**People Locator Email Processing Task (PLIET)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**NLP Implementation and Processing Accuracy**

***(Preliminary)***

**December 2013**

**Table of Contents**

[1 Introduction 1](#_Toc375131101)

[2 PLIET Problem Definition and Challenges 1](#_Toc375131102)

[2.1 Scenario 1](#_Toc375131103)

[2.2 Problem Definition 2](#_Toc375131104)

[2.3 Challenges 3](#_Toc375131105)

[3 Use of NLP for Extracting Reported Person Information 4](#_Toc375131106)

[3.1 Publicly Available NLP Tools used by PLIET 4](#_Toc375131107)

[3.2 PLIET Developed Components 5](#_Toc375131108)

[3.2.1 Person/Location Recognition 5](#_Toc375131109)

[3.2.2 Health Status Interpretation 5](#_Toc375131110)

[3.2.3 Coreference Resolution 5](#_Toc375131111)

[3.2.4 Sentence Structure Analysis 6](#_Toc375131112)

[3.2.5 Text Matching 6](#_Toc375131113)

[4 Overview of PLIET TEXT Processing 6](#_Toc375131114)

[4.1 High Level Design 6](#_Toc375131115)

[4.2 Basic Annotation Generation 8](#_Toc375131116)

[4.3 Annotation Processing 8](#_Toc375131117)

[5 Processing Details 10](#_Toc375131118)

[5.1 Sentence Decomposition and Clausal Hierarchy Determination 10](#_Toc375131119)

[5.2 Clausal Analysis and Determination of Unique Subjects 12](#_Toc375131120)

[*5.3* Interpretation of Predicates and Identification of *Reported Person* 13](#_Toc375131121)

[5.3.1 Ranking 15](#_Toc375131122)

[5.3.2 Data Transformation 16](#_Toc375131123)

[5.4 Generating the ReportedPerson Record 16](#_Toc375131124)

[5.4.1 Fragment Processing 16](#_Toc375131125)

[5.4.2 Looking for Special Annotations in the entire Message 17](#_Toc375131126)

[5.4.3 Merging all Information 17](#_Toc375131127)

[6 Implementation 17](#_Toc375131128)

[6.1 Extraction 18](#_Toc375131129)

[6.1.1 Anchor 18](#_Toc375131130)

[6.1.2 Clause 18](#_Toc375131131)

[6.1.3 Clausal Dependency 18](#_Toc375131132)

[6.2 Analysis 19](#_Toc375131133)

[Message Analyzer 19](#_Toc375131134)

[6.2.1 Attribute Analysis 20](#_Toc375131135)

[6.2.2 Name Resolution 20](#_Toc375131136)

[6.2.3 Anaphora Resolution 21](#_Toc375131137)

[6.2.4 Coreference Resolution 22](#_Toc375131138)

[6.2.5 Gender Determination 22](#_Toc375131139)

[6.3 Inference 23](#_Toc375131140)

[6.3.1 Structure and Contents of the PL Lexicon 23](#_Toc375131141)

[6.4 Use Cases 25](#_Toc375131142)

[7 Testing and Accuracy 27](#_Toc375131143)

[7.1 Text Corpora 27](#_Toc375131144)

[7.2 Accuracy 28](#_Toc375131145)

[8 Issues and Needs for Further Enhancement 28](#_Toc375131146)

[8.1 Parsing and Dependency Detection 28](#_Toc375131147)

[8.1.1 POS Tagging Errors 29](#_Toc375131148)

[8.1.2 Case Dependency 29](#_Toc375131149)

[8.1.3 Use of Interjection 29](#_Toc375131150)

[8.2 Anaphora Resolution 29](#_Toc375131151)

[8.3 PLIET Deficiencies 30](#_Toc375131152)

[8.3.1 Propagating inferences from a Clause to higher or lower levels 30](#_Toc375131153)

[8.3.2 Handling of Interrogative sentences 30](#_Toc375131154)

[8.3.3 Determining Locations 30](#_Toc375131155)

[8.3.4 Absence of a Subject in a Sentence 30](#_Toc375131156)

[8.3.5 Ranking 30](#_Toc375131157)

[8.3.6 Determining the Grammatical Accuracy of a Sentence 31](#_Toc375131158)

[8.3.7 Preliminary Nature of the PL Lexicon 31](#_Toc375131159)

[8.3.8 Dealing with Multiple Reported Persons 31](#_Toc375131160)

[9 Conclusion 31](#_Toc375131161)

[10 References 32](#_Toc375131162)

People Locator Email Processing Task (PLIET)

# Introduction

This is a summary report of the People Locator (PL) Email Processing Task (PLIET), intended to receive and process information about affected persons lost or found in a disaster in the form of emails, and send it to the People Locator [1] ReUnite database. PLIET is intended to provide an alternate way of sending information to PL, instead of through the People Locator Web-based Apps [2].The PLIET extracted information, entered into the ReUnite database, may then be searched using a Web browser. Currently, PLIET is not required to handle image attachments.

The text of the email is expected to be composed in the English language, but could be informal in nature. The emails may be sent from any part of the world, including regions with non-English native language. The main focus of the research-oriented task is to process and interpret the email text, using Natural Language Processing (NLP) techniques, to recognize and retrieve embedded information useful to PL.

The rest of this report is organized in the following manner:

1. PLIET problem definition and challenges
2. Use of NLP for extracting reported person information
3. Overview of text processing in PLIET
4. Processing details
5. Implementation
6. Test cases and accuracy
7. Issues and enhancement needs
8. Conclusion
9. References

# PLIET Problem Definition and Challenges

## Scenario

Following a disaster, a person at any location around the globe may send an email message to a dedicated ReUnite email account (such as [disaster@mail.nih.gov](mailto:disaster@mail.nih.gov)) to report or inquire about the condition of an affected person. The PLIET application, running on a server at CEB, continuously monitors (at pre-established, configurable time intervals) that mailbox for arrival of new messages, and reads and processes each new message as it arrives. The PL relevant information, retrieved and interpreted from the free text of the message is then formatted into an encoded XML stream, correlated to a specific disaster event, and sent to the PL Web Service to be entered into the PL’s ReUnite search database. Furthermore, PLIET also sends a response message back to the email sender with the value of each field as interpreted by the system.

## Problem Definition

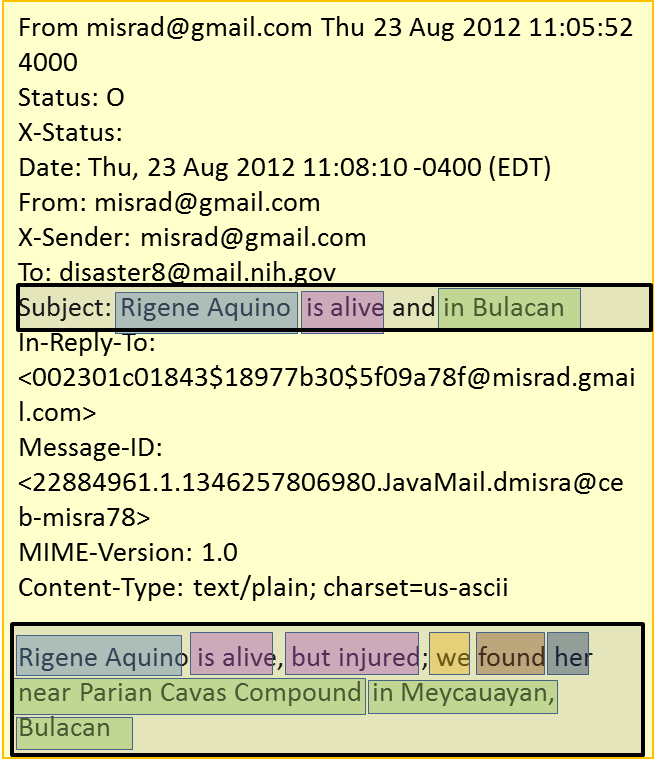
The PLIET task, corresponding to the above scenario, requires the following functionalities:

1. Monitoring and retrieving email messages from a specific email account and sending reply to each email originator
2. Processing the message text to extract a reported person’s ***name, health status, last known geographical location, gender*** and ***age***
3. Formatting and sending the results to the PL Web Server over a SOAP interface

Technical solutions for functions 1 and 3 are well-established, and simply require PLIET-specific implementation, whereas function 2 requires research activity involving NLP. Specific areas to focus in this regard are:

1. Processing of informal text
2. Recognizing names of persons and locations in the text
3. Identifying all references to the same person expressed in various ways in the text
4. Differentiating between a reporter and a reported person
5. Interpreting various phrases (single or a group of words) to recognize a health condition
6. Interpreting the underlying meaning of a recognized phrase in the context of the message

Figure 1 provides an example of extraction and interpretation of phrases from a disaster related email message, and contents of the corresponding Reported Person record.



Given name: Rigene

Family name: Aquino

Gender: Female

Age: Unknown

Health Status: Injured

Location: Meycauayan, Bulacan

Country: Philippines

***Reported Person record***

first name: Rigene

last name: Aquino

gender: female

age: unknown

address:

city: Meycauayan

region: Bulacan

country: Philippines

status: injured

***Email received at PL mailbox***

***PL NLP extracted data***

Figure 1 – Generation of a Reported Person record from an email using PLIET

## Challenges

The following paragraphs highlight the challenges regarding the extraction of pertinent fields from the PLIET-received emails mentioned in section 2.2.

1. **Processing of informal text:** The style of informal writing [3], such as the ones used in writing letters to a friend, sending an email or communicating among peers varies greatly from that in formal writings such as in a journal or newspaper article, a text book or an official letter. It may contain use of colloquial words, word contractions, use of first persons in the text, and may lack rigorous grammar. Other factors to be taken into account are: sentences with no explicit subject (usually derived implicitly from the context), as well as sentence fragments with no verbs. However, meaningless and incohesive sentences, poor syntax and grammar, spelling mistakes and lack of sentence breaks are not regarded here as characteristics of as informal text.
2. **Recognizing names of persons and locations:** Identifying a phrase as a person or a location, in NLP systems, is often based upon gazetteer lookups containing person first names, last names, important city/state look ups or recognition of keywords such as Street, Center, College etc. Most of these look-up tables are language dependent. Although PLIET email messages are in English, due to global origin of such messages, they are expected to contain non-English names. Therefore, their identification and person/location distinction have to be based upon syntactical analysis of sentences rather than on simple lookups and text matching.
3. **Resolving coreference of a person:** Coreference between entities in a discourse [4] includes both pronominal (anaphoric) and orthographic coreference. The former is resolved by identifying antecedent-anaphora/cataphora pairs within the text, and the latter is established for well-known persons, organizations and places using specific gazetteers created for such purpose [5]. (For example: *Barack Obama* and *President* or *New York* and *Big Apple*). In PLIET messages, orthographic coreference is not expected to occur, but contextual coreference between persons, expressed in different ways in sentences, needs to be addressed. Furthermore, pronominal coreference, connecting phrases referring to the same person in a message, need to be handled carefully rather than simply relying upon their grammatical resolution. It also requires resolution of first person anaphora, which is usually ignored by the available tools dealing with formal text.
4. **Recognition of health condition:** This requires a PLIET-specific lexicon - not only to define the health status related terms and their meanings, but also to specify related contextual conditions to perform word sense disambiguation for the detected terms.
5. **Interpretation of a person’s status:** This is a situation in which the meaning of a set of words recognized as a health condition and found within a lower level clause might be negated or otherwise changed based upon its connection to its higher level ancestor. In addition certain adjectives, determiners and keywords associated with a person or health condition in a sentence may also change the person’s status.
6. **Choosing the right elements from a list of potential candidates:** Both the subject and body fields of an email may contain information that would not necessarily be complementary or repetitive, but imply different or conflicting meanings. It is necessary to combine or chose the right elements from such message test to arrive at most probable outcome.
7. **Disambiguating a reported person from a reporter:** This function requires contextual analysis of information related to a person that is gathered from a message. It should also take into account that the same person may play both roles when he/she reports about self.

# Use of NLP for Extracting Reported Person Information

It is evident from the nature of the problem that tools based upon natural language processing are most suitable for dealing with the PLIET messages, and several NLP-based tools are available publicly for dealing with free text. Yet, most text processing problems pose their own complexities that cannot be fully resolved through generic tools. Such tools provide a broad platform on which new tools need to be built, using domain-specific techniques, to arrive at results with acceptable accuracy. Therefore, we have adopted the following procedure in processing the email text:

* Leverage upon publicly available, reliable open source NLP tools wherever possible
* Develop our own components, as necessary, to address PLIET-specific problems.
* To compensate for the informal (and possibly poorly constructed) sentences, utilize other text recognition techniques to supplement the NLP output.
* Integrate all tools to operate in a seamless manner and combine their outputs to get the best possible results.

## Publicly Available NLP Tools used by PLIET

Some of the well-known open source NLP tools used in text processing are as follows:

1. NLP Toolkits: A number of reliable open source NLP toolkits are currently available to aid in furthering NLP work; and are based upon machine learning using large corpora of formal text. They combine a number of lower level basic tools for standard operations, and allow user plug-ins for handling more complex situations. Well-known among them are OpenNLP [6], GATE/ANNIE [7] and the UIMA framework [8].
2. Stand-alone NLP Tools: These NLP tools deal with specific aspects of text processing. They include different types of parsers, gazetteer readers, and anaphora/coreference resolvers.

The open source NLP components used by PLIET are listed below:

1. **GATE/ANNIE –** This framework [9] was chosen by PLIET to provide basic NLP capabilities. GATE, in conjunction with its annotation generation engine ANNIE,handles most frequently required NLP functions such as tokenization, sentence segmentation, part-of-speech tagging, parsing, morphological analysis, named entity extraction, verb grouping and standard pronominal and orthographic coreference resolution. It is a well-supported and extensively used NLP platform, and incorporates an easy-to-use NLP Annotation pipeline into which user-defined tools may be plugged in using prescribed program interfaces or adapters. In additions to using language-specific gazetteers, GATE also uses simple rule-based analysis, using a Java-based scripting language called JAPE, to group words for named entity recognition (NER) of persons, places, etc.
2. **Stanford Dependency Parser** – The Stanford Dependency Parser (SDParser) [10][11] was selected by PLIET, in lieu of GATE’s standard parse, for parsing the email message text and annotating the dependency of a word with other words in the sentences. We found it most suitable for detecting sentence structure, performing clausal analysis, and retaining the contextual information in a sentence. Stanford Dependency Parser is also directly pluggable to the GATE/ANNIE System, with GATE provided adapters.
3. **Phil Gooch Anaphora Resolver** – This [12] is a more powerful pronominal coreference resolver than the GATE anaphora resolver, and comes as an external plug-in for ANNIE. It uses progressively pruned, double-linked lists to create in-document pronominal coreference chains for both backward and forward looking manner (anaphora, cataphora respectively). It also performs limited first person anaphora resolution, which is not useful for PLIET.

## PLIET Developed Components

A set of text processing modules were developed under PLIET to bridge the gap between the offerings of publicly available NLP resources and PLIET specific needs. They are used for PL related named entity recognition as well for resolution/disambiguation of key elements. Their overall functions are described below. Detail discussions are provided in Sections 4 through 6.

### Person/Location Recognition

This is a PLIET specific NER module that recognizes a name and/or a location, and disambiguates a location (which might be named after a person) from a person based upon several criteria. It examines the initial annotation of specific categories (such as *Person, Location, Lookup* and *Unknown*) generated by GATE/ANNIE using gazetteers and JAPE grammar, and refines/modifies those annotations based upon their syntactical usage in a sentence, as well as their usage in relation to specific disaster related nouns in the sentence. Location annotations are further verified, with latitude, longitude and country/state information using Google Geocode service.

### Health Status Interpretation

Terms and associated rules, which are generally used to indicate a person’s status and health condition following a disaster (flood, earthquake, tsunami etc.) through direct mention by a person or through enquiry/statements made by another person, are defined in a PL-specific lexicon. A set of PLIET-developed modules interpret these terms in the context of a message by following rules encoded in the PL Lexicon.

### Coreference Resolution

PLIET Coreference resolver combines different types of identity matching techniques mentioned earlier, such as first and third person singular anaphora matching, partial name matching, matching through appositional phrases and the *name* keyword.

### Sentence Structure Analysis

This function is implemented to analyze the structure of a sentence using grammatical features of component words and to lay it out in the form of hierarchical clauses to see the relationship among components, which is helpful in understanding the context, and interpreting the contextual meaning of a recognized entity.

### Text Matching

PLIET uses text matching to recognize and extract information from email text, based upon PL-specific keywords and regular expression matching. Keywords pertinent to different fields are specified in a set of gazetteers and are added to the GATE/ANNIE standard set - to be marked as annotations of the designated types in text look-up operation during the front-end annotation generation process.

Figure 2 shows how various NLP techniques, described above, are used by PLIET to extract pertinent fields in the email message shown in Figure 1.

Rigene Aquino is alive, but injured. We found her near Parian Cavas Compound in Meycauyan, Bulacan.

1. Person:

Unknown name;

derived through dependency tracing of parsed tokens

2. Parian Cavas: Unknown,

Resolved as Location

3. Meycauyan, Bulacan.

Resolved as Location:

Confirmed (with details) from Google Geo-coder service

6a. alive, injured → Aquino [*subject*]

6b. found → her (Aquino) [*object*]

5. female

4. her → Aquino (antecedent of unknown gender)

Figure 2 - Entity resolution in a sample message in PLIET using various NLP techniques

# Overview of PLIET TEXT Processing

Email processing operations in PLIET are modularized such that the three main functions mentioned in section 2.1 are performed independent of each other. The overall PLIET system design is presented in Appendix A. The rest of this document deals with the second function: namely, processing of the textual contents of an email using NLP, and the recognition of information specific to a lost or found person reported in the email.

## High Level Design

Text processing of disaster emails within PLIET is split into two parts: (a) Generation of basic annotations (b) Processing of annotated text for information extraction – both parts performed in a pipeline manner in sequential steps, as shown in Figure 3. **Email Process Controller** in the figure represents the highest level module, which controls the end-to-end processing of an email.

The PLIET-specific PL/*GATE Annotation Pipeline*, shown as the RHS component in Figure 3, is initialized once and then executed by the **PL ANNIE Pipeline Executor** for each email message. It generates the basic annotations for each email’s text, which are then used as the primary input for the in-house developed modules *in the PL NLP Pipeline*. The PLIET developed module, called the **Person-Location Recognizer,** is plugged into this GATE pipeline to better recognize and disambiguate Person and Location annotations than what is produced by the basic GATE tools.

The annotation processing modules within the *NLP Pipeline* perform the following sequential functions: (1) data extraction (2) information analysis (3) inference generation and (4) record augmentation, and provide the essential PLIET functionalities discussed in Section 3.3. If more than one person is inferred to be an affected person reported in an email, a ranking scheme is used to select the most likely candidate from this set - to be entered into the PL database as the *Reported Person*.

**Email Text**

**Data Extractors**

**Inference Generators**

**Ranker**

**Analyzers**

***PL NLP Pipeline***

**Email Process Controller**

**Email**

**Processed Results**

**PL ANNIE Pipeline Executor**

**ANNIE Annotations**

**Legend:**

Email/text contents

Annotations *(Tokens, Anaphors, Person, Location…)*

Intermediate output

Final result (Reported Person data)

PLIET developed tools

External tools

**Stanford Dependency Parser**

**Pronoun Annotator (Phil Gooch)**

**PL Person/Location Recognizer**

***PL/GATE Annotation Pipeline***

**GATE/ANNIE**

**(Standard plug-ins)**

**Field Augmenters**

Figure 3 – Functional modules governing the flow and processing of email data in PLIET Pipeline

Note that in this design, the dependency on GATE/ANNIE is limited to the front end only, so that system may use a different NLP Platform in future, if necessary, with minimal impact.

## Basic Annotation Generation

The subject and body fields of an email are processed by the GATE/ANNIE pipeline, resulting in the standard ANNIE annotation of the text. In addition, certain other PL-specific annotations are also generated and put into the output annotation set to be used in subsequent *text matching* operations. Besides the standard GATE tools, the SDParser, the Phil Gooch Pronoun Annotator and the PL-developed Person-Location Recognizer are plugged into this pipeline to either generate new annotations, or add additional features to an existing annotation. The annotations subsequently referenced by PLIET are the following:

* Tokens – A Token comprises a single word with a list of features indicating its text, morphological value, part-of-speech-tag, and relation to other tokens in the sentence, the last two features being output by the SDParser.
* Person and Location annotations: Phrases (one or more words) identified as a person or place are also added to this annotation stream by the PL Person-Location Recognizer module. Other features added to it by the Phil-Gooch Pronoun Annotator are the pronominal relationship between two Person annotations indicating noun, representing a person, and a third person singular pronoun.
* Health status and Personal attribute annotations: These annotations are generated by ANNIE lookup of PL-specific gazetteers, which are created and submitted by PLIET to ANNIE as special resources. This is used in recognizing keywords such as *name, age, location*, health conditions, familial relations, as well as for text matching operations for fragmented sentences.

Figure (a) in Appendix B provides an example of the set of Tokens generated by the PL’s GATE/ANNIE pipeline for a simple sentence, and Figure (b) provides a visual representation of the dependency between Tokens, as determined by the SDParser, in another sentence.

## Annotation Processing

Basic annotations, generated by ANNIE, are used to perform the following sequential functions:

1. **Extraction:** This function consists of extracting both phrasal and clausal structure of a sentence, as well as anaphoric relations and coreference among the Tokens in all sentences of the messages – using the ANNIE annotation set. Anaphora involves resolving both third person and first person singular words. (Note that a Token corresponds to a single word, and the Tokens in a relationship are the *head* or *governing* Token of the corresponding phrases.) The outputs of this step are the clausal hierarchy of each sentence, and coreference tables connecting Tokens, and directly specified personal attribute (name, age…)
2. **Analysis:** The analysis of the extracted information in step (a) is performed using a set of Analyzer modules, Its main purpose is to analyze the clauses to establish the subject/predicate/object triples (called *assertions*), establish coreference between noun phrases, and then group all information for a single subject, after anaphora/coreference resolution, using the corresponding assertion set.
3. **Inference Generation:** This refers to the task of generating inferences from the results of steps (a) and (b) above to determine unique person(s) in the email *which are linked to a status or health condition* term, and to further classify them as either as a reported person or a reporter. It makes use of the PL Lexicon to determine the meaning of identified verbs and adjectives, based upon a set of rules including the verb’s voice, objects, surrounding terms, and negative/conditional modifiers. In case of multiple status conditions detected for a person, the highest priority one is chosen. However, if more than one person is considered to be a reported person, they all are stored as a *candidate set* for ranking.
4. **Ranking:** Although the current version of PLIET is designed to accept emails about a single person (especially due to lack of tools to resolve plural anaphora), it is possible to have multiple persons reported in an email. Furthermore, PLIET might misidentify the role of other persons mentioned in a message as reported persons. Thus, an ad hoc ranking algorithm is used to assign a confidence factor to each inferred person from step (c).
5. **Augmentation:** Augmentation involves the process of checking for additional information from the email text that could not be fully or properly processed by NLP clausal analysis, and augmenting the analyzed results with such information. This includes the analysis of sentence fragments (that is: sentences with no verbs), and grammatically incorrect and poorly constructed sentences. This task is based upon the observation that certain entities may be recognized in email text through keyword lookups and regular expression matching, and although less reliable, may still be used in the absence of proper NLP results. Augmentation is applied to the highest ranking candidate of step (d) to generate the Reported Person record. In the absence of any such inferred person, a new record is created using the recognized entities found in this step. The final record is then sent to the PL Website.

The operations for each of the above five steps are summarized in Table 1. They are explained in further detail in Section 5.

Table 1 –Text processing steps and related operations in PLIET

|  |  |  |
| --- | --- | --- |
| ***Step*** | ***Function*** | ***Operations*** |
| **Information Extraction** | Building of Clause Trees | 1. Decompose Sentences and determine the clausal hierarchy |
|  |  | 1. Store anaphoric information |
|  |  | 1. Disambiguate Person vs. Location |
| **Analysis** | Generation of Unique | 1. Resolve coreference |
|  | “Subjects” | 1. Retrieve a Person’s explicit attributes (name, age…) |
|  |  | 1. Assemble Assertions for each unique Subject from all clauses |
|  |  | 1. Determine reporting category from each clausal assertion using PL health status lexicon |
| **Inference Generation** | Identification of one or more Reported | 1. Identify unique Persons from all unique Subjects and Person objects |
|  | Person(s) | 1. Group all clausal assertions for each unique Person |
|  |  | 1. Disambiguate between a *candidate* reported person (CRP) and a reporter person |
|  |  | 1. Determine and select the appropriate health condition for each CRP |
| **Ranking** | Selecting the most likely Reported | 1. Rank each field of a CRP based upon its features using pre- defined rules |
|  | Person | 1. Rank the CRP based upon weight of each field |
|  |  | 1. Get Person, Location, Health condition and Personal Attribute annotations from input stream. |
| **Augmentation** | Addition of missing or new PL | 1. Get associated values in the message through keyword matching and text search. |
|  | specific data | 1. Add missing values in the highest ranking candidate person (CRP) from this secondary source. |

# Processing Details

This section provides further details on the important processing steps mentioned in sections 4.3. Note that in the follow up discussions, object names are used interchangeably with or without embedded blanks, as well as in upper or lower cases, to improve readability.

## Sentence Decomposition and Clausal Hierarchy Determination

The main function performed in this stage is the detection of the grammatical structure of each sentence in the message text. PLIET extracts the structural composition of a sentence at two levels: (a) Phrase level and (b) Clause Level, starting with the lexical (part-of-speech based) information generated by the SDParser, and using the reverse grammatical hierarchy, namely: *Word → Phrase → Clause → Sentence*

For this analysis, a Phrase is defined as a group of words, or a single word, that function as a single unit in the syntax of a sentence. The clausal decomposition of a sentence, starting with GATE generated Token annotations is shown in Figure 4.

The main data structures and functional modules associated with this decomposition are as follows:

* **TextAnnotation** – This is an encapsulation of a Token annotation with additional attributes and methods to help tracing dependencies and build higher level data structures such as TextAnchors.
* **TextAnchor** – Class TextAnchor (henceforth simply referred to as Anchor) represents a set of one or more contiguous Text Annotations in a sentence performing as a single unit, and is linked to others such entities through designated relationships. Grammatically, it is a wrapper around a Phrase, and it stores important information including the IDs of governor (header) and leading tokens of the phrase. Following English grammar, three types of Anchors are generated, namely: Noun, Verb and Adjective, each type with its own additional attributes. Pronoun and Adverbial phrases are incorporated into Noun and Adjective anchors respectively.
* **AnchorLink** - Each Anchor saves its connections with other Anchors in the sentence using a data structure called AnchorLink. An AnchorLink maintains bi-directional information regarding the source and target Anchors, and the link type/subtype. (For example: A parsed collapsed dependency of *preposition\_for* corresponds to the link type *preposition* and subtype *for.*) Note that l*eaf* Anchors do not contain forward links to other Anchors.
* **Anchor Generator** – A module that generates the three types of Anchors, along with their connection to other Anchors as AnchorLinks, by tracing the dependency features of each Token in the sentence as determined by the SDParser.
* **Clause** – A Clause object matches its grammatical interpretation, containing a subject and a verb and their various types of connected data, with the subject usually being represented by a Noun Anchor. The verb is a Verb Anchor, and the related data correspond to other Anchors linked to the Verb or the Subject as a various types of Objects. A Clause may also have child Clauses, linked to the parent Clause through its Verb Anchor by specific *clausal markers*, thus creating a *clausal hierarchy* - the top level clause being the primary clause. Note that a sentence may contain more than one primary clauses connected by conjunctions or punctuations



Figure 4 – Decomposition of a Sentence into Clauses using Token dependency

* **Clause Tree Builder** - The clausal hierarchy (referred to as the Clause Tree by PLIET) is generated by the module Clause Tree Builder, starting with the Verb Anchors and traversing its relations to other Anchors that are designated as its subject or various types of objects. Each type of Anchor has its own set of objects, including prepositional chains, which are included in the object lists of the corresponding Clause.
* **Fragment Sentence** – This structure incorporates a sentence which does not have a verb and therefore cannot be expressed as a Clause. In addition, the corresponding Parser Tree, generated by a standard parser based upon formal text, might not be reliable to determine phrasal relations. Therefore Fragment sentences are processed by the PL TextMatcher module to extract named entities through annotation look-ups and regular expressions.

Further information on the *implementation* of Anchors, Clauses and Clause Trees, using the dependency features encoded within each Token, are provided in Section 6.1.

## Clausal Analysis and Determination of Unique Subjects

The main function performed through Clausal analysis is the identification of the *unique* Subjects in the extracted Clauses without regard to their type and role, and gathering all information, known as Clausal Assertions, related to each unique Subject. This overall function is partitioned into three primary sub-functions as shown in Figure 5. These sub-functions and corresponding data structures are mentioned below. The primary modules performing these functions are discussed in Section 6.2.

* **Coreference Resolution**: Resolves coreference between two Noun Anchors based upon a set of rules involving *partial and equivalent* *name matching*, anaphoric relationship, and other copular relationships. It also handles appositional, *chained and indirect* coreference*.*
* **Coreference Table:** Each entry in this table corresponds to an object (known as the *mention*, usually a person) that is coreferenced in any form in the message, and the list of all objects that coreference it. This coreference table is essential in identifying the unique Person in a messages and combining all information from different sentences for that Person.
* **Clausal Analysis and Assertion Generation:** Analysis of extracted Clauses is performed by the **Clause Analyzer** module,which analyzes each member in the Clausal hierarchy of a sentence, starting with the *primary Clause* and descending downwards. It determines the Subject, Verb and Objects within each Clause and builds the corresponding ClausalAssertion object.
* **ClausalAssertion:** A ClausalAssertion is a structure based on a Clause, with a Subject and a Predicate (Verb and Objects). Objects (Noun Anchors) include all direct and chained Objects, and are explicitly matched against the Person and Location annotation lists extracted earlier. Note that the same *mention* may be the Subject in one Clause and an Object in another Clause.
* **Unique Subject Identification:** This is a crucial step in grouping all information related to a single entity, used as the Subject in one or more Clauses in the entire message, which is further analyzed later to identify a Reported Person.
* **SubjectEntity:** This object represents a unique entity which is used as the *Subject of one or more Clauses* in a message, either directly or through coreference. This may or may not represent a Person. (For example: *Information* sought for John Smith). Each SubjectEntity contains a Noun Anchor (the *mention* in the Coreference chain, if any) identifying the Subject, and the list of all Predicates from the corresponding set of ClausalAssertions. It is generated by the **Subject Analyzer**, from the group of ClausalAssertions, using the Coreference table. All Clauses that don’t have a Subject are grouped under a special SubjectEntity, called the *NullSubject*, with a null as its Subject.

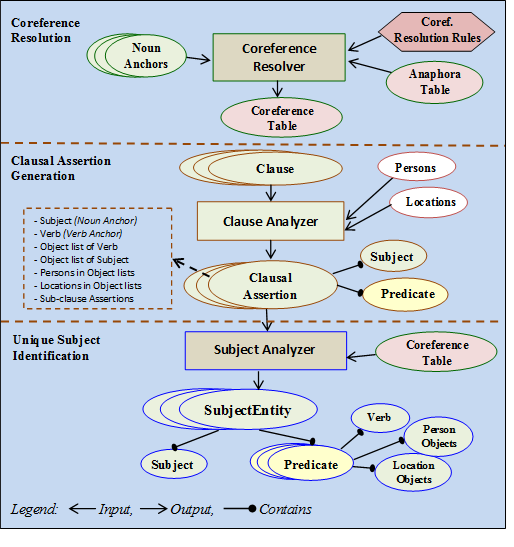


Figure 5 - Generation of SubjectEntity(unique Subjects) with related Predicates.

## Interpretation of Predicates and Identification of *Reported Person*

The most important aspect of processing an email in PLIET is the interpretation of the contents of the Predicates associated with each unique SubjectEntity to determine what information is pertinent, and to draw reasonable inference from this applicable data about a missing or found person and his/her health condition. This function is achieved by using the PLIET developed lexicon defining the vocabulary related to a disaster. The lexicon is referenced at this stage to collect information related to each unique Person, referred either as the Subject or an Object in the SubjectEntity set, and to evaluate all health status information from the corresponding Predicates to make the final assessment. This lexicon and other important components participating in these operations are described below.

* **PL Lexicon:** This is a file-based structure with a set of formatted entries, each entry specifying either a single term (verb and/or adjective) or a group of terms, which defines if the term directly indicates a person’s health condition, indirectly relates to it through an enquiry, or an ambiguous verb indicating either of these cases. For the last set, one or more criteria including *voice, tense, associated object type* and *text of the object* are specified for performing disambiguation. The health status related to the negative of the term (e.g*. not found* denotes *missing*) is also specified. Examples of PL Lexicon entries are provided in Section 6.3.1. This lexicon is read by the **PLLexiconReader** module at system initialization time.
* **LPFEntity:** This is the base class containing all named entities (Noun Anchors), which occur either as the Subject of a Clause or an Object representing a Person in a Clause. It is created from the SubjectEntity set, by including all the members of the set, as well as the Person Objects in each member’s Predicate. (Note that this includes all Persons Objects from the NullSubject structure mentioned earlier.) Each LPFEntity maintains a distinction to indicate if the represented entity was created from a Subject, an Object or both.
* **Entity Classifier:** This module classifies each LPFEntity entry to one of four possible types:

1. Person: In the Person list of the email
2. PersonalAttribute: Words *Name, Age* etc. which may be used as the subject of a clause.
3. DomainTerm: Words that may be used as the subject for an enquiry/statement related to a person. (*Information* sought for…, *Nothing* is known about…). These terms are explicitly defined in the PLIET vocabulary.
4. Unknown: Does not match any of the above. (e.g. The *river* was known to be rising near Badrinath. The *sky* was blue before the tsunami in Thailand)

LPFEntity objects of *Unknown* class are eliminated from further processing since the information is regarded as extraneous. A *PersonalAttribute* object is removed from the list and attached to the Person whose attributes it indicates. Hence only the LPFEntity objects in the final set, classified as *Person* or *DomainTerm*, need to be examined to get the handle over all Persons and determine the Reported Person(s).

* **Verb Interpreter:** The Verb in each Predicate may be one of three categories: (a) health status verb (b) reporting verb (c) unknown verb (i.e. not found in the lexicon). The Verb Interpreter analyzes the Verb, its properties and context in the Predicates of an LPFEntity using the PL Lexicon, and assigns it the proper category after resolving possible ambiguities following a set of rules. It also determines if the verb has a negative or conditional modifier, and the final health status represented by the Predicate.
* **InferredPerson:** An InferredPerson is the final result indicating a possible Reported Person as determined through structural analysis of a disaster email determined through NLP as discussed so far. It is created by examining each (unique) entry in the LPFEntity set, and selecting the one where one or more Predicates contain a *health status* verb. The PL-specific fields such as gender, age, location are filled out from the corresponding Personal Attribute object and the Location Objects in the Predicate. When multiple health status verbs are present, a simple weighting algorithm is used to select the final status. If multiple InferredPerson objects are found at this stage, they are all considered as *candidate persons*, with the final one to be chosen though a ranking algorithm.
* **Inference Engine:** This is the controlling classwhich oversees the transformation from SubjectEntity to InferredPerson by invoking lower level modules.

### Ranking

A confidence factor is assigned to each InferredPerson by assigning a relative weight to each field of the object, checking for the presence of certain fields, matching the value of a field against a pre-defined feature set, and statistical data related to the message. These are:

* Fields: Name, health status, gender, age, location
* Name Features: (First name, last name, isInDictionary, isaRelative…)
* Statistical: Number of predicates for the person / Total number of predicates in the message



*Figure 6 – Generation of an InferredPerson object from the SubjectEntity Set*

Note that this is useful in elimination of certain nouns and relatives such as *friend, son, neighbor*… (which are legitimately included in the Persons annotation list), in the event there is a more likely person with name as a Proper noun. This would happen, most frequently, due to error in determining the implied antecedent for an anaphor by the Anaphora extractor, but may also happen due to algorithmic errors in PLIET.

The operations involving the determination of an InferredPerson (the candidate person with the highest confidence factor) from the SubjectEntity set are shown in Figure 6. This InferredPerson is then used to create the *Reported Person* record in the follow-up processing.

### Data Transformation

An overview of transformation from ClausalAssertions, consisting of a Subject and a Predicate, to an InferredPerson, discussed in Sections 5.2 and 5.3 above, is shown in Figure 7. The entries in the Coreference Table translate a person’s reference from its various forms to the final name to help grouping all correlated data for each person into a single set.



Figure 7 - Assembling an InferredPerson record from ClausalAssertions (Subject and Predicate)

## Generating the ReportedPerson Record

An InferredPerson, generated above, may or may not contain all the PLIET fields, either due to absence of such information in the message, or in the full sentences processed as above, or due to errors in sentence structure. Thus, preliminary attempts are made to recognize missing data, if any, in the message contents through special annotation look-ups, mentioned below.

### Fragment Processing

FragmentSentences were extracted and stored separately during Clause generation (see Section 5.1) to *complement* the results derived from processing full sentences. To process such text, PLIET provides lookup tables to ANNIE specifying the Annotation types and keywords related to each field type. Depending upon the annotation type the annotation text itself is the field value (as for health status), or is retrieved through matching of text around the keyword using a set of applicable regular expressions (for age, name etc.) and stored in a special data structures. Multiple values of a field may be retrieved through such annotation matching.

### Looking for Special Annotations in the entire Message

Examples showed that it is not always feasible to retrieve correct information about a person from a message by NLP tasks through rigorous grammatical analysis, even though that can be easily recognized by humans in short messages. Two main reasons are: (a) ambiguity in the use of words, especially pronouns, in the text and (b) missing or erroneous placement of prepositions, adjectives etc. that result in wrong dependency output by the Parser. Therefore, the entire message is finally sent through the TextMatcher to pick up all recognized fields, as done for individual sentence fragments.

### Merging all Information

Once all data is retrieved from clausal analysis as well as through annotation matching described above, they are merged by the module **PLReconciler** following a set of simple rules.

First, if no InferredPerson was found through clausal analysis, a new ReportedPerson record was created with empty fields. Otherwise data from the InferredPerson record are copied to it. Next, data from fragment search are copied to the Reported Person’s missing field – except for the Location field, which is treated differently. The same procedure is repeated next for data fron full text match operation, excluding the location field. For multiple hits in text search data, either the first one or the highest priority one (for health status field) is used.

If the name in the InferredPerson record is not a proper noun, but is so in the text search data, the latter is used as the name in the ReportedPerson record.

For the Location field, since the individual values may be linked to each other (such as: near *ChristChurch* in *New Zealand*) each value is analyzed through the Google Geocoder and the final one is generated by merging the individual components, if applicable.

This final ReportedPerson record, after proper formatting, is then sent to the PL Web Server as mentioned in Section 2.2.

# ****Implementation****

This section discusses the actual implementation of the system.

*Note: In the parsing examples in this report the representation “*t1 →d:t2” *implies the Token* t1 *in a sentence has a dependency of type* d *with the token t2. The dependency relations are explained in the Stanford Dependency Parser reference manual [11].*

## Extraction

### ****Anchor****

**The three types of Anchors, namely: Noun, Verb and Adjective anchors are phrasal objects, created by analyzing the POS tag (as defined in the Pen TreeBank Project[13]) and the dependency relationship of each Token with other ones in a sentence, added by the SDParser to the Token’s feature list. The SDParser recognizes 55 such relationships. Note that all dependencies start from the head or governor Token in an Anchor, and that Tokens are not shared between Anchors. The Stanford dependencies used in generating the three types of Anchors are provided in Appendix C.**

#### ****Relationships between an Anchor and its subordinate Object****

**An Anchor may be related to another Anchor as its subject or object. We define four types of objects based upon their relation to the source Anchor.**

* **Direct object: *dobj***
* **Indirect object: *iobj***
* **Prepositional object: *[prep, pobj]*(phrases and clauses respectively)**
* **Copular object: copobj**

The first three types are directly defined the SDParser relationships. Prepositional objects may be linked to other prepositional objects (e.g. for research about NLP at NLM in Bethesda, the prepositional chain is: research → NLP → NLM → Bethesda)

Copular objects are PL-defined objects, generated through the copular relationship between two Anchors, for convenience in Clausal analysis. (For example: “*Mary is my niece*.” where *Mary* is the subject and *is* is the copular verb of *niece*. Here, we mark *niece* as the copular object of *Mary*.)

### ****Clause****

A Clause represents the grammatical clause in terms of the Anchors described above. Central to each Clause is the “single” Verb Anchor to which other Anchors within it are related as *subjects*, *modifiers*, or *objects*. A Verb Anchor is related to other Verb Anchors (belonging to other Clauses) in the sentence through hierarchical parent/child relations (thereby establishing the Clausal hierarchy) or as a sibling defining another clausal branch. The top level Clause is called the *Primary Clause*; a sentence may have more than one Primary Clause, related by conjunctions or other non-word Tokens.

### Clausal Dependency

The relationships through which a subordinate Clause is connected to its parent or master Clause are defined by the SDParser. They are of two categories: (a) direct relations and (b) indirect relations - through an adjective or noun subject/object of the Verb of the main clause. They are presented in Tables 9a and 9b respectively in Appendix C.

In a Clause, related conjunction (*conj*) and complementary (*xcomp*) clauses are maintained as a separate list and treated as equivalent, and other types are regarded as lower level ones. A conjunction Clause, which has its own subject, is regarded as an independent clause, and has the same level as the governor clause.

#### Subject of a Clause

The subject of a Clause is not always an internal component of the sentential clause, and is established through the following relationships in the feature list of the head token of the Verb Anchor (where *x/xpass* refer to *active* and *passive voices* respectively).

* nsubj/nsubjpass – direct subject
* xsubj/xsubjpass - subject of the parent clause for xcomp clauses
* csubj/csubjpass – subject of a “Subject Clause” of the Verb Anchor.

#### Objects of a Clause

A Clause may have several Objects – directly attached to its Verb or Subject, or indirectly as nested Objects to the higher level ones. (The Object types are discussed in Section 6.1.1.) The Object Anchors corresponding to the Verb and the Subject are maintained in two separate lists, within the Clause structure for semantic analysis.

#### Genre of and Purpose of a Clause

For semantic interpretation of a sentence and drawing inferences, the genre and purpose of each clause with respect to its parent are established, and stored as a Clause’a attributes. In addition, for prepositional clauses, the connecting marker (prep: *in, out, near, after, before…)* are useful for generating temporal and other information.

#### Dealing with Clausal Hierarchy

The clausal hierarchy is used to determine implicit Subject or Object of a Clause, as follows:

* A lower level Clause with no explicit Subject inherits the Subject of a parent Clause.
* Objects of lower level Clauses are included in the Object list of the parent.

## Analysis

### Message Analyzer

This is the top level class which analyzes the extracted clauses in the message to generate various data structures useful in building unique Subjects in a message. It uses a set of lower level Analyzer and Resolver modules, shown in Figure 8, to perform such function. The function of the *Clause Analyzer* and *Subject Analyzer* are mentioned in Section 5.2; rest are described below.

* *Location Analyzer* **- t**races through the list of Objects for each Verb in a Clause and determines which of them, if any, represent Locations - by analyzing the link type (for example: *preposition\_in* links). Examines chained links too, and performs the same for the *Subject* of a Clause.
* *Person Analyzer:*Similar to the Location Analyzer but determines the Objects in a Clause that represent a Person.
* *Person Resolver* - resolves coreference and other ambiguities and find the best match Noun Anchor representing a Person from a list, corresponding to a given name.

*Figure 8 – Analyzers and Resolvers used for Named Entity Recognition within PLIET*

**Person Resolver**

**Anaphora Resolver**

**Coreference Resolver**

**Name Resolver**

**Location**

**Analyzer**

**Person Analyzer**

**Clause Analyzer**

**Attribute Analyzer**

**Subject Analyzer**

***Analyzers***

***Resolvers***

The functions of the other modules and the techniques used are discussed in subsections 6.2.1 through 6.2.4.

### Attribute Analysis

This function, performed by the Attribute Analyzer, results in the generation of a map of all PersonalAttribute annotations in the message and the corresponding TextAnchor with which each one is associated. Personal Attributes represent the named properties or location/status of a person, expressed explicitly by known terms or keywords, as presented in Table 2.

### Name Resolution

This function is used to establish the equivalency between the texts of two Noun Anchors representing a person’s names through a set of rules, and is used for generating coreference and determining the unique *SubjectEntity* set in a message. It takes into account the first name, last name, middle name, title, and other common extensions to the name (Sr., Jr., etc.), and supports exact, equivalent, partial and longest match options*.*

Table 2: Personal Attribute annotation list

|  |  |  |
| --- | --- | --- |
| **Attribute Type** | **Annotation Keywords** | **Example** |
| Name | name, named | His *name* is John. The boy is *named* Alan. |
| Age | age, aged, year(s) old | His *age* is 15. Laura is *aged* 19. John is 15 *years old*. |
| Gender | male, female, man, woman, boy, girl | Searching for a ten year old *boy*. A 50 year old *female* named Rigene Aquino is missing. |
| Location | address, location, contact | His last known *location* was Bethesda, MD.  His *contact* place is unknown |
| Health status | condition, health, status, situation, prognosis | His *situation* is better. John’s *prognosis* is bad |

### Anaphora Resolution

Anaphora resolution is the problem of resolving references of pronouns to earlier or later *noun phrases* (anaphora/cataphora respectively) in the discourse*.* The Anaphora Resolver retrieves the third person singular anaphora relations from the output of the Phil Gooch Annotator, which are then matched against the Noun Anchors to build the *Anaphora resolution table*. In addition, it performs two other types of anaphoric matches (for singular person only) that are essential for PLIET:

1. Establishes antecedence for Persons of unknown gender, such as the entries in the Person annotation list that were found through dependency analysis (Section 6.3). (Note that the three Pronoun Annotators we tested, including the Phil Gooch one, can only match pronouns against Persons of known gender.) To perform this function, the Anaphora resolver checks each Proper Nouns in the Person List against their usage in the sentence (as subject, object or both), matches a set of properties between these nouns and the pronoun term, and finally selects the last Noun preceding the antecedent with such property match. This is illustrated in the following example:

We are still looking for Amnuay Chantarat. He was last seen wearing a blue jacket.

(*He →*antecedent: *Amnuay Chantarat*)

Here *Amnuay Chantarat* was resolved as the antecedent for *He*, even though the gender of the former was not explicitly known.

1. Resolves anaphora-antecedent relations for first person singular pronouns. This is performed in the two passes, through the detection of:
2. Intra-sentential relations - correlation between words in the same sentence
3. Inter-sentential relations - resolution of reference to the same person from different sentences

Table 3 provides examples of PLIET first person anaphora resolution in sample messages.

Note that anaphora resolution, based upon implied reference rather than grammatical rules are not yet available. See Section 8.2 for further discussion.

Table 3: First Person Anaphora resolution algorithm

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Technique Used** | **Example sentence** | **Coreference** |
| Intra-sentence | 1. **Person annotation** through Copular relation | **I am John Smith**.  Smith -> *copula*: am.  am -> *subject*: I | *I => John Smith* |
|  | 1. **Possessive pronouns** through “name” Attribute annotation | **My name is John Smith**. Smith -> *copula*: is.  is -> *subject*: My name. name -> possessive: My (root: I) | *I => John Smith* |
| Inter-sentence | Coreference to the same Annotation through root word | **My name is John Smith. I am fine.** | *(I, My) => John Smith* |

### Coreference Resolution

Coreference resolution is the process of determining whether two expressions in a discourse refer to the same entity [4]. In PLIET, we resolve explicit (direct or indirect) coreference between singular nouns in a sentence, rather than implied coreference based on context, such as discussed in [14].

The Coreference Resolver uses several techniques to establish coreference relation between Persons represented in noun phrases (i.e. Noun Anchors) and to build the coreference table connecting the *mention* with one or more *co-referrers*. These techniques are mentioned below, with follow-up examples

1. Use of Anaphoric relations: from the output generated by the Anaphora Resolver
2. Use of *Name* Attribute: from the output of the Attribute analyzer with explicit “name” keyword
3. Name Matching: Using the Name Resolver for *partial, equivalent*, and *exact* name matching
4. Use of appositional relation: Two phrases expressing the same person in appositional phrases
5. Use of *dep* relationship: This dependency relation between two words is generated by the SDParser and stored as a Link between the corresponding Anchors at Anchor generation time
6. Use of copular relations. A copular object of a name may also be its referrer.
7. Resolution of indirect coreference: where the referrer in one case becomes either referrer or the referent in another.

***Example:*** Table 4 shows how coreference is resolved, using some of the techniques mentioned above, for the following messages.

1. *John found a man lying on the road. He said his name was David Miller. David was injured.*
2. *I am looking for my niece Jane. She is missing after the earthquake.*
3. *I am looking for my son John Smith. My son is 10 years old.*
4. *I am searching for my friend. She was in Haiti. Her name is Jane McCall.*
5. *Amy is my sister. She is 10 years old*.

Note that coreference in messages (b) and(c) although similar in nature, are resolved differently die to difference parser-generated dependencies between the phrases.

### Gender Determination

A person’s gender is determined in one of several ways:

* From the keyword attribute (*boy, girl*…shown in Table 2)
* Pronouns (*he, she*) in the Anaphora table
* Applicable relationships (son, daughter, mother, nephew…)
* Gender feature in the Person list, output by ANNIE through gazetteer look up

Table 4 – Coreference resolution techniques used for the sample messages

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Message | Technique | Text segment | Relationship | Coreference |
| (a) | Anaphora Resolution | John found a man lying on the road. He said his name was David Miller*.* | *(He, his) => man* | Mention:  *David Miller* |
|  | Name Attribute | …his *name* was David Miller. | *he (*root of *his) => David Miller* | Coref:  *He, his, man, David* |
|  | Name Resolution | *David was injured.* | *David => David Miller* |  |
| (b) | “appos” relation from Parser | *…for my niece Jane…* | *niece -> appos:Jane* | Mention: *Jane Miller*  Coref*: my niece, she* |
| (c) | “dep” relation from Parser | *…for my son John Smith...* | *(Smith ->*dep*: my son)* | Mention: *John Smith*  Coref*: my son, he* |
| (d) | “copular” relationship | *Amy is my sister* | *(Amy-> copula: my sister)* | Mention: *Amy*  Coref: *my sister, She* |
|  |  | *She is 10 years old*. | *She =>my sister* |  |
| (e) | Indirect coreference | *I am searching for my friend. She was in Haiti.* | *She =>my friend* | Mention:  *Jane McCall* |
|  |  | *Her name is Jane McCall.* | *She (root of her) =>Jane McCall* | Coref:  *my friend, she* |

## Inference

The most important inference drawn by the PLIET is determining the person who is being reported as lost or found and determining the health status of the person, both of which are resolved, if possible, by the use of rules in the PL Lexicon.

### Structure and Contents of the PL Lexicon

As mentioned earlier, the PL Lexicon is used to interpret and/or disambiguate the meaning of certain terms (verbs, adjectives, adverbs) in a *clause* based upon their general meaning, associated determiners, words implying negation, linked words such as direct/indirect objects, and then map it to one of the six PL defined health conditions. (To get the overall meaning the genre of the clause in the clausal hierarchy has also to be considered.) The six types of health conditions are ALIVE, DECEASED, FOUND, INJURED, MISSING, and UNKNOWN.

There are two types of entries in the PL Lexicon: (a) Standard type: Entries defining the unambiguous disaster-related terms directly indicating a person’s health status; (b) Ambiguous type: Entries indicating ambiguous verbs, along with rules to disambiguate them, to yield a person’s health status.

There are seven fields in each standard entry. These fields are:

1. Word
2. Part of speech tag (Verb/Adjective)
3. Morphological value (default: same as the word)
4. Alternate forms of the word (different spelling, contraction…)
5. Information Category (Health Status/Reporting)
6. Implied health status or condition
7. Health status implied by negation of the term (e.g. seen => found, *not* seen => missing)

Note that the adjective terms include words such as *alive, well, stable, lost*… used as copular terms for a main verb; and henceforth referred to as *verb* in this discussion. Information category specifies whether the verb is used to indicate the actual health status of a person (as the subject: *John is missing*) or a term used to make a statement about the condition of a person (as the object: *I am searching for John*), which is crucial in determining if the corresponding person is a reported person or a reporter.

There are six fields in an ambiguous entry, which are:

1. Word or word group
2. Resolution criteria
3. Context Specification
4. Information Category (Health Status/Reporting)
5. Implied health status or condition
6. Condition implied by negative of the term (e.g. *not* found => missing)

The first field of the entry usually specifies a set of words which can be resolved by applying the same rule to produce the same status

Resolution criteria imply the types of rule to be applied to resolve the ambiguity, presented in Table 5. Tables 6 and 7 present sample entries from the PL Lexicon for standard and ambiguous cases respectively.

Table 5: Ambiguity resolution techniques as specified in the PL Lexicon

|  |  |  |
| --- | --- | --- |
|  | **Ambiguity Resolution Criteria** | **Options** |
| 1 | Voice of the Verb | Active/Passive |
| 2 | Tense of the Verb | Present/Past |
| 3 | Object of the Verb | Specific words |
| 4 | Type of Object | Person/non-Person |
| 5 | Complement verb connected with word “to” *(POS Tag: TO)* | Specific Verbs |
| 6 | Default resolution | None of the above |

Table 6: Examples of PL Lexicon Standard entries

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Word** | **POS Tag** | **Root** | **Other forms** | **Information Category** | **Health Condition** | **Negative Condition** |
| okay | Adjective | okay | ok|o.k. | HEALTH\_STATUS | ALIVE | INJURED |
| unconscious | Adjective |  |  | HEALTH\_STATUS | INJURED | UNLNOWN |
| released | Verb | release |  | HEALTH\_STATUS | ALIVE | INJURED |
| searching | Verb | search |  | REPORTING | MISSING | UNKNOWN |
| worried | Verb | worry |  | REPORTING | UNKNOWN | UNKNOWN |

Table 7: Examples of PL Lexicon Ambiguous entries

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Word set** | **Resolution Criteria** | **Resolving option/Entity** | **Information Category** | **Health Condition** | **Negative Condition** |
| want|need|  seek|wait | OBJECT | news|status|  information|conditio | REPORTING | UNKNOWN | UNKNOWN |
|  |  | help|assistance|  rescue|treatment… | HEALTH\_STATUS | INJURED | UNKNOWN |
| find|locate | VOICE | Active | REPORTING | FOUND | MISSING |
|  |  | Passive | HEALTH\_STATUS | FOUND | MISSING |
| want|try|  unable | To\_Compl. | know|find|contact|  locate|check | REPORTING | MISSING | UNKNOWN |
| concerned|worried|distreesed | OBJECT\_TYPE | Person | REPORTING | UNKNOWN | ALIVE |
|  |  | Non\_Person | HEALTH\_STATUS | INJURED | ALIVE |

Note that the PL Lexicon only makes the best possible guess about the health status of a person from the utterance in the email, and often it may not be easy to resolve ambiguities through definitive rules described above.

## Use Cases

In this section, we present several examples of processing an email from end-to-end within the PLIET NLP pipeline. They show the final results, leading to the creation of the Reported Person record, performed by the PLIET software as described in Sections 4 through 6.3.Note that these examples pertains to the cases where we get correct results. The deficiencies of PLIET text processing are mentioned in Section 8.

**Message 1**

<subject/>

<body>We are unable to find Amnuay Chantarat after the flood in Surat Thani. He was last seen wearing a blue jacket.</body>

**Reported Person:**

First name: Amnuay Last name: Chantarat

Gender: male Age: unknown

City: Surat Thani Region: Surat Thani Country: Thailand

Latitude: 9.1382389 Longitude: 99.3217483

Reported Status: Missing

*NLP Confidence ranking: 65/100*

This email highlights the retrieval of: (a) names and location in non-English speaking region (b) recognition of status as *missing* from the phrase “unable to find.” Also shows the recognition of “Amnuay Chantarat” (with original gender = *unknown*) as the antecedent for *H*e by the PLIET Anaphora Resolver, and thus setting of the gender to *male.*

**Message 2**

<subject>David Miller </subject>

<body> David has used a friend's cell phone to call us to tell he's safe, but their house is in poor condition.</body>

**Reported Person:**

First name: David Last name: Miller

Gender: male Age: unknown

Reported Status: Missing

*NLP Confidence ranking: 31/100*

In this message, the person originally identified as safe (through anaphora) was *a friend*. Since it is not a Proper noun, the (only) name David Miller was used instead. Extraneous information related to the house was ignored. However this substitution (and no *Location*) resulted in a low confidence ranking for the record.

**Message 3**

<body>We cannot find our niece Anita Satpathy following the cyclone in Odisha. She is 20 years old, has brown eyes and black hair. She was last seen in Gopalpur market by a friend before 6 pm on the day of the cyclone. Any information about her will be very much appreciated. </body>

**Reported Person:**

First name: Anita Last name: Satpathy

Gender: female Age: 20

City: Gopalpur Region: Odisha Country: India

Latitude: 19.2582880 Longitude: 84.9057430

Reported Status: Missing

*NLP Confidence ranking: 69/100*

This message, more complex in nature, shows the extraction of gender, age and location through proper coreference resolution. (Note that changing the voice or occurrence of certain words in complex sentences may change the outcome.)

**Message 4**

<subject>Injured man found </subject>

<body>name: Michael Cane, Age: 55 City: Rockville, MD </body>

**Reported Person:**

First name: Michael Last name: Cane

Gender: male Age: 55

City:Rockville Region:Maryland Country: United States

Latitude: 39.0839973 Longitude: -77.1527578

Reported Status: Injured

*NLP Confidence ranking: Unknown*

This message highlights the usage of annotation matching and text search with regular expressions to find information in sentence fragments and the full message. All the fields above were found only by text matching through PL-specific annotations *name, age, city* and the health status keywords *injured, found.* Note that no ranking is provided here as nothing was inferred from PLIET’s standard NLP processing.

# Testing and Accuracy

## Text Corpora

Due to the lack of any real disaster related email corpus, attempts were made to build test sets in two different ways:

(a) Internally, through request to CEB staff members as well to the Sahana community. These two sets, consisting of about 140 messages, were too general in nature and did not represent messages that would be sent from a disaster affected area, especially from a non-English speaking region.

(b) From actual PIPF *notes* sent to Google PersonFinder site for the New Zealand earthquake and Philippine flood of 2012, stored in the PL’s database. These messages, being additional notes supplementing fielded data, are not always very specific and lacked Subject headers (which often contain a useful summary of what is discussed in the body of an email), but still are very useful go have a general idea about the content and genre of such messages.

A corpus of messages were selected from these two sets and manually modified to create (what we believe to be) a good representation of disaster emails, and then manually annotated to provide ground truth to test the accuracy of PLIET results.

A number of test sets were also developed to study the dependency relations between words as generated by the Stanford Parser, and to test specific aspects of PL algorithms during implementation. Those results were validated through manual inspection and were used for improving the NLP algorithms implemented in PLIET. They are not discussed here.

## Accuracy

The accuracy of the system was measured by the PLIET-created *Accuracy Evaluator* tool, accepting “partial match” of Name and Location field values, with the annotated data in the ground truth of two test sets. The accuracy of the “age” field was not measured due to insufficient occurrences in the test sets. The set marked “General*”* included messages selected at random, whereas “Selected” contained simpler messages that PLIET should be able to handle. A subset of the messages in the “Selected set: are shown in Appendix D.

Table 8: Accuracy score of sample test sets

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Type** | **# of Cases** | **Accuracy** | **Person Name (F/I)** | **Health Status (F/I)** | **Gender (F/I)** | **Location (F/I)\*\*** |
| General | 50 | precision | 0.62/0.60 | 0.59/0.59 | 0.96/0.77 | 0.27/0.17 |
|  |  | recall | 0.60/0.55 | 0.49/0.36 | 0.96/0.83 | 0.23/0.04 |
|  |  | F1\_score | 0.61/0.57 | 0.54/0.45 | 0.96/0.80 | 0.25/0.06 |
| Selected | 25 | precision | 0.91/0.77 | 0.90/0.81 | 0.87/0.73 | 0.61/.50 |
|  |  | recall | 0.91/0.74 | 0.83/0.74 | 0. 91/0.73 | 0.79/.36 |
|  |  | F1\_score | 0.91/0.76 | 0.86/0.77 | 0. 89/0.73 | 0.69/.42 |

Note: F/I => Final and Initial results (after/before merging NLP and Text Search Results)

\*\* => Results are too low as “components of a Location” are not combined here and partial match of text does not work well.

The ground truth set would be expanded, with more varies data, as we refine the PLIET algorithms.

# Issues and Needs for Further Enhancement

There are many factors, apart from the inherent ambiguity in a message, affecting the accuracy of NLP text processing in PLIET. The PLIET system is not matured yet, and some algorithms should be improved in future by conducting more tests and gaining more experience and with a better understanding of the idiosyncrasies of the underlying tools. Still other factors impose problems that are difficult to surmount easily. Both these subjects are discussed in the following sections

## Parsing and Dependency Detection

The Stanford Dependency (SD) Parser, although found to be most suitable for our work, still produces incorrect results in parsing sentences with no verbs, common words that may be used as nouns, verbs and/or adjectives. It also yields different sorts of dependencies between words in a segment depending upon their position or phrase length in the sentence, making the analysis difficult. For example: Consider the following three sentences:

My niece Jane Miller is missing. (Miller → poss: my, Miller → nn: Jane → nn: niece)

I am looking for my niece Jane Miller. (niece → poss: my, niece → dep: Miller → nn: Jane)

I am looking for my niece Jane. (niece → poss: my, niece → appos: Jane)

In case (a) *My niece Jane Miller* is regarded as a single phrase with *Miller* as the governor token, where as in cases (b) and (c) *My niece* and *Jane [Miller]* are regarded as two phrases connected through *dep* and *appos* relationships respectively, where *dep* means an unresolved dependency and *appos* means a modifier phrase. This type of difference makes the coreference detection between *My niece* and *Jane [Miller]* difficult and error-prone.

Similar examples are also found in relationship between verbs of different clauses.

### POS Tagging Errors

Many oriental last names, and certain English first/last names as well, match English nouns, verbs, pronouns etc. are often assigned wrong POS tags, making the parsing of a sentence erroneous. Examples are: Last names: *He, Hunt,* ?, month names: *May, June* - interpreted as female names and vice versa. Phrase:  *till today* - *till* is regarded as a verb, Last names such as Smith or Baker are found to be treated differently in different cases.

### Case Dependency

It is very important that the names of Persons and Locations in a sentence should start in uppercase, following English grammar rules. Although the GATE/ANNIE pipeline may be instructed to ignore the *case* in annotating such entities in Gazetteer lookups, a Parser cannot recognize lowercase words as Proper nouns, and thus cannot parse the sentence correctly.

### Use of Interjection

Stanford Dependency Parser (the GATE Parser as well) cannot correctly handle interjections in a sentence. In messages relate to lost and found persons, we expect the interjection word *please* to occur occasionally (for example: *Please let us know…*). This word is interpreted as a verb, and results in erroneous dependency among tokens, even though V2.xx of the SDParser is supposed to recognize interjections. To accommodate this, we remove the word “please” from a message before it is sent through the GATE annotation generation pipeline.

## Anaphora Resolution

Two of the main problems in resolving anaphors (and hence coreference) are:

1. Lack of interpretation of the overall context of the message. For example: Consider the messages: *John Smith is missing. James Miller last saw him yesterday.* Here the antecedent for *him* is resolved to be *James Miller,* although we can see that it is really John Smith, who is being referred to in the next sentence
2. Problem with “unknown gender” associated with a name: Anaphora resolver tools (Phil Gooch and others) establish the antecedent for a pronoun by matching the gender in a Person annotation. If the Person’s first name is not in a gazetteer (as the case with many non-English names), this connection is missed. PLIET partially corrects for this problem as explained in Section 6.2.3.

## PLIET Deficiencies

### Propagating inferences from a Clause to higher or lower levels

Although PLIET NLP processing algorithms can determine a clausal hierarchy and clause types with good accuracy, it is not yet capable of drawing inference from a lower level clause and passing it on to the higher levels for further interpretation, especially for clauses that work as the subject or object of a verb in another clause. For example: in the message:

We want to know that Julie is fine.

the health status of Julie is interpreted as FINE, rather than as UNKNOWN. More studies and improvements are also needed in dealing with other aspects of Clausal relations.

### Handling of Interrogative sentences

Although PLIET recognizes a sentence to be interrogative or to pose a question (*Does anyone know where Michael is?*), it is not clear how to develop rules to deal with such sentences.

### Determining Locations

The name of a place, or a city or town often consists of more than one word, and the full location is specified not as a single phrase but as a set of words linked by prepositions. In other cases, only the village/city name might be specified without reference to a state or province or country. (For example: Bombing in *Cambridge.)* In the former case, attempt is made by PLIET to determine the full location name, by carefully piecing the individual components together and verifying with the Google Geocoder – but still needs some enhancements. In the latter case, the real location must be chosen by checking with an external reference such as the location where a disaster happened recently, which is not yet implemented.

### Absence of a Subject in a Sentence

Currently, if no subject is associated with a sentence, PLIET puts the predicate under a special structure called the NullSubject, and later checks the predicate only for Person type objects. If the objects contain other information, with the subject or an object in the previous sentence as the implied subject, that information gets lost. This is illustrated in the following example:

Mary Smith is looking for Jane Miller. Twenty years old. Last seen in Christchurch.

Here *Jane Miller*’s age and location are not picked up in building the InferredPerson object, although they are later recognized by the TextMatcher and integrated with the former to generate the ReportedPerson record. Nevertheless, Text matching is less reliable, and PLIET should develop algorithms to assign the proper subject (that is: *Jane Miller*, not *Mary Smith*) for such sentences.

### Ranking

The ranking algorithm presently used to determine the reliability of the contents of an InferredPerson and to choose the highest ranking one as the ReportedPerson is ad hoc in nature, and should be improved in future releases. Sentence distance and other factors should be considered in selecting and/or merging the results of the TextMatcher with that of the NLP processor.

### Determining the Grammatical Accuracy of a Sentence

A sentence with incorrect grammatical structure would result in incorrect parsing and erroneous output. For example the incorrect sentence (a) (from a Google Note) and its grammatically correct version (b) yield very different parsed outputs.

1. Mrs. Sarla devi kain has spoken to me last 12th june 2013 after that till date no news about her.
2. Mrs. Sarla devi kain has spoken to me last 12th June 2013. After that till to date we have no news about her.

However, PLIET has not developed any classification scheme to determine if a sentence is grammatically sound and treat it accordingly.

### Preliminary Nature of the PL Lexicon

At present, the PL Lexicon is built as a prototype rather than a real resource to be used in a production environment. A more thorough understanding of disaster vocabulary and examination of field cases would be necessary to incorporate more rules into the lexicon and enhance existing rules and build an operational version.

### Dealing with Multiple Reported Persons

Presently, PLIET deals with only a single reported person. Even if multiple persons are reported individually (through separate clausal hierarchies), such as *Mary is fine, but Jane is injured* in the same email, only the highest ranking one is selected. If they are mentioned using conjunctions and plural nouns/and pronouns (*Mary and Jane are found. They are injured.*), the results become incorrect.

# Conclusion

During the development phase of this system, a large percentage of time was spent in understanding the nuances of the English grammar, NLP, and various tools - by building appropriate test cases and analyzing the results rather than implementing algorithms in areas that are not well understood, or refining algorithms in areas that are better understood, but is time consuming. This was mainly due to the fact that so many aspects of NLP processing are to be addressed in this simple task to have a broad understanding of the problem that we could not explore any one aspect in full depth as warranted by a research topic.

Furthermore, that as new open source NLP tools are being released or existing ones are being upgraded with more accurate results, they may solve some of the underlying problems currently faced by PLIET. For example:

* Coreference Resolver that comes with the newly released Stanford Parser V3 performs better anaphora resolution than the Phil Gooch tool. (However this V3 Parser is not yet pluggable into the GATE/ANNIE system.)
* A newly released tool called ClausIE, released by the Max Planck Institute in May 2013, [15], yields the clauses in a sentence based on the Stanford Parser V2. It may be compared or integrated with the PLIET system for better accuracy.
* PLIET architecture is modular enough such that it can be plugged into other suitable NLP platforms besides GATE with relative ease.

From the discussions above, we believe that PLIET can be developed into a useful tool for interpreting simple disaster related inquiries and reporting about affected persons, with acceptable accuracy. Twitter or other sites, which inform public about the existence of such services, may be tailored to instruct people to send short English messages, one per a reported person.

Nevertheless it should be noted that the most difficult issue in predicting the usefulness of PLIET is the lack of real emails from disaster sites, and especially from areas where English is not the native language. Often, habitually, people also use a lot of contractions in emails (e.g. U => you, R => are, 4 => for etc.), and start proper nouns in lower case, which is not acceptable by parsers trained on formal text corpora and would therefore result in lower accuracy. In any case, there should be manual review of the contents of each Reported Person Record prior to its entry into the ReUnite database.

***Acknowledgement:***

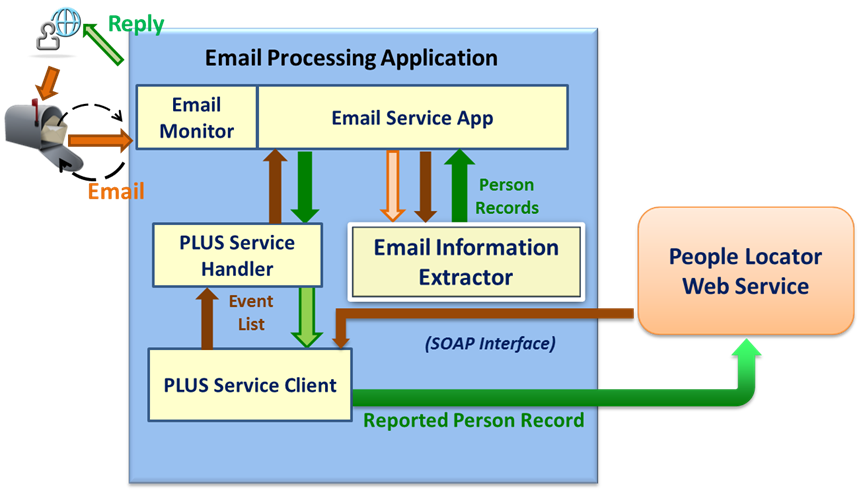
The front part of this work, namely the interface with the email server using Messaging protocols, and the interface with the GATE/ANNIE system was previously implemented by Ajay Kanduru under a different PL task. He had also implemented the module to interface with the Google Geocoder. His software was adapted for use with PLIET.

# References

1. Lost Person Finder and People Locator <http://archive.nlm.nih.gov/proj/lpf.php>
2. The ReUnite: iPhone APP <http://archive.nlm.nih.gov/proj/reunite/faq/index.html>
3. Formal and Informal Writing: <http://word-mart.com/html/formal_and_informal_writing.html>
4. On Coreferring: Coreference in MUC and Related Annotation Schemes: *Kees van Deemter and Rodger Kibbley* <http://acl.ldc.upenn.edu/J/J00/J00-4005.pdf>
5. Message Understanding Conferences [http://www-nlpir.nist.gov/related\_projects/muc/](http://www-nlpir.nist.gov/related_projects/muc/%20)
6. Apache OpenNLP Natural Language Processing Toolkit <http://opennlp.apache.org/index.html>
7. [General Architecture for Text Engineering](gate.ac.uk) and [ANNIE: a Nearly-New Information Extraction System](http://gate.ac.uk/sale/tao/splitch6.html" \l "x9-1270006)
8. Apache Unstructured Information Management applications (UIMA) <http://uima.apache.org/index.html>
9. [Developing Language Processing Components with GATE Version 7 (a User Guide)](https://gate.ac.uk/sale/tao/tao.pdf)
10. [Stanford Dependencies](http://nlp.stanford.edu/software/stanford-dependencies.shtml) and [Stanford CoreNLP](http://nlp.stanford.edu/downloads/dependencies_manual.pdf)
11. Stanford typed dependencies manual <http://nlp.stanford.edu/downloads/dependencies_manual.pdf>
12. [Phil Gooch Pronoun Annotator](http://vega.soi.city.ac.uk/~abdy181/software/#pronoun)
13. The Penn Pen TreeBank Project <http://www.cis.upenn.edu/~treebank/>
14. Information Extraction from Recipes: *Rahul Agarwal and Kevin Miller* <http://nlp.stanford.edu/courses/cs224n/2011/reports/rahul1-kjmiller.pdf>
15. ClausIE: Clause-Based Open Information Extraction: *Luciano Del Corro and Rainer Gemulla* <http://www.mpi-inf.mpg.de/~rgemulla/publications/delcorro13clausie.pdf>

**Appendix A – System Architecture of People Locator Email Processing Task**

**Figure 9 below provides an overview of the PLIET system architecture. The modules shown in the diagram are described below.**



Legend:

→Email Text, → Disaster Event List, → Extracted data for a Reported Person

*Figure 9 – PLIET system architecture*

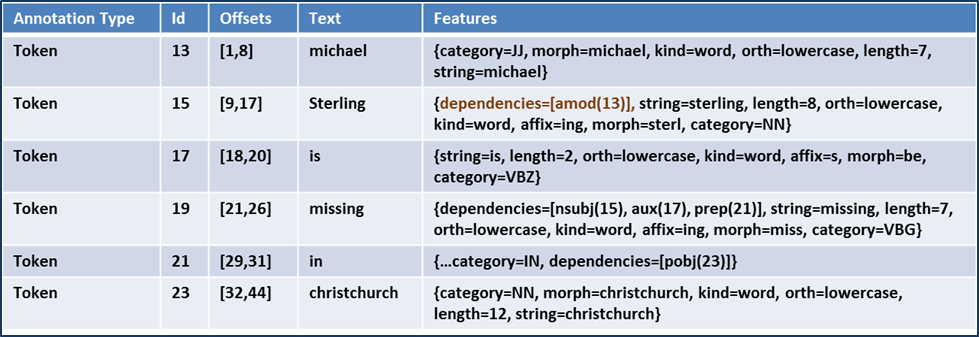
* **Email Service App: This is the top level PLIET module, which starts up the Email Monitor to retrieve emails from the designated PLIET-specific disaster mailbox, and processes each one by sending it to the Email Information Extractor. At system initialization time, it gets the list of all open disaster events supported by PL and also sends that to the Extractor. (Note that this approach forces the system to be restarted when a new PL event is added, and so should be modified later.)**
* **Email Monitor: This module reads the set of new email messages in the mailbox and sends each email to Email Service App (as a callback) for processing. The returned results are then sent as a reply to the email originator. After all the messages in the mailbox are processed it waits for a pre-specified interval and then checks the mailbox again for new messages.**
* **Email Information Extractor:** This module is invoked for each incoming email, along with the PL “disaster event list”. It creates the GATE/ANNIE NLP Pipeline once at startup phase, and sends each email to be processed by the NLP related modules described in Section 3-5. The extracted fields for a *Reported Person* are retrieved after the processing is complete **Plus Service Handler: This is the interface between the Email Service APP and the PL Web Service for exchange of data using the Plus Email Client.**
* **Plus (People Locator User Service) Email Client: This module communicates with the PL Web Service, running on a dedicated server, using SOAP protocols. It obtains the disaster event list from the server and sends the formatted PL Reported Person record to it at system initialization time.**

**Appendix B – Token features and Stanford Dependency relations**

1. **GATE Token annotations with Stanford Parser generated dependency features**

**Sentence: *Michael Sterling is missing in Christchurch.***

***(Note: all Token text is output in lowercase)***

**

**b)** **Visual representation** **of Stanford dependencies between words**

**Sentence: *My family in London is looking for John Smith at Christchurch.***



**Dependency relations shown here are among the 55 different types of relationships recognized by the SDParser [11]. The visual output is generated by the online tool Stanford CoreNLP [10].**

**Appendix C - Dependency relations used for building Anchors and Clauses**

1. **Phrasal Relationships (used in building Anchors)**

**Verb Anchor:**

* **POS Tags: {*VB, VBD, VBG, VBN, VBP, VBZ*}**
* **Governor: Head token or copular word if exists *(John is alive: Gov =>alive)***
* **Conjunction: *cc***
* **Phrasal Array: *aux, auxpass (Mary had found John. Mary was hurt. Mary had been* *hurt.)***
* **Phrasal Subject: *nsubj, nsubjpass***
* **Adverbial Modifier: *acomp* (Mary was found alive in Bethesda.)**
* **Auxiliary Verb*: aux, auxpass, cop (*Ex. John is missing*: missing → cop: is)***
* **Relationships to other Verbs in a sentence: See section ???**

**(Note: In issues: Discuss the difference between: *Mary was alive in Bethesda,* and *Mary was found alive in Bethesda*.)**

**Noun Anchor (includes Pronouns):**

* **POS Tags: {*NN, NNS, NNP, NNPS,PRP, PRP$, WP, WP$, CD}***
* **Head token: Last word of a phrase (TBD: John Smith, Jr.)**
* **Phrasal Array: *nn***
* **Possessive (Pronoun): *poss, possessive***
* **Conjunction chain to another Noun Anchor: *[cc, conj]***
* **Adjectival modifier: *amod***
* **Copular object: *cop* (usually another noun)**
* **Appositional modifier: *appos*  (comma separated relation *Jack, my brother*)**
* **Participal modifier: *partmod* (leading to a *dobj)***
* **Subject of a Verb or a copular Adjective: *nsubj, nsubjpass***

**Adjective Anchor (includes Adverbs):**

* **POS Tags: *{JJ, JJR, JJS, RB, RBR, RBS , WRB}***
* **Phrasal chain: *xcomp, npadvmod* ??**
* **Adverbial modifier: *advmod ( e.g.“Mary is not well.”)***
* **Conjunction: *cc* (introductory word, but no connecting chain) (*e.g. John is alive and well*)**
* **Copular Verb: *cop***
* **It should be noted that in case of Verb Anchors where the verb is the copula of the adjective (e.g. I am fine: (fine → copula: is, fine → nsubj: I), the governor is the adjective rather than the verb itself.)**

**Common Phrase modifiers**

* **Negative: *neg***
* **Unresolved dependency: *dep***
* **Noun modifiers of a Phrase: *amod,* *det, neg***
* **Adjectival modifiers: *advmod, tmod, acomp, amod, infmod, npadvmod***

1. **Clausal Relationships in a sentence**

Table 9a – Direct relations between Clauses

|  |  |  |  |
| --- | --- | --- | --- |
| **Relation to subordinate** | **Function** | **Marker Type** | **Example** |
| conj | Conjunction (Complementary) | [conj, cc] | 1. John was found in Bethesda, but he is still very weak. *(found → weak)* 2. John is staying with a friend and waiting for help. (*staying → and, staying → waiting*) |
| advcl | Adjectival modifier | mark | 1. Mary went home so she could help others.*(went → help, mark: so)* 2. John was released because he was fine. *(released → fine, mark: because)* |
| ccomp | Object Clause (with internal subject) | Mark | 1. Mary went home so that she could help others*.(went → help)* |
| xcomp | Object Clause  (without internal subject) | aux/cc/to | 1. John is staying with a friend, waiting for help. (*staying → waiting*) 2. John went to the shelter to stay overnight. *(went → stay\*)* |
| csubj/csubjpass | Subject Clause |  | 1. If you find John, send an email.(*send → find*) |
| pcomp | Prepositional Clause | IN | 1. John is staying with a friend, while waiting for help. *(staying → while → waiting*) 2. We are searching for John, without having any success. |
| dep | Auxiliary/  dependent Clause | aux | TBD |

Table 9b - Indirect relations between Clauses

|  |  |  |  |
| --- | --- | --- | --- |
| **Relation to subordinate** | **Function** | **Connector** | **Example** |
| partmod | Particle Modifier | Noun (Subject) | 1. The man found near the building was identified as John Smith. (identified → man → found) |
| rcmod | Modifier | Noun (Object) | 1. John was taken to the shelter where he would be treated. (taken → shelter → treated) |
| prepc | TBD |  |  |

Note that certain relationships (such as advcl, ccomp, xcomp, etc.) between the Clauses are based upon the syntactic or lexical structure of the sentence rather than any semantic difference.

**Appendix D – Email messages used for Accuracy evaluation**

The following XML formatted messages are a subset of email messages, created from the Google Notes in the PFIF database and used for accuracy evaluation of PLIET, with the presented under the **Selected Set**in Section 7.2, Table 8

**<m id="1"><**body>I am looking for my gran's brother in Christchurch. His name is Billy William McDaniel. He lives at 45 Sinclair Street New Brighton, Christchurch </body> </m>

**<m id="2"><**subject>Justin Bieber believed alive. </subject><body>He was outside ChristChurch, New Zealand. He is ok now. Everything is fine. </body> </m>

**<m id="3">**<subject>Mark Wilder</subject><body>We have had text messages from Ken and he has spoken to Mark by phone. He is shaken but fine, currently with his friends at 24 Maple Street, Bishopdale.</body></m>

**<m id="4">**<subject>Mariana Alves Rossi.</subject><body>Notified Kaplan International Colleges Christchurch staff that she is safe.</body></m>

**<m id="5">**<body>Todd Woods has made contact with his brother James and he is ok.</body></m>

**<m id="6">**<subject>Angela O'Brien. Angela is fine.</subject><body>She is at home in Auckland with her family. They have no power, internet or phone (at this time, tuesday night 8.15pm) but the house is more or less standing.</body></m>

**<m id="7">**<subject>seaching for Maureen Griffiths and Trevor Griffiths. </subject><body>Tourists in Christchurch from WEst Australia </body></m>

**<m id="8">**<body>My name is Elena Pavel. I am looking for Diego Pavel. We live in the Chikidee Estates.</body></m>

**<m id="9">**<subject>Mitchell Gregory.</subject><body>Embassy of Japan in Jamaica would like to know he is safe.</body>

**<m id="10">**<body>Bailey Wallace has huge gash in head, is on the Oregon coast and bleeding a-lot, He needs HELP!!! please!</body></m>

**<m id="11"><**subject> Mr. Al-Safwan is dead</subject><body>I have received information from Mr. Al-Safwan's friend Abdullah Isaac that he was killed during a random drive-by shooting in Damascus.</body>

**<m id="12">**<subject>Andrea Johnson is lost</subjec<body>I do not know where I am right now. Somewhere in Cambridge? There is no food. I feel exhausted and helpless.</body></m>

**<m id="13"><**subject>found safe</subject><body>David Miller has used a friend's cell phone to call us to tell he's safe. Their house is a shambles.</body></m>