

# **TIPSTER Text Phase II Architecture Requirements**

**Prepared by :  
Architecture Committee  
for the  
TIPSTER Text Phase II Program**







# TABLE OF CONTENTS

<b>1.0 INTRODUCTION</b>	<b>1</b>
1.1 Requirements Traceability	1
1.3 Requirement Document Notes	2
1.4 Architectural Concept	2
1.5 Architecture Description Language	2
1.6 TIPSTER Functional Areas	3
1.7 Testability	3
1.8 User Interface	3
1.9 User Interaction	3
1.10 Relationships of Major Functional Areas	3
1.11 Technical References	3
<b>2.0 GENERAL TIPSTER FEATURES</b>	<b>5</b>
2.1 Task Handling	5
2.2 Multi-Lingual	5
<b>3.0 PERSISTENT KNOWLEDGE REPOSITORY</b>	<b>7</b>
3.1 Persistent Knowledge Sharing	7
3.1.1 Common Lexicon	7
3.1.2 Common Gazetteer	7
3.1.3 Common Parts of Speech Word Lists	7
3.1.4 Common Grammar Rules Library	8
3.1.5 Common Document Structure Library	8
3.1.5.1 Basic Near Term Requirement	8
3.1.5.2 Enhanced Far Term Requirement	8
3.1.6 Common SGML Tag Sets Dictionary	8
3.1.7 Common Complete Template Library	8
3.1.8 Common Template Object Library	8
3.1.9 Common Pattern Library	9
3.1.10 Common Detection Criteria Library	9
3.1.11 Common Stemming Library	9
3.1.12 Common Stop Word List	9
3.1.13 Common Phrase Lists	9
3.1.14 Common Predicate-Argument Dictionary	9
3.1.15 Common Term Expansion Dictionary	9
3.1.16 Common User Annotation Library	10
3.1.17 Common Document Inverted Index	10
3.1.17.1 Legacy Applications	10
3.1.17.2 New Applications	10
3.1.18 Machine Readable Dictionaries	10
<b>4.0 DOCUMENT MANAGEMENT</b>	<b>11</b>
4.1 Document Attributes	11
4.2 Other Attributes	11
4.3 Original Document	11
4.4 Document Sets	11
4.5 Corrections	11
4.6 Adaptive Document Structures	11
4.7 Annotation	12
4.7.1 Permanent Annotations	12
4.7.2 Multiple Usage	12
4.7.3 Annotation Structure	12
4.7.4 Annotation Groups	12
4.7.5 User Annotation	12
4.8 Document Control	12

4.8.1 Access Control	12
4.8.2 Document Ownership	12
4.8.3 Version Control	13
4.8.4 External Information	13
4.9 Large Corpus	13
4.10 Document Code Set Conversion	13
<b>5.0 USER INFORMATION REQUEST</b>	<b>15</b>
5.1 User Defined Detection Criteria	15
5.1.1 Selection Statements	15
5.1.2 Keyword Criteria	15
5.1.3 Other Detection Criteria	15
5.1.4 Detection Criteria Retention	15
5.1.5 Defining Document and Criteria Zones	15
5.1.5.1 Basic Near Term Requirement	16
5.1.5.2 Enhanced Far Term Requirement	16
5.1.6 Query	16
5.1.7 Multi-term Item Criteria	16
5.1.8 Foreign Language Document Detection	16
5.1.9 Foreign Language Document Detection Criteria Assistance	16
5.1.10 Query and Detection Criteria Refinement	16
5.1.11 Rapid Query Evaluation	17
5.1.12 Prioritization of Selection Statements	17
5.2 User Defined Extraction Criteria	17
5.2.1 Template Schema	17
5.2.2 Fill Rules	17
5.2.2.1 Basic Capability	17
5.2.2.2 Enhanced Capability	17
5.2.3 Constructing New Template Schema	18
5.2.3.1 Manual Processes	18
5.2.3.2 Automated Processes	18
5.2.4 Template Slot Filling	18
5.2.5 Multi-lingual Extraction	18
5.3 Document Clustering	18
<b>6.0 USER INFORMATION OUTPUT</b>	<b>19</b>
6.1 Document Viewing	19
6.2 Document Ordering	19
6.3 Merged Results	19
6.4 Document Grouping	19
6.5 Marking of Significant Text	19
6.6 Viewing and Editing Filled Templates	19
6.7 Component Processing Confidence	20
6.8 Processing Log	20
6.9 Document Processing Statistics	20
6.10 Error and Diagnostic Messages	20
<b>7.0 SCOPE OF PROCESSING SERVICES</b>	<b>21</b>
7.1 Detection	21
7.1.1 Compiled Query Creation	21
7.1.2 Attribute and Text Retrievability	21
7.1.3 Query Grouping	21
7.1.4 Relevance Estimates	21
7.1.5 Multiple Document Sets	21
7.1.6 Routing Considerations	22
7.1.7 Prioritization of Documents	22
7.2 Information Extraction	22
7.2.1 Extraction Input From Detection	22

7.2.2 Criteria Processing By Extraction	22
7.2.3 Abstracts	22
7.2.4 Extraction Objects	23
7.2.5 Output of Annotations or Filled Templates	23
7.2.6 Finding Identical Documents	23
7.2.7 Extraction Evaluation	23
<b>8.0 INTERFACE CONTROL DOCUMENT</b>	<b>25</b>
8.1 Modularity	25
8.2 Interchangeability	25
8.3 Specific Interfaces and Protocols	25
8.4 Application Language	25
8.5 Extensible Architecture	25
<b>9.0 OPERATING ENVIRONMENT</b>	<b>27</b>
9.1 Research Framework	27
9.2 Transportability	27
9.3 Scalability	27
9.4 Tools	27
<b>10.0 HIGH LEVEL GOALS</b>	<b>29</b>
10.1 Robustness	29
10.2 Plug & Play Components	29
10.3 Common Functions	29
10.4 COTS Packages	29
10.5 Security Considerations	29
10.6 Development Time	29
10.7 Maintenance Cost	29
10.8 Application Lifetime	29
10.9 Minimize Skilled Labor	30
10.10 Manual Intervention	30
10.11 Multi-media	30
10.12 Response Time	30
<b>APPENDIX A - DOCUMENT ATTRIBUTES</b>	<b>31</b>
<b>APPENDIX B - GENERIC INFORMATION TYPES</b>	<b>35</b>
<b>APPENDIX C - GENERIC INFORMATION TYPES</b>	<b>37</b>



## 1.0 INTRODUCTION

### 1.1 Requirements Traceability

The requirements herein are derived from several Government agencies. Some requirements may be traced to specific documents given below. Interviews with Government personnel were also a source. When possible, the source documents, shown as (n), indicate the basis for the TIPSTER requirement:

0. Derived Requirement.
  1. BAA 93-36 and Scenarios
  2. Architecture Requirements (draft), Sarah Taylor, 13 March 1994
  3. FBIS CONOPS (draft), MITRE, 28 February 1994
  4. ADEPT CONOPS (working draft), MITRE, 4 February 1994
  5. Requirements Overview, W. B. Schultheis, 8 April 1994
  6. Requirements Document, W. B. Schultheis, 20 July 1994
  7. Analyst Requirements for TIPSTER Phase II, Based on BAA Scenarios, 15 March 1993
  8. TIPSTER Phase II Architecture Requirements, Terry Hand, 8 March 1994
  9. TIPSTER Architecture Concept, 5 July 1994

### 1.2 Requirement Document Terms

**Application** is a group of components and modules, both internal and external to the TIPSTER Architecture, that operates on documents to answer a User's request for information from documents. An Application is a "complete package" including any processing necessary to setup documents for TIPSTER processing and the User interface component. An Application may contain a Detection Component, Extraction Component or Clustering Component or any combination thereof.

**Compiled Query** is the Detection Component specific form of a query generated by a Detection Component from a Statement of Relevance and is understandable only by the Detection Component.

**Component** is equivalent to a Computer System Component (CSC) in the conventional life-cycle definition. A CSC is a distinct part of a Computer Software Component Item(CSCI). CSCs may be further decomposed into other CSCs and CSUs. A detection component is an example of a CSC. The size, function and complexity of an item is a guide to calling the item a 'component'. Components are major parts of an Application; they are built from modules. Note: The use of the term 'component' in the Architecture Design Document is in a different context, namely, part of the structure of a generic object.

**Detection Component** is a component that selects documents from a Collection based upon the Detection Criteria.

**Detection Criteria** is the User statement of information need and may include Selection Statements , short form free text queries, Boolean keyword queries or example documents

**Document Attribute** is a characteristic of a document as represented by a single specific value or a set of values of the same type e.g., date received or authors.

**Document Collection** is an unordered set of documents.

**Document List** is an ordered list of document identifiers and optional document attributes or computational result for each list entry.

**Extraction Component** is a component that selects information from a document or a list of documents based upon the Template structure, Fill Rules and Patterns.

**Fill Rules** are a collection of criteria which describe the constraints used to select information for Template slots and the conditions under which Template Objects are instantiated. The specific formats for the fills is also included.

**Module** is equivalent to a Computer System Unit (CSU) in the conventional life-cycle definition. A CSU is an element specified in the design of a CSC that is separately testable. A parser is an example of a module. Modules are used to build components. Generally modules are composed of no more than 300 lines of code.

**Multi-lingual** is considered to be multiple languages in one document or multiple documents in different languages.

**Pattern** is an expression of a specific form that is used for matching text during the Extraction process.

**Query** is the Detection Component specific form of a query generated by a Detection Component in the component's unique structure and language. It is understandable by a User and processable by the Detection component.

**Selection Statement** is a high level textual description of the needed information as specified by the User.

**Template** is a form or structure which identifies one or more objects that are associated together as the result of an extraction process. The specific values, for each type, are placed in template slots to fill the template.

**Template Object** is a group of associated slots. One or more linked objects form a Template.

**User and User Group** refer to the analyst(s) controlling and interacting with the Application through the User Interface.

### 1.3 Requirement Document Notes

Note 1: This document presents Requirements for the TIPSTER Architecture. Every Application implemented under the TIPSTER Architecture will need an Application Requirements Document that selects specific features allowed by the TIPSTER Architecture as well as specification of any non-TIPSTER capabilities. The Application will also need a User Interface Document that defines User commands, screen layouts and the sequence of User operations. See Paragraph 1.8.

Note 2: Not all Requirements supported by the Architecture will necessarily be included in an Application. The Architecture provides a 'shopping list' of capabilities that may be included in the Application Requirement Document, as appropriate.

Note 3: A key concept of the Architecture is that any implementation of a TIPSTER Architecture Requirement shall be fully compatible with the TIPSTER Requirements that have not been implemented.

### 1.4 Architectural Concept

The Architecture shall provide for the development of common, shared, logical modules, components and interfaces that can be configured to support various text document processing tasks in different Applications and natural languages. More specific background may be found in Reference 9.

### 1.5 Architecture Description Language

The Architecture shall be documented in a formal language that describes the components and modules and how they interact using an object oriented methodology. The Interface Control Document (ICD) shall unambiguously define the Architecture component and module interfaces so that these parts of the Architecture may work together to produce the intended results. (3, 5)

Verification Method: Inspection.

## **1.6 TIPSTER Functional Areas**

The TIPSTER Architecture can be divided into the following major functional areas:

- Persistent Knowledge Repository
- Document Management
- User Information Requests
- User Information Outputs
- Scope of Processing Services
  - Detection
  - Extraction

Requirements will be assigned to either the above functional areas or the supporting areas of:

- Interface Control Document
- Operating Environment
- High Level Goals

## **1.7 Testability**

Implementation of requirements must meet verification criteria either through demonstration tests or by inspection.

## **1.8 User Interface**

The User Interface provides the mechanism for the user to interactively control various events and processes, and to examine results. The User Interface functional area is NOT part of the TIPSTER Architecture. However, the User Interface, that is concerned with screen layout, windows, command choices, output displays and the sequence relationships between these items, is closely tied to User Information Requests and User Information Outputs areas. The User Information Request area accepts, for TIPSTER processes, inputs from the User Interface and the User Information Outputs that sends TIPSTER results to the User Interface for display. This boundary is the interface between the TIPSTER Architecture and the User Interface implementation for a specific Application.

## **1.9 User Interaction**

User interaction shall be part of an Application through modular interfaces. (3,4,5)

Verification Method: Demonstration and inspection.

## **1.10 Relationships of Major Functional Areas**

In addition to the relationships identified in Paragraph 1.8, Document Management, Detection and Extraction accept inputs from User Information Request and provide outputs to User Information Output. The Document Management area is also closely related to Detection and Extraction by providing documents for these two areas as well as performing document accounting and control.

## **1.11 Technical References**

- (a). Guidelines for Electronic Text Encoding and Interchange, TEI P3, Vol.1 & 2, ACH, ACL, ALLC
- (b). ISO 8879: 1986, SGML standards



## **2.0 GENERAL TIPSTER FEATURES**

The Architecture shall provide appropriate information to support TIPSTER features that are included in specific Applications. Features developed as a result of the requirements specified in this document will generally, but not always, be initiated by the User through User Interface commands that may be passed to the User Information Request for processing. The results from such processing, or other internal TIPSTER processes, will be passed to the User Interface via the User Information Output. Supporting details, for those requirements considered to be an internal part of the Architecture, will be found in the remainder of this document.

### **2.1 Task Handling**

Processing tasks shall operate interactively with progress status shown to the User or in the background with progress status available upon User request. Depending upon the specific task the User may direct the task to operate interactively or in the background mode. Tasks may be canceled or the mode changed. The nature of this requirement implies co-operation between the Operating System, the Application and the Architecture. The Architecture shall facilitate this co-operation by allowing status information to be updated and passed between components.(5, 6, 7)

Verification Method: Demonstration.

### **2.2 Multi-Lingual**

The Architecture shall allow documents to be processed in any human language, provided the appropriate knowledge bases have been built and language specific modules are available. This includes the processing of documents which contain multiple languages in a document. The Architecture shall be based on multi-lingual layers so that additional languages can easily be added, without recreating the basic Application. The existence and use of multi-lingual layers shall not have significant, adverse effect on single language processing. (5, 6, 7)

Verification Method: Inspection.



### **3.0 PERSISTENT KNOWLEDGE REPOSITORY**

The Persistent Knowledge Repository is a common place (storage device) where information may be retained.

The goal of Persistent Knowledge is to further the use of common knowledge items as new Applications are built. A significant pay-off may accrue through common usage, particularly in the development of operational Applications. It is recognized that there may be some reduction in capability when common knowledge items are used in an Application instead of unique, customized knowledge items; however, the history of other technical areas indicates that common items and sharing has merit and possible pay-off.

Generally, Persistent Knowledge is that knowledge which is retained from one run to the next of an Application(s). However, there are degrees of persistency. Some items, such as, lexicons, gazetteers, glossaries, dictionaries, marking rules and grammars rules, are comparatively stable, subject to normal maintenance operations. A Query Library, on the other hand, may be modified frequently. Results from Detection, Extraction or User actions may be used to modify the items in the Persistent Knowledge Repository.

Specific algorithms, even those which use items in the Persistent Knowledge Repository, are not considered part of the Repository, but instead are included in the particular instance of an implemented requirement; however, the structure of the particular knowledge item is an Architectural item.

It should be noted that while Persistent Knowledge requires a place to keep knowledge items and the format of the item or storage area, not all items must be completely filled initially under the TIPSTER project. Some items will be filled, grow or be augmented as Applications are implemented under the Architecture. As a minimum the format and access method of each Persistent Knowledge item shall be established. Since Persistent Knowledge items will require frequent access the format and access design shall place a high priority on efficiency.

Requirement 2.2 for multi-lingual capability is applicable for this section and Persistent Knowledge items should be designed to apply to multiple languages.

#### **3.1 Persistent Knowledge Sharing**

The Architecture shall allow for sharing of Persistent Knowledge items. This includes sharing by different components in an Application and by different Applications built in accordance with the TIPSTER Architecture. (2, 8)

Verification Method: Inspection.

##### **3.1.1 Common Lexicon**

The Architecture shall provide for the use of a common Lexicon that can support various document processing tasks in different Applications. The intent of this requirement is to allow new Applications to adopt and modify existing TIPSTER lexicons, where suitable. The common lexicon format should cover a frequently used set of data fields and be applicable across languages. (5, 8)

Verification Method: Demonstration and inspection.

##### **3.1.2 Common Gazetteer**

The Architecture shall provide for the use of a common Gazetteer that can support various document processing tasks in different Applications. (5, 8)

Verification Method: Demonstration and inspection.

##### **3.1.3 Common Parts of Speech Word Lists**

The Architecture shall provide for the use of common parts of speech word lists with a range of different tags to support various document processing tasks in different Applications. These are the word lists used by a part of speech tagger. (5)

Verification Method: Demonstration and inspection.

### **3.1.4 Common Grammar Rules Library**

The Architecture shall provide for the use of common grammar rules that can support various document processing tasks in different Applications. This may be considered a far term requirement. (0)

Verification Method: Demonstration and inspection.

### **3.1.5 Common Document Structure Library**

The Architecture shall provide for the use of a common Document Structure Library which will store formal descriptions of the structures of specific document types to aid in their processing. Reference (b), SGML standards, defines the concept of a DTD, Document Type Definition. Reference (a), TEI, addresses the concept and retention of common document structures as well as methods for assembling complex structures, including graphics, multi-media and text, from basic structure definitions. This is accomplished through the use of DTDs. While much of the scope of Reference(a), TEI is directed to embedding the associated tagging information during document creation in order to support publication needs, it also provides supporting concepts that can be used to dynamically mark documents. Document structure definitions are not restricted to the TEI methodology; however, whatever method is used, the document structure definition shall be stored in the Library so as to facilitate use by new Applications(0)

#### **3.1.5.1 Basic Near Term Requirement**

A basic library defining commonly used message types and standard document divisions, such as communication header and text, shall be available for Applications implemented in the near term. The near-term requirement shall be met through the use of the Core Tag Sets and the Base Tag Sets as defined in Reference (a), TEI, or optionally and less desirable, by document structure definitions not restricted to the TEI methodology;

Verification Method: Demonstration and inspection.

#### **3.1.5.2 Enhanced Far Term Requirement**

An enhanced library with richer membership which covers more complex document structures than the basic library shall be implemented over the long term. The long-term requirement shall be met through the use of Additional Tag Sets as defined in Reference (a), TEI.

Verification Method: Demonstration and inspection.

### **3.1.6 Common SGML Tag Sets Dictionary**

The Architecture shall provide for the use of a Common SGML Tag Sets Dictionary The basis for these Tag Sets shall be Reference (a), TEI. and any augmentation provided by various Applications. (5)

Verification Method: Demonstration and inspection.

### **3.1.7 Common Complete Template Library**

The Architecture shall provide for the use of a common Complete Template Library that can support various document processing tasks in different Applications. Templates in the library shall be composed of template objects, fill rules and patterns associated with them. This is a library of templates already defined and used by TIPSTER applications (0, 8)

Verification Method: Demonstration and inspection.

### **3.1.8 Common Template Object Library**

The Architecture shall provide for the use of a common Template Object Library that can support various document processing tasks in different Applications. The Library shall contain common objects composed of slot definitions with fill rules but without patterns. The purpose of this library is to allow construction of templates using pre-defined objects. (0, 8)

Verification Method: Demonstration and inspection.

### **3.1.9 Common Pattern Library**

The Architecture shall provide for the use of a common Pattern Library that can support various document processing tasks in different Applications. The purpose of this library is to provide patterns strings for use in building new template filling capability. (0, 8)

Verification Method: Demonstration and inspection.

### **3.1.10 Common Detection Criteria Library**

The Architecture shall provide for the use of a common Detection Criteria Library containing statements of user information needs as well as the associated Application translated, user understandable, queries that can support various document processing tasks in different Applications. Reuse of such criteria can facilitate the building and modification of requests for retrieval and routing. (0, 8)

### **3.1.11 Common Stemming Library**

The Architecture shall provide for the use of a common Stemming Library that can support various document processing tasks in different Applications. The Library shall include word stems, prefixes and suffixes; thus a stem may be identified by direct look-up or the word may be parsed and parts compared with prefixes or suffixes, as appropriate, to identify a word stem. Stems, prefixes, suffixes may reside in separate parts of the Library so as to improve lookup efficiency.(0)

Verification Method: Demonstration and inspection.

### **3.1.12 Common Stop Word List**

The Architecture shall provide for the use of common Stop Word Lists that can support various document processing tasks in different Applications. Different Stop Word Lists shall be applicable to different parts of a document to allow for different usage/meaning of the same word. (0)

Verification Method: Demonstration and inspection.

### **3.1.13 Common Phrase Lists**

The Architecture shall provide for the use of common Phrase Lists that can be shared by various document processing tasks in different Applications. This requirement means that domain specific Phrase Lists can be a shareable resource. (0)

Verification Method: Demonstration and inspection.

### **3.1.14 Common Predicate-Argument Dictionary**

The Architecture shall provide for the use of a common Predicate-Argument Dictionary that can support various document processing tasks in different Applications. (0)

Verification Method: Demonstration and inspection.

### **3.1.15 Common Term Expansion Dictionary**

The Architecture shall provide for the use of a common Term Expansion Dictionary that can be used to look up equivalent terms, variation terms, synonym terms or abbreviation expansions to support various document processing tasks in different Applications. In general, there may be Application dependencies with the same term having different meanings, depending upon the Applications(0)

Verification Method: Demonstration and inspection.

### **3.1.16 Common User Annotation Library**

The Architecture shall provide for the use of a common User Annotation Library that provides a repository for pre-defined User Annotations in the form of partially or fully completed Annotations for pre-defined document locations or associated with particular attributes that a User may use for comments about the document or the Annotations created by the Application. Specific Annotation type(s) shall be assigned for User Annotations. (6)

Verification Method: Demonstration and inspection.

### **3.1.17 Common Document Inverted Index**

The Architecture shall provide for the use of a common simple Document Inverted Index for existing and future Applications. (0)

#### **3.1.17.1 Legacy Applications**

New Applications shall share existing indices from legacy applications to the maximum extent possible.

Verification Method: Demonstration and inspection.

#### **3.1.17.2 New Applications**

New Applications shall provide Application Program Interfaces (APIs) to a simple Document Inverted Index so that future Applications may use the indices to access the Application's Collection.

Verification Method: Demonstration and inspection.

### **3.1.18 Machine Readable Dictionaries**

The Architecture shall support the use of Dictionaries in machine readable form (e.g. CD-ROM) to support document processing tasks in different Applications.

## **4.0 DOCUMENT MANAGEMENT**

The Architecture Concept, Reference (9), identifies several typical forms of documents, Form 0 through Form 4. Form 0 is the original document and Forms 1 and 2 are intermediate forms of the document. The Document Management process is concerned with Form 3 and Form 4 documents. A Form 3 document is the input to TIPSTER and Form 4 is the internal TIPSTER form. Document sources shall be in machine readable form and may be from communication lines or from computer files.

### **4.1 Document Attributes**

There are a variety of document attributes that may be used by TIPSTER processing functions. Some of these attributes such as Date of Information, Author or Source may be used directly by Detection to select documents. Other attributes such as Original Language or Code Set may be used to control the internal TIPSTER processing. The specific, necessary group of attributes is Application dependent. See Appendix A for a list of the most common attributes that might be used by an Application. (5)

Verification Method: Inspection and demonstration.

### **4.2 Other Attributes**

Other document attributes may be specified by the Application, by the operating system, by the installation, by the user group and by the user. These may include such varied items as: processing network configuration, access/security control information or document collection names. (5)

Verification Method: Inspection and demonstration.

### **4.3 Original Document**

A document shall always be available in an original form. The Application may define this to be a Form 1, Form 2 or Form 3 type of document. See Reference (9) (5)

Verification Method: Demonstration.

### **4.4 Document Sets**

Documents shall be managed in such a way that ordered and unordered groups can be created. These groups may be comprised of sets with various characteristics such as, source, publisher or discourse. The document sets can have access controlled through Access Control that limits usage to the specified User or User Group.

### **4.5 Corrections**

Corrections to a document shall be applied so as to leave the original document, as defined in Paragraph 4.3, intact. Corrections are made through the use of Annotations. Revision numbers shall be associated with correction Annotations. A new persistent document may be created by applying corrections and assigning the appropriate revision number. Revised, corrected documents may be processed as any other Form 3 document. It shall be possible to make corrections Annotations to corrections Annotations. (5, 8)

Verification Method: Demonstration.

### **4.6 Adaptive Document Structures**

Adaptive identification of formats and document structure may be made based upon representative documents. This implies a degree of learning about document structures as they change over time. Representative document structures are maintained in the Document Structure Library. (4).

Verification Method: Demonstration and inspection.

## **4.7 Annotation**

Annotations are information added to a document by User or computer processing. The Architecture shall recognize various types of annotations associated with specific passages of text as specified by a text span with begin and end values or the whole document. An Annotation to an Annotation is permitted. Also, it should be possible to obtain all Annotations associated with a particular document location through specific begin and end location values. Retrieval of Annotations should also be possible by type, by Annotation Group and for the entire document. (5, 7, 8)

### **4.7.1 Permanent Annotations**

Provisions must be made for permanent Annotations. These may be stored separately from the document to which they apply. In such cases appropriate references or links shall tie the Annotations to the document. (5, 7)

Verification Method: Demonstration.

### **4.7.2 Multiple Usage**

A single annotation may be associated with multiple document locations (spans). (5)

Verification Method: Demonstration.

### **4.7.3 Annotation Structure**

Annotations shall be extensible, in accordance with the Architecture Maintenance Policy, as the Architecture matures. In addition to a reference mechanism(span and document ID) that identifies the scope of the Annotation, the two principal parts are the Annotation Type identification and the specific information in the Annotation. Common Annotation Types shall be defined and these definitions shall be maintained as part of the Architecture Configuration Policy. New Annotation Types may be created. (5)

Verification Method: Inspection.

### **4.7.4 Annotation Groups**

Annotations may be treated together as sets of the same or differing types. These groups are necessary for retrieval purposes and to control processing. (0, 7, 8)

Verification Method: Inspection

### **4.7.5 User Annotation**

A type of Annotation is required to allow the User to make notes about a document, other Annotations or the Detection and Extraction processing. User Annotations may have access controlled by the mechanism specified in Access Control. (6, 7, 8)

Verification Method: Inspection.

## **4.8 Document Control**

Basic Document Control includes the maintenance of document collections, document lists, correction and version records, ownership record, access control and security. (0, 8)

### **4.8.1 Access Control**

Access control mechanisms shall recognize multiple levels such as individual Users and hierarchical User Groups. (3, 5)

Verification Method: Demonstration.

### **4.8.2 Document Ownership**

Access to documents and identification of ownership shall be via one control module so as to maintain access and ownership integrity. (5)

Verification Method: Demonstration.

### **4.8.3 Version Control**

Successor versions to the original document shall be marked with a revision number and the document ID and the cause for revision recorded as a document history attribute. (5)

Verification Method: Demonstration.

### **4.8.4 External Information**

Applications shall not be prevented from interfacing to external source control information. In general, the Architecture shall not restrict an Application from using:

- a. Any usage control or authorization required by a particular electronic feed to facilitate determination of usage charges
- b. Any access control mechanism required by a legacy system
- c. Any privacy or user ownership authorization required to add extractions to a database not normally part of the Application
- d. Any network access control parameters
- e. Any security requirements imposed by a host computer system. (3, 4, 5, 8)

Verification Method: Inspection.

## **4.9 Large Corpus**

Document Collections shall not have arbitrary size limits so as to enhance retrieval and extraction. Detection and extraction shall be possible from a very large corpus. Any sub-setting of the Corpus must be transparent to the User during detection and extraction operations. (3)

Verification Method: Demonstration.

## **4.10 Document Code Set Conversion**

The Architecture shall support conversion of document files written in various character encodings to a standard encoding before TIPSTER processing. The conversion shall be from a selected set of conversion tables and a set of conversion algorithms for the documents. The Architecture permits a standard Code Set for all internal operations. Retention of the original document is still required. The Architecture shall recognize on-going work of Reference (a) in the area of Code Sets. (5, 7)

Verification Method: Demonstration.



## **5.0 USER INFORMATION REQUEST**

User Information Requests define the manner in which the Application shall operate and perform its various tasks. These criteria are usually originated by the User; however, they may be integral to the specific Application. They are embodied in Selection Statements, other Detection Criteria, Queries, Routing Query and Templates. User commands from the User Interface component pass through the User Information Request area to initiate specific TIPSTER operations; for example, show a document, create a private collection, show a document list, etc. User information requests shall be presented in the language of the document except for two cases noted below.

### **5.1 User Defined Detection Criteria**

The User shall be able to establish the document detection criteria in a variety of formats. These include creating a free text Selection Statement, a shorter free text need statement, example document or keyword Boolean statements with negative operators to specify the desired documents or sub-sets of documents. Previously created and saved detection criteria may be obtained from the appropriate library and modified. Criteria used for retrospective search and for routing shall have the same formats available. Different types of criteria may be used together, such as a Selection Statement with keywords. (3, 4, 6, 7, 8)

#### **5.1.1 Selection Statements**

The Application should provide any necessary assistance for Selection Statement generation. Selection Statements shall describe, in free text form, the kind of information the user requires. The statement shall be sufficiently complete so that relevance can be determined with reasonable certainty. The format for Selection Statements must be common and sharable between Applications. (0, 6, 7, 8)

Verification Method: Inspection.

#### **5.1.2 Keyword Criteria**

Detection criteria may be stated as Boolean keyword criteria and negative operators to exclude documents. Keyword stated criteria may include statements of document attributes, such as author, source, date of composition, date of receipt, country of origin, etc. (0, 6, 7, 8)

Verification Method: Demonstration.

#### **5.1.3 Other Detection Criteria**

Other Detection Criteria may be of two types, a natural language short form query or example documents. A natural language short form query may consist of a sentence(s) or phrase(s). (0, 6, 7, 8)

Verification Method: Demonstration.

#### **5.1.4 Detection Criteria Retention**

All forms of detection criteria shall be storable in the appropriate library, should the user desire, to be retrieved and modified by that or other users.

Verification Method: Demonstration.

#### **5.1.5 Defining Document and Criteria Zones**

The Architecture shall accept statements identifying the portion of a document that can be used for matching and which portions of the criteria shall be used to match a specific portion of a document. This includes the specification of title, abstract and paragraphs as the document portion. Weights may be assigned to signify relative importance of a portion. (6, 7, 8)

Verification Method: Demonstration.

#### **5.1.5.1 Basic Near Term Requirement**

A basic capability shall be available for Applications implemented in the near term. This shall include the ability to differentiate between different communication header fields and between communication headers and message text.

Verification Method: Demonstration.

#### **5.1.5.2 Enhanced Far Term Requirement**

An enhanced capability with additional and more advanced zoning than the basic capability, such as referencing titles, abstracts, introductions, conclusions and captions, shall be available for Applications implemented over the long term.

Verification Method: Demonstration.

#### **5.1.6 Query**

The user shall be able to view a detection component's interpretation of the submitted Detection Criteria. This query shall be in a format that is understandable by an interested user. The user shall be able to relate this query to both the criteria and the document it retrieves. The user shall be able to modify this component version query directly and shall be able to create new Detection Criteria in this component's format if he learns the format.

Verification Method: Demonstration.

#### **5.1.7 Multi-term Item Criteria**

In the detection process, items such as personal names, organization names, equipment names, locations, dates and identification numbers, may be treated as single units when specified as such by the user. (0, 8)

Verification Method: Demonstration.

#### **5.1.8 Foreign Language Document Detection**

The Architecture shall support a detection component that allows foreign language documents to be searched or distributed using English language Selection Statements. A multi-lingual retrieval or routing Application may have a separate module for each language it handles, but builds an index that is language independent. The component shall return a single ranked list of documents in multiple languages. Foreign language documents in the list may be annotated with English glosses of selected words and phrases. The Application can provide a custom browsing interface to display annotated documents to the user. (5, 6, 8)

Verification Method: Demonstration.

#### **5.1.9 Foreign Language Document Detection Criteria Assistance**

The Architecture shall allow assistance to the native speaker of English in formulating detection criteria in a foreign language. This includes providing interfaces for a program that shall take an English Selection Statement and shall return a version of the Selection Statement in another language or several versions in several different languages. The Architecture shall maintain the logical association between the English version of the query and the foreign language version of the query. (6, 7, 8)

Verification Method: Inspection.

#### **5.1.10 Query and Detection Criteria Refinement**

Refinement of all types of Detection Criteria, including the queries for retrospective search or routing, shall be supported by the Architecture. Relevance tags may provide guidance for the refinement; also, as an option, documents already seen may be suppressed from the re-run. (2, 3, 8)

Verification Method: Demonstration.

### **5.1.11 Rapid Query Evaluation**

Rapid and semi-automatic evaluation of query and profile changes with test information shall be allowed. It shall be possible to compare the results from pairs of Detection Criteria so as to evaluate query specifications. (4, 6, 7, 8)

Verification Method: Demonstration.

### **5.1.12 Prioritization of Selection Statements**

The Architecture shall allow prioritization to affect the manner in which documents are retrieved and presented for review. Priorities may be attached to Detection Criteria, including Selection Statements or portions thereof. In particular, the User shall be able to prioritize references to personalities, events, objects, times, or locations as well as identifying the priority of Detection Criteria in a submission of multiple criteria statements. (6, 7, 8)

Verification Method: Demonstration and inspection.

## **5.2 User Defined Extraction Criteria**

Presently, this process is labor intensive and requires a co-operative effort between Users and Application developers. The objective is to allow the End User or Developer to establish the criteria for extracting information from documents. This may include creating Templates, Patterns and Fill Rules needed for extraction and keeping items in libraries for future use. Toward this objective the Architecture should support more automation with interactive assistance to the User and Developer in preparing these items. Sample or training Templates may be used to assist the User. (0, 4, 7, 8)

Verification Method: Demonstration.

### **5.2.1 Template Schema**

The Architecture shall accept as input Template Schema with empty slot relationships and treated as formatted information. Templates may be of varying complexity, from just single entities to Message Understanding Conference (MUC) type Templates. The language used to specify Template Schema should be that which is evolving under MUC. (0, 5, 8).

Verification Method: Demonstration.

### **5.2.2 Fill Rules**

The Architecture allows the specification of criteria for correct filling of each template slot and object. These criteria may be provided as a "Fill Rules Document" or as a sufficient number of examples of correct fills with context obtained from sample text. The extraction component shall determine how to fill the Template based upon these criteria. (0, 8)

Verification Method: Inspection.

#### **5.2.2.1 Basic Capability**

The basic capability shall involve developers manually (or through appropriate toolsets) building an Extraction component, based upon narrative descriptions of the Fill Rules and/or the examples of correct fills.

Verification Method: Demonstration.

#### **5.2.2.2 Enhanced Capability**

The enhanced capability shall be based upon machine tractable Fill Rules and/or automatically developed rules from examples.

Verification Method: Demonstration.

### **5.2.3 Constructing New Template Schema**

The Architecture shall permit complete Templates, Template Objects and Patterns to be stored in their respective libraries. These items may be retrieved, modified and stored as new items. When Templates Objects are retrieved the Fill Rules associated with each slot shall also be retrieved to assist in understanding existing Templates, constructing new Templates or modifying the fill criteria. Template Objects and Patterns may be selected, modified and re-combined or attached to different Templates to establish new extraction criteria. (2, 4, 7, 8)

Verification Method: Demonstration.

#### **5.2.3.1 Manual Processes**

In the near term, Templates Objects and Patterns, that is the component specific structures which allow the extraction component to recognize slot fills, shall be constructed/combined by manual or semi-automated processes and the appropriate libraries updated by normal maintenance operations. (2)

Verification Method: Demonstration.

#### **5.2.3.2 Automated Processes**

In the future, the Architecture shall allow construction/combining of Templates Objects and Patterns by automated processes and the libraries updated automatically. The goal is to permit the development of Template Schema and fill criteria connected to component specific patterns via an interactive process by a user with subject domain knowledge. (2, 4, 7, 8)

Verification Method: Demonstration.

### **5.2.4 Template Slot Filling**

This requirement is applicable to Requirements 5.2.1, 5.2.2 and 5.2.3. The Architecture shall support, as a minimum, the following information types as slot fillers:

- a. String fills
- b. Set fills
- c. Hierarchical set fills
- d. Normalized fills
- e. Pointers to other entities.

The Architecture shall specify how to store and pass filled Templates between components as well as the detail representation of these types. Filled templates shall appear as Annotations with links to relevant text spans in the source document. Security requirements are applicable to this requirement. (6, 7, 8)

Verification Method: Demonstration and inspection.

### **5.2.5 Multi-lingual Extraction**

The Architecture shall allow a multi-lingual extraction component that represents Template definition in any combination of multi-lingual Fill Rules, Template Objects, slots and documents. For example, Fill Rules may be in a language different than the language of the source document. The Architecture shall support Template, Object and slot-level language and code set identification, as necessary, either in the Template Schema (5.2.1) or in individual filled Templates. (6, 7, 8)

Verification Method: Inspection.

## **5.3 Document Clustering**

The Architecture shall allow requests for document clustering provided the appropriate clustering algorithms and interfaces to Document Management and User Information Output are established. (0)

Verification Method: Inspection.

## **6.0 USER INFORMATION OUTPUT**

User Information Outputs are all of the selected or computed information that is created by the processing initiated by the User Information Input in the form of criteria and commands. User Information Outputs are passed to the non-TIPSTER User Interface component that provides services identified in Paragraph 1.7. (0, 3, 4, 5)

### **6.1 Document Viewing**

The Architecture shall allow an Application to define which parts of formatted or structured documents, including document attributes, summaries, abstracts, subject, word delineated zones, etc., get displayed when a document is shown to the user. The Architecture shall allow viewing of documents to commence before a query operation is completed, if appropriate for the particular detection component. (3, 4, 5, 6, 7, 8)

Verification Method: Demonstration.

### **6.2 Document Ordering**

Architecture shall allow the User to order document lists to support document viewing by any document attribute(s), result(s), Annotation(s), Template(s), slots or combination thereof related to document Detection or Extraction processing. For example, view by date, source, relevance rank and template slot. (4, 5, 6, 7, 8)

Verification Method: Demonstration.

### **6.3 Merged Results**

The Architecture shall allow detection results from different collections or different detection components to be combined for viewing purposes. A common sorting and priority ordering may be applied by the User. (6, 7, 8)

Verification Method: Demonstration.

### **6.4 Document Grouping**

The Architecture shall allow the grouping of documents based upon common occurrences of specific strings, nearly identical passages of text or similar Template Objects. The selection specification may identify the length of the string that is considered for "identical" selection. Identical or nearly identical documents may be viewed together or removed from the viewing list. The output shall result from either finding all "identical" documents in a collection which match a specific document or finding all "identical" documents in a collection based only on the selection specification. (6, 7, 8)

Verification Method: Demonstration.

### **6.5 Marking of Significant Text**

The Architecture shall support tagging of the text strings in a document, that caused selection of the document so that when the document is passed to the User Interface for viewing highlighting or other notification may be applied. The Architecture shall also support tagging of text that caused a particular slot to be filled. Additionally, text that caused template or object instantiation shall be tagged. (6, 7, 8)

Verification Method: Demonstration.

### **6.6 Viewing and Editing Filled Templates**

The Architecture shall make filled Templates available to the User Interface for viewing, editing and disposition. Editing includes modification or deletion of any tags or links. Any assessment of the component confidence of the slot fills shall also be available for viewing and if tags or links were edited by the user the confidence field may also be edited. (5, 6, 7)

Verification Method: Demonstration.

## **6.7 Component Processing Confidence**

The detection and extraction components may provide an estimate of the component confidence level about each document selected or each piece of information that has been extracted. In the case of the extraction component estimate, it is the component's confidence in its process for filling each slot. (6, 7)

Verification Method: Demonstration and inspection.

## **6.8 Processing Log**

The Architecture shall allow an Application to create a Processing Log which records each individual process step of an Application run and any related errors. The log may be turned on or off or particular levels of reporting may be selected, e.g. critical error, warning error, etc. (4)

Verification Method: Demonstration.

## **6.9 Document Processing Statistics**

The Architecture shall allow an Application to collect document processing and error statistics. Processing statistics shall include various counts related to document processing and progress status during runs. Error statistics shall include counts of the number of times particular errors occurred. The specific level and scope of operational statistics is Application dependent, but may include Document Management, Detection and Extraction statistical items. (3, 7)

Verification Method: Demonstration.

## **6.10 Error and Diagnostic Messages**

Error reporting shall be done through an error message format which is common to all Applications. The diagnostic portion of message shall indicate the source and cause of the error and if possible the correction necessary to fix the error. (0)

Verification Method: Demonstration.

## **7.0 SCOPE OF PROCESSING SERVICES**

The Architecture shall address document detection and information extraction. The statistical, linguistic and semantic analysis performed by Detection and Extraction may be considered the key work of TIPSTER processing. It is here that document information is manipulated, in various ways, to support the identification of a desired sub-set of documents or extracted information. This is accomplished by applying Persistent Knowledge information in conjunction with matching algorithms, annotations and template/pattern matching techniques to reduce a Document Collection to the desired sub-set or to extract information from the Document Collection. This sub-set or its extracted information is then presented to the user, passed to other components or stored for later use.

The analytical methods and tools representative of the techniques to be applied to this functional area are given in Appendix B; however, this list should not be considered either exhaustive or restrictive.

The Architecture shall support various generic information types that are applicable to TIPSTER processes. The use of any particular type of information is dependent upon the Application. Appendix C gives a list of the more common generic types that may be used by an Application. (0, 5, 7, 8)

### **7.1 Detection**

The Detection process shall use the Detection Criteria in conjunction with Document Management and the Persistent Knowledge Repository to select and route documents from document sets. The specific search and identification algorithms are dependent upon the particular Application. Annotations may be created and/or used by the Detection component. The Detection component shall be compatible with other components and modules of the TIPSTER Architecture. (0, 7, 8)

#### **7.1.1 Compiled Query Creation**

Compiled Queries are the forms of the Detection Criteria, generated by the detection capability from the Selection Statement and used for document selection. Compiled Queries operate for both retrospective retrieval and routing applications. Compiled Queries shall indicate and retrieve the document sub-sets that are of interest to the User. Query generation may be done automatically by the Application or with help from the User. (3, 6, 7, 8)

Verification Method: Demonstration.

#### **7.1.2 Attribute and Text Retrievability**

Retrieval must be allowed on any document text string or attribute, such as source, author, date, etc. Retrieval must also be allowed on Annotations by type, Annotation Group and User Annotations. (5, 6, 7, 8)

Verification Method: Demonstration.

#### **7.1.3 Query Grouping**

Multiple queries may be applied in one pass against a specific collection. (6, 7)

Verification Method: Demonstration.

#### **7.1.4 Relevance Estimates**

The query process shall determine and report relevance estimates. (3, 7, 8)

Verification Method: Demonstration.

#### **7.1.5 Multiple Document Sets**

Queries may be simultaneously applied to multiple document sets. (6, 8)

Verification Method: Demonstration.

### **7.1.6 Routing Considerations**

Every document detection component shall be able to serve as part of a document routing function that compares new documents from a specified source to, potentially, very large numbers of profiles from many users. The routing function is expected to have compared, within some specified interval that may be minutes to hours, each new document with all profiles. Certain categories of documents shall be processed at higher priority than others, if required by the Application, for example, FLASH messages before ROUTINE messages. Depending upon the Application, as soon as a document has been routed, it must be available for retrospective search. In these cases the routing process shall coordinate its operations with the process that controls the indexing of documents for search and retrieval. (3, 6, 7, 8)

Verification Method: Demonstration and inspection.

### **7.1.7 Prioritization of Documents**

Documents may be assigned priorities based on matches between documents and prioritized portions of queries, profiles and Selection Statements. Various algorithms for combining weights, when there are multiple matches in one document, may be selected. Priority scores can be weighted, based on where in the document the match occurs, such as title, source attribute, etc. If any portion of a document matches a portion of a query that has a priority attribute, then the document is assigned that priority. The priority can be scaled, based on where in the document the match occurs. If several such portions of the query match, then an algorithm can combine priorities to get a net priority for the document. The priority, once calculated is available as an attribute for sorting documents for subsequent processing, whether viewing by the user, information extraction, or subsequent routing. (6, 7, 8)

Verification Method: Demonstration and Inspection.

## **7.2 Information Extraction**

The Extraction process shall use Templates, Fill Rules, Patterns or other methods in conjunction with Document Management, Detection and the Persistent Knowledge Repository to extract desired information from documents in Document Collections. The specific extraction methodology and algorithms are dependent upon the particular Application. Annotations may be created and/or used by the Extraction component. The Extraction component shall be compatible with other components and modules of the TIPSTER Architecture. Typical factual information to be extracted includes, but is not limited to, entities, objects, events, entity relationships and event relationships. (6, 7, 8)

### **7.2.1 Extraction Input From Detection**

The Architecture shall support the concept of using Detection to filter documents as the input to the Extraction process, however, the Extraction process may operate independently from the Detection process, if required by the Application. (0, 8)

Verification Method: Demonstration and inspection.

### **7.2.2 Criteria Processing By Extraction**

The Architecture shall support the processing of Selection Statements and the natural language portions of the user defined Detection Criteria by the Extraction component, i.e. the desired information to be extracted may be specified through the use of natural language and the specifics identified by an Extraction component. The resulting specifics can be used to aid in the query formulation. (0)

Verification Method: Demonstration.

### **7.2.3 Abstracts**

The Architecture shall allow Extraction to provide Abstracts or document summaries. The specific quality and content of the Abstract are Application dependent. (0, 8)

Verification Method: Demonstration.

#### **7.2.4 Extraction Objects**

The Architecture shall recognize a standard set of objects. All information extraction components are expected to be able to extract instances of these objects. These objects are expected to be part of an expanding library that shall be augmented as Applications are implemented. See Paragraph 3.1.8. (6, 8)

Verification Method: Demonstration and inspection.

#### **7.2.5 Output of Annotations or Filled Templates**

The output of an information extraction component may include a set of document Annotations and optionally, one or more lists of filled templates. Each template should support a tabular or spreadsheet view of extracted information. The building of a data base is beyond the scope of the Architecture. However, it shall support the access to specific information to build a data base by allowing an Application to use filled templates. (6, 7, 8)

Verification Method: Demonstration and inspection.

#### **7.2.6 Finding Identical Documents**

The Architecture shall provide for identifying documents as identical or nearly identical through Detection, Extraction or other methods. (6)

Verification Method: Demonstration and inspection.

#### **7.2.7 Extraction Evaluation**

The Architecture shall provide an interface to existing standard extraction module evaluation tools. This shall include the use of a test corpora, although actual evaluation is not part of the Architecture. (6, 8)

Verification Method: Demonstration and inspection.



## **8.0 INTERFACE CONTROL DOCUMENT**

The Interface Control Document is the defining document identifying specific inputs and outputs for TIPSTER components and modules.

### **8.1 Modularity**

The Architecture shall provide for modularity. The ICD shall unambiguously define the Architecture component interfaces. These interfaces shall describe all the external information needed by the component. This serves to bound the component and there-by modularizing it. (2, 5, 8)

Verification Method: Inspection.

### **8.2 Interchangeability**

The Architecture shall provide for interchangeability. The ICD shall unambiguously define the Architecture component interfaces. The inputs and outputs shall be defined with sufficient detail to allow an Application's TIPSTER components and modules to be exchanged with similar TIPSTER components or modules. This shall also allow vendors to develop alternative components that also meet the specifications of the ICD. In this way, the components shall be Interchangeable. (2, 8)

Verification Method: Inspection.

### **8.3 Specific Interfaces and Protocols**

The Architecture shall provide for specific interfaces and protocols. The ICD shall unambiguously define the Architecture component interfaces. This shall include any specific standards and protocols allowed in the Architecture. Included in this requirement is the specification of Application Program Interfaces (API). (0)

Verification Method: Inspection.

### **8.4 Application Language**

Interfaces should be specified to facilitate future development of an Application language. Such a high level language would allow the construction of Applications by use of APIs corresponding to various modules and components of the Architecture. (5)

Verification Method: Inspection.

### **8.5 Extensible Architecture**

The Architecture shall provide for extension and adoption of new implementation approaches. The ICD shall unambiguously define the Architecture component interfaces. This shall provide a basis for any future enhancements to the Architecture. Enhancements can only be envisioned and designed when the base Architecture is well defined. (0)

Verification Method: Inspection



## **9.0 OPERATING ENVIRONMENT**

The Operating Environment is concerned with such items as client/server schemes, file handling methods, operating systems, communications and support items.

### **9.1 Research Framework**

The Architecture shall provide a design that can serve as an efficient research framework. (2)

Verification Method: Demonstration.

### **9.2 Transportability**

The Architecture shall provide a design that maximizes platform transportability. The use of capabilities which are Operating System or environment dependent must be clearly identified and modularized so as to isolate them from transportable components and modules. (2, 3, 4, 8)

Verification Method: Inspection and demonstration

### **9.3 Scalability**

Components shall be scaleable to a large number of documents and a high document flow rate; up to a maximum of 1,000,000 documents per day with access to 2 terabytes of text. (2, 3, 4, 5, 8)

Verification Method: Inspection.

### **9.4 Tools**

Tools and enhancements to assist in applying the Architecture to new tasks, applications and languages should be identified and, where possible, developed. (1, 7)

Verification Method: Demonstration.



## **10.0 HIGH LEVEL GOALS**

These requirements represent goals that are applicable to the entire Architecture but are not assigned to specific functional areas.

### **10.1 Robustness**

The Architecture shall recognize the importance of robustness[doesn't "break"]. (3, 4)

Verification Method: Inspection.

### **10.2 Plug & Play Components**

The Architecture shall recognize the need, where possible, to have modularity for the purpose of plug-and-play, reuse of software, ease of development, minimum maintenance and extended Application life time. (2, 3, 4, 7, 8)

Verification Method: Inspection and demonstration.

### **10.3 Common Functions**

The Architecture shall support the use of common functions that are Application independent for Application implementation. (5, 8)

Verification Method: Inspection and demonstration.

### **10.4 COTS Packages**

Applications shall use COTS packages to the maximum extent possible. (2, 3, 4, 5, 8)

Verification Method: Inspection and demonstration.

### **10.5 Security Considerations**

Architectural choices should recognize the desirability of incorporating multi-level security in component implementation. The Architecture shall support the Application security requirements of the organization responsible for the Application, for example, security labels on processes and/or data items. In such cases labels shall not be separable from the process or data item. (5, 8)

The architecture shall support physical and software boundaries to devices where documents and data are stored. The boundaries shall ensure that persons only have access to the appropriate level of classified information. These may be implemented at the API level of the software components.

Auditing and administrative support shall be available for marking and filtering data to ensure proper document/data access, distribution and viewing and also to record improper access attempts. The scope of marking of data may be at the document, paragraph, data item or object level.

Verification Method: Inspection.

### **10.6 Development Time**

The Architecture shall recognize, as a goal, the need to minimize Application development time. (2, 8)

Verification Method: Inspection.

### **10.7 Maintenance Cost**

The Architecture shall recognize, as a goal, the need to minimize Application maintenance cost. (2)

Verification Method: Inspection.

### **10.8 Application Lifetime**

The Architecture shall recognize, as a goal, the need extend an Application's lifetime through Application upgrading. (2)

Verification Method: Inspection.

### **10.9 Minimize Skilled Labor**

The Architecture shall minimize the need for special skills in programming or information preparation when operating and adjusting Applications. (3, 4)

Verification Method: Inspection.

### **10.10 Manual Intervention**

The Architecture shall minimize the need for manual intervention, when appropriate. (3)

Verification Method: Inspection.

### **10.11 Multi-media**

The Architecture shall recognize the future need for multi-media processing and transport. (3)

Verification Method: Demonstration.

### **10.12 Response Time**

The Architecture shall recognize the importance of appropriate response times, particularly in the interactive mode, and not impede implementations from meeting accepted standards. The goal is 2 seconds for interactive activities such as document or list displays and a few tens of seconds for activities such as query or search that require significant computer resources. (2, 3, 8)

Verification Method: Inspection and demonstration.

## APPENDIX A - Document Attributes

The list below identifies typical Document Attributes that may be encountered in any TIPSTER Application. This list is open ended. The actual attributes are dependent upon the specific TIPSTER Application. Descriptions of attributes marked with \* may be found in Reference (a).

- Form 1 Document ID
- Form 1 Document Location
- Form 2 Document ID
- Form 2 Document Location
- Form 3 Document ID
- Form 3 Document Location
- Document owner
- Primary source
- Secondary source
- Original languages
- Other languages
- Character code set
- Date/time received
- Date published
- Date of information
- Author(s)
- Publisher
- Discourse
- Category
- Document type
- Revision number
- Reason for revision
- Document condition flag
- Security classification
- \*Title
- \*Sponsor
- \*Funder
- \*Principal
- \*Intellectual responsibility person
- \*Edition,
- \*Date of information

- \*Publication date
- \*Distributor
- \*Authority to distribute
- \*Place published
- \*Publishing address
- \*Publishing idno
- \*Publishing restrictions
- \*Series title
- \*Series idno
- \*Series intellectual responsibility
- \*Notes
- \*Bibliographic title reference
- \*Bibliographic extract type
- \*Editor
- \*Transcribe recording method
- \*Transcribe recording equipment
- \*Encoding description
- \*Encoding project description
- \*Encoding sampling
- \*Encoding editorializing
- \*Encoding tags
- \*Encoding references
- \*Profile description,
- \*Language usage
- \*Text description
- \*Text description channel
- \*Text description constitution
- \*Text description derivation
- \*Text description domain
- \*Text description factuality
- \*Text description interaction
- \*Text description preparedness
- \*Text description purpose
- \*Text classification
- \*Text classification keyword
- \*Text classification code

\*Text classification category reference

\*Editorial correction

\*Editorial normalization

\*Editorial quotation

\*Editorial hyphenation

\*Editorial segmentation

\*Editorial standardization values

\*Editorial interpretation

The identification and location of major document parts such as:

Graphics

Tables

Text body

Multi-media part

The identification and location of major document structural elements such as:

Beginning of a Form 3 Document

End of a Form 3 Document

Communication Header

Title

Abstract

Names of saved information that is associated with a document as a result of TIPSTER processing, such as:

Annotation group names

Annotation group locations

Private collection names

Data Base Name for extracted information

Record identifier for extracted information



## APPENDIX B - Generic Information Types

The list below identifies Typical Analytical Methods that may be encountered in any TIPSTER Application. This list is open ended. The use of a particular method is dependent upon the specific TIPSTER Application.

- Matching conditions and exceptions
- Coreference tagging
- Document annotating
- Document corrections (spelling)
- Dot products
- Field and range comparison
- Likelihood computation
- Part of speech tagging
- Probability computation
- Relevance ranking
- SGML tagging
- N-gram retrieval
- Sorting
- Statistical metrics
- TF/IDF scaling
- Template and pattern building
- Threshold limits
- Statement of Relevance analysis
- Vector representation



## APPENDIX C - Generic Information Types

The list below identifies typical Generic Types that may be encountered in any TIPSTER Application. This list is open ended. The use of particular types is dependent upon the specific TIPSTER Application.

Annotation  
 Collection list  
 Correction  
 Document attribute  
 Attribute list  
 Document collection  
 Document list  
 Document part tag  
 Fill Rules  
 Form 3 Document  
 Form 4 Document  
 Index (inverted, general)  
 List (general)  
 Persistent Knowledge Repository Item  
 Pattern  
 Profile  
 Query  
 Relevance value  
 Saved item (File)  
 SGML Tag  
 Template  
 Statement of Relevance  
 User request command  
 User output information

## INDEX

abstracts ..... 16, 19  
 algorithms ..... 7, 13, 21, 22  
 analytical methods ..... 21  
 annotation ..... 10, 12, 19, 33, 37  
 annotation group ..... 12  
 annotation types ..... 12

API.....	25, 29
application.....	1, 2, 3, 5, 7, 9, 10, 11, 13, 15, 16, 17, 19, 20, 21, 22, 23, 25, 29, 31, 35, 37
application program interfaces.....	10, 25
attribute.....	1, 21, 37
auditing.....	29
background.....	2, 5
Boolean.....	1, 15
clustering.....	18
code set.....	11, 13
communication header.....	8, 16
compiled query.....	1, 21
component.....	1, 2, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 29
confidence level.....	20
configuration policy.....	12
conversion.....	13
corrections.....	11
COTS.....	29
data base.....	13, 23
detection.....	1, 2, 3, 7, 9, 11, 12, 15, 16, 17, 19, 20, 21, 22, 23
detection criteria.....	1, 9, 15, 16, 17, 21, 22
developer.....	17
diagnostic messages.....	20
dictionaries.....	7
document.....	1, 2, 5, 7, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 20, 21, 22, 23, 25, 27, 29, 30, 33
document collection.....	1, 13, 21
document history.....	13
document id.....	12, 13
document list.....	1, 12, 15, 19
document management.....	3, 11, 20, 21, 22
document ownership.....	12
document structure.....	8, 11
DTD.....	8
English.....	16
environment.....	27
error statistics.....	20
evaluation.....	17, 23
example document.....	1, 15
extraction.....	2, 17, 18, 20, 21, 22, 23
fill rules.....	1, 2, 8, 17, 18, 22, 37
fills.....	2, 17, 18, 19
foreign language.....	16
form 0.....	11
form 2.....	11, 31
form 3.....	11, 31, 33, 37
form 4.....	11, 37
free text.....	1, 15
gazetteers.....	7
glossaries.....	7
grammar.....	8
grouping.....	19
ICD.....	2, 25
identical documents.....	19
information types.....	18, 21
interchangeability.....	25
interface.....	2, 3, 5, 15, 19, 25

interface control document .....	2, 25
inverted index .....	10
keyword .....	1, 15, 32
language .....	2, 5, 15, 16, 17, 18, 22, 25
legacy applications .....	10
lexicons .....	7
library .....	7, 8, 9, 10, 11
links .....	12, 18, 19
maintenance operations .....	7
maintenance policy .....	12
marking of data .....	29
marking rules .....	7
merged results .....	19
Message Understanding Conference .....	17
modularity .....	25
module .....	2
MUC .....	17
multiple document locations .....	12
negative operators .....	15
network .....	11, 13
objects .....	2, 8, 17, 22, 23
operating system .....	11, 27
operational statistics .....	20
order document lists .....	19
ordered .....	1, 11
ownership record .....	12
parts of speech .....	7
pattern .....	2, 9, 37
patterns .....	8, 9, 18
permanent annotations .....	12
persistent knowledge .....	3, 7, 21, 22, 37
phrase .....	9
prioritization .....	17, 22
private collection .....	15
processing log .....	20
processing statistics .....	20
protocols .....	25
query .....	1, 2, 7, 15, 16, 17, 21, 22, 37
relevance .....	1, 16, 21, 35, 37
research .....	27
response time .....	30
retrieval .....	16, 21, 22, 35
retrospective search .....	15, 16, 22
reuse of software .....	29
revision number .....	11, 13
robustness .....	29
routing .....	15, 16, 21, 22
scaleable .....	27
security requirements .....	18
sets .....	8, 11, 13, 21
SGML .....	8, 35, 37
size limits .....	13
slot .....	8, 17, 18, 19, 20
sorting .....	19, 22
span .....	12, 18

statistics.....	20
stemming.....	9
stop word.....	9
synonym terms .....	9
tagging.....	8, 19, 35
template.....	1, 2, 8, 9, 17, 18, 19, 21, 23, 35, 37
template object .....	2, 8, 18, 19
template objects .....	8
template schema .....	17, 18
text span .....	12, 18
tools.....	27
transportability .....	27
unordered .....	1, 11
user annotations.....	10, 12
user group.....	11, 12
user information output .....	3, 5, 19
user information request.....	3, 5, 15
user interface .....	2, 3, 5, 15, 19
viewing of documents .....	19
weight.....	15
zones .....	15