**LPF Information Extraction from Email Messages (PLEIT Task)**

*Date: June 01, 2013*

This document describes the PLEIT approach for extracting Lost Person related information from informal text messages received as emails at the LPF disaster site, and classifies the messages into three different categories, based upon their complexity for information extraction. Note that the discussions here pertain to *work in progress* and subject to updates in future.

1. **Background**

***Email Characteristics***

The textual contents of an email, categorized under “informal text” pose a set of challenges, mentioned below, in addition to the ones encountered in processing formal text found in regular documents.

* Punctuations, capitalizations and detection of sentence boundaries may not always be correct.
* A sentence could be grammatically incorrect; missing subject, or verb. [May also contain “interjections” that don’t provide any useful information.]
* Adjectives may be used as verbs (John alive.).
* A statement may be split into several sentences, with implied subject/object (from previous or follow up sentences.)
* Some sentences may be extraneous, not containing any useful information for People Locator.
* The information contained in both the Subject Header and the body field of the email need to be processed. One of these fields may be empty. If not, data in both fields should to be merged and reconciled to arrive at the final results.
* Text in subject header may be cryptic in nature with a few disjoint words summarizing the message, or it may span full sentences.

***Techniques***

Several different methodologies may be used, either individually or in combination, to locate and extract information from textual documents, using a domain-specific vocabulary. They include:

1. Sending a document through a natural language processing (NLP) pipeline, such as UIMA, GATES, etc. The basic functions are tokenization, parsing (to determine Parts of Speech tags), gazetteer lookup, ortho-matching, etc.
2. Refining the results obtained above through context analysis. This is important to properly identify persons, locations etc. in a disaster area, independent of its any native language.
3. Recognition of relevant information through pattern matching, either as standard regular expressions, or more advanced ones such as the GATE/JAPE tool
4. Determining names, genders, locations, numbers etc. through dictionary/gazetteer lookups
5. Applying machine learning techniques for “Named Entity Recognition” in the document

Because of the nature of the text in an email, especially one sent from a disaster site either by a victim or an associate (friend/relative/relief worker), it may be desirable to use all these techniques in combination to extract as much information as feasible (with an option for the operator to edit the final record.) However, because of our limited test dataset and the wide variation in the structure of these messages, there is difficulty in training a model to yield reasonable accurate results; and therefore won’t be used in the initial release of PLIET.

* 1. **Related Work**

There are no readily available tools or techniques dealing with text extraction from email messages as such, though various aspects of formal/informal text processing are discussed in literature, including determining names in informal text using machine learning.

* Since we have adopted an NLP-based approach for this task, a crucial aspect was the choice of the most suitable parser. The following parsers were experimented with: (a) Open NLP   
  Parser (b) GATE Text Parser (c) Linked Parser (d) Stanford Text parser, and (e) Stanford Dependency Parser. We used the Stanford Dependency Parser, V 2.1 (SD parser) as the most suitable one for our approach. Example of its parsed outputs are shown in Appendix A.

The NLP pipeline that we used is the GATE 7.1 release, with a number of customized user plug-ins. They include the Phil Gooch Pronoun Annotator and the LPF Named Entity Resolver.

1. **PLIET Task Description**

Extraction of Lost Person related information from informal email text messages involves the following functionalities:

1. Extraction of specific data elements (text components/fields) from the email header/body
2. Analysis of data elements to build cohesive Lost Person-related datasets
3. Interpretation of above datasets to determine the identity, status and location of a reported person, differentiating from or merging with that of the Reporting Person.

These functionalities and their implementation approaches are described below. The reference numbers there refer to the corresponding discussions and/or examples in Section (b).

Note that in the following section, the term parser refers to the Stanford Dependency Parser (V2.1) unless stated otherwise.

* 1. **General Approach**
  2. **PLIET Data Structures**

The important text-based data structures, defined by the PLIET task, are as follows:

* **LPFLexicon:** This static data defines the keywords, and their classification, to be used for recognizing key information in a PL email message. It includes various data sets including:
  + Verbs andadjectives describing the condition of a person and their PL-established category (e.g. Alive and Well, Deceased, Found, Injured, Missing, Unknown)
  + Keywords denoting an enquiry or report by a person related to a lost person
  + Various criteria to resolve ambiguity of certain verbs depending upon the voice and/or context
  + Interpretation of negative verbs/phrases
  + Weight of different category of person condition
* **TextAnnotation:** A wrapper class around a GATE Annotation (Sentence, Token, Location, Person etc.) with a number of additional methods for relationship tracing with other Annotations in the document. The Annotations and their basic features (type: noun/verb/preposition/adjective…), and dependencies with other Tokens, are determined by the underlying Stanford Dependency Parser.
* **TextAnchor:** A set of consecutive TextAnnotations grouped together to denote an entity such as a person’s name, a verb, an address etc. which function together as an entity, such as subject, verb, object, etc in sentence. Its derived classes are: Noun Anchors, Verb Anchors and Adjective Anchors. Important attributes of a TextAnchor are its relationships (links and link types) to other anchors in the text. All relationships are intra-sentence except for pronominal (Pronoun-related) ones, which generally span sentence boundary.
* **LPFTriple:** An LPFTriple is equivalent in concept to an RDF Triple, representing a of <Subject><Verb><Object> structure in a sentence, where each member of the triple is a Text Anchor. The Triple is Verb-centric; that is: it must have a verb, whereas either the Subject or the Object (but not both) could be null. Thus a sentence can be decomposed to one or more Triples, starting with each Verb Anchor, and tracing the relationships with other Anchors.
* **Clause:** A Clause represents a grammatical clause in a sentence, the Verb being the head of the Clause with links to other data. It is hierarchical in nature - with a parent Clause, and one or more subordinate Clauses. The relationship of a subordinate Clause to its parent Clause is indicated by the relation between the main verb of the Clauses, the clause genre, and the marker word (preceding the clause) (following English grammar by WREN and Martin). The top Clause of a clausal hierarchy is called a primary Clause. The subjects and objects of a clause may be phrases as well as subordinate clauses. The relation of a subordinate Clause to its parent Clause, and the Clause genre, is determined by analyzing the enclosed Tokens’ features and dependencies returned by the parser. Note that either the subject or the object (or possibly both) of a Clause may be null.

A Clausal modifier of a noun (e.g. “rcmod” clause) is regarded as independent *Primary Clause* since it is not related to any verb as its parent. *(I know Mary, who lives here.)*

In PLIET, a sentence (containing a verb) is represented by a data structure, called the **ClauseTree,** consisting of one or more independent primary Clauses and the corresponding clausal hierarchies.

* **ClausalAssertion:** A ClausalAssertion is a data structure built around a Clause, and concept-ually similar to a triple with subject, verb and object. The Verb consists of a VerbAnchor. Although for general Text processing, the Subject could itself be a Clause, for this task, we consider NounAnchors only for the Subject of a ClausalAssertions.

In addition, ClausalAssertion stores the list of TextAnchors, representing all Persons or Locations, associated with this clause hierarchy as Objects.

Just as a Clause, each ClausalAssertion object contains the set of subAssertions corresponding to the subClauses of the Clause.

* **LPFVerbModel:** This class describes a Verb with all its properties, as derived from the context of the corresponding CasualAssertion. Specifically, it denotes the verb category (REPORTING/HEALTH\_STATUS/UNKNOW, its health status category, voice, negative/positive, its (adjectival) modifier etc., in addition to reference to the container clause.
* **AnaphoraMap**

An AnaphoraMap is a Java TreeMap structure that connects a pronoun to the TextAnchor representing a Name or Location it references to. This map is generated by PLIET based on the antecedent/anaphoric relations between various Annotations, spanning multiple sentences, corresponding to annotated Pronouns in a document. (The Pronominal generator tools we studied deal with singular pronouns only.) Antecedents for a plural pronoun are determined by PLIET using algorithms based upon gender and other properties of Noun anchors – as explained in Section 3 c.

* **LPFSubject:** An LPFSubject represents a unique Person which is the Subject of a ClausalAssertion. All ClausalAssertions with the same TextAnchor as the Subject are listed as a set for the corresponding LPFSubject. The attributes of an LPFSubject are: The NounAnchor representing the Subject, the set of ClausalAssertions, the set of corresponding VerbModels, and the set of TextAnchors in the test representing Persons and Locations.

Note that in creating the LPFSubjects, all ClausalAssertions (irrespective of their clause level) are added to create a full listof assertions (assentially flattening the hierarchial structure to a list), and their Subjects are searched to create the unique LPFSubject instances for the document. Note that Anaphoric relations are resolved (and Pronoun Subjects are replaced by the corresponding Noun Anchors), and Partial Name matching is performed for generating unique LPFSubjects.

* **LPFPerson:** An LPFPerson represents either a ReportedPerson, or a Reporter, with corresponding attributes, by checking verb category in the set of VerbModels (and Assertions) in each LPFSubject.

1. **Important Notes:**
2. **Choosing a Parser**

The initial analysis of the PLIET task indicated that in order to interpret the information in an email, it was necessary not only to identify relevant words, but also to interpret the information they convey through context analysis. To capture the context, one may go down to the word (Token) level Annotations in a sentence and examine the relationship of each word with other words in that sentence.

We considered two alternate approaches to parse the text to decompose a sentence to manageable and meaningful chunks:

* Building a Penn Treebank (PTB) style Phrase Structure Tree, by recursively dividing a sentence’s text into a Noun Phrase and Verb Phrase - supported by OpenNLP (Example: *A rare black squirrel has become a regular visitor to a suburban garden)*
* Building other data structures (Triples, Clause Trees, etc.) by analyzing the relationship and dependency between individual words in a sentence – supported by the Stanford Dependency Parser.

Since the PTB format does not directly represent the grammatical functions between words, it is difficult to use it directly in applications. In PTB examples, it is not clear how the syntactic arguments (e.g. the subject and the object) of verbs are identified. (Reference: *Evaluating contributions of natural language parsers to protein–protein interaction extraction: by Yusuke Miyao, Kenji Sagae… Vol. 25 no. 3 2009, pages 394–400 doi:10.1093/bioinformatics/btn631).* Thus, for this work, where it is important to identify such arguments and interpret the text, we chose the SD parser.

There are about 55 different relationship types through which two related words in a sentence may be connected, though only a subset of these relationship types applies to each specific Parts-of-Speech type. Such dependencies may be further used to chunk a sentence into clauses, and to determine the genre of clause by analyzing the dependency of one verb on another.

1. **Triple vs. Clausal Approach**

We examined two different ways of connecting the word sets (Anchors) in a sentence: as Triples and through Clauses. The first approach was easier and was experimented with initially. However, for complex sentence structures, it became difficult to establish the context of a Triple, and its exact meaning, without having additional supporting information. In a Clause-based approach, however, it was feasible to maintain the context though the genre, the clause level (position of a clause in the clausal hierarchy) and other associated properties of a clause.

1. **Anaphora/Cataphora Resolution**

In a textual data containing one or more sentences, the relationship between a pronoun and its referent is established by linking the pronoun with an appropriate noun term(s), usually representing a person, or place or objects.

In an anaphoric relation, the noun (antecedent) precedes the pronoun (John is looking for his brother [John, brother]), whereas in a cataphora the referent noun follows the pronoun (When he was found, John was unconscious [he, John]). Note that in informal text, cataphoric occurrences are much rarer compared to anaphoric ones.

The Pronominal anaphora resolver in the GATE pipeline resolves only singular pronouns (he/she/his etc.) in anaphoric context. It (and other tools we investigated) does not resolve plural pronouns and cataphoric references. Therefore, additional Anaphora resolution is performed in PLIET, in two passes, as follows:

In the first pass, all Pronoun Annotations are stored in a list, along with any antecedent established by the Pronoun Annotator application. In the second pass, all NounAnchors are matched against the unresolved Pronouns in terms of gender, number (singular/plural), etc. Both anaphoric and cataphoric relations are treated the same way to establish the referent. Note that currently, the PLIET Anaphora Resolver does not span multiple NounAnchors to satisfy the relation.

To be implemented in future:

* Span multiple NounAnchors to satisfy a relation: For example: *John went to school, Mary went to work. They both had breakfast*, but handles: *John and Mary went to school. They both had breakfast.* In the second case, John and Mary are conjunction terms in the same Noun Anchor.)
* Resolve familiar relationships: For example: John Smith is looking for his son Michael. He was last seen in Christchurch. [His son => John Smith’s son]

1. **Dealing with Interjections**

Interjections are a category of words in the Parts-of-Speech representation, which are simply used to draw attention to, or to emphasize, some statement. They are not used in “formal text” and most parsers, including the Stanford Dependency Parser V1.?? / (which are trained using “formal text” based corpora) do not deal with Interjections. The SD Parser V2.1 (released in April 2013) is enhanced to identify Interjections through a new dependency relation called “discourse”, we switched to this newer version for use in PLIET. However, initial experiments indicated that interjection word “Please” id not always captured properly, and often regarded as the verb in the sentence, there by rendering the parsed results inaccurate. To overcome this problem (without altering the meaning of a sentence) such interjections are now deleted from an email message in a pre-processing step.

1. **Interrogative Sentences**

In parsing a sentence, the SD Parser cannot difference between normal sentences and interrogative sentences (e.g. “you know that he is missing”, vs. “Do you know that he is missing?”) though they impart very different meaning to the sentence. In an email message one may not assume that a sentence would end in “question mark”, to be easily recognized as interrogative. PLIET establishes certain grammatical rules to identify interrogation in processing the text.

1. **Named Entity Resolution**

The modules in the GATE Pipeline create the *Person* and *Location* annotations by analyzing the parsed words in a sentence. The property of each word, words in its neighborhood are used along with Gazetteer lookups to denote if a term is a person or a place. (A Term refers to an Annotation comprising a single word or a set of consecutive words representing an Entity)

Terms that cannot be resolved are annotated as *Unknown*, and added to the output Annotation set.

Although PLIET determines if a term is person or location based upon many factors, including contextual relationship between words in a sentence, Person and Location annotation are used by the Pronoun Annotator processes to link pronouns to corresponding Noun terms. Thus, it is necessary to further analyze the Unknown annotations and classify those to Person/Location if possible. This is performed by the PLIET-developed LPFNamedEntiryResolver application Note that this application is added to the processing pipeline prior to the PronounAnnotator.

An Unknown annotation is changed to a Location annotation through the following sequential tests:

* The Unknown term is related to a known location through an “appos” relation (such as Bethesda, Maryland (Bethesda*: appos Maryland*)
* If an Unknown term is part of a phrase where one of the other terms is a known Location, then it is merged with that Location Annotation and deleted from Unknown list. ( elementary school Bethesda)
* The Unknown term is object of preposition link conveying a location (such as *in, from, near, towards, around* etc.) and then validated as a geo-coder location by invoking the Google Geocoder Service

If it is not a location, it is tested for a person.

**Ambiguity Resolution of Common Verbs**

A set of verbs, which may be used to represent the health status of an affected person or a statement/enquiry by another person, are resolved by applying various criteria specified in the lexicon, as well as by analyzing other contextual data as mentioned below, in the following order:

* Voice - active/ passive (*I found vs. John was found*)
* Complement word related by “TO” (*want to know, called to inform*)
* Object keywords *(get information, want news)*
* Object type of the verb - Person/nonperson *(Looking for John vs. John is looking fine)*
* Most probable usage – Default (*Found in Bethesda.* Default *-> Health status*)

**Handling Clauses with Null Subjects**

In informal text, a sentence or a clause may not have an associated subject (e.g. *Looking for Jane in Bethesda*.) All Assertions (consisting of verbs and objects) with no Subject are grouped under a special structure called NULLSubject, which is an LPFSubject instance (with NULL as its subject), to hold the corresponding clausal information. This data is searched in searching for reported persons (Jane) and/or locations (Bethesda) if not available otherwise in te text.

>>>>>>>>>>

* 1. **Dataflow**

1. **Extraction of data elements from a message’s text stream**

**Task:**

**Mark the sentences in the text stream and divide each sentence into a number of chunks, consisting of one or more words *that operate as a unit*. Identify the type (Noun, Verb, or Adjective) and relations (Link types) between these chunks.**

**Approach:**

Step 1 – Use NLP processing, with GATE framework integrated with the Stanford Dependency parser, to perform the following functions

* Determine sentence boundaries.
* Tokenize the text in each sentence.
* Parse the sentence and assign a parts-of-speech (POS) tag to each word token.
* Determine the grammatical relations of a token (*governor*) with other tokens (*dependents*). [1]

Step 2 – Generate LPF Triples by PLEIT software modules to:

* Analyze the tokens and their relations to other tokens, if any, to generate textual chunks called *anchors*. [2] An anchor has an assigned type (Noun, Verb, Adjective), and other type-specific properties such as singular/plural, gender, tense, voice, qualifier etc. Note that anchor generation is a generic text processing function of PLEIT.
* Determine relationships between various anchors in a sentence. This is a LPF-domain specific function, which marks the verb anchors as related to: (1) a reported person’s health status, (2) a reporting person’s action or (3) Unknown, and traces how other anchors are related to each verb anchor to generate subject-verb-object *triples.*

An object may be null, and there could be multiple triples in a sentence. [3] A sentence may also simply comprise a triple with no verb or object (for example: just a person’s name in the “subject field” of an e-mail.)

1. **Analyzing Triples and building LPF-related datasets**

**Task:**

**Perform Anaphora resolution and determine Equivalency of Noun anchors to create an information structure (LPFSubject) for each unique Subject in the message.**

**2.1 Anaphora resolution**

Determine the Anchors that constitute anaphor (pronoun/adverb) and antecedent (noun) pairs - referring to the same Subject or Object in different triples, *which may span multiple sentences.* Combine information from different anaphors for the same antecedent anchor, and from equivalent Noun anchors to create a data structure for each unique Subject in the message.

**Approach:**

Find anaphoric relations for *persons and locations only*, (but not for currency, time, etc.) which are of interest to the LPF task. It involves a preprocessing and a post-processing phase, performed before and after step 2 of Function 1 above.

1. ***Preprocessing: Generation of anaphoric annotations for Locations and Persons -*** using the following modules in sequence

* GATE/ANNIE Transducer**:** Generate *Location* and *Person* annotations within a sentence. (It does a gazetteer lookup, and terms not there are marked as UNKNOWN)
* LPFLocationResolver (developed in-house, integrated with GATE): Check the UNKNOWN annotations and resolve/disambiguate Location/Person using Google Geocoder service along with analysis of grammatical relationships.
* Phil Gooch PronounAnnotator (integrated with GATE): to annotate possible anaphoric terms, and indicate their antecedent Location/Person annotations, when that could be resolved.

1. ***Post-processing: Generation of LPFSubject data structures***

* Based upon the anaphoric annotations generated in (a) develop an anaphora map connecting the anchor pairs that enclose each anaphor and its antecedent.
* Develop grammar rules and detect additional antecedents, especially for plural pronouns, as they may be missed or erroneously identified by the PronounAnnotator, using inhouse-developed **AnaphoraExtractor** module, and add them to the anaphora map. [4]
* Combine data in each set of anaphoric Subject anchors in triples of all sentences, and build a common set – consisting of a single Subject anchor (the antecedent) with its multiple verb-object pairs.
* Substitute the anaphor in the object fields of the set with their antecedents, if present.

**Equivalency Determination - Approach:**

This is a text matching operation which determines if the texts of two Subject anchors are equivalent, either through exact match, partial name match or component (conjunction) match.

1. **Interpret LPFSubject dataset for required information on Reported Persons**

**Task:**

**Interpret information in individual LPFSubject structures to determine Reported Person(s), their latest health status, and location. Fill the missing elements from other known data. Assign a confidence label to each message.**

**Approach:**

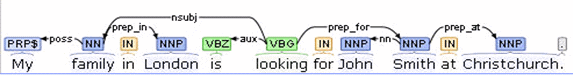
This involves a number of LPF rule-based operations, most of them being context-sensitive. The final health status of the reported person is determined based upon the following actions:

* Implement *Verb* *Rules* to mark *Reported Person*, *Reporter,* *either* or *none.*
* Analyze timelines of verbs for a Subject to determine the latest set of health status. [5]
* Implement rules to establish relative weight of different verbs in a set. [6]
* Assess impact of negative qualifiers in verbs and subjects (*not* found, received *no* words).
* Handle *interrogative and conditional* sentences (***Is*** *he okay*, ***if*** *he is okay* vs. *he is okay*).
* Reconcile for missing subject and/or object in a sentence from neighboring data (TBD).
* Examine environmental parameters (disaster location, date) to fill missing elements.
* Establish rules to assign confidence level (high, medium, low) (TBD).
* Generate the final LPFPerson records, one per reported individual, following the above steps.

Note: *It may be noted here that except for certain anaphora resolution, this task does not depend upon gazetteer lookups to recognize a person’s name - which is useful in dealing with disasters in countries with English is not the native tongue.*

1. ***Examples and Discussions***

**[1]** **Collapsed Dependency of words in a sentence -**  from Stanford Dependency Parser



Each pointed arch joins two related tokens, showing the type of their grammatical relation, as defined by the Stanford Parser. The origin is the *governor* and the arrow points to the *dependent*. (In the GATE pipeline, the *feature map* in the annotation of a token shows the relations and dependents for which that token is the *governor*.)

**[2] Dividing a sentence into Anchors** (NA=> Noun Anchor, VA=> Verb Anchor)

**I am looking for John Smith and Mary Lee who were last seen at Christchurch.**

| NA| |--- VA ---| |-------- NA -------| |NA| |--- VA ---| |--- NA ---|

The relationships between two anchors (links) are determined from the type of arch joining the governor tokens of those anchors.

**[3] Triples** (Subject –> Verb –> Object)

(1) I –> am looking –>John Smith and Mary Lee

(2) who -> were seen -> Christchurch

**[4] Anaphora resolution**

I experimented with three different pronominal resolvers, including the one from GATE/ANNIE, and found the Phil Gooch PronounAnnotator to be most suitable to work with the GATE framework. It assumes the Locations and Persons annotations to be already present in the input *annotation set* for the text stream. However, it works fine for singular pronouns only, but fails to associate the proper antecedents for plural pronouns (the case with the other two resolvers also), as shown in the following examples

* I saw Mary. She was injured. *(She -> Mary):* correct
* I saw John and Mary. They were injured. *(They->John and Mary):* missing
* I talked to my friends. We are concerned about John and Mary. They are missing.

(*They -> friends*): incorrect

The last two cases are properly resolved by the in-house *AnaphoraResolver* module. However, this still cannot process sentences where the antecedent consists of multiple words with no direct relation between them, as in the following example:

* I saw Mary, and talked to John. They were both injured.

Here Mary and John are not related by any conjunction links, unlike the second case in the previous group. (Handling such cases would involve more complex algorithms and linguistic rules. However, I did not find such examples in the *PL Notes* database on a casual search.)

**[5] Timeline analysis and Event selection**

Analysis of event timelines is important to understand the sequence of events and determine the latest one, although in our case it is not necessary to analyze the sequence itself. So a simple precedence-based algorithm is proposed here:

1. Choose present tense over past tense
2. Within the latest tense – choose higher priority status verb over lower priority
3. Within same priority – choose the last one in the utterance

However, step 2 need further refinement, since the message may contain explicit time specification such as *yesterday*, *three days back*, *on the 19th*, *afterwards*, and such. This problem would be addressed later.

**[6] Rules in choosing Health status verbs**

a) Establish explicit rules for choosing the highest order verb out of a set.

Example: (*seen*, *located*), (*injured*, *rescued*)

b) Assign weight to qualifiers of a verb in a statement, (such as assumed, considered, thought, believed ,…) for selecting the highest weight one.

Example: *John was only* ***injured****, although we* ***believed*** *he was* ***dead****.*

**[7] Dealing with multi-word Terms**

Dealing with multiple consecutive words representing a single component of a tuple is tricky, since it can have many forms. For example, consider the following cases and the relations shown in []:

* *John and Mary Smith* were in Bethesda. [*Conjunction*: Smith, John]
* *John Smith and Mary Lee* were in Bethesda. [*Conjunction*: Lee, Smith]
* *John Smith, Mary Lee, and their son* were in Bethesda. [*Appos*: Smith, Lee]
* *John Smith, my brother*, was in Bethesda. [*Appos*: Smith, brother]
* *My brother John Smith* was in Bethesda. [*Dep*: Smith brother]

In each of these cases, the subject of the sentence is shown in Italics, but the relationship among the words and the technique to connect the individual Tokens are very different.

Thus, to get the full subject/object of a verb, various possible relationships between the subject of the verb (*Smith* or *Lee* in all the above cases) and other Tokens have to be checked.

***c)******Classification of Email Messages in terms of Complexity***

Based upon the discussions above, we may classify the email messages into three categories in terms of their complexities. This classification scheme and example messages (as adopted from the Google PL Notes) are provided below:

1. **Simple messages:**

*One or two sentences - with well defined structures, recognized status verbs, with standard anaphora resolution. Single or multiple subjects. No ambiguities.*

**Examples:**

* Michael Brown is missing in christchurch.
* Rigene Aquino is injured and staying with her friends in Meycauayan, Bulacan.
* Steve Warren's family in London is very worried for him.
* I am looking for my brother Michael Sterling, who was last seen in Christchurch.
* Jessica Taylor. This is me. My family and I are fine.
* Jayna Richards had contacted her sister, and said she is ok.
* I am a teacher with LSI Christchurch, and I have received news that Rie Fujimoto is alive and safe in Christchurch. This is second-hand news from another student.
* Daryll Patco in Philippines has reported on twitter as safe now.

1. **Medium complexity messages:**

*With sentences that the PLEIT task would be able to process correctly using various rules discussed earlier. Also includes most interrogative and conditional type messages.*

**Examples:**

* Michael and Ann Sterling alive but injured. Their friends contacted them. They are waiting for assistance in their home near Nakhon Sawan.
* Krupskaya Valila trapped on the rooftop of a house in the specified neighbourhood. They have no more water or food left. They've been up there since 3pm. Send help immediately.
* Matthew Esguerra. Please inform us if he has been rescued, and provide water n food.
* Clara Rigor does not reply to any form of contact. Cannot be reached.
* Will anyone with information about Merv and Bella Dickinson get in touch with Muriungi? Will be glad to know if they are well and safe.
* Fiona Wilcox not in Christchurch when earthquake hit. Currently fine and in Geraldine. Partner/boyfriend Tim (from the UK) is also fine.
* Anybody trying to reach Hazel, she is ok and currently in Auckland.
* Andrew Holmes usually is in this vicinity of cbd and he has not been seen or heard from by family, will leave messge with red cross.
* Rose Mia Salazar and Cherry Rojas. Please help they were stuck in the 2nd floor of their home in 121 Harvard Street, Provident Village.

1. **Complex messages**

*These messages involve: Inadequate anaphora resolution, more complex timeline analysis, text with indiscernible or rambling sentences, or complex conditional phrases, Invalid structure, grammar, with ambiguity*

**Examples:**

* Marc Celis went to work around 5pm because his boss asked him to despite the heavy rains and unforgivable flood. After that we never heard from him.
* I do not know where Stuart was when the earthquake hit, but family in the UK is hoping to hear from him soon. Please get in touch Stuart if you see this so that we know you are well.
* Alan Gordon Armstrong,"5ft 10 in, 63 years old, known to work at Christchurch Polytech or University, wife called Jane. Please call brother Bruce in Australia 0061738235456. He is looking for you."
* I have spoken to Louise and she is driving home to her flat with husband Ben. Is doing well but last time i spoke to her, was making slow progress in car. Hopefully home by now. Ben and dog are also okay.
* Aaron Moses Infante. The baby needs to be transferred to the nearest hospital with electricity because of the power outage in DLS-STI. The baby needs to stay inside the incubator.

Some form of complexities can be handled, perhaps partially, by further research into the field and additional enhancements to the task. Still, if a message contains a large amount of extraneous data for determining the person/status, it might confuse the analyzer/interpreter modules to yield any meaningful result.

>>>>>>

**Choosing between the Open NLP vs. Stanford Dependency Parser**

There are two alternate approaches to parse the text to decompose a sentence to manageable and meaningful chunks:

* Building a Penn Treebank (PTB) style Phrase Structure Tree, by recursively dividing a sentence’s text into a Noun Phrase and Verb Phrase supported by OpenNLP (Example: *A rare black squirrel has become a regular visitor to a suburban garden)*
* Building other structures (Triples, Clause Trees, etc.) by analyzing the relationship and dependency between individual words in a sentence – as supported by the Stanford Dependency Parser.

Since the PTB format does not directly represent the grammatical functions between words, it is difficult to use it directly in applications. In PTB examples, it is not clear how the syntactic arguments (e.g. the subject and the object) of verbs are identified. (Reference: *Evaluating contributions of natural language parsers to protein–protein interaction extraction: by Yusuke Miyao, Kenji Sagae… Vol. 25 no. 3 2009, pages 394–400 doi:10.1093/bioinformatics/btn631).* Thus, for this work, where it is important to identify such arguments and interpret the text, we chose the SD parser.

**Important Notes:**

**Any time there is an anticipated disaster in a new location*, immediately create a new gazetteer for location (provinces, towns, facilities, etc.)***, and store it in the directory $GATE\_PLUGIN/ANNIE/resources/gazetteer. Then edit the “list.def” file in the same directory and enter this file name in the specified format (with ‘:’ separated entry types). Then restart the Web application.

**This will greatly help in recognizing name of disaster-hit places as locations by PLIET.**