

Extracting Spatial and Temporal Information from Natural Language

author1
address1
address1
address1
email: email1

author2
address2
address2
email: email2

author3
address3
address3
address3
email: email3

Abstract—Dealing with temporal and location information in text is difficult but is highly desirable for Intelligence tasks. We present in this paper a system that analyzes natural language documents and focuses on temporal and location information. We overview the system, our approach to sentence analysis, and our framework for dealing with information on time and (geographic) location. Our classification of temporal and location expressions allows us to develop, in an organized manner, sets of patterns to locate these expressions, and is the basis for normalization procedures for said expressions.

I. INTRODUCTION

Natural language analysis tools are very important for Intelligence tasks. A considerable amount of information is available in document of various types (mail messages, technical reports,...). The recent increase on use of open source (OSINT) has made the number of documents available for analysis grow at a very high rate. Hence, Intelligence analysts require tools to help inspect, classify and analyze the large quantity of documents available.

Many programs exist that analyze natural language. Most of them are based on Information Extraction (IE) tools ([23], [3]). Such tools are able to locate and extract some of the most salient features of a narrative: entities involved, relationships among them, events described. However, many IE tools are limited to a certain domain, while others require considerable setup time and effort. Finally, very few of them seem especially dedicated to dealing with temporal and location information. Handling time is known to be a complex problem that requires sophisticated AI tools, including reasoning ([2]). However, this additional information is very useful for Intelligence Analysis. Situating events is vital to put actions in their proper geo-strategical context; this in turn is an important part of the complex task of interpreting events ([4], [5], [15]). Having these details can help analyst do a more thorough analysis: for instance, it can help relate events (events that happen at the same time and/or the same place; or some events preceding others -usually a necessary condition for causality analysis). Unfortunately, such analysis is hard due to the flexibility and expressivity of natural language.

In this paper we present our system for natural language analysis, with especial emphasis on our tools to extract and analyze temporal and location information. We analyze text at

the sentence level, and for each sentence we output a structure that is anchored by the triple Subject-Action-Object, but has also additional optional components to store information about aspects of the event. Because of its recursive nature, the schema can accommodate very complex sentences, while remaining simple. Thus, the schema is easy to store (it can be represented by a simple DTD in XML [27]) and to query (XML query languages can be used for this purpose). Our contributions are:

- A simple, yet flexible and powerful schema to store (semantic) information from natural language sentences, that extends the known (but limited) Subject-Action-Object triple. The representation includes slots for temporal and location information, as well as additional information that may or may not be present. It also has a recursive nature to accommodate complex sentences with embedded sub-sentences.
- A classification of temporal and location expressions that provides a framework for capture, representation and reasoning with this types of information.
- A software module for recognition of temporal and location expression, based on regular expressions and gazetteers. The module focuses on prepositional phrases started by certain prepositions (like *before*, *after*, *in*, *on*, *at*), but can also recognize other types of expressions. The module is able to cope with absolute expressions (dates) but also with relative expressions (like *a week ago*).

In the next section, we provide an overview of the overall system. Then, in section III we show how our analysis proceeds at the sentence level. Section IV introduces our classification of temporal and location expressions and describe our software module for dealing with them. Some preliminary experimental results are also presented and discussed. In section V we discuss related research. Finally, we close with some conclusion and a discussion of further work.

II. SYSTEM OVERVIEW

The overall workflow of our system is summarized in the following six steps:

- Step 1: Splitting input text into separate sentences. The input text is processed using an English parser to mark sentence boundaries.

- Step 2: Parsing. Connexor ([9]) is used to parse the input text. When a sentence is successfully parsed, the parser tags each word with its part of speech (POS), and gives syntactic details such as tense and number. It also marks syntactic relationship among words. When a word or sequence of words is not identified by the parser, it does not get a POS tag and is not covered by the parse tree. A word or sequence of words with no tags attached form a *Lost Chain* and are assigned a node marked LOST by our program. The parser output is constructed as an XML tree for efficient processing. This XML data is further processed using DOM.
- Step 3: Splitting sentences with coordination. The XML output from Step 2 is examined to determine if the sentence has a coordination under the main structure of the sentence. Coordinated sentences are split into smaller sentences to simplify them and extract all relevant details without having to chunk many details within one sentence component. The separated sentences will have their own complete subject-verb-predicate structure. Coordination inside of components such as subject or object does not cause a sentence to be split.
- Step 4: Sentence Re-construction. Sometimes the parser fails to successfully parse a sentence. We distinguish between *partial* and *total* parser failures. A partial failure happens when at least one word in sentence is not tagged and not covered by the parse tree. A total failure happens if the parser fails to recognize the main verb in sentence. Fortunately, total failures are very rare. In both cases, we use several heuristics to try to obtain a good parse. Many times the sentence can be “reorganized” by moving some parts around. The LOST chains are examined and re-arranged to simplify the sentence structure. The reconstructed sentence is then processed through steps 2 and 3 again.
- Step 5: Extraction of details. We define a successful parse as one where the parser identifies the main verb in the sentence (i.e. no total failure). After a successful parse of the sentence, the XML output is processed to extract details from the sentence. Details on this processing are given in the next section.
- Step 6: Formatting Output. Once the details are identified from the sentence, they are formatted into a certain XML DTD. The format includes the main verb, subject, object, as well as location, temporal, and manner information, with recursive, complex subcomponents if needed. This format can then be used for analysis, or eventual storage into a relational database for future use. The DTD is described in detail below.

We next give details about steps 5 and 6, the fundamental steps in our approach.

III. SENTENCE ANALYSIS

A few programs break up natural language sentences into triples made up of Subject-Action-Object components. Clearly, such tuples contain valuable information but leave out much

additional detail that may be available in a sentence. For instance, the sentence

Rebels attacked an oil rig in Nigeria last week using explosives and automatic assault rifles

gives us not only an action with a subject and an object but also additional information about the action: temporal information (*when* the action happened: *last week*), location information (*where* the action happened: *in Nigeria*), manner information (*how* the action happened: *using explosives and automatic assault rifles*).

Like past research (see section V), we assume each sentence contains, at least, a Subject-Action-Object triple (although it is perfectly possible to have sentences with less than this, such a case is infrequent). However, we also assume that, often times, sentences have more information attached to this core.

We start our analysis of parser output by locating the main verb of the sentence. It is known that the nature of the main verb often determines the content, number and structure of the arguments of the verb ([22]). We distinguish three types of verbs:

- Communicative. Sentences with verbs that are communicative in nature have a complex object that refers to the content of the communication. This complex object contains most of the details of the event. The communicative verbs we have explored include *say, tell, explain, believe, think, observe, recommend, inform*. As an example, consider the sentence *Around 30 firefighting machines were used for over an hour to control the flames, said a fire official*. The content of the communication contains most of the information in the sentence, for instance the temporal information *for over an hour*. The object in such sentences is recursively considered as a sentence on its own subject, object, main verb and possibly complements. Our program analyzes the object to extract all the details from it.
- Attributive. When the main verb is *be* or *have*, then the sentence can be interpreted as stating that an entity (or class of entities) has a certain property with a certain value. The entity is denoted by the subject, and the property and value by the object. For instance, the sentence *This contract has a potential value of \$300 million* states that the entity *This contract* has the property *potential value* with value *\$300 million*. Note that the verbs *have* and *be* also appear frequently as an auxiliary to another verb and in such instances this interpretation would be incorrect. Thus, it is necessary to reliably distinguish between uses of *be* and *have* as main verbs or as auxiliaries.
- Other. All other verbs are lumped together since we did not use concrete case information for different verbs, instead relying on the parser analysis.

We capture all information in a structure which has a triple at its core, but offers room for optional arguments. To further accommodate the complexity of natural language, several components of the triple and some optional arguments can be further down decomposed into parts, which recursively

may contain a triple of their own. This allows us to analyze quite complex sentences within a uniform and (relatively) simple structure. The DTD for the output template is shown in Figure 1. For simplicity, all elements without subparts (i.e. just PCDATA) do not have their own entry. The optional elements are indicated by '?', and the choices by '|', as usual in DTDs. Note that adverbial elements are given a type, indicating the kind of information they contain (duration, frequency, goal, etc.). Note also the *Lost* element, which is our scape hatch for sentences that cannot be analyzed completely.

```
<!ELEMENT Sentence
  (Verb Subj Obj Tmp? Loc? Agt?
   SubjComp? ObjComp?
   Advl? CC? App?
   PP? PostMod? Lost?)>
<!ELEMENT Subj (#PCDATA (Tmp? Loc? Mod?
  Quantifier))>
<!ELEMENT Obj (#PCDATA | (Sentence? Mod?))>
<!ELEMENT Mod (#PCDATA | ( Sentence?))>
<!ELEMENT PP (#PCDATA | Prep (Pcomp Mod |
  Sentence))>
<!ELEMENT CC (#PCDATA | (Sentence)>
<!ATTLIST Advl Type (dur | man | frq | goa |
  sou | pth | cnt | meta |
  cla ) #REQUIRED>
```

Fig. 1. DTD of the output format

Text is only attached to leaf nodes (those with no subparts). The complex components of the sentence implicitly cover the text attached to the leaves under them.

IV. TEMPORAL AND LOCATION INFORMATION EXTRACTION

Most language statements are *situated*, that is, they refer to a certain context in which they should be interpreted. As part of this context, most sentences contain temporal and location information in one way or another. In fact, it is difficult to think of a natural language statement that does not have an implicit or explicit time component. Sometimes a sentence simply expresses time with verb tense: *The train left the station* clearly indicates an action that occurs in the past (with respect to the time of utterance) in the use of past tense in the main verb *to leave*. In this paper, we disregard this implicit time and focus on explicit time information in sentences. The main reason is that implicit time information is usually vague, while explicit information tends to be more precise. All we know about the train is that it left the station sometime in the past (maybe one hour ago, maybe days ago¹). However, explicit expressions tend to give more information *The train left the station two hours ago* give us a concrete point in time. Of course, explicit information may be vague too: *The train left the station some time ago* does not tell

¹*Pragmatics* would lead to disregard the last case, but we do not enter here into pragmatical considerations ([11]).

us much. Fortunately, such vague statements are infrequent in many sources, especially the type followed by Intelligence agencies (news wires, etc.).

Similar remarks can be made about location information: it is very often present, and it can be implicit or explicit. However, while location information is also pervasive, it seems more likely to be absent than temporal information. Also, when present, the implicit type seems more frequent than for temporal information. For instance, a sentence like *They bought a ticket* has implicit temporal information, but no location information -unless the context makes clear a location where tickets are bought. This makes gathering temporal information more difficult. Another point of difficulty in dealing with location information, as we will see, is that there are certain patterns that are often used to express temporal information, but location information relies more on world knowledge -hence, in *open sets* of words, which cannot be captured very efficiently by simple patterns like regular expressions.

We consider that temporal expressions can be classified as follows:

- *absolute expressions*. Such expressions indicate time with respect to a calendar (i.e. dates). This include expressions like *November 1945*, or *September 11, 2001*. Note that the expressions can have different levels of *granularity*, that is, the first one denotes a whole month, the second a single day. Such expressions may be accompanied by an indication of uncertainty, e.g. *sometime during November 1945*.
- *relative expressions*. Such expressions indicate time with respect to some additional information. They can be subdivided into two types:
 - *indexicals*. These expressions can be seen as mappings or functions that take one argument and return an absolute date. The argument is usually the value of the current time, that is, *today* or *now*. Examples include *yesterday*, *last summer*, *the night before last*. Note again that there are different levels of granularity. There can also be a modifier for uncertainty (*approximately two hours ago*).
 - *event-based*. These expressions can also be seen as a mapping that allows us to compute an absolute time given enough information. However, this information is complex in itself, usually an event description. A simple example is *before the fall of Baghdad*. The event description that follows is called the *reference event* and can be either a single name, an NP (as in our simple example), or a more complex description, including a whole sentence (complete with its own triple structure). It is even possible to have another temporal expression nested within the event description of the temporal expression (an example is shown below). It is important to point out that expressions like *Two weeks later* are also included here. We consider that this expression refers to an

implicit event, which must be found in the context (surrounding text) of the expression. For instance, in the text “*The rebels attacked the capital. Two weeks later, the city fell.*” we consider the temporal expression equivalent to *Two weeks after the rebels attacked the capital.*

Location information can be expressed by several types of expressions too. We distinguish two types:

- *absolute expressions.* These denote a certain geographical location by proper name. Linguistically, however, the expression may comprise a whole NP that includes the proper name: *in Baghdad, in the city of Baghdad, on the banks of Yamuna.*
- *relative expressions.* These give a location by connecting it to another location via a spatial relationship: *north of Baghdad, in the outskirts of Baghdad, 100 miles from the Pakistani border.* We note that expressions like *northern Iraq* also belong here.

Even after temporal and location expressions are identified, there is a very important task left: *normalization.* This task identifies the information contained in the expressions in some standardized form, which makes reasoning with said information much simpler. The process is different in each case:

- *Time Normalization.* Ideally, a standard reference for time should be absolute and complete. Complete means that a point in time can be identified, usually by having a year-month-day-hour-minute-second representation. This way, all events can be placed on a common timeline, compared and ordered. Most temporal references in text do not contain the complete representations. This, completion may be needed even for absolute references. Also, relative references must be translated into absolute ones. We assume that the document itself has a certain reference time, which gives value to the special constant *now*, which is expressed in standardized form (another special constant, *today*, is obtained from *now*). A temporal reasoning module is then applied to transform expressions like *a week ago* or *last May* to an absolute, normalized value. Note that event-based expressions are also relative, but to be normalized we must first associate the event itself (the fall of Baghdad, in our example) with an absolute time (and even then this type of expression tends to have poor granularity). Such module is currently under development.
- *Location Normalization.* This involves two processes: *disambiguation* and *locating.* The first one is needed because geographic names may be ambiguous, e.g. two cities with the same name. We attempt this by examining the context (surrounding text) of a given location for clues that will help trim a list of possible referents (obtained from a gazetteer). We also use the closest other location expression to a given one for indicators. Even after a certain place (say, a city) is located, it may be desirable to use a standardized format to express the information.

We may use, as in the time case, a representation of the form city-area-country, or we may resort to a longitude-latitude format. Note that sometimes we will be dealing with *points* in space, sometimes with *areas.* If all we know is that an event happened by a river, all the area covered by the river is the possible location for the event. Note also that some events call for points (a gunshot), while others call for whole areas (a military offensive). The process of *locating* refers to pinpointing a location for relative expressions. In a manner similar to relative time expressions, we need to calculate a geographical referent for a relative location expression by considering it a function that must be applied to an argument to yield a value. The process consists of finding an absolute (normalized) location for the reference event, and applying a function that corresponds to the spatial relationship that is being used. This again may result in a point or an area, but the result is usually a (broad) area, and is considered as set of constraints that determine what falls within the area. For instance, *north of Baghdad* can only result in an area, calculated by obtaining an absolute (standard) location for Baghdad and then applying a function that is basically a test (returning true for any location north of the one for Baghdad, false otherwise). Modules of location normalization are also being currently developed.

We have pointed out that both temporal and location information come in different *granularity.* We consider that both time and geographical information can be organized in a hierarchy. Thus, for location, a city is contained in a county, an county in a state, a state in a country. Likewise, for time an hour exists within a day, that exists within a week, etc. A given expression refers to a location/time at a certain level in the hierarchy; granularity refers to the lowest level in the hierarchy that an expression can be associated with. Many expressions in documents have *low* granularity (they refer to a level *high up* in the hierarchy). We tend to prefer lower levels in the hierarchy because they denote more smaller time intervals/geographic areas, hence giving more accurate details about the involved events. Sometimes context may allow refining the granularity of a temporal/location expression; however, we do not attempt this refinement in our system.

In order to obtain as much temporal and location information as possible, we follow two strategies: we use hints from the parser, and we develop specific sets of syntactic patterns. The parser does not indicate temporal and location information but in a few cases; however, in others, it provides some clues as to where the information could be located. When it is absolute, the parser marks temporal information correctly most of the time. When the temporal information is relative, the parser does not recognize it. We have used syntactic patterns to recognize temporal information embedded inside prepositional phrases and modifiers. In the case of location, the parser correctly identifies proper names found in its dictionary, if such information is found in the main phrase of the sentence. To identify location information inside sub-phrases we use syntactic pattern recognition ([10]) along

with world information. In order to focus our search, we have concentrated on prepositional phrases (PP), based on the observation that many of the expressions with time and location information are started with prepositions ([7]). Thus, *before*, *after*, *during* are indicators that commonly are followed by a temporal expression. However, there are exceptions; even when followed by the right expressions, some prepositions are difficult to interpret due to ambiguity. For instance, if *on* is followed by a time expression (like a day of the week: *on Tuesday*) it refers to time, but followed by a location or type of location (like *on the docks*) it refers to location. Yet, followed by phrases like *such a large scale*, *on* is indicative of manner. Hence, a combination of syntactic pattern recognition as in [8], [22], [21] and examination of prepositions are used here. The preposition gives an indication of what may follow; based on this, different patterns are tried in order.

Patterns for temporal expressions include regular expressions made up of constants, POS tags, and grammatical categories. Patterns for location expressions include those using geographical features, like *in the region*, *in the city*, etc. This can be used alone or together with a proper name (*in the region of Kazakhstan*). Our patterns include regular expressions that in the end reduce to either atomic markers (a certain letter, punctuation symbol) or to a basic category. A basic category is a type of thing as it occurs in the real world: for instance, a city name, or a last name, are basic categories. Most of the time, such categories constitute an *open set*, that is, a set whose elements are not identified by some kind of pattern, but simply form part of our world knowledge. For this cases, we rely on *gazetteers*, explicit lists of values. For instance, we use a list of city names, one of country names, and so on, as part of our pattern recognition schema. Note that such lists are largely domain-independent but are language-dependent.

We are still expanding our set of patterns to deal with more and more complex expressions. For instance, in the sentence *Not since the torrential floods from Tropical Storm Allison, which badly damaged the Texas Medical Center in 2001, has scientific research been disrupted on such a large scale*, we failed in a first attempt to recognize *Not since the torrential floods from Tropical Storm Allison*, which is a relative, event-based expression in our classification. This has been solved by using the pattern *[Not] since NP*, where *NP* refers to a Noun Phrase. The problem here is that it is hard to put boundaries on the event description, since one can argue that the whole expression should be *Not since the torrential floods from Tropical Storm Allison, which badly damaged the Texas Medical Center in 2001*. This has embedded into it an absolute reference (*in 2001*), which is independently recognized by our program. Ideally, however, one would like to see all this expression as one, since this allows reasoning that the temporal expression denotes *Not since 2001*.

We point out a methodological issue that has come up in this research. It is clear that creating lists of regular expressions and gazetteers can cover only so much: there is always a new, unknown name. Places may change names; man-made artifacts are created anew. Thus, many researchers

have turned to machine learning techniques in an attempt to deal with the shortcomings of the pattern approach. In fact, most recent research in Information Extraction ([23], [3]) relies on such approaches. This is indeed a worthwhile avenue of research, and we plan to investigate the use of learning techniques in the near future. However, we point out that most learning techniques perform better when restricted to a certain domain, and not so well in domain-independent tasks. Domain independent learning is an exciting new area of research ([12]), but still in its infancy. We set our approach to be domain independent from the start, which made adapting a particular learning approach difficult. Also, the most powerful learning techniques are the (semi)supervised ones, and require *training* -usually in the form of annotated data that makes up a training set. Coming up with such sets is a tremendous amount of work (whole research efforts have been devoted to tools to alleviate this task; see the Alembic workbench at MITRE ([1]). We also set our approach to be usable without any training. Hence, we have decided to exploit the full power of syntactic pattern matching ([10]) and gazetteers before delving into learning efforts. One possible avenue of research is to combine both approaches by using patterns and gazetteers to provide training sets to learning algorithms in an automatic manner.

A. Preliminary Experimental Results

We ran some preliminary test of our program, using texts from news websites www.cnn.com, www.bbc.co.uk and www.timesofindia.com. We chose text on different subjects and from different sources to make sure our processing was truly domain independent. The whole system was tested using 20 paragraphs of text containing a total of 100 sentences. Such sentences had an average of 24.14 words; the shortest one had 9 words and the longest sentence had 67 words. There were 6 total failures; our heuristics for reconstruction dealt with 3 of those 6 failures.

Unfortunately, gathering standard precision/recall numbers turned out to be a complex task, as there is considerable ambiguity present. It can be quite difficult for humans to agree on exactly what constitutes location information. For instance, a sentence stated that *Barcelona will not align his best player*. Barcelona is certainly a city name, but the context makes clear that the sentence refers to a soccer team. Is the name *Barcelona* (which any program with a decent gazetteer will catch) a false positive? What about a reference to the *University of Bangalore*? While *Bangalore* is certainly a location, here it is part of a larger name that denotes an institution. In the expression *China's larger mines*, is the whole expression the correct location (as opposed to just the country name)? Complex expressions that may have a location expression embedded within them (as in *Nigeria's southern state of Bayelsa*) pose another problem: do they count as one or two?

To fight ambiguity, we adopted a simplistic solution: two humans analyzed a sample of 64 sentences looking for temporal and location expressions, and counted as part of the ground truth only those expressions in which the two agreed. This yielded a total of 50 location expressions (plus 11 in

dispute) and 39 temporal expressions (there were no disputes on temporal expressions). Our program recognized 50 location expressions, of which 5 were considered mistakes. Hence, for location precision can be considered as 90% (number of correct expressions found by the program divided by number of undisputed expressions located by humans), and recall at 90% (number of correct expressions located by program divided by total number of expressions located by the program). Note that if we include the expressions in dispute, the numbers go down to 73% and 74%, respectively. For temporal information, our program recognized 37 temporal expressions, of which 31 were correct. Hence, precision and recall for time was 79.5% and 84%.

We believe that our results show promising performance, although it is clear that much more extensive testing are needed: we regard such results as temporary, and are continually expanding our test set. However, examination of mistakes also reveals that improving performance may be quite difficult: note that the only way to distinguish the reference to a soccer team (as opposed to a city) is to do an in-depth semantic analysis of the sentence. Also, 3 of the 6 mistakes in our temporal information recognition were motivated by a paragraph about the Today show (an US television show); again, the only way to recognize that this is not a temporal expression may be a semantic analysis of the sentence.

V. RELATED RESEARCH

There is a very large body of research in natural language analysis and Information Extraction, which we cannot hope to cover here. hence, we only mention a (subjective) selection of closely related references.

A line of research has focused on the role of verbs. The Proposition Bank developed by Palmer et al. ([22]) incorporates semantic roles to the syntactic structures of verbs in the Penn TreeBank. By representing the semantic roles of verbs, the PropBank aims at providing a domain independent resource for natural language processing systems. Each verb is considered to form a roleset with the arguments it can take corresponding to a distinct usage. In the same vein, Klavans and Kan ([14]) present an effective way to classify documents based on the verbs found in them. The study focuses on both the frequency and distribution of verbs within the documents. The set of verbs in a document form a conceptual map of the events and actions. This body of work has clearly influenced our approach, in that we also depend on locating and analyzing a main verb.

Hideo and Sanderson ([12]) have developed a system that extracts descriptive phrases (dp) of text from documents for a given user query noun (qn). The system aims at finding appositives, which are noun phrases that describes a given noun or gives more information about it. This has influenced our analysis of failed parses.

Several efforts have attempted to capture natural language content in a fixed framework. Liang et al. ([18]) develop a framework to extract data frames from text. A data frame is a table that represents data in a table-like (rows and columns)

format. Unlike conventional data mining techniques, they use NLP tags for indexing and text storage and retrieval. The proof of concept system, called *InFact*, uses syntactic parsing and performs clause-level indexing, thus capturing syntactic roles, relationships, inter-clause relationships etc. The result is a set of inter-linked subject-action-object triplets. Leskovec et al. describe the use of semantic graphs for document extraction ([19]). They have used two types of syntactic analysis for the semantic graph representation of sentences a simple linguistic analysis that extracts noun phrases and verbs (NPV) or adjectives, nouns and verbs (ANV) and a slightly sophisticated linguistic analysis which extracts logical form (LF) triples of subject-predicate-object. The START (SynTactic Analysis using Reversible Transformations) ([13]) natural language processing system is a question-answering system, consisting of an understanding module that generates a knowledge base and a generating module that produces English sentence answers in response to questions. START breaks up sentences into smaller units called kernels. After analyzing the kernels, START rearranges the elements of the parse tree into representational structures, to form Ternary expressions or T-expressions. The most salient structure is the subject-relation-object triple. Litkowski ([17]) has developed a prototype question-answering system that uses semantic relation triples to store answer documents as well as questions. The system uses a parser to break documents into sentences, obtain parse trees for these sentences and then extract triples from the parse trees to be stored into a database. A semantic relation triple consists of a discourse entity, a semantic relation in which the entity participates and a governing word that relates the entity to the semantic relation. The semantic relation includes such roles as agent, location, theme, manner, modifier, purpose and time. This system supports time-related and location-related questions by allowing the semantic relationships TIME and IN to represent, respectively, temporal and location information.

There is also a long tradition of using simple finite-state approaches to deal with text. Pugeault et al. ([24]) aim at extracting predicate-argument structures from texts. Their study is very domain-specific, focused on extracting details from research project descriptions. Byrd et alia ([8]) use a series of extractors called *Textract* to identify salient concepts, entities and their relationships from large collections of text. The frequency of occurrence of concepts and entities are used to identify salient concepts. Concepts are identified by looking for grammatical patterns. A Name Extractor is used to determine names of persons and organizations. Since the study deals with large collections of text, speed is a critical consideration. Therefore, instead of relying on extensive syntactic parse results, the extractor uses special built-in finite state automata. However, no special attention is paid to temporal and location information in this body of work.

Work focusing specifically on temporal or location information is relatively recent and not abundant. [16] is devoted to location normalization using a combination of techniques. It is noted there that location information can be highly ambiguous. For each location expression, a set of possible referents is

created. This set is then explored and one referent is chosen as the most likely one, based on local information (words co-occurring with the location expression) and global information (other information in the document, especially information related to the location expression). The system described in [20] has a Location Extraction module and time extraction routines, but they are both severely limited. Location extraction is restricted to using a large gazetteer, and ambiguity is not dealt with (in fact, only names that can be unambiguously determined are considered). Time extraction is limited to a short list of patterns for dates. The work in [26] extracts location information but focuses on images, not text. The work in [7] extracts temporal information using prepositions as clues to finding such information, and is a predecessor to our own approach.

Barzilay et al. come up with a technique to extract sequences of related words, called lexical chains, to produce summary of documents ([6]) Cohesion is the principle by which different parts of a sentence stick together. Lexical chains help represent the cohesive structure of the text. We plan to apply this work in order to related different events in a document, helping produce a single *timeline* of events for each document.

VI. CONCLUSION AND FURTHER WORK

We have described our system to process natural language, with special emphasis on the extraction and analysis of temporal and location information. Our framework provides the basis for our view of temporal and location expressions as functions that can be analyzed part by part to obtain a value. The scope of this work is limited by the fact that we attempt to remain domain-independent and use limited but efficient methods that require no additional setup.

This is work in progress, and therefore we continue to expand our sets of patterns and gazetteers, as well as the testing sets used. We are also considering a number of enhancements that can be added to our system to ensure more accurate extraction of details from sentences. Currently, we do not address co-reference resolution. Implementation of co-reference resolution prior to the storage of details in a relational database will facilitate extraction of information by question-answering. Note that our strategy of splitting sentences may hinder co-reference resolution efforts; we plan to keep track of splitted sentences and share this information with the co-reference resolution module. Many different verb types can be considered to increase the kinds of analysis possible, as in [14]. However, many verbs are polysemic (some, highly so), so this extension must be considered together with ambiguity fighting modules. One of our most ambitious goals is to make do without full parsing information. In particular, we want to rely only on *chunking*, the splitting of the sentence into phrases (NP, VP, and so on), since this analysis is more resilient and efficient. However, not having the main verb of the sentence available requires an in-depth transformation of our approach.

Finally, we are working on the ability to, given a document and the set of all temporal references extracted from it, organize the references in a *timeline* that will give us a date

for the document and a date for the main events in the text, as well as their (temporal) relationship. Such organization is bound to prove very useful for Intelligence Analysis, but is also highly complex and challenging.

REFERENCES

- [1] Alembic Project, MITRE Corporation, www.mitre.org/technology/alembic-workbench/.
- [2] Allen, J. *Maintaining knowledge about temporal intervals*, Communications of the ACM, v.26 n.11, p.832-843, Nov. 1983.
- [3] Israel, D. and Appelt, D. *Introduction to Information Extraction*, tutorial in the IJCAI-99 Conference, available on the web at www.ai.sri.com/appelt/ie-tutorial/IJCAI99.pdf.
- [4] *A Consumer's Guide to Intelligence*, Washington, DC, CIA Public Affairs Staff, 1995.
- [5] *A Compendium of Analytic Tradecraft Notes*, Washington DC, CIA, Directorate of Intelligence, 1997.
- [6] Barzilay, R., Elhadad, M. *Using Lexical Chains for Text Summarization* in Proc. of the Intelligent Scalable Text Summarization Workshop (ISTS'97).
- [7] Bree, D.S. and Pratt, I.E. *Using prepositions to tell the time*, Proc. of the 19th Conference of the Cognitive Science Society.
- [8] Byrd, R. J. and Ravin, Y. *Identifying and Extracting Relations in Text* In NLDB'99 - 4th Intl. Conf. on Applications of Natural Language to Information Systems, 1999.
- [9] Connexor oy, www.connexor.com.
- [10] Fu, K. S. *Syntactic Pattern Recognition and Applications*, Prentice-Hall Englewood Cliffs, NJ. 1982
- [11] Grice, P., *Logic and Conversation: Syntax and Semantics*, volume 3, Academic Press, Cole, P. and Morgan, J., editors, 1975.
- [12] Hideo, J. and Sanderson, M. *Retrieving Descriptive Phrases from Large Amounts of Free Text* Proc. of the 9th Intl. Conf. on Information and Knowledge Management, 2000.
- [13] Katz, Boris *Annotating the World Wide Web using Natural Language* Proc. of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet, 1997.
- [14] Klavans, J. and Min-Yen Kan *Role of Verbs in Document Analysis*, Proc. of the 17th Intl. Conf. on Computational linguistics, 1998.
- [15] Krizan, L. *Intelligence Essentials for Everyone*, Occasional Paper n. 6, Joint Military Intelligence College, Washington DC, 1999.
- [16] Li, H., Srihari, R. K., Niu, C. and Li, W. *Location Normalization for Information Extraction*, Proc. 19th Intl. Conf. Comput. Linguistics (COLING 2002).
- [17] Litkowski, K. C. *Question-Answering Using Semantic Relation Triples*, in Proc. of the 8th Text Retrieval Conference (TREC-8), 1999.
- [18] Liang, J., Koperski, K. Nguyen, T. and Marchisio, G. *Extracting Statistical Data Frames from Text* ACM SIGKDD Explorations, June 2005. Vol 7, Issue 1, pgs. 67-75.
- [19] Leskovec, J., Milic-Frayling, N. and Grobelnik, M. *Impact of Linguistic Analysis on the Semantic Graph Coverage and Learning of Document Extracts*, 2005, pgs. 1069-1074.
- [20] Moon, N., Shu, Y. W. and Singh, R. *A Multiple-Perspective, Interactive Approach for Web Information Extraction and Exploration*, in Proc. of the 22nd Intl. Conf. on Data Engineering Workshops (ICDEW'06).
- [21] Ogonowski, A., Herviou, M. L. Dauphin, E. *Tools for Extracting and Structuring Knowledge from Texts* Proceedings of the 15th Intl. Conf. on Computational linguistics 1994.
- [22] Palmer, M., Paul Kingsbury and Danile Gildea *The Proposition Bank: An Annotated Corpus of Semantic Roles*, Computational Linguistics, March 2005. Vol 31, Issue 1, 71-106.
- [23] Patienza, M.T. (ed.) *Information Extraction*, International Summer School, SCIE-97, 1997; LNAI 1299, Springer-Verlag.
- [24] Pugeault, F., Saint-Dizier, P. and Monteil, M. *Knowledge Extraction from Texts: a method of extracting predicate-argument structures from texts*, Proc. of the 15th conference on Computational linguistics, 1994.
- [25] Surdeanu, M. and Harabagiu, S. *Infrastructure for Open-Domain Information Extraction*, in Proc. of the Human Language Technology Conference (HLT 2002).
- [26] Toyama, K. et alia *Geographic Location Tags on Digital Images*, ACM Multimedia, 2003.
- [27] Extensible Markup Language (XML), 4th edition <http://www.w3.org/TR/xml/>.