic entity in a CD is the event, or conceptualization whereas a basic entity in a MIR is the node, which could be an event or an entity. The CD has only a limited number of cases such as ACTOR which is similar to ARG0 of the MIR. It also has a limited number of action types or primitives based on the abstract notion of transfer: ATRANS, abstract transfer (such as of possession of an object); PTRANS, physical transfer; and MTRANS, mental transfer (such as talking). It also includes primitive actions of bodily activity such as PROPEL (apply force), MOVE (move a body part), GRASP, INGEST, and EXPEL, as well as a few mental actions such as CONC (conceptualize or think) and MBUILD (perform inference).

### 3. Domain ontology

The second important piece of knowledge required in an information extraction system is the knowledge of the domain application, i.e. the domain ontology. A *domain ontology* defines the set of basic terms comprising the vocabulary of the domain area and the relationships that bind these terms. For example, in an automobile domain, the set of vocabulary includes motorcar, motorist, engine, chassis, body, wheel, steering, light, brake, fuel system, gearbox, wiper, etc. These terms have some relationships to one another. For example, a motorist is a person operating a motorcar; the glove compartment and the boot are reserved spaces in an automobile for storage of things.

In this research, for pragmatic reason, we have selected to model a single restrictive domain, using a simple yet processable ontology. The domain knowledge is organized in object-centered hierarchies with inheritance, with the objects representing the terms and concepts in the domain, and the relationships of these concepts through the hierarchy. In this hierarchy, the subordinate concepts, in addition

Table 1  
Details of semantic relationships of MIR links

[1] MIR-node with verb category	
ARG0->	Subject or agent of the action (e.g. “The manager sent the document”.)
ARG1->	The direct effect of the action (e.g. “The manager sent the document”)
ARG2->	Beneficiary of the action (e.g. “The clerk gave the form to John”.)
CIRC->	Loosely-coupled relationship to the MIR-node (e.g. “John sent the form through the fax”.)
ATG->	Descriptive modifier to the MIR-node (e.g. “The clerk sent the paper urgently”.)
COORD->	Joining relationship to the MIR-node (e.g. “John faxed and copied the paper”.)
[2] MIR-node with noun category	
ARG0->	Subject that resulted in a state (e.g. “The approval of the loan by the manager”.)
ARG1->	The direct effect of the state (e.g. “The approval of the loan by the manager”.)
CIRC->	Loosely-coupled relationship to the MIR-node (e.g. “The manager who approved the loan”.) (e.g. “The manager, attending a meeting, will be late”)
ATG->	Descriptive modifier to the MIR-node (e.g. “An urgent document”)
POS->	Possession of the MIR-node (e.g. “My application is being processed”.) (e.g. “The clerk is working on John’s case”.)
COORD->	Joining relationship to the MIR-node (e.g. “I have the form and the envelope”.)
REF->	Reference to the MIR-node (e.g. “The client inquired about his application”.)
ISA->	Is-A-Kind of relationship to the MIR-node (e.g. “Mary is a clerk”.)
[3] MIR-node with adjective category	
ARG1->	The direct effect of the modifier MIR-node (e.g. “He is capable of this action”.)
ATG->	Descriptive modifier to the MIR-node (e.g. “The application is very complicated”.)
COORD->	Joining relationship to the MIR-node (e.g. “The clerk is efficient and friendly”.)
[4] MIR-node with adverb category	
ATG->	Descriptive modifier to the MIR-node (e.g. “The client talked very loudly”.)
COORD->	Joining relationship to the MIR—node (e.g. “The clerk processed the documents efficiently and carefully”.)

to having their own attributes, will inherit characteristics and features from the superordinate concepts. However, if the subordinate concept has an attribute that has a different value from that inherited from its superordinate parent, its attribute value takes precedence.

Fig. 3 shows an ontology on the concept of delivery node of documents. The term “delivery-mode” comprises of all the various concepts which are lower level in the hierarchy, such as, mail, airmail, surface mail, registered mail, express mail, courier, federal express, DHL, local urgent mail, and LUM. All these subordinate concepts will inherit all the characteristics of the term “delivery-mode”.

Such an ontology facilitates reasoning. For example, the system through its inference reasoning “understands” that

the concept “DHL” is a kind of “courier”, which is in turn a kind of “express mail” of “mail”, and a type of “delivery-mode”. From the larger ontology that encompasses this ontology, when “DHL” is being used in a different sense, such as a verb in “Pls DHL the documents to the issuing bank”, the concept of “send” is implied.

This ontology representation also facilitates the knowledge specification of the target information to be extracted. As superordinate concepts of a concept could be automatically inferred from the ontology, there is no need to specify all the possible concepts in the knowledge specification of the target information to be extracted.

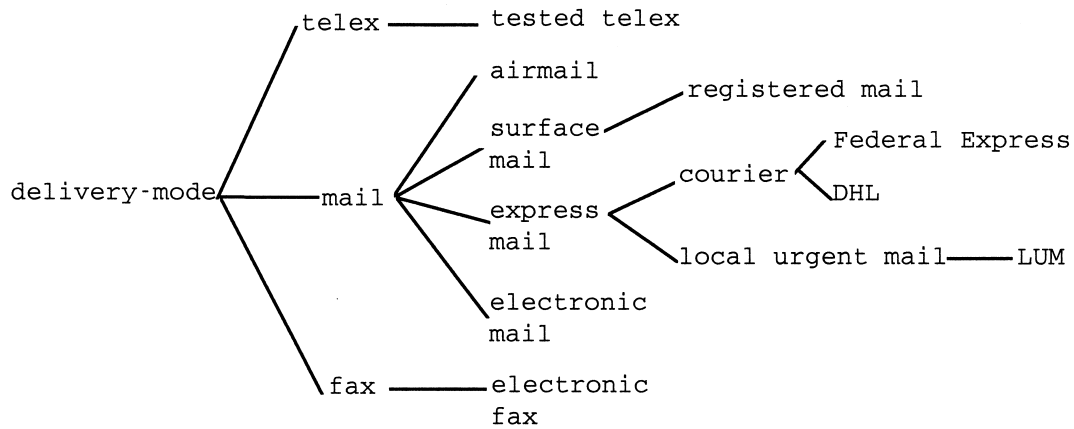
#### 4. Frame for extracting information from messages

After having represented the meaning of the input text and the application domain field, the third piece of information that needs to be represented in an information extraction application is the target information to be extracted (refer to interchangeably as *target-information*). In this research, a frame approach to represent the target information to be extracted has been chosen. The frame system is called FEIM.

Frames or templates are common knowledge representation tools in AI applications (Bobrow and Winograd, 1977; Brachman and Schmolze, 1985; Charniak, 1978; Nyberg, 1988). However, the FEIM system employs templates for the specific purpose of extracting information from input messages. The design of FEIM is very much influenced by the Framekit (Nyberg, 1988) GUS (Bobrow et al., 1977) and KRL (Bobrow and Winograd, 1977) frame representations. It has similar basic features such as slot, demon and inference as well as similar operators such as creation, access and update of slot values. In particular, the FEIM structures have been designed to handle practical issues in an information extraction application. FEIM provides convenient means for the knowledge coder (or the user) to specify the common phenomena in information extraction applications like relationships between target-information.

Knowledge on what constitutes relevant information that is to be extracted from the input text is stored as a set of FEIM specifications. A FEIM specification can be viewed as a template of slots with each slot corresponding to a piece of relevant information that is to be extracted. Associated with each slot is a set of attributes with values of one of the following types: *descriptive* that describes the slot; *status* that contains characteristics of the target-information; *constraint* that spells the type specifications of the slot value; and *demon* that contains the function and macro calls that manipulate the slot value. Table 2 summarizes the attributes of the FEIM system.

Fig. 4 shows a sample of a FEIM slot specification. The FEIM slot named *dispatch-of-document-slot* specifies that the target-information to be extracted could correspond to a MIR subgraph of the input text, if this MIR subgraph is



**A sample ontology**

```

(telex delivery-mode)      ((registered mail) (surface mail))
(mail delivery-mode)      (courier (express mail))
(fax delivery-mode)      ((local urgent mail) (express mail))
((tested telex) telex)   ((Federal Express) courier)
(airmail mail)           (DHL courier)
((surface mail) mail)    (LUM (local urgent mail))
((express mail) mail)    ((electronic fax) fax)
((electronic mail) mail)
    
```

**Implementation of Ontology**

Fig. 3. Domain ontology.

identified and matched by a GML-rule *document-to-send-rule*. GML is a graph manipulation language cum graph pattern matcher developed for matching the specified condi-

tions against the graph structures such as the MIR structure. The FEIM slot also specifies that the information may be absent in the input text, but a warning is to be reported if the GML rule identifies more than one such information in the input text. Also specified is the relationship between the dispatch-of-document-slot and the other slots, *available-document-slot* and *delivery-mode-slot* in the when-filled demon. The information for these two slots is to be inferred from the extracted information of dispatch-of-document-slot.

The design of FEIM has incorporated practical operational requirements of *inferencing*, *cross-checking*, *discrepancy highlighting*, and *change propagation*. cross-checking is a special type of inferencing but with a different purpose. The aim of cross-checking is for verification of consistency across slots. A special slot, known as the *cross-checking-slot* (similar to inferree slot) is created to verify consistency of one or more of the *To-Be-cross-checked-slots* (similar to inferer slots). If the consistency check fails, the discrepancy attribute value will be filled. For example, in a loan application domain, a cross-checking-slot “check-credit-limit” is a inferree slot derives its value of whether credit limit extended to customer is exceeded by cross-checking that the inferer slot “credit-limit-extended” is less than or equal to two times the inferer slot “monthly-salary”. If the cross-checking finds that the

Table 2  
Attributes of the FEIM system

Attributes	Explanations
Slot-name	Name of the slot
Slot-description	Description of the slot
Slot-content	Content value of the slot
Default	Default value to slot-content
Invalid-content	The value of slot-content will be transferred here and the slot-content will be reset to nil if its value fails the validation test.
Discrepancy	Records the failed validation test
Single	Determines if the slot-content is single-valued or multi-valued
Compulsory	Determines if the slot-content value is optional or compulsory
Validate	Tests to validate the slot-content value
TO-FILL	Demon on how to fill the slot-content value
Normalize	Demon on how to manipulate the slot-content value
When-filled	Demon on what instructions to perform after the Slot-Content value is determined
Modifiable	Determines if the slot-content value can be manually modified

```
(FEIM-SLOT Despatch-of-Document-Slot
  (content nil)
  (to-fill (gml-rule document-to-send-rule
    (a r1 b r2 c r3 d r4 e)
    ((a despatch)
     (r1 object->)
     (b document)
     (r2 beneficiary->)
     (c (or person organization))
     (r3 means->)
     (d delivery-mode)
     (r4 agent->)
     (e (or person organization)))
    (extract)))
  (when-filled (infer FEIM-SLOT
    (Available-Document-Slot
     Delivery-Mode-Slot)))
  optional
  single
  modifiable)
```

Fig. 4. A sample FEIM slot.

check fails, the “check-credit-limit” will have its discrepancy attribute value filled.

This will serve to highlight to the user to do a check on the credit limit that is extended to the customer.

Similar to the inferee slots, the cross-checking-slot does not have a TO-FILL attribute. Instead, it has a special attribute of *cross-check* and the TO-FILL attribute will be automatically generated during compilation of the FEIM specification. The following shows a cross-checking-slot “type-of-tenor” which cross-checks the to-be-cross-checked-slots “type-of-tenor” and “tenor” through a predicate function “type-tenor-agree-p”.

```
(FEIM slot type-of-tenor
  (cross-check type-tenor-agree-p type-of-tenor tenor))
```

FEIM highlights any discovered discrepancies in the slot-content of slots through its discrepancy attribute of the slots. Three types of discrepancies can be discovered through the FEIM system. They are *missing information* (through the compulsory and slot-content attributes), *excess information* (through the single and slot-content attributes) and *wrong information type* (through the invalid-content attribute). This discrepancy feature of FEIM conveniently highlights to the user of the potential incorrectness in the financial transaction or inaccuracy in the information that has been automatically extracted by the system.

The discrepancy feature enables the user to more quickly detect incorrectness and allow him to make the necessary modifications. Only the slots with the modifiable attribute can be modified by the user. For ensuring the integrity of some of the slot-content, the slots are marked intentionally with unmodifiable attribute and hence the user will not be able to modify these slots. Most of the inferee slots are intentionally marked unmodifiable as their values are

inferred. As FEIM caters for the flexibility of inferencing, once the slot-content value of a inferer slot is modified, FEIM has a change-propagation mechanism to propagate changes to other inferee slots that are affected by this change. The change-propagation mechanism will re-infer and re-validate the values of all the inferee slots. Without this mechanism, it will be difficult to ensure the consistency and integrity of the slot-content values of all the inferer and inferee slots through manual changes.

## 5. The generic information extraction architecture

The GIE architecture system was developed using the English parser of the Tapestry NLP toolkit (Tong, 1995). Fig. 5 shows the GIE architecture.

The GIE architecture has four main processes:

1. Text pre-processing;
2. Surface text analysis;
3. Deep text analysis;
4. Template filling.

The text pre-processing module will decompose the input text into useful text segments for subsequent text analysis. The module consists of six steps depicted in Fig. 6. The format handling tool is a collection of shareware as well as word editor routines that convert the text documents with different formats (e.g. propriety word editor formats or mark-up languages, etc) to plain ASCII text. The user is required to indicate to the system the format of the input text document in order to convert it correctly.

The text layout analysis I module segments a text into the following categories:

1. Basic text segments, such as address, are extracted immediately without further analysis.

## GIE Information Extraction Architecture

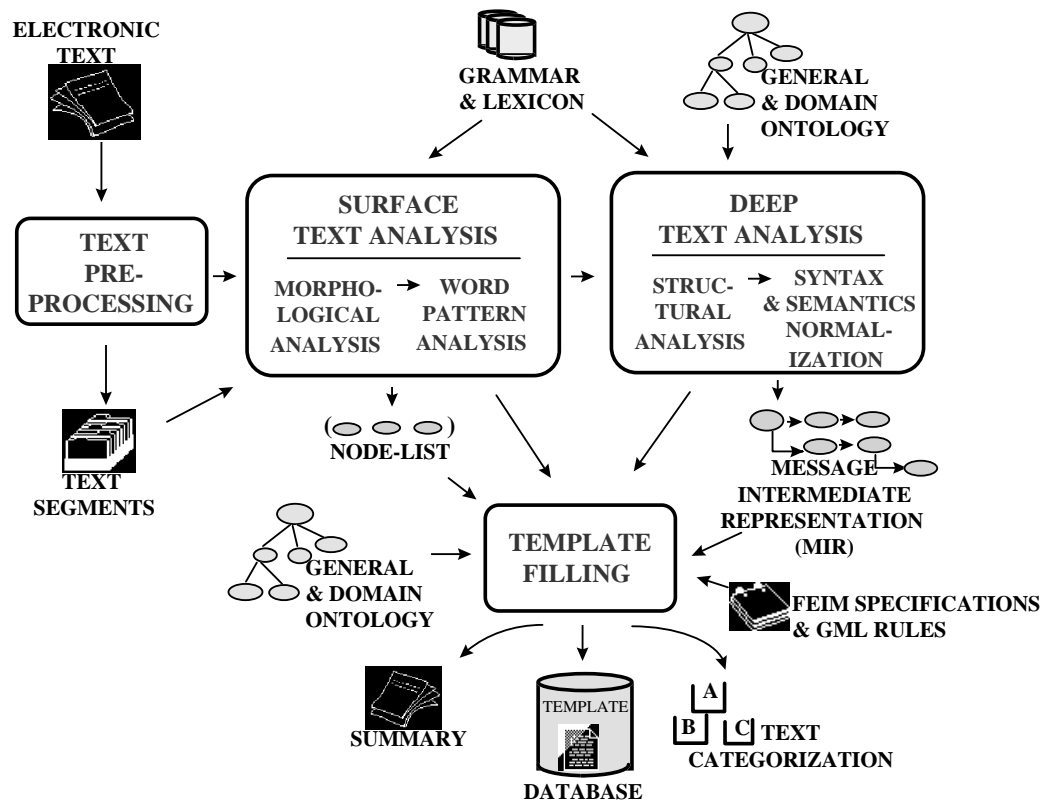


Fig. 5. The GIE architecture.

2. Simple text segments, such as organization names, will be subjected to simple text analysis.
3. Complex text segments, such as instructions and descriptions, will be subjected to text layout analysis II module.
4. Redundant or irrelevant text segments that will not contribute to subsequent processing are identified and filtered out of the system. E.g. tabular data in appendices. This is handled by the text segment filtering module.

The text layout analysis II works on each complex text segment by segmenting it into a section–paragraph–sentence hierarchical structure depending on the input text document layout heuristics. This hierarchical structure is useful to the GIE information extraction system to enable it to more intelligently determine the relevant segment of the entire text as the desired extracted information in the template filling module, i.e. whether the entire section, or specific paragraphs or sentences are to be extracted.

After the input text is segmented into logical text segments by the text pre-processing module, relevant text segments are sent to the surface text analysis module for text analysis. There are two phases to surface text analysis—morphological analysis and word pattern analysis.

Morphological analysis decomposes the input text segment into its basic word constituents and normalize these constituents to their baseforms. In addition, it assigns potential part of speech as well as other linguistic information to the normalized words. For example, the string “The system functionalities are documented” is analyzed as shown in Fig. 7.

Word pattern analysis will attempt to form lexical patterns from the morphological analyzed node-list by looking into its left and right context. The word pattern analysis uses a set of pattern lexicons, each entry in its lexicon

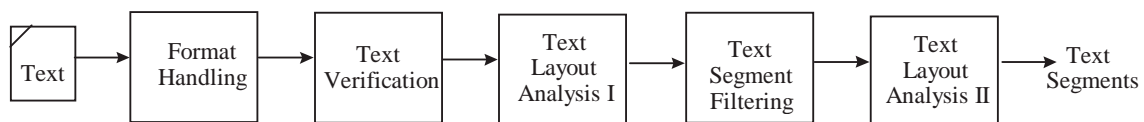


Fig. 6. Text pre-processing.

```

**Morphological Output** of
"The system functionalities are documented." :
#<Node: The> #<Node: system>
#<Node: functionalities> #<Node: are> #<Node: documented>
#<Node: .>)

#<Node: The>                #<Node: functionalities>          #<Node: documented>
(BASEFORM "The"            (BASEFORM "function"          (BASEFORM "be"
  POS D                    POS N                          POS A
  D DDEF                  SEM ABST                          AFFIX-FOUND _ED
  NUM (SIN PLU)          NUM PLU                          DRV NA
  S-BEGIN T              AFFIX-FOUND (IES _ITY _AL)  A ADJ
  TEXT-END-POSN 3        DRV AN                          TEXT-END-POSN 40
  ALPHACASE INITCAP      ALPHACASE LOWERCASE          ALPHACASE LOWERCASE
  T-TYPE ALPHABETIC      T-TYPE ALPHABETIC           T-TYPE ALPHABETIC
  TEXT-START-POSN 0)     TEXT-END-POSN 26             TEXT-START-POSN 31)

#<Node: system>           #<Node: are>                        #<Node: .>
(BASEFORM "system"        (BASEFORM "be"                (BASEFORM "."
  POS N                   POS V                          POS P
  NUM SIN                 INVARIANT T                      CL-END T
  NEND N1                 V VB                              S-END T
  TEXT-END-POSN 10       NUM PLU                          JOINB4 T
  ALPHACASE LOWERCASE    TENSE PRES                       TEXT-END-POSN 41
  T-TYPE ALPHABETIC      VL1 (NX WH)                      ALPHACASE LOWERCASE
  TEXT-START-POSN 4)     COP T                             T-TYPE PUNC
                        TEXT-END-POSN 30                          TEXT-START-POSN 41)
                        ALPHACASE LOWERCASE
                        T-TYPE ALPHABETIC
                        TEXT-START-POSN 27)

```

Fig. 7. Morphological analysis output.

describing a particular word sequence pattern that is to be matched with the node-list output from the morphological analyzer. It does not make use of any grammar rules. The output of the word pattern analysis is a new node-list of entities formed from the combined word sequences.

The following are some examples of more complex word pattern entries developed in GIE. The word sequences need not be adjacent to each other.

```
((DBS (opt *Bank) (opt (opt of) *Singapore))) nil (:-
company_name (:= node *bank)))
```

will capture all the following sequences of words and add a 'company\_name' feature to the captured concept:

```

"DBS",
"DBS BANK",
"DBS BANK OF SINGAPORE",
"DBS BANK SINGAPORE",
"DBS OF SINGAPORE" and
"DBS SINGAPORE".

```

```
((or @credit @pay @remit) (opt "to") (or ?dollar
?customer (opt "to") (or ?dollar ?customer) (opt
!account))) nil (:- transaction))
```

will capture all the following sequences of words and add a 'transaction' feature to the captured concept:

```

"pay Mr Tan $10 000",
"pay to Mr Tan $10 000",
"remit $10 000 to Mr Tan", and
"credit $10 000 to Mr Tan's account",

```

assuming that "\$10 000" has DOLLAR attribute and "Mr Tan" has CUSTOMER attribute.

In some cases, the target-information is not as straight forward as identifying well-defined word sequences. Hence, the analyzed node-list of words and entities from the surface text analysis has to be subjected to further deep-level of analysis in the deep text analysis module to discover more complex concepts and the relationships between the concepts.

There are two phases to deep text analysis—structural analysis followed by syntax and semantic normalization. The structural analysis matches the word and pattern node-list from the surface text analysis with appropriate grammar rules, assigns meaning to them and verifies sentence processing to produce the 'text meaning' in the

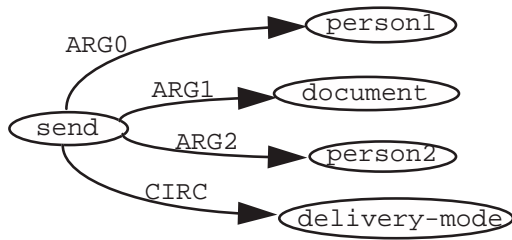


Fig. 8. MIR for “Someone sends a document to another through some delivery mode”.

form MIR discussed in Section 2. The syntax and semantic normalization normalizes variant forms of the MIR with the same text meaning for ease of extraction during the Template Filling process.

For example, the following sentences:

- “All documents are to be DHLed to the personnel department”.
- “Pls send us the original transcripts through surface mail”.
- “Dispatch to us by express mail the certificates to the aforementioned effect”.
- “Fax us the letters as soon as possible”.
- “All disclaimers are to be signed and forwarded to us by courier”.

state the same request of wanting someone to send some kind of documents to a person or an entity through some

means. The purpose of syntax and semantic normalization is to transform all the variant but semantically equivalent surface structures into the same MIR structure. The aforementioned sentences will have very similar MIR structures as in Fig. 8.

After the text has been pre-processed and analyzed, it is now ready for extraction by the template filling module. This module uses the FEIM discussed in Section 4. Knowledge on what constitutes relevant target-information from the input text is stored as a set of FEIM specifications, which can be viewed as a template of slots; each slot corresponds to a piece of relevant information and contains the descriptions of the information to be captured. It also contains information on how to search for the required information.

There can be more than one template but at any one time, there is only one master template which drives the extraction process. The master template will essentially contain information on the sequence of filling up the subordinate templates. The subordinate templates will in turn contain the target-information knowledge. If there is only one template, then the template itself will contain the target-information knowledge. For multiple templates, the template filling process will iterate through each template, in the order accorded by the master template.

The template filling process will go through each FEIM slot within a template and attempt to fill up the value for the CONTENT attribute of the slot. In the process, it will trigger the filling mechanism to execute the functions or macros embedded in the TO-FILL demon attribute. The functions

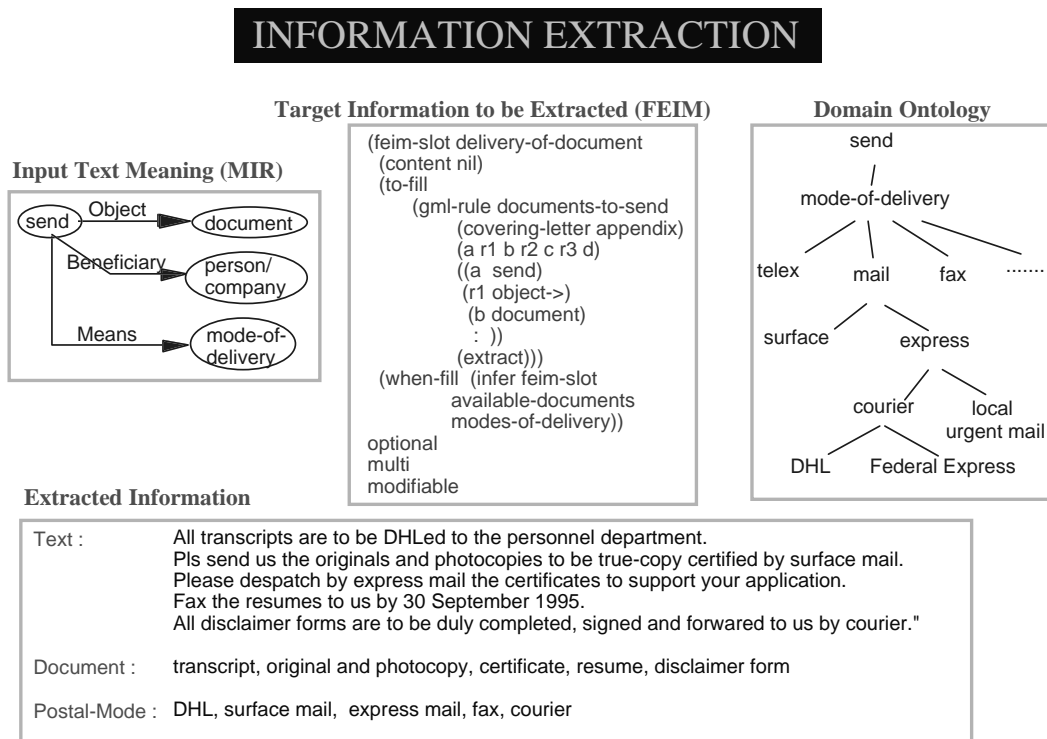


Fig. 9. The interplay of the knowledge structures in information extraction.

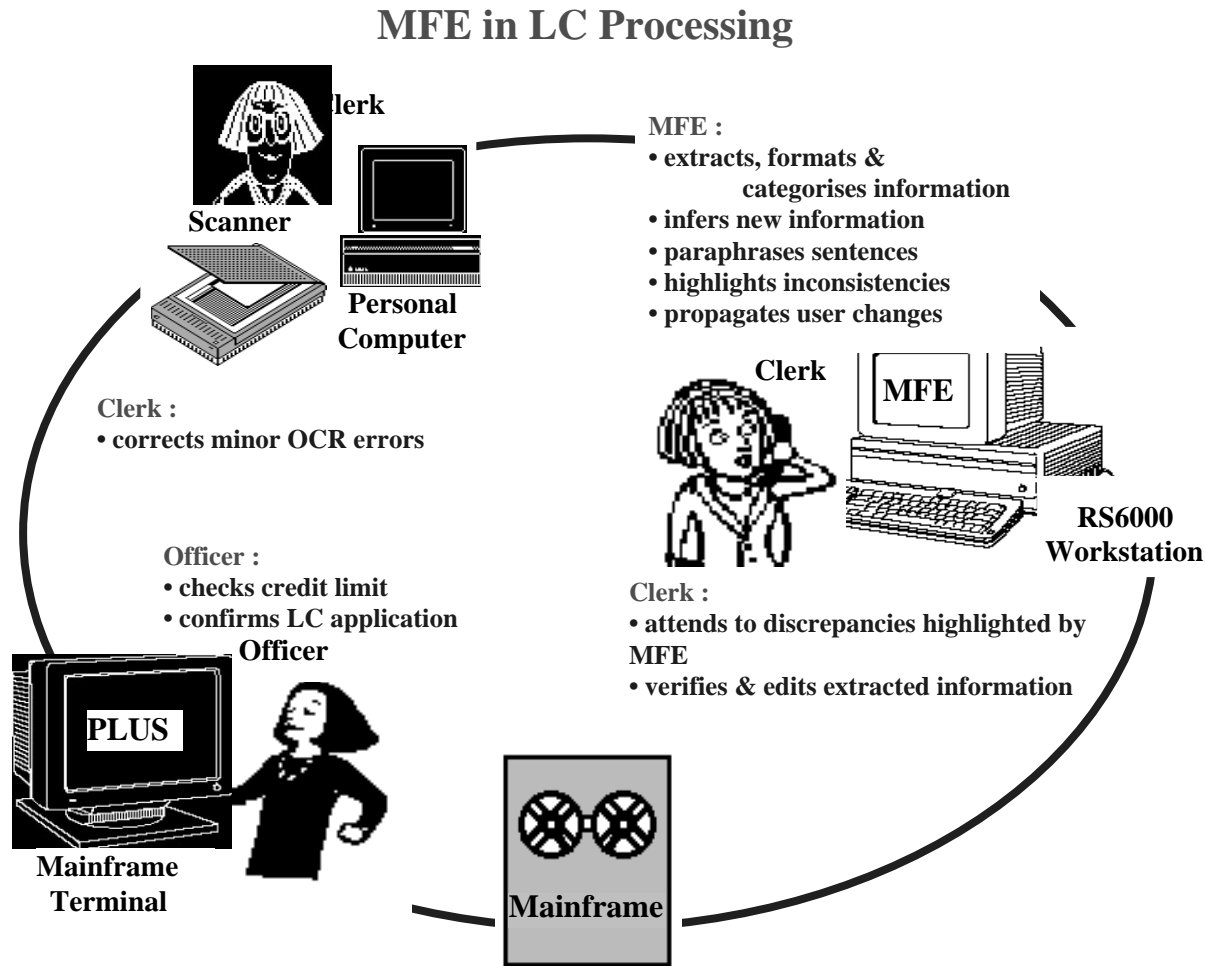


Fig. 10. Message formatting expert in action.

and macros can access the following data structures—node-list from surface text analysis, MIR structure from deep text analysis and database fields of specified databases.

Fig. 9 shows how the various knowledge structures inter-play within the GIE architecture in information extraction. For example, the text “All transcripts are to be DHLed to the personnel department”, it would be pre-processed and analyzed to a MIR with entities of ‘transcript’, ‘DHL’ and ‘personnel department’. Through the general and domain ontology, it is derived that ‘transcript’ is a type of ‘document’, ‘DHL’ is a mode of ‘send’, and ‘personnel department’ is part of a ‘company’. When the system attempts to fill up the FEIM slot ‘delivery-of-document’, it searches the MIR through the GML-rule and matches successfully the MIR represented by the text. Hence, the relevant portion of the MIR is extracted.

The relevant portion of the MIR to be extracted is determined by the text document layout hierarchical structure which is discussed earlier in this section. However, the user can overwrite the default by specifying in the ⟨return-function⟩ of the GML-rule to return the entity discovered instead of the determined section, paragraph, or sentence of the text document layout hierarchical structure. For example, if the user is only interested to know the documents that were discussed in the text, he can specify in the GML-rule to return the particular node corresponding to the matched document. In the example shown in Fig. 9, the ‘document’ target-information is ‘transcript’ for the text “All transcripts are to be DHLed to the personnel department”.

The extracted information can either be updated into databases, or could be used to generate a summary or even for subsequent text processing such as text categorization.

## 6. Message formatting expert

The GIE architecture has been successfully deployed in a practical information extraction system—the MFE, a large-scale information extraction system in the finance domain. The MFE system has been operational in one of the largest local banks in Singapore, DBS Bank since early 1997.

The MFE system processes electronic applications of letter of credit (LC) and its accompanying annexes and extracts relevant information for update into the database as well as generate the letter of offer. The MFE LC application is a useful application to reduce the data entry work as well as to automate the generation of the first draft of letter of offer to the customers, thus reducing turnaround time for LC application.

LC is a facility that guarantees payment for shipment of goods. The exporter of goods require the importer to apply for a LC from an issuing bank before they effect the shipment of the goods. Once the goods are received by the importers in good order, the issuing bank will pay the exporter. As the importers and exporters not using the same banks and also different banks have different banking facilities, a few intermediary banks could be involved in this LC process. Hence, the instructions can get very long and complicated.

Before the MFE–LC automation, the bank processes LC manually. The trained clerks would first go through the application forms, analyze the forms and map and key in the LC data into the relevant fields of the database and then format the data accordingly. After which, an officer would check through the database fields and verify the credit limit of the customer. After approving the LC application, the officer would then create a letter of offer from the database fields.

The manual LC data capture is not a simple mapping process from the document to the database. The mapping requires good knowledge of the LC domain. The challenges are:

1. The LC domain is complex. There are many ways to formulate the LC clauses depending on the many conditions. For example, there are a few reimbursement procedures and hence different ways to formulate the reimbursement clauses.
2. Information stated in the LC may be conflicting or invalid as the customers do not usually ensure correctness and consistency. There are a few fields within the LC application that are required to be counter-checked. For example, the expiry date of the LC application as well as the latest shipment date of the goods.
3. Mapping of data from the LC form to the database fields is not a simple one-to-one function. There are several one-to-many (e.g. beneficiary country) and many-to-one mappings (e.g. reimbursement clause).
4. There is also inter-dependent fields whose information may affect one another (e.g. type of tenor).

The objective of automating the manual LC process is first to encapsulate and internalize the LC domain knowledge of the clerks and the officers as well as to automate the data capture and mapping process. The scope of MFE LC are:

1. Automate the data entry process by scanning the hard-copy LC application form and subjecting it through

Table 3  
Customization needed for different modules

Text pre-Processing <sup>a</sup>	Word pattern entries <sup>a</sup>	Grammar rules <sup>b</sup>	Ontology lexicon <sup>b</sup>	FEIM specifications <sup>b</sup>
----------------------------------	-----------------------------------	----------------------------	-------------------------------	----------------------------------

<sup>a</sup> Slight customization.

<sup>b</sup> Much customization.

OCR, if it is not already applied through electronic means.

2. Analyze, extract and format the information through:
  - Classify the fields into the right categories;
  - Extract information from the text fields;
  - Formulate clauses from the extracted information by paraphrasing;
  - Infer further information from the extracted information.
3. Check and highlight invalid or conflicting information or discrepancy detected in the LC.
4. Assist updating of information by change propagation mechanism, highlighting any discrepancy detected.

Fig. 10 shows the automated MFE LC processes. An LC application comes in a standard application form with 80% of structured fields and 20% of free-form fields. There are 20 structured fields and four free-form fields which could contain any volume of free-form text. If the form runs out of space, an annex is attached to the application form. Each annex can range from one to three pages of free-form text.

The 20 structured fields are simple fields pertaining to the information about the applicant (e.g. applicant's name, address, account number, contact person), advising bank information (i.e. bank name, address, account number), beneficiary (exporter) information (i.e. beneficiary's name, address, contact person), transaction information (i.e. currency, amount, price basis such as free on board) and shipment information (i.e. shipment source, destination, latest shipment date, partial shipment or transshipment).

The four free-form fields are free-form description of documents required, merchandise, special instructions and additional instructions on the annex. Though only 20% of the LC form are free-form fields, these fields contain the important bulk of the LC letter of offer. The extraction of these free-form fields required 61 FEIM slot specifications and 143 GML rules for comprehensive extraction.

A total of 75 representative LC samples was selected for use in fine-tuning the GIE architecture for the MFE LC system. There were much customizations due to the complexity of the LC domain as shown in Table 3. There is only slight customization to the text pre-processing module as the application forms are relatively structured and it is easy to segment the various fields. There is some customization done to the word pattern entries and much customizations to the grammar rules for the text analysis modules as the text is more instructional than the formal english style. As the domain knowledge of LC is complex, there is much effort required to craft both the ontology domain knowledge and the FEIM specifications of extraction.

The delivered MFE LC system underwent three sets of due diligence user acceptance testing. A total of 710 unseen actual documents, with an average of one A4 page of 200 words per page were used for the testing (200, 226 and 284

documents, respectively, for each user acceptance testing). The accuracy of MFE was measured to have a high accuracy of 90–95% (an average of recall and precision) for extraction unit of words and word patterns and 80–92% for extraction unit of sentences and concepts.

The following shows the sequence of extraction. The free-form text is first analyzed to its MIR. The relevant FEIM specifications, GML rules and ontology are triggered during the template filling process. The GML rules matched the MIR of the free-form text and extract those that fit the specifications. After which, further processing such as paraphrasing or inferencing is done.

Free-form text:

1. The details of container number and the engine and chassis number of vehicles stuffed into each and every container must be shown on the packing list. For example, the prefix and the identifier must accompany the container number.
2. Bills of lading must not contain documentary credit.
3. Bills of lading must indicate Shipper as SING INTERNATIONAL PTE LTD, SINGAPORE.
4. All documents to be sent to opening bank in one lot.
5. 10% more or less on credit amount and merchandise quantity is allowed.

Triggered extraction rules:

(FEIM–GML–RULE items-must-show

```

nil
(a r1 b)
((a (& (mem ul '(show indicate state mark accompany
include)) (n == svl 'ing)))
(r1 (or arg0 arg1 circ))
(b (or (mem ul '(document (credit number) (P.O.
number) (contract number) letter-of-credit note)) pack-
ing-document shipping-certificate)))
(SENTENCE a)

```

(FEIM–GML–RULE documents-from-nego-bank

```

nil
(a r1 b r2 c)
((a (== ul 'send))
(r1 arg1)
(b (== ul 'document))
(r2 circ)
(c (& (mem ul '(lot cover mail set)) (== svl 'in))))
(PARAPHRASE "All documents must be dispatched
direct to DBS Bank, 6 Shenton Way, DBS Building,
Singapore 068809, in
" (c card-ul) (if (== (c card-ul) 1) "lot" "lots"))
(cond ((true (atg-> delivery-mode))
(funcall #'format-delivery-mode (atg-> token)))
(true ((c atg->) delivery-mode))
(funcall #'concat-string "by" ((c atg->) token)))
(t "by courier service"))'.')

```

(FEIM-GML-RULE tolerance-clause-2

```
merchandise-description
(a r1 (b r2 c))
((a (mem ul '(allow accept)))
(r1 (or arg1 arg0))
(b (== ul 'percent))
(r2 circ)
(c measure))
(concat (sentence a) (card-ul b)))
```

Extracted information:

FEIM-GML-RULE items-must-show returns the following text:

1. The details of container number and the engine and chassis number of vehicles stuffed into each and every container must be shown on the packing list [*note that the accompanying sentence has been dropped*].
2. Bills of lading must not contain documentary credit.
3. Bills of lading must indicate Shipper as SING INTERNATIONAL PTE LTD, SINGAPORE.

FEIM-GML-RULE documents-from-nego-bank returns the following text:

4. All documents must be dispatched direct to DBS Bank, 6 Shenton Way, DBS Building, Singapore 068809 in 1 lot by courier service [*note that the sentence has been paraphrased*].

FEIM-GML-RULE tolerance-clause-2 returns the following text:

5. 10% more or less on credit amount and merchandise quantity is allowed.
6. Tolerance-limit = 10%.

Paraphrasing:

All documents to be sent to opening bank in *one* lot by *courier service*.

Paraphrased to:

All documents must be dispatched direct to DBS Bank, 6 Shenton Way, DBS Building, Singapore 068809 in 1 lot by courier service.

Information inference:

MFE:

1. identifies the tolerance clause from the free-form text;
2. extracts the tolerance clause:
3. “10% more or less on credit amount and merchandise quantity is allowed.”
4. infers the value of the tolerance limit from the tolerance clause: tolerance limit = 10%;
5. extracts the face amount value;
6. calculates the liability amount: liability amount = tolerance limit × face amount.

Change propagation:

Extracted tolerance clause:

“10% more or less on credit amount and merchandise quantity is allowed.”

Clerk modifies clause to:

“A tolerance of *twenty percent* on credit amount is acceptable.”

MFE automatically:

1. infers a new tolerance limit value: tolerance Limit = 20%
2. re-calculates the new liability amount: liability amount = tolerance limit × face amount.

## 7. Conclusion

The GIE architecture relies much on the existence of large, comprehensive lexicons as well as grammar rules to be able to analyze the text to accurate MIR structures. Though the GIE architecture already has a large, near complete and comprehensive set of lexicons and grammar rules for standard and typical english sentences, it is still impossible for complete coverage of the syntax one encounters in real-world texts.

However, as demonstrated in the MFE system for financial application, the GIE architecture works well when applied to well-defined, restricted and specific domain applications because of its knowledge-based nature. Other applications of the GIE architecture may be found in (Wee, 1998). Unlike the statistical method, the knowledge base is incrementally crafted and can be fine-tuned for better accuracy. In the initial development of the system, relatively less effort of fine-tuning is required to raise the accuracy of the information extraction system. However, when the knowledge base accuracy reaches a certain threshold, much effort in fine-tuning is required for achieving a small percentage improvement in accuracy. This threshold value varies for different domain applications and the size of the knowledge bases.

As the GIE architecture models after the expert’s train of thoughts, the knowledge base is intuitive and it makes adding or modifying the knowledge rules relatively easier. Further, changes to the knowledge base can be immediately tested unlike a statistical system where the entire system has to be retrained in order to test. The training time for such statistical systems vary for different applications and the time can be non-trivial especially during the development as well as live operation.

The GIE architecture offers two levels of text analysis for different complexity of information to be extracted, namely, the surface text analysis phase and the deep text analysis phase. Simple information that is easily identified for extraction can already be extracted during the surface text analysis phase and need not undergo extensive text analysis. Only relevant amount of text analysis is required depending on the complexity of the information to be extracted. This

makes processing more efficient. Moreover, the deep text analysis phase offers some form of text understanding and this is essential for complex information.

The architecture unifies the domain ontology with the rest of the knowledge structures in order to facilitate the knowledge specification of the target information to be extracted. This reduces the amount of redundant extraction rules and significantly reduces the size of the knowledge base. Further, the domain ontology can be reused for other applications within that same domain.

The design of the GIE architecture has incorporated practical operational requirements such as being able to infer or derive other information from the extracted information. Extracted information is also cross-checked to highlight discrepancies of either derived or extracted information. Further, when a user modifies a wrongly extracted information, the system can automatically re-verify itself based on the modified information.

## References

- Allen, J. (1987). *Natural language understanding*. Menlo Park, CA: Benjamin/Cummings.
- Bobrow, G. D., & Winograd, T. (1977). An overview of KRL a knowledge representation language. *Cognitive Science*, 1 (1), 3–46.
- Bobrow, G. D., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., & Winograd, T. (1977). *GUS, a frame-driven dialog system artificial intelligence*. Amsterdam: North-Holland pp. 155–173.
- Brachman, R. J., & Schmolze, J. G. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9, 171–216.
- Charniak, E. (1978). On the use of framed knowledge in language comprehension. *Artificial Intelligence*, 11, 225–265.
- Costantino, M., Morgan, R.G., Collingham, R.J., Garigliano, R. (1997). Natural language processing and information extraction: qualitative analysis of financial news articles. *Proc. of conference on computational intelligence for financial engineering (CIFEr 1997)*, New York City, 23–25 March.
- DARPA (1991). *Proc. of third message understanding conference (MUC-3)*. Morgan Kaufmann Publishers Inc.
- DARPA (1992). *Proc. of fourth message understanding conference (MUC-4)*. Morgan Kaufmann Publishers Inc.
- DARPA (1993). *Proc. of fifth message understanding conference (MUC-5)*. Morgan Kaufmann Publishers Inc.
- DARPA (1995). *Proc. of sixth message understanding conference (MUC-6)*. Morgan Kaufmann Publishers Inc.
- Jensen, K., Heidorn, E. G., & Richardson, D. S. (1993). *Natural language processing: the PLNLP approach*. Dordrecht: Kluwer Academic Publishers Ch. 16 pp. 203–214; Ch. 21 pp. 273–283.
- Kupiec, J., Pedersen, J.O., Chen, F. (1995). A trainable document summarizer. *Proc. of the 18th ACM/SIGIR Conference, 1995* (pp. 68–73).
- Nyberg, H.E. (1988). *The FrameKit User's Guide Version 2.0*, Carnegie Mellon University.
- Riloff, E., & Lehnert, W. (1994). Information extraction as a basis for high-precision text classification: special issue on text categorization. *Journal of ACM Transactions on Information Systems*, 12, 296–333.
- Schutze, H., Hull, D.A., Pedersen, J.O. (1995). A comparison of classifiers and document representations for the routing problem. *Proc. of the 18th ACM/SIGIR Conference, 1995* (pp. 229–237).
- Terry, D., & Loeb, S. (1992). In D. Terry & S. Loeb (Eds.), *Special section on information filtering, communication of the ACM*, 35(12), 26–81. New York: ACM.
- Tong, L.C. (1995). *Tapestry: a toolkit for practical NLP applications, technical report*, Kent Ridge Digital Labs, Singapore.
- Wee, L. K. A., Tong, L. C., & Chng, T. J. (1997). DeNews - a personalized news system. *Journal of Expert Systems with Applications*, 13, (4) 249–257.
- Wee, L.K.A. (1998). *A generic information extraction architecture*. MSc Thesis, School of Computing, National University of Singapore, to be submitted in end of 1998.