LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-209871

Natural Language Processing as a Discipline at LLNL

M. A. Firpo

February 18, 2005

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Natural Language Processing as a Discipline at LLNL

Michael Firpo

Abstract

The field of Natural Language Processing (NLP) is described as it applies to the needs of LLNL in handling free-text. The state of the practice is outlined with the emphasis placed on two specific aspects of NLP: Information Extraction and Discourse Integration. A brief description is included of the NLP applications currently being used at LLNL. A gap analysis provides a look at where the technology needs work in order to meet the needs of LLNL. Finally, recommendations are made to meet these needs.

Executive Summary

This paper presents an introduction to Natural Language Processing, describing five primary tasks: Morphology, Syntactic Processing, Semantic Processing, Discourse Integration, and Pragmatics. The state of the practice is described with emphases on the two primary tasks of interest to LLNL scientists: Semantic Processing (in the form of Information Extraction), and Discourse Integration (including coreference). There are gaps between the technology that is available and the technology that is needed by LLNL researchers. The most reasonable approach for the Lab and ASC to take in pursuing an NLP discipline is then recommended.

Appendices provide examples of the various aspects of NLP and a digest of selected papers.

The Laboratory should not be concerned with developing NLP tools to support Morphology, Syntax, or Pragmatics. Morphology is solved, Syntax is essentially solved, and Pragmatics is beyond our scope.

Information Extraction is prohibitively the most prevalent manifestation of Semantic Processing in the NLP community. Scientists at LLNL are specifically interested in extraction of Entities, Relationships, and Events. Entity Extraction has essentially already been solved for many useful domains. Though some work is needed in less mainstream domains, this work is expected to be an application of known principles (i.e. this is more of a development issue than a research issue). There are a variety of teams in the community that already have the infrastructure and momentum and are closing in on solutions for Relationship and Event Extraction. Thus, LLNL should partner with an existing team to meet our remaining IE needs.

The area of Discourse Integration needs work. ASC/LLNL should invest in R&D in the field of Discourse Integration.

LLNL can build a good tool to meet its NLP needs by getting an existing Syntactic Processor, co-developing an Information Extractor with an experienced partner, developing our own Discourse Integration, and integrating the pieces using Systems Engineering practices.

There is a team of NLP scientists at Pacific Northwest National Laboratory (PNNL) who are funded directly by DHS. Their work focuses on Relationship Extraction. LLNL could collaborate with them to meet the needs of DHS.

1. Introduction

1.1 Purpose

The purpose of this document is to discuss my work over the past several months in Natural Language Processing (NLP), describe the state of the practice in NLP, indicate the gaps in current technology as they impact LLNL's need to utilize NLP, and recommend means to develop the necessary NLP technology.

1.2 Scope

For the scope of this document, NLP will refer to the processing of free-text. In other words, it will consider neither spoken language nor structured or semi-structured text. Furthermore, while many of the individual tools referenced in this document work with multiple natural languages, the intended use of this technology is for English language texts.

An important aspect of NLP not addressed in this document is the annotation, or tagging, of texts. During annotation the user tags the text (syntactically, semantically, or for coreferences) for purposes of training or testing a software package. Annotation forms a bottleneck in the NLP process. It is considered in a separate paper entitled A Novel Approach to Semantic and Coreference Annotation at LLNL.

1.3 Elementary Natural Language Processing

NLP is composed of five components, or primary tasks:

1. Morphology: how a word relates to its root.
2. Syntactic Processing: parts of speech, and phrase identification.
3. Semantic Processing: identifying meaning; this is primarily done via Information Extraction (IE) which utilizes a set of rules for pattern matching, which in turn make use of various resources, such as gazetteers, lexicons, shallow syntax parsing, etc.

4. Discourse Integration: connecting meaning between sentences, handling coreference, managing other idiosyncrasies of the English language that make it useful for humans, but difficult for machines. For the purpose of this paper, DI is a separate entity from IE. This is done for the sake of clarity. In practice, DI is often incorporated into IE tools for usability and often to make use of iterative techniques of performing IE and DI.
5. Pragmatics: integration of real-world knowledge with semantic meaning of words and phrases. While there are external efforts to develop computational pragmatics systems, LLNL is more interested in using NLP to fill the Semantic Graph. Therefore pragmatics is outside the scope of this document.

The data used in NLP tasks is typically in the form of plain text, emails, or HTML text documents. A body of such documents is referred to as a *corpus* (more than one body are *corpora*). The syntactic, semantic, and discourse tasks all utilize human input in the form of an annotated corpus. The annotations typically appear in the form of XML tags, or as metadata in a separate file with offsets to the location of data in the original file.

NLP tools can be divided into two types, expert systems and adaptive systems (a.k.a. machine learning systems). Expert systems require an expert user. Adaptive systems learn from annotated data. Both adaptive and expert systems require annotated data for verification and validation.

1.4 Measuring Accuracy

Accuracy in NLP is considered in terms of two basic measures: precision and recall.

Precision is the number of correctly identified items (words, phrases, etc.) divided by the total number of identified items. Increased precision results in a reduced rate of false positives.

Recall is the number of correctly identified items divided by the number of items available (i.e. the number that should have been identified). Improvements in recall reduce the rate of false negatives.

To form a convenient single measure on the same order of magnitude of both precision and recall, the community uses F1, which is the harmonic mean of the two measures (or twice the product over the sum):

$$F1 = (2 * P * R) / (P + R)$$

where P is precision and R is recall. All three measures can be reported as a number between 0 and 1, or as a percent.

While F1 gives equal weight to precision and recall, a more general form of the F-measure allows the user to weight the two unequally:

$$F_{\alpha} = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha)\frac{1}{R}}$$

where α is the weighting factor of precision, it is less than one, and the weighting factor of recall is its compliment 1 - α. By setting α to 0.5, the elements are of equal weight, and the formula simplifies to the harmonic mean; F1 = $F_{0.5}$.

2. My Work in NLP

In addition to educating myself on NLP, my work has involved: evaluating existing academic tools, tagging corpora and working through associated challenges, interacting with the Louisiana Project team, familiarizing myself with their needs, and considering alternative solutions to meet those needs.

2.1 Evaluating Existing Academic Tools

A variety of academic NLP tools are available, some are available to the public, and some are not. Table 1 contains a list of tools I encountered during this assignment. Appendix C contains an evaluation of selected tools.

Table 1: Academic NLP Software Tools

Package	Scientist	Institution	Purpose
Porter Stemmer	Martin Porter	Cambridge, UK	Find root of each word.
Brill Tagger	Eric Brill	MIT	Tag parts of speech (POS)
Marmot		U. Massechusetts, Amherst	Tag POS and bracket phrases. (a.k.a. syntax parsing)
Crystal		U. Massechusetts, Amherst	Learns Text Extraction Rules
Badger		U. Massechusetts, Amherst	IE
WHISK	Stephen Soderland	U. Washinton, Seattle	IE
CORA		U. Massechusetts, Amherst	Various tools
MALLET	Andrew	U. Massechusetts,	Various tools,

	McCallum	Amherst	Java object library
GATE		Sheffield U., UK	Various tools, Java object library, IDE
Annie		Sheffield U., UK	Syntax parsing
Amilcare	Fabio Ciravegna	Sheffield U., UK	Relationship Extraction
Melita	Fabio Ciravegna	Sheffield U., UK	Semantic Tagger with active learning
Armadillo	Fabio Ciravegna	Sheffield U., UK	Information Harvester (beyond IE)
TRex	Fabio Ciravegna	Sheffield U., UK	Event Extraction
AutoSlog	Ellen Riloff	Utah University	Learn Text Extraction Rules
AutoSlog-TS	Ellen Riloff	Utah University	Learn Text Extraction Rules from Keywords instead of using a tagged corpus
Sundance	Ellen Riloff	Utah University	IE and DI (I think)
LTChunk		Language Technology Group*	Shallow Syntax Parsing
Lexicalized Parser	Christopher Manning	Stanford U.	Deep Syntax Parsing
TextToOnto	Philipp Cimiano	Karlsruhe, Germany	Extract Ontology from Corpora
MnM		Open University, UK	Semantic Tagging
Alembic Workbench	David Day Lisa Ferro	MITRE**	Semantic Tagging with active learning
Callisto		MITRE*	Semantic Tagging with active learning
MinorThird	William Cohen	Carnegie-Mellon	IE
Raptor	David Beckett	U. Bristol, UK	RDF Parser

*Language Technology Group is a commercial, not an academic institution.
** MITRE is a non-profit organization that manages government funded R&D centers (http://www.mitre.org).

2.2 Tagging Corpora

I have tagged documents from FBIS (Foreign Broadcast Information Service of the CIA) and from CNS (Center for Non-proliferation Studies, Monterey Institute of International Studies, Monterey, CA). The CNS corpus is compiled with the specific intent of accumulating information about terrorist activities and weapons of mass destruction activities in a single source. As such, it contains a wealth of interesting text that is worth tagging. By contrast, the FBIS corpus is more sparsely populated with tag-worthy information.

Because I experienced tagging to be an incredibly tedious and difficult task in which to maintain consistency, I spent time looking for pre-tagged corpora. There are pre-tagged corpora available including some from Defense Advanced Research Projects Agency's (DARPA) Message Understanding Conferences (MUC) and Automatic Content Extraction (ACE) program (described in section 3.8). Some academics offer pre-tagged data with their tools for evaluation purposes. I also obtained pre-tagged Pubmed data. After further investigation and discussion with wise individuals, it became evident that pre-tagged corpora would offer limited help for the following reasons:

- Any tool used to perform IE will have to be tuned to the corpus from which information will be extracted.
- It is of no utility to tune a system to a corpus purchased from ACE, and then expect it to extract information from CNS.
- A system must be trained to recognize LLNL's data, using LLNL's ontology.

The only real use for pre-tagged data is to evaluate software tools. The data from Pubmed was of limited value. Because it had been auto-tagged, it captured whole noun-phrases instead of simple entities, and was not of the highest quality. None of the data obtained was tagged for Relationship or Event Extraction, or for coreference.

2.3 Interacting with Louisiana

Over time, it became clear that my work was related to the Louisiana Project. Everett Wheelock, Information Fusion Group Leader, was my customer, and his interest was to improve the process of the Louisiana team, lead by Anton Fulmen (a.k.a. Homer Briggs).

I showed Fulmen and his team tools I downloaded from Sheffield University: Amilcare and Melita. Amilcare performs relationship extraction and Melita is an annotation tool with active learning. An active learning system does a number of things:

- it learns from initial user-provided tags,
- it divides untagged text into two categories: those that the system can tag on it's own with high confidence, and those that the system cannot confidently tag; and
- it prompts the user for further human annotations on certain documents that the system can not confidently tag.

Active learning makes the most efficient use of both user time and processor time.

Anton was pleased with the active learning system and requested that a similar facility be added to the Information Operation and Analysis (IOA) Center's existing tagging tool, SPUD. Eventually ThingFinder, a very good Entity Extractor, was integrated with SPUD.

At this time, my focus was on breaking the NLP process down and improving the individual parts. Tina Eliassi-Rad prompted me to get good IE before considering DI. Fulmen, however, was concerned more with the short-term deliverables than the long-term process improvement. Hence, he was invested in the commercially available all-in-one expert systems, and not in the parts and piece-meal tools coming from academia.[1]

Some disagreements arose about the best way to annotate text; I suggested tagging to each specific step, and Fulmen preferred tagging to the end-to-end process. We held a series of meetings on the topic of tagging, and in-between meetings we amassed tagging experience. Because tagging is such a repetitious and non-stimulating task, we first hoped to be able to tag once for all parts of the process. This was called the uber-tagging-rules. The hope was that annotators could sit down with each document once and tag for entities, relationships, events, and coreference.

The deeper we got into tagging, however, the more it became clear that this uber-tagging scheme was of limited utility. Entity tagging called for the tagging of each entity (once per document, or ubiquitously) where event tagging called for the tagging of entities strictly within the contexts of the text and the ontology. I believe there is a way to implement an uber-tagging scheme, however, the size of LLNL's ontologies would be too much for the typical human tagger to maintain in his or her frontal lobe while reading text. It seemed best to perform entity tagging first, and follow that with event tagging. As it became clear that there were good Entity Extractors available and not good Event Extractors, our attention turned to the tagging of events.

Other topics that were discussed at meetings included performance of the various all-in-one tools, the various tools for tagging and other aspects of NLP, features of the ontologies, President of ClearForest Ronnen Feldman's comments at the 2004 Conference on Knowledge Discovery and Data Mining, ACE tagged corpora, the pros and cons of establishing a team of corpus annotators, and the nature of ADVISE and its relationship to Louisiana and the Biodefence Knowledge Center (BKC).

[1] I have since concluded that Fulmen's approach is, in fact, the best for the short term; expert systems do currently provide the highest accuracy. However, experts in the field, such as Andrew McCallum [1] believe that adaptive systems will surpass the expert systems, even as Big Blue was able to beat the reigning human chess champion.

3. State of the Practice

The state of the NLP components: Morphology, Syntactic Processing, Semantic Processing, and Discourse Integration, are discussed in sections 3.1 – 3.4 respectively. These are followed by section 3.5 on current work at LLNL and sections 3.6 – 3.8 on NLP in the commercial, academic, and governmental sectors.

3.1 Morphology

The most well established morphology code is the Porter Stemmer, published in 1980 (http://www.tartarus.org/~martin/PorterStemmer/). Porter's stemming algorithm (or some variant thereof) is widely used throughout the NLP community and is integrated into the majority of available NLP codes. To my knowledge, there are no appreciable efforts to improve the state of morphology in the English language.

3.2 Syntactic Processing

There will continue to be incremental advances in Syntactic Processing, but usable systems are freely available with F1 measures in the high 80s and low 90s. LTChunk from Language Technology Group (LTG: http://www.ltg.ed.ac.uk) is an effective shallow parser. It labels each word with its part of speech, and 'chunks' the sentences into non-nested noun phrases and verb phrases. The Stanford Lexicalized Parser (SLP) (http://nlp.stanford.edu/downloads/lex-parser.shtml) is an excellent deep parser. SLP utilizes two methods of tagging parts of speech (one that makes use of lexical information, and one that does not) and merges the results [2]. As a deep parser, it annotates prepositional phrases in addition to the noun and verb phrases, and it nests phrases within other phrases, for a more precise parse than the shallow parsers. Examples of shallow and deep syntax parses are provided in Appendix A.

3.3 Information Extraction (Semantic Processing)

There is a concerted effort underway in the NLP community to solve the semantic problem, that is, to automatically find the meaning within a corpus. The prohibitively most popular approach to Semantic Processing is Information Extraction (IE). IE picks out interesting information from a document by utilizing a set of rules for pattern matching.

The user defines concepts that are of semantic interest, and codifies these in an ontology – a specification of semantic constructs of interest. Currently most ontologies are written in some form of XML. An example ontology in DAML (DARPA Agent Markup Language) format is provided in Appendix A.

There are four main types of information generally extracted from the text, which are listed in Table 2.

Table 2: Types of information to be extracted and expected accuracy [3]

Type of Information	Description	Expected Accuracy
Entity or Concept	an object of interest such as a person, organization, location, or date	90–98%
Attribute or Property	a property of an entity such as its name, alias, gender, employer, or model	80%
Relationship or Fact	a relationship between two or more entities such as Position of a Person in a Company	60–70%
Event	an activity, potentially involving multiple entities such as a terrorist act, airline crash, management change, new product introduction	50–60%

There are commercial products on the market (described in 3.6) that achieve very high accuracy when extracting entities. Little is said about attribute extraction, this seems to get subsumed into relationship extraction.

The relationship and event numbers still lag below a useful level of accuracy (F-measures of roughly 80% to get started according to domain experts; this number is expected to rise as people grow accustomed to using the tools, and as the corpora continue to increase in volume). Note that LLNL researchers have observed a qualitative trend of inflated self-reported accuracy in the commercial arena. According to Anton Fulmen, all vendors claim to extract relationships in the 90s or high 80s, but those numbers are meaningless without a detailed understanding of what relationships are being extracted, what documents they are being extracted from, and what criteria are being applied to discern correct extraction. Often effectiveness claims are backed up by nothing more than an informal eyeballed assessment of a few documents, sometimes the same documents that were used to train the system being evaluated.

One freely available relationship extractor is Amilcare, which surpasses 80% in published tests, but relies on later Discourse Integration to sort out the extracted relationships.[2]

While it may be tempting for LLNL to delve into Event Extraction, I believe this task is better–suited to teams of people who have been in the field for a while, who already have the infrastructure, and who are aggressively moving toward a solution.

[2] Amilcare was developed by Fabio Ciravegna, Sheffield University, UK. (http://www.dcs.shef.ac.uk/~fabio/Amilcare.html)

3.4 Discourse Integration (DI)

The discourse problem is still an open field of research, as attested to by Andrei Popescu-Belis: "As more and more language resources become available to computers, the numeric evaluation of coreference/anaphora resolution on unrestricted texts becomes possible, but perfect scores are still far from reach." [4] As such, it would make sense for the Lab to form a research team of software developers and natural language processors to investigate and develop DI tools.

The term *Discourse Integration* is often viewed as a synonym for the term *coreference*, but a broader definition would also include other features of the language, features that can be quite difficult for the computer to pick up even though the fluent human reader handles them without effort.

The coreference problem can be illustrated with the following sample discourse:

> George W. Bush invited Diane Feinstein to tea.
> She accepted his invitation.
> The President was happy to see the Senator.

In this text sample, two people are referenced by name, by pronoun, and finally by title or description. The fluent human reader is automatically aware that "Diane Feinstein", "she", and "the Senator" all refer to the same individual (this is true even if the reader is not familiar with Ms. Feinstein). The challenge of coreference is to teach the computer to recognize these truths, even in more complicated cases.

Other features of the language that can be considered aspects of DI include proper handling of indexical expressions (e.g. this, here, when) and additive expressions (e.g. also, as well, with, too). In a cursory literature search, I did not uncover any proposed methods or means of approaching these tasks. However, Antonio Sanfilippo mentions the great usefulness of these expressions to resolve relationships and coreference.[3] They are less challenging than the coreference problem. It would be reasonable to begin with the information provided by the deep Syntactic Parser as a way to disambiguate meaning within such expressions.

In practice, especially in the commercial world, DI is performed very closely with IE. This has its benefits, as much can be gained from an iterative approach to the two tasks. When considering how to improve overall performance, it is also useful to think about them separately, to ponder how best to improve each as an identifiable task. This is especially true of DI, which suffers lower accuracy than IE.

[3] Comment made by Antonio Sanfilippo (PNNL) during a conversation on November 11, 2004.

3.5 Current Work at LLNL

LLNL would be better described as a consumer of NLP, rather than as a developer. This is not a core competency.

I am aware of two research programs at LLNL with an interest in NLP, both are in the IOA Center. One is the Louisiana Project and the other is the Biodefense Knowledge Center (BKC).

The IOA Center maintains the Semantic Graph, which is fed information from a variety of sources, including databases, data feeds, structured and semi-structured text, and unstructured or free text. The Louisiana Project is the Center's primary means by which free text is processed for loading into the Graph.

Louisiana can be described as a pipeline with two primary components: a Knowledge Extractor (KE, described below) and a Graph Translation/post processor. The information also goes through a process of Canonicalization on its way into the Graph, but this is not unique to the free-text. Louisiana provides this service for a number of clients, including the Threats, Vulnerability, and Incidents Service (TVIS) and others who are external to LLNL. Louisiana will soon provide this for the BKC as well.

The BKC is in the early stages of assembling an NLP process. They have formed a partnership with Ellen Riloff at University of Utah, who is looking at relationship extraction. They expect it to be a multi-year effort, which may expand from Relationship Extraction in to other parts of IE and into DI as well.

The NLP portion of Louisiana is the Knowledge Extractor (KE), which performs Morphology, Syntactic Processing, Information Extraction, and Discourse Integration. To date, the KE used by Louisiana has been one of the various commercial products available. Three tools have been looked at: Attensity, ClearForest, and NetOwl. Other packages are waiting to be considered, including AeroText by Lockheed Martin, FactFinder by Inxight, Intelligent Miner by IBM, and Cicero by Language Computer Corporation (LCC) (see Table 3).

Scott Kohn has been named to head ASC; he will find someone to lead NLP research at LLNL.

3.6 NLP in the Commercial Sector

Because practical use of an automated system typically requires both IE and DI (which, in turn, depend on Morphology and Syntax), commercial producers of NLP tools generally produce all-in-one systems called Knowledge Extractors (KE). KE systems vary in the number and degree of adjustments the user can make to the process (e.g. in some cases the user writes all the IE pattern matching rules while in some the user cannot even edit the rules which may be provided by the vendor or

which may be determined via machine learning based on training data). The user feeds the corpus to the KE tool, adjusts what he or she will, and receives the completely–processed information as output.

Some of the vendors in contact with LLNL personnel are listed in Table 3.

Table 3: Commercial vendors in contact with LLNL

Company	Tool	Self-reported Accuracy	Tested At LLNL	LLNL users impression
Attensity	Malta Server	>90	Yes	Not worth it
Attensity	Relational Extraction Server	>90	No	Not worth it
ClearForest	ClearForest	>90	Not really	undecided
Systems Research & Applications	NetOwl	>90	Yes	Pretty good for now
Lockheed Martin	AeroText	>90	No	undecided
Inxight	ThingFinder (entity extraction only, no coreference)	>90 (we expect this one is true)	Yes	Terrific
Inxight	FactFinder	>90	No	undecided
LCC	Cicero	>90	No	undecided

LLNL-determined accuracies for some of these tools will be forthcoming in early 2005, others will trickle in as work proceeds.

3.7 NLP in the Academic Sector

There are academics who continue to pursue Syntactic Processing solutions (Christopher Manning at Stanford University, for example), but the field is largely dominated by IE and DI. It appears that IE is getting more attention than DI, but this is misleading. As iterative approaches become more prevalent, systems performing both tasks are often only referred to as Information Extractors.

The field of players is too large to address in the present scope. A selected list of academic researchers in the field is:

- Ellen Riloff, of Utah University, is an acquaintance of LLNL's Terence Critchlow; she is partnering with LLNL's BKC on Relationship Extraction.
- Andrew McCallum, of the University of Massachusetts at Amherst, is an acquaintance of LLNL's Tina Eliassi-Rad, and is a domain expert at the DHS

Information Sciences Workshop in September; his code, MALLET, is publicly available.
- Fabio Ciravegna, of Sheffield University in England, and I have been exchanging email during the past several months as I have evaluated his publicly available tools, Amilcare and Melita; I am looking forward to beta testing his new Event Extractor, TRex, soon.

Appendix B includes a digest of selected papers by these scientists on IE.

3.8 NLP in the Government Sector

The government has been involved in NLP for decades. I will not consider information prior to 1987, when the Navy sponsored the first Message Understanding Conference (MUC). In 1990, the DoD's Defense Advanced Research Projects Agency (DARPA) sponsored MUC-3, and continued with these conferences through MUC-7 in 1995. MUCs were an opportunity for IE developers, and in the later years for DI developers as well, to try their products blind on well–prepared data. DARPA has continued its participation in NLP with such notable items as the DARPA Agent Markup Language (DAML) for specification of ontologies, and its involvement in the Automatic Content Extraction (ACE) program. Both MUC and ACE data continue to be de facto standards for comparing various IE and DI tools.

As mentioned briefly in Table 1, there is a not-for-profit organization called MITRE (a spin-off of MIT Lincoln Labs) that has been tasked by the federal government to provide technical solutions to various problems including information science. MITRE runs the Center for Integrated Intelligence Systems (CIIS), which has produced software tools, such as Alembic Workbench and Callisto. MITRE also runs the "Northeast Regional Research Center (NRRC) for the Advanced Research and Development Activity (ARDA). ARDA is a U.S. Government entity that sponsors and promotes IT research and development for the Intelligence Community."[4] ARDA has three focus areas: Quantum Information Science, Information Exploitation, and Global Information System Access.

Perhaps the first national laboratory to answer the DHS's call for NLP technology was the Pacific Northwest National Laboratory (PNNL), which has a small team of NLP researchers including chief scientist Dr. Antonio Sanfilippo. Dr. Sanfilippo has been in the field of NLP for over 15 years, and has worked at SRA (makers of the NetOwl Knowledge Extractor) and in academic settings as well. His current work focuses on Relationship and Event Extraction, and will soon include DI as well.

A partnership between LLNL and the PNNL team may lead to valuable work. Because NLP is not a core competency at LLNL, we could gain from their expertise in the field. LLNL could contribute software development and customers. The PNNL team is more concerned with the theory and science of NLP than with actually

[4] http://www.mitre.org

developing a computer code; they could benefit from a LLNL team of software developers to implement their work. Also, although they have funding to do NLP, they lack a customer who is interested in utilizing the tools they produce. The ADVISE Program at LLNL would clearly benefit from such work.

4. Gaps in the Present State of NLP that Impact Current Efforts at LLNL

4.1 Information Extraction

While there are good Entity Extractors available, LLNL will need to make use of Relationship Extraction (a.k.a. Fact Extraction) and Event Extraction. So far, high quality Event and Relationship Extractors have been evasive, but the community is making advances, and better tools are likely to be developed in the near future.

4.2 Discourse Integration

As mentioned previously, Discourse Integration is an immature field. There are no tools currently available with the requisite accuracy to meet LLNL's needs. The NLP community is making investments in this area, but I recommend that LLNL also contribute to development for our unique (and classified) needs.

4.3 Available Options

I have identified three main alternative approaches to addressing LLNL's NLP needs: a) buy commercial solutions, b) build our own solution from scratch, or c) use a systems engineering approach to form a solution by taking the best of what currently exists for Morphology and Syntax, partner with an expert to provide IE in an LLNL-specific way, and form a team of researchers to address DI from the ground up.

Option a) has its merits, but it does not put NLP technology directly into the hands of DHS or LLNL. It ties us to a commercial interest and could lead to the sharing of sensitive or classified information with outside entities.

Option b) keeps the technology in-house, and protects classified data from outside interests, but it does involve large investments in technology that already exists.

Option c) is the middle path, providing the most technology and security for the least investment. This option might include a partnership with PNNL for the IE and DI components. I believe this is the best approach for LLNL because:

- DHS is interested in establishing an NLP program within the National Lab structure;
- LLNL has programs which need to use NLP;
- NLP is not a core LLNL technology; and

- Ramping up will take time; in the mean time there are tools available for use today.

In the following section I recommend an approach to accomplish this Systems Engineering option. The overarching Systems Engineering challenge is to integrate these components together to solve the NLP problem.

5. **Recommendations**

LLNL should take a Systems Engineering approach to researching and developing a Knowledge Extraction system by utilizing available technology to solve the Morphology and Syntax problems, collaborating with an expert team to solve the Information Extraction problem, and developing our own solution to the Discourse Integration problem, perhaps in collaboration with PNNL.

Good DI depends on having good IE, so IE is the prior concern. That being said, work can start on a DI system concurrently with work on the IE system. DI can initially be applied to Entity Extraction. DI can also be integrated into an IE system.

Work in both IE and DI utilizes pattern matching, which uses a rather light-weight syntactic parse, but seems to avoid the use of deep syntactic information. The primary reasons for this seems to be that deep syntax information is difficult to use and expensive to generate, in terms of processor time. Fortunately, here at LLNL, we have access to virtually unlimited processor time. Furthermore, I believe that the deep syntactic parse has a wealth of information we can use. While it has its limitations (e.g. handling typos and slang), a syntactic approach may be the perfect complement to the pattern matching approach.

5.1 Information Extraction

As indicated previously, the NLP community is working toward good solutions to the IE problems. I recommend that the LLNL NLP team partner with an external NLP team who is already invested in and working toward Relationship Extraction and Event Extraction. The exact form of this partnership is not yet clear; a partnership with PNNL would look markedly different from a partnership with a university team.

It would be reasonable to expect LLNL to pay a university partner for its expertise. Applying their nearly mature IE infrastructure to our unclassified data and ontologies will benefit both parties. They receive data and funding to continue their work in IE. We receive a mature software tool (source code included) and the understanding behind it, so that we can make the necessary modification for future work (including the application of this tool to classified data and ontologies).

In contrast, the NLP team at PNNL has funding through DHS. I would not expect LLNL to pay for technology. In fact, LLNL would likely be on the receiving end of both the technology and the funding, as PNNL has NLP scientists and is in need of a team of software developers and of a user-base, which LLNL can provide in the form of the ADVISE Project. Also, the problem of sharing sensitive and classified data is simpler with a sister DOE lab than with most universities.

Selecting a partner should be done with an eye on the big picture. There are a variety of interested parties at LLNL. A working group could be formed to get input from all sources and form a consensus. Possible candidates include: Antonio Sanfilippo (PNNL), Ellen Riloff (University of Utah), Fabio Ciravegna (Sheffield University), and Andrew McCallum (University of Massachusetts at Amherst).

5.2 Discourse Integration

A team of three or four FTEs should be assembled to research Discourse Integration and develop a DI tool. Such a team will require NLP expertise, knowledge of Discourse Integration, the ability to perform Systems Engineering, and connectivity with the users in the IOA Center.

More information about DI will come with a more profound literature study and by hiring and/or collaborating with people trained and experienced in the field. The direction of the LLNL DI system will be dependent on and influenced by the direction we go with the IE system.

5.3 Systems Engineering

Forming a Knowledge Extractor out of a free Syntax Parser, an Information Extractor built in collaboration with a partner, and a home-grown Discourse Integrator would require integration.

Fortunately, the output of the Syntax Parser is straightforward and easily shared with the other tools.

Because we would have the source code for the IE system, we could utilize the object libraries in the design and implementation of the DI system.

System Engineering of this project would consist primarily of insuring that the various components are written to make efficient use of the large volume of data, and to allow easy data exchange between parts. Algorithms need to scale well. Annotations may some day be performed without human involvement [5], but for now, human taggers would use active learning systems to make the most efficient use of their time, and strict tagging guidelines would be needed to insure compatibility between the human taggers and the codes. And of course the output of such a system would need to be ready to feed what the customers need it to feed; in the case of the Louisiana Project, that would be either the Graph Translation or the Canonicalization.

5.4 Political Concerns

A NLP R&D team will not be viable without adequate funding. The various sources of potential funding available to this project each have their own concerns and requirements.

It seems reasonable to expect a certain level of funding from the IOA Center, who would be the primary beneficiary of the project's effort. The IOA Center would want to see usable tools in place in the not-too-distant future.

DHS is requesting NLP technology; it may fund work at a higher level than the IOA Center through ASC. As a funder, ASC would be more interested in producing a research effort than tools development.

LDRD may fund the project at a later date. I expect LDRD to be interested in both the research component, and the technology development for the sake of other LLNL programs.

Finally, PNNL has funding for NLP, and they may be interested in spending some of that at LLNL in order to build a collaboration that meets their needs. Not enough is known yet about their needs, but such information will be forthcoming.

At this point in the process, the primary concern is the disparity between ASC's requirement for a research organization and IOA's need for tools. This does not seem to be an insurmountable barrier, as there is common ground. I believe both ASC and IOA would agree that their best interests are served by the success of the other.

Appendix A: Examples

A1 Examples of Morphology

The following list contains words whose morphological root is 'determin'. Morphology tools making use of the Porter Stemming algorithm taking such words as input, would return 'determin' as the stem.

```
determin ativeness
determin e
determin ed
determin edly
determin edness
determin er
determin ers
determin es
determin ing
```

```
determin ism
determin ist
determin istic
determin istically
determin ists
```

Source: http://www.scientificpsychic.com/paice/paice.html.

A2 Examples of Shallow and Deep Syntactic Processing

A shallow parse and a deep parse are shown below, given the following sample sentence: "Even though Pat had asked Terry to give the letter to Sam, Terry gave it to Taylor."

An example shallow parse is provided by LTCHUNK of Language Technology Group (LTG: http://www.ltg.ed.ac.uk):

<S>Even_RB though_IN [[Pat_NNP]]
((had_VBD asked_VBN)) [[Terry_NNP]]
((to_TO give_VB)) [[the_DT letter_NN]] to_TO
[[Sam_NNP]],_, [[Terry_NNP]] ((gave_VBD))
[[it_PRP]] to_TO [[Taylor_NNP]]._.</S>

where <S> and </S> denote sentence boundaries, double square brackets ([[and]]) encapsulate shallow noun phrases, double parentheses (((and))) encapsulate shallow verb phrases, and the underscore (_) separates each word in the input sample sentence from the part of speech tag provided by the parser.

A deep parse is provided by the Stanford Lexicalized Parser (http://nlp.stanford.edu/downloads/lex-parser.shtml):

```
(S
 (SBAR (RB Even) (IN though)
  (S
   (NP (NNP Pat))
   (VP (VBD had)
    (VP (VBN asked)
     (S
      (NP (NNP Terry))
      (VP (TO to)
       (VP (VB give)
        (NP (DT the) (NN letter))
        (PP (TO to)
         (NP (NNP Sam)))))))))))
 (, ,)
 (NP (NNP Terry))
```

```
(VP (VBD gave)
  (NP (PRP it))
  (PP (TO to)
    (NP (NNP Taylor))))
(. .))
```

Where parentheses encapsulate words, phrases, clauses, and sentences; S or SBAR denotes the beginning of a sentence or sub-sentence; NP, VP, and PP indicate noun phrases, verb phrases, and prepositional phrases respectively; and the other capitalized markings tag the part of speech.

Note that the deep parser nests phrases, an example of nesting is highlighted in blue. The deep parser has the advantage of providing a more precise parse which can be used in IE and DI. In practice, IE and DI generally make use of a shallow parse, because text is not always written in proper syntax, because the deep parse hogs processor time, and because it is more error prone than the shallow parse.

A3 Semantically Annotated (Tagged) Text

A brief sample of semantically annotated text:

<victim>15 people</victim> have been <care>admitted to hospital</care> in Kuzbass [Kemerovo Region] with suspected <condition>tickborne encephalitis</condition>, the <actor>Kemerovo Regional Administration's press service</actor> told RIA on Mon 17 May 2004, referring to the latest figures supplied by the <actor>State Health Inspectorate's Regional Centre</actor>. Over <victim>3000 local people</victim> have already sought <care>medical help</care> for tick bites. Out of that 3000, over 2000 have done so during the past week [mid May 2004].

The text is provided by Promed Mail. The color coded XML tags delimit the information of interest. This sample was annotated by a human (myself) but the goal of IE is for the software to learn to discern the information of interest, and be able to accurately annotate it with minimal user assistance.

A4 Ontology

Below is an example ontology in DAML format, which is built on top of RDF:

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<!DOCTYPE rdf:RDF [
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <!ENTITY rdfs "http://www.w3.org/2000/01/PR-rdf-schema-
19990303#">
   <!ENTITY daml 'http://www.daml.org/2001/03/daml+oil#'>
]>
```

```
<rdf:RDF
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns = "http://www.daml.org/2001/03/daml+oil#"
  xmlns:daml = "http://www.daml.org/2001/03/daml+oil#"
  xmlns:base =
"http://www.cs.umd.edu/projects/plus/DAML/onts/base1.0.daml#"
>

<Ontology rdf:about="">
   <versionInfo>Firpo-nonsense-ontology, v.0.1</versionInfo>
   <comment>My first shot at a DAML ontology.</comment>
</Ontology>

<!-- Classes -->

<Class rdf:ID="Party">
  <comment>Base party class</comment>
</Class>

<Class rdf:ID="Shower">
  <subClassOf rdf:resource="#Party"/>
</Class>

<Class rdf:ID="Reception">
  <subClassOf rdf:resource="#Party"/>
</Class>

<Class rdf:ID="Holiday">
  <subClassOf rdf:resource="#Party"/>
</Class>

<Class rdf:ID="HouseWarming">
  <subClassOf rdf:resource="#Party"/>
</Class>

<Class rdf:ID="Venue">
  <comment>address</comment>
</Class>

<Class rdf:ID="Guest">
  <comment>person attending the party</comment>
</Class>

<Class rdf:ID="Time">
  <comment>a point in time</comment>
</Class>

<Class rdf:ID="Honoree">
  <comment>honoree</comment>
  <subClassOf rdf:resource="#Guest"/>
```

```
</Class>

<Class rdf:ID="Date">
  <comment>date</comment>
</Class>


<!-- Relationships -->

<Property rdf:ID="on-date">
  <label>is on</label>
  <domain rdf:resource="#Party"/>
  <range rdf:resource="#Date"/>
</Property>

<Property rdf:ID="begin-time">
  <label>starts at</label>
  <domain rdf:resource="#Party"/>
  <domain rdf:resource="#Reception"/>
  <domain rdf:resource="#Shower"/>
  <domain rdf:resource="#Housewarming"/>
  <domain rdf:resource="#Holiday"/>
  <range rdf:resource="#Time"/>
</Property>

<Property rdf:ID="end-time">
  <label>ends at</label>
  <domain rdf:resource="#Party"/>
  <domain rdf:resource="#Reception"/>
  <domain rdf:resource="#Shower"/>
  <domain rdf:resource="#Housewarming"/>
  <domain rdf:resource="#Holiday"/>
  <range rdf:resource="#Time"/>
</Property>

<Property rdf:ID="at-venue">
  <label>is at</label>
  <domain rdf:resource="#Party"/>
  <domain rdf:resource="#Reception"/>
  <domain rdf:resource="#Shower"/>
  <domain rdf:resource="#Housewarming"/>
  <domain rdf:resource="#Holiday"/>
  <range rdf:resource="#Venue"/>
</Property>

<Property rdf:ID="honoree">
  <label>in honor of</label>
  <domain rdf:resource="#Party"/>
  <domain rdf:resource="#Reception"/>
  <domain rdf:resource="#Shower"/>
  <range rdf:resource="#Honoree"/>
</Property>
```

```
<Property rdf:ID="host">
  <label>throws</label>
  <domain rdf:resource="#Party"/>
  <domain rdf:resource="#Reception"/>
  <domain rdf:resource="#Shower"/>
  <domain rdf:resource="#Housewarming"/>
  <domain rdf:resource="#Holiday"/>
  <range rdf:resource="#Guest"/>
</Property>

<Property rdf:ID="attender">
  <label>attends</label>
  <domain rdf:resource="#Party"/>
  <domain rdf:resource="#Reception"/>
  <domain rdf:resource="#Shower"/>
  <domain rdf:resource="#Housewarming"/>
  <domain rdf:resource="#Holiday"/>
  <range rdf:resource="#Guest"/>
</Property>

</rdf:RDF>
```

Appendix B: Digest of My Cursory Literature Search

B.1 Information Extraction

The following are selected papers by the academics listed in section 3.7. The first four are authored by Ellen Riloff and tend to focus on descriptions of methods of IE that provide meaningful results with minimal user effort. The next four are authored by Andrew McCallum and tend to focus on the mathematical/probabilistic aspects of IE. The last one is by Fabio Ciravegna, and it explains in some detail the workings of his algorithm for generating rules for IE.

- Ellen Riloff and Mark Schmelzenbach [6] describe case frames for use in Event Extraction. A case frame recognizes a 'trigger-word' indicating an event of interest. It also has slots for various roles and expected syntactic patterns for filling the roles with data in proximity to an instance of the trigger-word in text. E.g.:
  ACCUSATION (active-verb "blamed")
  Accuser= subject
  Perpetrator = direct object
  Perpetrator = prepositional phrase (on)
  where "ACCUSATION" is the event type, "blamed" is the trigger-word, "Accuser" and "Perpetrator" are roles. "Accuser" is filled by the subject of the trigger-word, while "Perpetrator" can be filled by either a direct object, or by the object of a prepositional phrase beginning with the word "on."
  They go on to claim that for a person to determine these case frames is an intractable problem, but that the case frames can be determined automatically from

the corpus. They describe "AutoSlog-TS, a corpus-based system for generating extraction patterns automatically." Then they present their new algorithm that builds case frame structures from the extraction patterns.

- Ellen Riloff and Rosie Jones [7] describe IE as a process based on "a dictionary of extraction patterns and a semantic lexicon." They describe a process of mutual bootstrapping; beginning with a set of seed words in the initial semantic lexicon, they run AutoSlog to generate an initial set of extraction patterns, which are used to extract more terms, which are used to generate more extraction patterns, and so on. Such an approach is clearly vulnerable to early errors throwing the whole process off in the wrong direction. To defend against errors in the lexicon, a second level of bootstrapping is applied that "retains only the most reliable lexicon entries from the results of mutual bootstrapping and restarts the process with the enhanced semantic lexicon." The extraction patterns are checked after each iteration of the mutual bootstrapping to account for the new lexical entries. They reported precisions for five entity types ranging from 0.67 for terrorist location (after 30 iterations) to 0.92 for web locations (after 10 iterations). The other three entity types were optimized at 20 iterations. These numbers seem low by today's standards, but given that this paper was published in 1999 and that this process utilizes neither expert tuning nor semantic annotations, these numbers are fairly high. Although, no recall is reported.

- Ellen Riloff [5] introduces AutoSlog-TS as an adaptation of AutoSlog. First she describes AutoSlog as a tool for generating extraction patterns from semantically–tagged corpora. AutoSlog uses the pretagged data and a set of heuristic rules based on a shallow syntactic parse. She then touches on the challenges of human tagging of texts to introduce AutoSlog-TS for the purpose of generating extraction patterns from an untagged corpus. The corpus does need to be preclassified, that is, the documents need to be separated into two classes: relevant and not relevant. In the first pass of a two-pass process, the tool parses the entire corpus syntactically and exhaustively generates extraction patterns for every noun phrase in the corpus. These extraction patterns are used in the second pass, wherein "the sentence analyzer activates all patterns that are applicable in each sentence." The 'relevance rate' is determined, that is, the probability that a given term is relevant. The extraction patterns are then ranked in order of the relevance rate times the log base 2 of their frequency. In the results, the F1 measures for both AutoSlog and AutoSlog-TS were in the high 30s and 40s. She acknowledges that these are low and expects that a new ranking function will improve these numbers. In time, this has the potential to develop into a method for performing IE without human tagging.

- Rosie Jones, et al. (Riloff) [8] introduce a framework for active learning of cotraining problems. They describe cotraining as having to do with data wherein "each example can be described by multiple feature sets, any of which are sufficient to approximate the function." Cotraining makes use of redundant information. Active learning systems utilize bootstrapping techniques to annotate

unlabeled data, requesting user-input of text that the system cannot confidently tag. They look at noun phrases using two feature sets: the contents of the noun phrases, and the context of the noun phrases. The system is used to disambiguate three common types of ambiguities: general polysemy (a word with multiple senses), general terms (can refer to entities of various types), and proper name ambiguities (e.g. organizations are often named after people). They employed a method similar to the one in [7] above, starting with seed words and inducing rules and bootstrapping words from rules and rules from words. Various strategies were considered for determining texts for active learning user-tagging: uniform random selection, density selection, feature set disagreement, and context disagreement. They found that feature set disagreement was the most efficient strategy for all entities except "people classes," where density selection performed the best. By merging active learning with bootstrapping the two approaches made up for weaknesses in each other. Active learning improved the accuracy of bootstrapping, and bootstrapping reduces the amount of time the user spends tagging.

- Wei Li and Andrew McCallum [9] discuss semi-supervised learning in which both tagged and untagged data are used to train a learning system (including active learning systems). It is noted that including untagged text in the training data sometimes helps and sometimes hurts the training effort. They consider a model for learning from labeled data (a purely supervised model), and derive a Markov random field (MRF: which like CRF involves summing feature functions weighted by conditional likelihood). To make this model work with untagged data, they add features and weights that introduce a connection between the set of unlabeled text, and the set of labels found in the tagged text. The model is applied to a classification task, which is not a task of interest in our IE domain. However, one generalizable point is that a good distance metric is necessary in making semi-supervised learning work with untagged text. "In many models the distance metric is implicit and treated as a given, but the right distance metric is task-specific and should be learned."

- Ben Wellner, et al. (McCallum) [10] present a method for integrating Information Extraction with data mining applications so that each task can learn from the accomplishments of the other. In this context, coreference seems to refer more to 'inter-document coreference' than 'intra-document coreference' (the latter being of prior concern for LLNL's current needs). They introduce Conditional Random Fields (CRF) as an approach to both information extraction and coreference. CRFs have certain advantages over the popular Hidden Markov Model (HMM) approach in that the conditional probability requires less work than the joint probability inherent in HMMs, and that CRFs allow for a determination of confidence of the resultant label. Very briefly, a CRF can be computed as the product of a set of summations of the potential functions of associated feature sets and their weighting values. Because they use semi-structured data in their experiments, their reported accuracy is presently unachievable in free text (F1 consistently above 93%).

Conversations with Dr. Andrew McCallum [1] indicate that the method can be made to work with free-text as well.

- Fuchun Peng and Andrew McCallum [11] try out different techniques with CRFs. While they apply their work to semi-structured data (citations and headers of research papers), their work is instructive in the use of CRFs. After introducing CRFs and specifying them to the problem space, they introduce the practice of regularizing them to a prior distribution. They try a Gaussian prior, an Exponential prior, and a Hyperbolic-L1 prior. They also experiment with various feature sets, including: state transitions up to third order, unsupported features (features not represented in the training data, these are actually given negative weights), and what they call "local features, layout features, and lexicon features," or features derived from the training set. Finally, they compared methods utilizing CRFs with methods using HMMs and SVMs. The results, in brief were as follows: CRF outperformed HMM and SVM in 12 of 13 fields (SVM did best in extracting publication number), the Gaussian Prior outperformed both the Exponential and the Hyperbolic Priors (the Exponential actually added error), the second order state transition features performed better than first or third order (they suspect third order would do better with more data), and using unsupported features improved results. Finally, taking the use of local features as a baseline, "the layout feature dramatically increases the performance … adding lexicon features alone improves the performance. However, when combining lexicon features and layout features, the performance is worse than using layout features alone."

- Aron Culotta and Andrew McCallum [12] use CRF to perform IE and then estimate the confidence of their extractions. Again, this IE is performed on semi-structured text, but it is instructive to see the confidence measures with CRFs, which are useful in free-text IE. They introduce the CRF as applied to the present problem. They mention that the Viterbi algorithm determines the most likely state sequence given an observed input sequence. They discuss the Forward-Backward algorithm, claiming that it generalizes the Viterbi algorithm by evaluating all possible state sequences, not just the most probable. They show how to estimate the probability that a given field is extracted properly with a Constrained Forward-Backward algorithm (CFB). They form three confidence metrics: "FieldProduct calculates the confidence of each field in the record using CFB, then multiplies these values together to obtain the record confidence. FieldMin instead uses the minimum field confidence as the record confidence. ViterbiRatio uses the ratio of the probabilities of the top two Viterbi paths, capturing how much more likely s* is than its closest alternative," where s* is the most likely extraction.

- Fabio Ciravegna [13] presents an adaptive IE algorithm called $(LP)^2$ (Learning Pattern by Language Processing). It is designed to handle both text that is structured by nonlinguistic mechanisms present on the worldwide web, such as "HTML tags, document formatting, and stereotypical language," and free text (with an eye to sparse data as well). $(LP)^2$ generates tagging rules, contextual rules, and correction rules. An individual tagging rule inserts a single SGML tag (either a

start tag or an end tag) where the text matches the associated pattern. That is, it recognizes the beginning of a slot, or the end of a slot, but not the slot as a whole. The user provides pre-tagged text as training data. $(LP)^2$ takes the SGML tags in the data as positive examples, and the untagged portions of the data as negative examples. For each positive example, an initial rule is generated and generalized. Other positive examples covered by the new rule are removed from the pool of positive examples. This induction and generalization of rules continues until all positive examples are covered. These tagging rules tend to provide high precision, but not high recall. Improved recall is provided by contextual rules. Some of the rejected tagging rules are reconsidered. They are later accepted as contextual rules if they can solve some recall problem, such as the presence of an unmatched tag. Correction rules are based on the same logic as the tagging rules, and are used to fix problems on the slot boundary, shifting an inserted tag by one or more words in either direction. The patterns include elements such as word tokens, case, lemma (root), lexical category, and semantic category. "It is ... important on the one hand to generalize over the plain word surface of the training example in order to produce rules able to overcome data sparseness. On the other hand it is necessary to prune the resulting rule set in order to reduce over fitting." Rules are generalized by relaxing constraints in the pattern, and testing them on the training data, considering fit for both positive and negative examples. Some pruning takes place at the time of induction. Rules that do not cover a minimum number of cases are simply removed from the rule set, as are rules with a high error rate. Rules are also pruned according to coverage. If the cases covered by one rule are all covered by another rule, then the rules are redundant, and one of them is pruned. The algorithm was implemented in a program called Amilcare, and compared to five other IE tools, including WHISK, on a corpus of seminar announcements. It outperformed them all in overall F1 scores for the slots: speaker, location, start time, and end time.

B.2 Discourse Integration

A limited literature search uncovers the following architectural approaches to addressing the coreference problem:

- Kokar, et. al. [15] use an annotation scheme based on an OWL ontology, and then use an ontology consistency checker to determine whether two references refer to the same concept/instance.

- Shankaranarayana and Cyre [16] identify two types of references: an initial reference (which they term a ‘definition’) and a subsequent reference (the coreference). They provide a set of rules to determine whether a given reference is a definition or a coreference. These rules account for things like the nature of the determiners or quantifiers used, the types of head concepts, restrictions on relationships, and others.

- Andrei Popescu-Belis [4] provides an approach similar to Shankaranarayana and Cyre in that he looks sequentially at each referring expression (RE) in a document and compares it to the REs previously considered. Instead of giving the first RE to reference a real-world entity the status of a "definition," he creates a separate object called a discourse entity (DE) to which the RE refers. This is useful where the antecedent carries less information than the anaphor, for example: "In their last meeting, the division leaders discussed SQA." His rules are based on eight "knowledge sources" that take into account: DE-RE compatibility, RE-RE compatibility, attachment criteria (for attaching an RE to a DE), evolution of the list of characteristic REs (those REs that provide characteristic information about the DE), whether pronouns are included in the list of characteristic REs, and the salience of an RE. This method has a learning component for optimization, but the bulk of the code uses "analytical knowledge based considerations."

- Soon, Ng, and Lim [17] take a machine learning approach. As such, it requires training data in the form of a corpus tagged for coreference in the format used in the MUC-6 and MUC-7 conferences. In their words, "The learning algorithm used in our coreference engine is C5, which is an updated version of C4.5 … C5 is a commonly used decision tree learning algorithm and thus it may be considered as a baseline method against which other learning algorithms can be compared." The system passes the data through a seven-stage-pipeline of NLP tasks that include Noun Phrase Identification and Nested Noun Extraction (an example of a nested noun would be, "seat belt," or an appositive like "Mike Anastasio, Director of LLNL"). The extracted nouns are compared with the tagged data, and C5 learns a set of rules. These rules can then be applied to untagged data. While they did not participate in the MUC conferences, they ran their system on the MUC conference data. They achieved F1 scores of 62.6% on the MUC-6 data and 60.4% on the MUC-7 data. These numbers are fairly high compared with the state of the practice, but are still too low to be useful.

Appendix C: NLP Software Packages I Have Evaluated

Porter Stemmer: finds the stem or root of each word. This is important technology, and has been folded into all the syntactic processors.

Brill Tagger: a part of speech (POS) tagger. It appends each word with the appropriate POS tag. This was obviously a very important piece of work when it came out in the early 90s, and remains a classic today, though it has been folded into more complex syntactic processors, which also parse each sentence into its constituent phrases.

Marmot: another syntactic processor. One advantage of Marmot is that it is integrated with Badger (a semantic processor) and Badger is, to some extent, integrated with WHISK (IE). It is a good POS tagger, but it is difficult to use and is not maintained or supported. It also has lower accuracy than the Lexicalized Parser from Stanford.

Crystal: finds semantic extraction patterns from syntactically processed text. It trains Badger.

Badger: a semantic processor, which has the advantage of being integrated with Marmot, and to a lesser extent with WHISK. It is no longer maintained nor supported. As semantic processing evolves, Badger will not take advantage of the increased accuracy and performance. Also, since Marmot is less accurate than SLP, we would be using Badger without its intended partner, which would increase the complexity of use.

WHISK: Information Extractor. It is not clear from the documentation what form the input should take. What is clear is that it will take the output of Badger. This code was written in 1999, and is not maintained nor supported. Other IE codes are available that have been responding to the NLP literature over the past five years, and are actively supported.

CORA IE and HMM: CORA IE is simply a tool to support manual tagging of corpora for semantic or IE training. HMM is an IE code based on Hidden Markov Models. The difficult documentation gave little evidence how to call the program, and the examples show it working only on semi-structured text, not on free text. When I wrote to Andrew McCallum of the University of Massachusetts at Amherst for clarity on HMM, he referred me to MALLET.

MALLET: a set of Java classes that can be used by developers for use in developing NLP code. I decided it would be more efficient to continue evaluating available codes, than to write my own semantic processor or IE. MALLET does come with some executables, which are used for document classification.

MnM: an annotation tool, it allows the user to mark-up a corpus with semantic tags for use in IE. It supports a variety of IE plug-ins, including the Amilcare plug-in, which I used. MnM streamlined the process of tagging the training and testing corpora for use with Amilcare. Unfortunately, Amilcare and MnM did not integrate as well as I would have liked. Amilcare performed poorly in the context of MnM. I eventually downloaded a standalone version of Amilcare, and wrote a little utility to convert my tagged text from the MnM style tags to the Amilcare style.

Amilcare: a Relationship Extractor. Using this I learned the importance of consistency in tagging a corpus semantically. It requires a semantically–tagged training corpus for training, and a tagged corpus for testing. It can be adjusted to increase accuracy by editing the rules or modifying the thresholds. With a fairly small data set, I was getting 60 to 100% accuracy on concepts that had reasonable coverage, and 0 to 50% accuracy on concepts that were not well covered in the training set. These numbers are expected to improve for larger training sets.

Melita: an active learning tool for semi-supervised tagging. This code has a GUI client and an engine server. The engine is Amilcare. The user tags text in the GUI, while the server makes use of the users tags to suggest tags of its own. The more the user tags, the

more accurate the server becomes. Melita looks ahead to untagged documents in the corpus, and presents the user with those documents which would most improve its tagging. This is a good tool, but it has its limitations. It doesn't seem to handle nested tags, which can be important in tagging relationships and events. It also doesn't handle coreference. Sometimes its logic gets stuck in patterns where it suggests inappropriate annotations in large numbers, and it can be inconvenient to reject all the bad tags.

LTChunk: a very good shallow syntactic parser. It tags parts of speech, and identifies noun and verb phrases. There is no nesting of phrases, and it doesn't identify prepositional phrases.

Stanford Lexicalized Parser (SLP): SLP is my choice for syntactic processing. It uses two independent methods and merges the results to increase accuracy. It is actively maintained, which means there is someone who is interested in keeping it current with the latest technology. It also means that they provide e-mail support. They let me know what works and how, and they let me know about their plans for the future. SLP also does not explicitly differentiate between subject and object. This code does have an upper limit on the number of words it can handle per sentence. Out of the box, this limit was set to 40 words. The code slows down considerably when this limit is increased and the code is fed longer sentences. The FBIS data, most of which is translated to English from other languages, many of which typically have longer sentences than English, had some very long sentences, some longer than 200 words. I found it best to set the limit to 200 words. In this way, three of the 1000 files had sentences that were too long, and required some massaging. Of these three, one had a sentence longer than 200 words, and the others simply had large tables with no periods. I was able to manually insert periods into these long tables in an intelligible way, and break the one long sentence into two without altering the meaning. Three files per thousand seems like a reasonable number of files to receive user attention.

TextToOnto: this code has a powerful ontology graphing capability. As the name implies, it can be used to develop an ontology from the text. The user can also provide an ontology in KAON format (a derivative of RDF), or build one graphically. These three methods can be combined. It seems to me that the primary strength of this tool is to provide maximum flexibility in working with the ontology. The ability to extract ontological information from the text is billed as a way to tune an ontology to the specific data. TextToOnto is being maintained and is supported. The documentation does not go into much detail about how specifically to use the code.

MinorThird: an Information Extractor. It works best with entities (as do most). Its author, William Cohen (http://www-2.cs.cmu.edu/~wcohen/), is respected in the field. I found the documentation confusing. It came with data (Enron email) that had been tagged for human names. I was very disappointed in my initial results with this code on a simple Entity Extraction task. I wrote to the author, who told me how to improve its performance. In the end, I only got it to perform slightly better than Amilcare on this task, though Amilcare is written for relationships, not entities.

References

1. McCallum, Andrew. Conversation held between Andrew McCallum, Anton Fulmen, Sarah Tyler, and Michael Firpo, November 16, 2004.

2. Klein, Dan, and Christopher Manning, "Fast Exact Inference with a Factored Model for Natural Language Parsing," *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, December 2002.

3. Feldman, Ronen, "Mining Unstructured Data, Tutorial Notes," *10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Seattle, WA, August 22–25, 2004.

4. Popescu-Belis, Andrei, "Evaluation-Driven Design of a Robust Coreference Resolution System," *Natural Language Engineering* **9** (3): 281–306, 2003, Cambridge University Press.

5. Riloff, Ellen, "Automatically Generating Extraction Patterns from Untagged Text," *Proc. 13th National Conf. on Artificial Intelligence (AAAI-96)*, pp. 1044–1049, 1996.

6. Riloff, Ellen, and Mark Schmelzenbach, "An Empirical Approach to Conceptual Case Frame Acquisition," *Proc. 6th Workshop on Very Large Corpora*, 1998.

7. Riloff, Ellen, and Rosie Jones, "Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping," *Proc. 16th National Conf. on Artificial Intelligence (AAAI-99)*, 1999.

8. Jones, Rosie, Rayid Ghani, Tom Mitchell, and Ellen Riloff, "Active Learning for Information Extraction with Multiple View Feature Sets," *20th Int. Conf. on Machine Learning (ICML 2003)*, Washington, D.C., August 21–24, 2003.

9. Li, Wei, and Andrew McCallum, "A Note on Semi-Supervised Learning using Markov Random Fields," Techical Note, February 3, 2004.

10. Wellner, Ben, Andrew McCallum, Fuchun Peng, and Michael Hay, "An Integrated, Conditional Model of Information Extraction and Coreference with Application to Citation Matching," *Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2004.

11. Peng, Fuchun, and Andrew McCallum, "Accurate Information Extraction from Research Papers using Conditional Random Fields," *Proc. Human Language Technology Conf. and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.

12. Culotta, Aron, and Andrew McCallum, "Confidence Estimation for Information Extraction," *Proc. Human Language Technology Conf. and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2004.

13. Ciravegna, Fabio, "$(LP)^2$, an Adaptive Algorithm for Information Extraction from Web-Related Texts," *Proc. IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the 17th Int. Conf. on Artificial Intelligence (IJCAI-01)*, Seattle, WA, August, 2001.

14. Ciravegna, Fabio, "Adaptive Information Extraction from Text by Rule Induction and Generalisation," *Proc. 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001),* Seattle, WA, 2001.

15. Kokar, M. M., C. J. Matheus, J. A. Letkowski, K. Baclawski, and P. Kogut, "Association in Level 2 Fusion," *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, pp. 228–237 SPIE, 2004.

16. Shankaranarayanan, S., and W. R. Cyre, "Coreference Detection in Automatic Analysis of Specifications," *Proc. 1994 Int. Conf. on Conceptual Structures*, College Park, MD, August 16–20, 1994.

17. Soon, W.M., H.T. Ng, and D.C.Y. Lim, "A Machine Learning Approach to Coreference Resolution of Noun Phrases," *Computational Linguistics* **27** (4), pp. 521–544, 2001.