PROGRESS IN INFORMATION EXTRACTION

Ralph Weischedel, Sean Boisen, Daniel Bikel, Robert Bobrow, Michael Crystal, William Ferguson, Allan Wechsler and the PLUM Research Group
BBN Systems and Technologies
70 Fawcett Street
Cambridge, MA 02138
weischedel@bbn.com
617-873-3496

0. ABSTRACT

This paper provides a quick summary of the following topics: enhancements to the PLUM information extraction engine, what we learned from MUC-6 (the Sixth Message Understanding Conference), the results of an experiment on merging templates from two different information extraction engines, a learning technique for named entity recognition, and towards information extraction from speech.

1. ENHANCEMENTS TO THE PLUM INFORMATION EXTRACTION ENGINE

1.1 What's New

Recently BBN has ported part or all of the PLUM system to new languages (Chinese and Spanish) and new domains (name finding, heterogeneous newswire sources, and labor negotiations). Coupled with this there have been several improvements which are described next.

1.1.1 Infrastructure

The message reader was improved. As before, it takes declarative finite state descriptions (regular expressions) defining document structures. We added a general parser for SGML to simplify text descriptions of documents that contain SGML markup. The overall impact has been more robust handling of documents with SGML and less effort to specify this format.

In MUC-5 and MUC-6 the output of information extraction has been a multi-level object-oriented data structure. Although appropriate for an object-oriented data base, the structures frequently were not straightforwardly mappable from linguistic structures. This was one of the reasons that prior to MUC-6, semantic inferencing was added to PLUM's discourse processor. The discourse processor, when invoked on a sentence, first adds all semantic predicates associated with a sentence to the semantic database. Reference resolution is performed, and the relevant semantic objects are unified in the database as a result. At this point, a set of inference rules are applied, and any that succeed will add more predicates to the database.

The semantic database is used as the repository of semantic information for all the objects mentioned in a message. It is queried frequently by the discourse processor. The semantic inference component performs two primary functions:

- Perform inferences to normalize different but equivalent semantic representations present in the database, where the differences may have stemmed from syntactic variations, or from incomplete knowledge at the time they were generated; and
- Generate more complex representations (closer to the template structure) from simpler, "flatter" semantics (closer to the linguistic structure).

In the ST task of MUC-6, the inference module was used, for example, to try to infer the corporations involved in a succession event. We represented each instance where a job position is mentioned as a JOB-SITUATION semantic object. JOB-SITUATIONs are a flattened version of the SUCCESSION and IN-AND-OUT objects. An inference rule queried the semantic database to determine whether the JOB-SITUATION PERSON was involved in multiple JOB-SITUATIONs, if so, it would add predicates for the JOB-SITUATION's OTHER-ORG and REL-OTHER-ORG.

1.1.2 Capitalizing on lightweight processes

We have begun making a distinction between *lightweight* techniques and *heavyweight* processing. Lightweight techniques are those that rely only on local processing, do not involve deep understanding, and can be optimized. Example lightweight techniques are for SGML recognition, hidden Markov models, finite state pattern recognition, text classifiers, text indexing and retrieval and SGML output. BBN's IdentiFinder™ (Figure 1) is made up solely of lightweight algorithms, and was entered in the MUC-6 Named Entity task (recognizing named organizations, named persons, named locations, monetary amounts, dates, times, etc.).

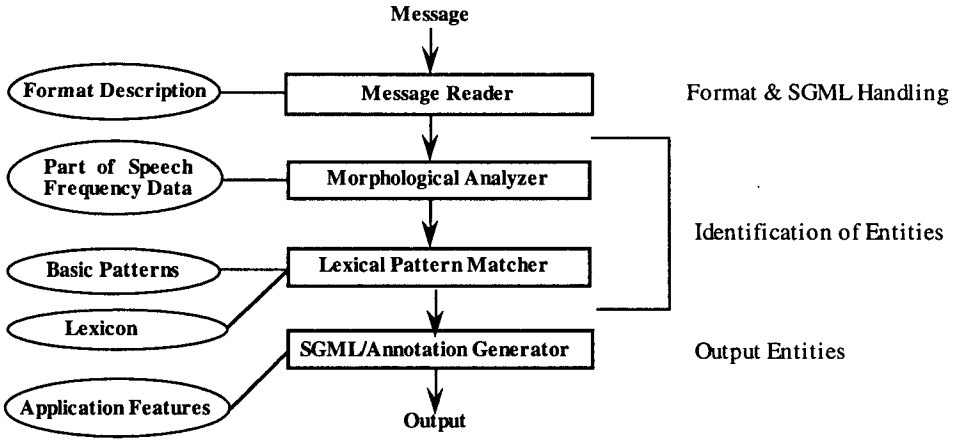

Figure 1: IdentiFinder System Architecture: *Rectangles represent domain-independent, language-independent algorithms; ovals represent knowledge bases*

Heavyweight processing includes procedures that depend on global evidence, involve deeper understanding, and are research oriented. Full parsers, semantic inference, and co-reference algorithms are examples. PLUM (Figure 2) employs both lightweight and heavyweight techniques. PLUM was employed in both MUC-6 template tasks (TE and ST).

Two new heavyweight algorithms were developed in the last year. One is a full parser of English, using a statistically learned decision procedure; SPATTER has achieved the highest scores yet reported on parsing English text (Magerman, 1995). Because the measurable improvement in parsing is so great (compared to manually constructed parsers), it appears to offer a qualitatively better parser. We are looking

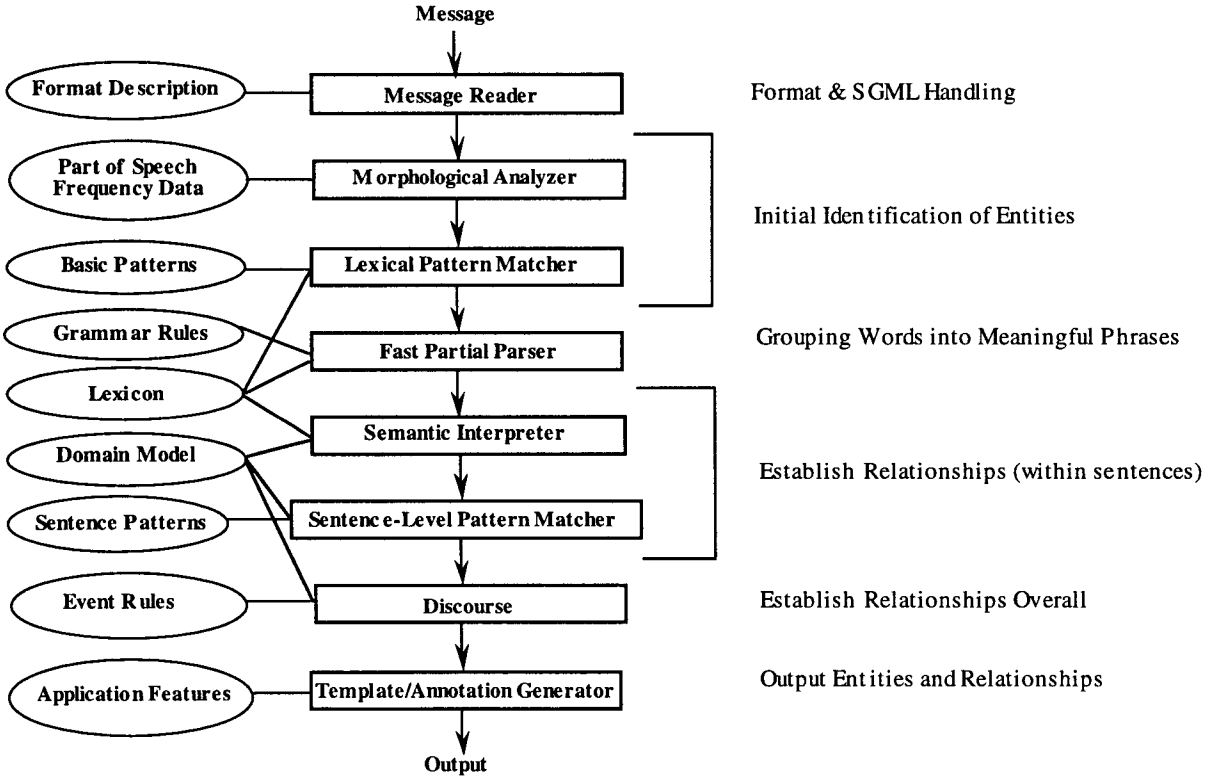

Figure 2: PLUM System Architecture: *Rectangles represent domain-independent, language-independent algorithms; ovals represent knowledge bases.*

forward to trying it on information extraction problems. The second is the semantic inference procedure discussed in the previous section.

1.1.3 Domain-independent systems

The NE configuration of IdentiFinder is stand-alone and domain-independent; no information regarding succession of corporate officers is employed. Similarly, the MUC-6 TE (properties of organizations and persons) configuration of PLUM uses no information regarding succession of corporate officers, and therefore can be used on other domains. We believe that development of other broadly applicable information extraction functionality such as NE and TE will be a win, maximizing the value of defining reusable knowledge bases for information extraction.

1.2 Tools for Porting and Maintaining the PLUM Information Extraction System

Customizing extraction systems to a given application domain is a difficult process. It is time consuming and requires highly specialized skills. BBN has developed the NLU Shell, which provides tools that simplify many of the tasks associated with porting information extraction to a new domain, and support maintenance once the port is complete. Such a tool kit would be unimportant if we did not anticipate the need for creating many extraction applications. In fact, however, message processing systems typically must deal with a large number of distinct applications.

The NLU Shell software is a collection of independent editors, some tools for managing a set of factbases, and a compiler for compiling the factbases into an NLU Application. Its major components are:

- *Top Level Executive (TLX):* Primarily a human interface tool for invoking the other NLU Shell components.
- *Text Description Editor (TDE):* Maintains the Text Description Factbase, a data file containing knowledge about the structure of messages, i.e. where word, sentence, and paragraph boundaries fall.
- *Output Record Editor (ORE):* Maintains the Output Record Factbase, a data file containing knowledge about how the NLU Application Output should be formatted.
- *Semantic Model Editor (SME):* Maintains the Semantic Model Factbase, a data file containing the concepts which the NLU Application uses to represent the information contained within messages.
- *Part of Speech Editor (POSE):* Maintains the part of speech models that are used to determine word categories in the face of ambiguity (e.g. is "hit" a verb or a noun). Generates a new word list of tagged words not found in the lexicon.
- *New Word Editor (NWE):* Updates the dictionary of words (lexicon) to be recognized by the NLU Application. Enters new words found by POSE into the lexicon.
- *Lexicon Editor (LE):* Maintains the dictionary of words (lexicon) to be recognized by the NLU Application. Adds and edits word meanings based on concepts and their attributes in the semantic model.
- *Case Frame Editor (CFE):* Maintains the lexical Case Frame Factbase, a data file of how to infer information by analyzing how verbs relate to logical sentence elements (e.g. subjects or objects) and to the objects of their prepositions.
- *Fill Rule Editor (FRE):* Maintains the Trigger Rule Factbase, a data file containing the knowledge describing when output records should be created given the existence of semantic concepts. Also, maintains the Fill Rule Factbase, a data file containing knowledge describing how to fill fields of output records when they are generated by a trigger in the Trigger Rule Factbase.
- *Consistency Checkers (CC's):* Report inconsistencies within and between factbases and, also, report states of the factbases that suggest areas for further work (e.g. output records that can never be created). There is one consistency checker for every factbase.
- *Application Generator (AG):* Compiles the different factbases into an NLU Application.

1.2.1 Evaluation procedure

A programmer with no computational linguistics background performed a usability evaluation of the NLU Shell toolkit. The evaluation exercised only three of the system's eight editors. First, the Fill Rule Editor was used to attempt to improve performance on an existing application. Second, the New Word Editor and the Lexicon Editor were used to add a list of new words to the application's dictionary. The results of the evaluation follow.

1.2.2 System strengths

The NLU Shell provides a way for non-programmers to build and maintain language processing applications. The use of a GUI and a database in place of files of source code and data (which must be edited as text) represents a fundamental advance in making natural

language technology widely available. While the NLU Shell users need detailed familiarity with the processes and knowledge needed for extracting formatted data from natural language, they do not need to be programmers. This is not only because the need to know a specific programming language is eliminated; the greater benefit of using NLU Shell is that the deep understanding of natural language processing algorithms and heuristics that went into the creation of the NLU Shell is not required for its use. This constitutes a qualitative change in the construction and use of language processing software.

On a more mundane level the NLU Shell GUI is structured as a set of cascading diagrams depicting the underlying data extraction process. Thus, the user is guided through the application building process. The use of a hierarchy of access screens keeps each one simple.

There is design and implementation level documentation for the system as well as a comprehensive user's manual.

1.2.3 Areas for Improvement

During the evaluation several areas needing improvement in the NLU system emerged. These are divided into three categories: user interface, system speed and missing features. The interface is from cascading menus, which allowed the initial screen to be simple, at times awkward to use when navigating among the tools. It was often necessary to move from one tool to another to complete an operation (e.g. using the semantic model editor during word definition to pick out the right meaning for a word). The need to pop up additional windows repeatedly made editing multiple entries time consuming. Common processes can be further streamlined in the interface.

Compiling the complete NLU Shell application in order to test changes is time consuming. Adding incremental compilation functionality would improve this.

A handful of missing features were identified, such as the need to "find" words in the lexicon rather then scrolling. An available filtering mechanism is valuable but does not replace a "find" function.

1.2.4 Conclusion

To the extent covered by this evaluation, the NLU Shell is a successful first prototype of a GUI-based natural language processing application builder. It enables aspects of the maintenance and construction of such systems to be performed without knowledge of specific programming languages and environments. Further work would make it even more effective.

In the test conducted here, vocabulary was successfully added and slot filling performance improved, using only the NLU Shell tools, however the process could be improved. With the improvements mentioned above, plus a more extensive review of the system for other enhancements, the NLU Shell could significantly reduce the time and effort needed to build a natural language processing application and make this process available to knowledge engineers who are not programmers as well.

2. LESSONS FROM MUC-6

2.1 Participation in MUC-6

For MUC-6, there were three different application dimensions that one could choose to participate in, as follows:

- Named Entity (NE), recognition of named organizations, named persons, dates and times, monetary amounts, and percentages. The task in principle is domain-independent, though in MUC-6 it was evaluated only on documents obtained by a query for documents about change in corporate officers.

- Template Element (TE), extraction of organizations, persons, and properties of them. For instance, organizations have full names, aliases, descriptions, locations, and types (company, government, or other). Persons have full names, aliases, and titles. Like NE, this really is domain-independent, though in MUC-6 it was evaluated only on documents obtained by a query for documents about change in corporate officers.

- Scenario Template (ST), a domain-specific, full template extraction task. For MUC-6, the task was change in corporate officers. This included three levels of objects; the lowest level of objects is identical to the TE objects. Information extracted above that level included the individual, the position they were taking on (if any), the corporation in which the position is held, the position they were leaving (if reported), the corporation of the former position, the reason for the change (e.g., retirement), whether the holder was acting in that position or not, etc.

One approach to the three tasks is to have a full information extraction system apply all its knowledge to all three tasks, simply providing three different levels of output. Certainly that would appear to offer the highest scores, since even the domain-independent tasks were evaluated only on domain-dependent data. Many groups chose exactly that route.

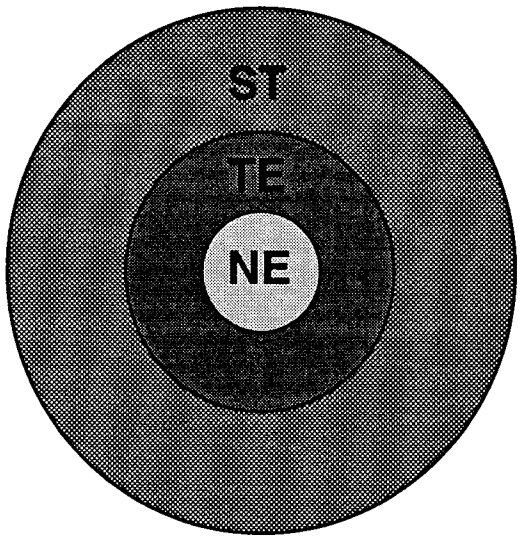

Figure 3: Relation of BBN Systems in MUC-6

However, we chose a path represented in Figure 3 below so that there would be domain-independent versions of the software for efforts other than MUC, such as the TIPSTER program. The more complex systems are built on top of the simpler systems in order to minimize duplication of effort and maximize knowledge transfer. The NE task is the simplest task and makes use of only lightweight processes, the first three modules of the PLUM system (the message reader, the morphological analyzer, and the lexical pattern matcher, which together form IdentiFinder™). This NE system is represented at the core of Figure 3, since it was the core for all three tasks.

The TE task takes the entity names found by the NE system, and merges multiple references to the same entity using syntactic and semantic information. Since the components are the same in the TE and ST system configurations, and since the knowledge bases of TE are inherited by ST, we have represented this in Figure 3 via making TE the next innermost circle. By this organization, the knowledge bases of TE do not include domain-specific knowledge, and domain-specific knowledge is localized only in ST.

As a consequence, we have a domain-independent low-level template extraction application in TE.

2.2 Lessons Learned

The most exciting lesson we learned is that near human performance in named entity recognition is within the state of the art for mixed case English. Several systems performed at 90% or above. Since MUC-6 we have improved IdentiFinder's prediction of aliases once a name has been seen and added rules for low frequency cases, e.g., for names that are quite unlike Western European names. Our NE and TE systems employ no domain-specific knowledge by design. They should therefore work well on other text, not specific to change in corporate officers.

Compared to PLUM's previous performance in MUC-3, -4, and -5, our progress was much more rapid and our official score was higher than in any previous template fill task. Figure 4 shows progress in porting PLUM to ST, evaluating periodically on blind test material.

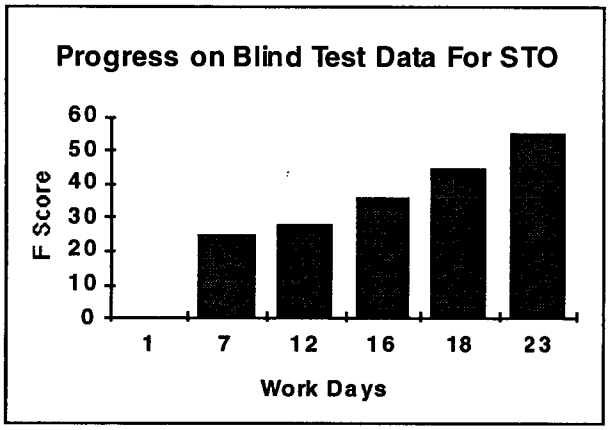

Figure 4: Measured Progress on the New Domain.

We make two general observations. First, the state of the art has progressed greatly in portability in the last four years. For MUC-3, some high performing groups invested a small number of person years. By contrast, several groups this year achieved an F in the 50s in 30 calendar days.

We suspect there are other basic objects that would be broadly applicable (as TE seems to be) and that there may be many generically useful event types and relationships that can be defined. This should broaden the utility of information extraction technology and cut its cost.

In retrospect, lightweight processes carried most of the burden in our TE system. We believe that with a little more effort, we could achieve an F score of 80 in TE with only lightweight techniques. It would be valuable to see how far only lightweight techniques can go, not just on TE, but on other tasks of extracting basic objects and direct relationships between them.

Yet, we believe that we are only beginning to understand techniques for learning domain-independent knowledge and domain-dependent knowledge. Far more

can be achieved. BBN particularly would like to investigate how statistical algorithms over large unmarked corpora can effectively extrapolate from a few training examples, such as in ST in MUC-6, to provide greater coverage. For example, statistical techniques may have suggested the importance of "hire," a verb which many groups did not happen to define.

Second, since there has been a marked improvement in the quality of full parsers, now achieving an F in the high 80s (Magerman, 1995), we believe it is now feasible to consider using full parsers again. The rationale is straightforward: for full templates (e.g., ST) scores have been mired with an F in the 50s ever since MUC-3 in 1991. Pattern matching has given us very robust, very portable technology, but has not broken the performance barrier all systems have run up against. Recent (statistical) full parsers (e.g., BBN's, IBM's, and UPenn's) have such quantitatively better performance that they are qualitatively better. We believe this offers the opportunity to again try heavyweight techniques to attempt deeper understanding. Pattern matching techniques will still have a crucial role for domain-specific details, but we believe overall improvement will be achieved by deeper understanding.

3. MERGING EXTRACTION OUTPUT FROM TWO SYSTEMS

The parallel structure of the MUC competition, with many different systems attempting to extract the same data from the same material, creates an opportunity to achieve even greater performance by combining the results of two or more systems. A combined system could use the outputs of several different systems to produce performance levels potentially superior to those of any one component alone. BBN has produced an architecture for system output combination.

The success of a data extraction system is judged by how well its results fit a hand-coded target set (the key) of structured template objects. For our target stories we selected the MUC-5 story set — one hundred English newspaper articles dealing with joint ventures.

Our combined system was based on the output of our own BBN PLUM system, configured for EJV, and the output of the Shogun system developed at GE, and made available to us through Lockheed-Martin. Combining the outputs of the PLUM and Shogun systems appeared promising because the two systems had different performance profiles in terms of precision and recall, but similar F-scores. Shogun had the better recall score — it extracted a higher proportion of the information contained in the key templates. PLUM had the better precision score — it extracted less spurious and incorrect information. Overall, Shogun had the higher F-score because it was optimized for maximal F while PLUM had been optimized for a mix of high F and low errors. We hoped that by combining the two systems we could produce a system with a high enough recall and precision to yield a higher F-score than either system could achieve alone.

In the hope that our results could be applied to combining any two template generating systems, we ignored the methods and mechanisms employed by the two extraction programs, and used only their output. For each story, this output consists of a set of interconnected frames specifying the content of the joint ventures as described in the source text. Each system produces an overall structure for each input story. This template organizes a set of joint-venture frames called Tie-Up-Relationships (or TURs), one for each joint venture mentioned in the story. These TUR frames, in turn, organize many smaller frames that describe the organizations, people, places, and activities involved in the joint venture.

3.1 Effort

Our first step was to massage the data from the PLUM and Shogun systems into a form that could be read into a merging program. In parallel with this data conversion we built an initial version of a data-combining program. This initial system matched the whole templates produced by the two systems for each story. It aligned the joint-venture frames and other frames from each system and then used various heuristics to combine the outputs frame by frame. This initial approach proved unwieldy as we had to simultaneously deal with the problems of aligning output from the two systems and merging that output. Additionally, we realized that we needed to gather more data to begin to learn about what each system was doing well compared to the other.

Useful dissimilarities in the detailed performance of the two systems proved hard to identify. We examined the score reports for PLUM and Shogun and found that their performance in almost every category matched their overall performance. PLUM had higher precision, while Shogun had better recall. Shogun's F-score was always slightly higher. This similarity rules out strategies that take advantage of contexts in which one system is superior to the other. [1]

We decided to simplify the combining task by using the scoring program (which had been used to test the PLUM and Shogun systems before and during the MUC-5 competition) to match the PLUM and Shogun frames.

[1] This similarity in performance profiles may indicate a similarity in the underlying methodology of the two systems.

In order to get a sense of how well we could potentially perform by treating the overall system-combining task as one of choosing templates, we performed an experiment as follows: we combined all the frames from both systems and scored the combined result against the key. We then took only the frames from this combined result that matched frames in the answer key, and we scored these selected frames as if they were the result of some ideal combining system. This score represents an upper bound on how well we could possibly do if our combining strategies allowed us to choose exactly the best frames provided by each system. This experiment showed a large improvement, since the no spurious objects would be produced, only the occasional spurious slot.

Our first step was to score the PLUM system against the Shogun system as if the Shogun system were the key. The mappings produced by this scoring were then used by filtering heuristics. Since the scorer's mapping was intended to maximize overall F-score, the alignment it produced was well suited to our purposes. Our first heuristic (and one of the best we found) was to take only the frames from the system with the best recall (Shogun) that also matched frames from the system with the best precision (PLUM). This subset of Shogun frames had a higher precision than Shogun's output alone, but its recall was low enough that its F-score was also worse than that of Shogun alone. The "opposite" strategy, taking all the Shogun frames and adding those PLUM frames that did not match Shogun frames, improved recall somewhat but lost on precision and resulted in a decrease in overall f-score.

We tried many variations on this theme for matching PLUM/Shogun frames, as well as combining various matching approaches with the use of simple statistical methods on individual frames to judge their likelihood of matching the key. While we uncovered many filters that had some predictive value, none of the tests we devised was of high enough quality to allow us to raise F-scores for the combined system over those of Shogun alone.

Our results indicated that the matching process used by the scorer was sensitive to perturbations of the internal structure of the joint-venture frames. The matching processes for related frames were interdependent, so that the removal of a "good" frame often caused other frames which pointed to it to fail to match. This occurred because these other frames relied on the match provided by the good frame in order to be matched to the key themselves. Thus, even though we found fairly good strategies for eliminating bad frames, the damage done by eliminating even a few good frames more than outweighed the benefit of eliminating many bad frames. To circumvent this difficulty, we decided to try using strategies that would select among whole joint-venture structures rather than selecting individual frames.

We tried another experiment in which we selected the Shogun TURs (entire joint-venture descriptions) having the highest percentages of individual frames that matched the key. Scoring just these templates produced a combined result with a better F-score than that of Shogun alone, though not nearly so good as the score for choosing just the right individual frames in the previous experiment. Thus, it would be possible to get an improved F-score by selecting TURs if we had a good enough selection heuristic or filter. The best filter we found was simply to select those Shogun templates that had the highest percentage of frame matches to the frames produced by EJV. By varying the acceptance threshold, we hoped to find a level at which we would get enough increase in precision to offset the decrease in recall. The results are graphed below in Figure 5.

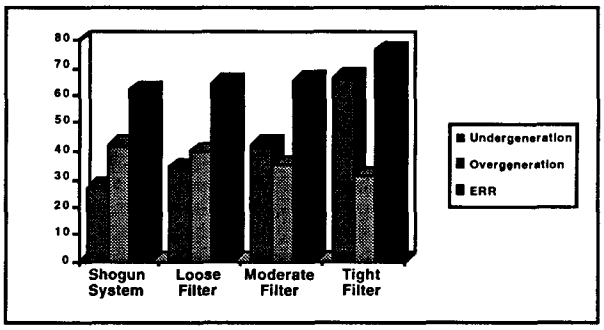

Figure 5: Trade-off in undergeneration and overgeneration in the combined system.

This graph refers to three measures: 1) under-generation (UG), 2) over-generation (OG) and 3) error (ERR). These are the values produced by the MUC-5 scoring system. All three are expressed as percentages. UG varies inversely with recall; OG varies inversely with precision; and ERR varies inversely with F-score. These inverse relationships are not linear, but this will not matter to the arguments presented here.

As the graph shows, the lowest error for the combined system occurs when the filter is loosened all the way and all Shogun frames are used. Undergeneration rises at the other points, but at each point it more than offsets the gain in overgeneration. As a result, ERR rises (usually) slightly in each case.

3.2 Conclusions

The template merging experiment provided a substantial range in recall versus precision, i.e., in undergeneration versus overgeneration. This is noteworthy itself.

Nevertheless, we did not achieve a breakthrough in overall F-score, i.e., in ERR. There are several potential contributing factors:

- The merging occurred after output was produced; perhaps results would have been better by combining results earlier in linguistic processing.
- The systems in our experiment perhaps produce too similar results.
- The scoring mechanism suffers from the linchpin phenomenon. In flatter template structure without the linchpin phenomenon, the penalty for a mistake in merging templates would be less severe.

One other possibility to investigate is to use the two systems in parallel (take anything that either produces) or in series (take only what both accept). The parallel system increases F-score only if the two systems have much better precision than recall, while the series case yields improvement only if the two systems have much better recall than precision. These two systems fell into neither category.

4. A LEARNING TECHNIQUE FOR NAMED ENTITY RECOGNITION

Like several other groups, we are pioneering research in automatically learning to extract information based on examples. Unlike other groups which have focused on case-based reasoning or on binary decision trees, we are focusing on statistical learning techniques.

4.1 The Application

The first extraction problem that we are tackling is learning to identify entities. Entity recognition and identification has been recognized by the community as a core problem and was evaluated in MUC-6 in November, 1995 for English and was also evaluated for Chinese, Japanese, and Spanish in the recently held Multi-lingual Entity Task (MET). An example for English is shown in Figure 6.

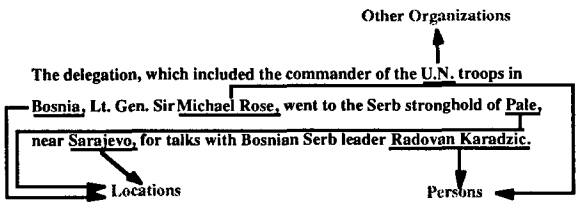

Figure 6: Named Entity Task. *An example of the named entity task for English.*

Nothing about our approach is restricted to the named entity task; other kinds of data could be spotted with similar techniques, such as product names, addresses, core noun phrases, verb groups and military units.

Previous approaches such as our own IdentiFinder system described earlier in this paper and evaluated in MUC-6, have used manually constructed finite state patterns. For every new language and every new class of new information to spot, one has to write a new set of rules to cover the new language and to cover the new class of information. Though the technique has clearly resulted in systems with very high performance and very high speed and has also led to commercial products, we believe that the technology based on learning is highly desirable for the following reasons:

1. Freeing a group's best people from manually writing such rules and maintaining them is a better use of the time of highly gifted people.
2. A learned system may be brought up with far less effort, since both manually constructed rules and learned systems assume a substantial collection of annotated textual examples.
3. In the ideal, the only knowledge of the language required for the learned system would be the examples of correct output. Therefore, once a linguist had defined the guidelines for correct output, potentially less sophisticated, less trained speakers of the language could develop the answer keys.

4.2 Approach

Our approach is represented in Figure 7. As the diagram shows, there are four requirements to our approach.

1. A set of training data must be provided; training data consists of sentences and annotations that represent correct output, i.e., an answer key.
2. A model must be defined that states the mapping from words to the annotations.
3. A training algorithm or module must estimate the parameters of the probability model to be learned from the example data, that is, from the sentences with their correct annotations.
4. A recognition algorithm or module must apply the probabilistic, learned model to new sentences to provide their annotations.

Under separate DARPA funding, we have applied this approach to the NE problem for English and for Spanish. Government supplied data in MUC-6 and in MET serve as the training data. The probabilistic model employed is in the general class of hidden

Markov models (HMM), though the HMM used currently is more complex than those traditionally used in speech recognition and in part-of-speech tagging. Since the SGML-marked text can be straightforwardly aligned with the answer keys, the training algorithm simply counts the frequency of events and normalizes with respect to the event class to estimate the parameters of the model. For the recognition algorithm, the Viterbi algorithm, which is typically used in hidden Markov models, is applied.

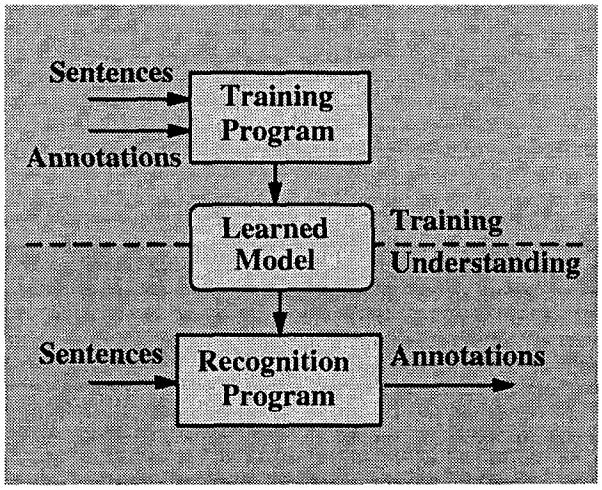

Figure 7: Components of a Learning Approach

The details of the probabilistic model will be documented separately.

4.3 Experiments

The training module and recognition module were first tested on English in early March. Scores on test material in preliminary testing have ranged between 87 and 89. Using the same language model and nothing specific to Spanish other than Spanish training examples, we are achieving scores even higher than in English.

These preliminary results are quite encouraging, since they are better than any previously reported scores for a learned system and since they are approaching the scores of the state-of-the-art for manually built, rule based systems. We would like to test the techniques on quite different languages, such as Chinese where performance for manually based systems is lagging behind systems in English and Spanish. We would also like to try languages where new word formation poses problems, such as in German.

The current version assumes three language-specific features: (1) that the system is provided with words as input (i.e., we have not tackled word segmentation), (2) that sentence boundaries have been found and (3) that simple, deterministic computations on words to determine word-features can be performed, to distinguish different sorts of numbers—which *are* largely language-independent—and different sorts of capitalization, in the case of Roman languages. As to segmentation, other groups such as the University of Massachusetts have explored probabilistic techniques for segmenting Chinese; Xerox has reported good results on learning to predict sentence boundaries from example data.

5. TOWARDS INFORMATION EXTRACTION FROM SPEECH

A pioneering effort started under DARPA funding (though not part of the TIPSTER program) is the goal of information extraction from speech. A goal is to identify objectives appropriate for future research and development. For example, can simple template information (e.g., who did what to whom, where, and when) be reliably extracted? Is name finding including recognizing unknown names feasible? Can a system reliably identify the topic of a speech segment? A second goal is to explore the architecture for such a system.

5.1 The Challenge

Schematically, the architecture being investigated is given in Figure 8 below; a system with this architecture has already been demonstrated. An audio signal is received from radio, television, or telephone. State-of-the-art large vocabulary continuous speech recognition (LVCSR) technology automatically transcribes speech.

The output of an LVCSR may be a single transcription (1-best), the n highest scoring transcriptions (n-best), or a chart of high scoring alternatives at each point. To date, we have only investigated 1-best.

As a preliminary experiment, we provided the single best transcription to PLUM configured for ST (the MUC-6 domain on succession of corporate officers) in order to determine the kinds of problems that speech input would pose. Consider the example below, where the source text is presented first. The second text is the result of automatic transcription. The following problems are evident:

- Lack of punctuation
- Lack of reliable mixed case to signal names
- Transcription errors when input is outside the 45,000 word vocabulary, which is problematic for infrequent names.

Text

Diane Creel, recently promoted to chief executive officer, said the return to profitability reflects the success of the company's "back-to-basics" strategy focusing on its core government and commercial hazardous-waste consulting businesses.

Automatic Transcription of Speech
DIANE COELHO RECENTLY PROMOTED TO CHIEF EXECUTIVE OFFICER SAID THE RETURN TO PROFITABILITY REFLECTS THE SUCCESS OF THE COMPANY'S BACK TO BASICS STRATEGY FOCUSING ON ESCORT GOVERNMENT AND COMMERCIAL HAZARDOUS WASTE CONSULTING BUSINESSES .

The effect on information extraction is most notable on names. Whenever a part of a name (e.g., a person's last name) is outside the vocabulary, current LVCSR technology will find the closest transcription within the vocabulary, causing one to misrecognize the name, and providing errorful input to information extraction. An example appears below: Text input is first; the automatic transcription of a spoken version appears next. Note how the company name is heard correctly (each word is in the vocabulary), but the person's last name "Barbakow" is misheard as "BARR NOW".

Text
As a managing director of Donaldson, Lufkin & Jenrette, Mr. Barbakow also was an investment banker for National Medical.

Automatic Transcription of Speech
AS A MANAGING DIRECTOR OF DONALDSON LUFKIN AND JENRETTE MR. BARR NOW ALSO WAS AN INVESTMENT BANKER FOR NATIONAL MEDICAL

5.2 Quantifying the Challenge

Using the evaluation methodology developed under the DARPA-sponsored Message Understanding Conferences (MUC), we measured information extraction performance on the MUC-6 template element (TE) task. TE requires recognition of organizations, persons, and properties of them. Like NE, this really is domain-independent.

To simulate the challenge of speech input, one speaker read Wall Street Journal texts from the MUC-6 corpora and automatically transcribed those texts via BYBLOS. Reading these texts in an office environment had a 20% error rate, which is better than the current best error rate of 27% on broadcast news. As summarized in Table 1 and in Figure 9, the effectiveness of information extraction (on training material) dropped from the high 80s to the mid 50s.

Condition	F
Text: Mixed Case	**86.76**
Text: Upper Case	**80.86**
Speech: with Commas	**76.57**
Speech: 0% Word Error	**71**
Speech: 20% Word Error	**53.28**

Table 1: Result of prototype experiment on information extraction from speech.

Part of the problem is the speech error rate. This can

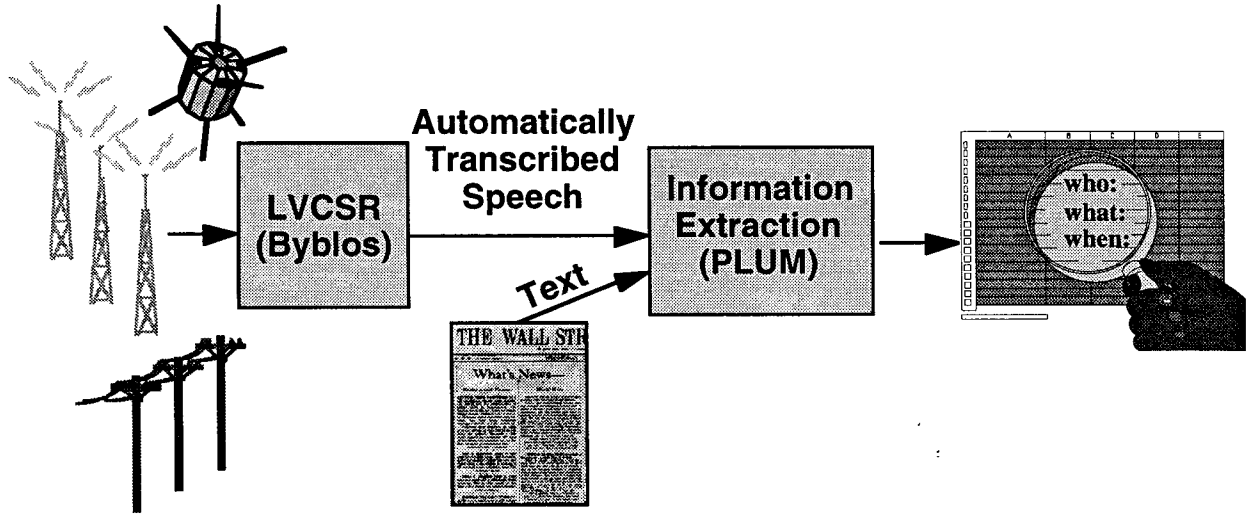

Figure 8: Architecture of Information Extraction from Speech

be seen in the drop in performance from 0% word error rate (perfect transcription) to 20% word error rate. A small part of the problem is the lack of capitalization (upper case only). The remainder of the degradation in performance compared to text input is due to other problems, such as the lack of punctuation; note how important commas are in information extraction.

5.3 Hearing New Names

There are at least three ways one could improve LVCSR for names. The one we believe most likely to succeed is vastly increasing the vocabulary size. Instead of only a 45,000 word vocabulary, which was derived by including all words occurring at least 50 times in seven years of The Wall Street Journal, suppose we add roughly 200,000 vocabulary items, focusing on last names, rare first names, and rare words in organization lists, e.g., companies listed by Dun & Bradstreet. Based on previous experiments (Asadi, 1991), the impact should be that

- About half of the occurrences of the additional 200,000 newly added words will be correctly recognized as the top choice in context.
- The computation for BYBLOS will increase.
- The overall word error rate will increase slightly.

That is quite promising, since it is relatively straightforward, might halve the error rate on names, and has only modest effect on speed and overall accuracy.

A second alternative is to allow the LVSCR system to enter into the transcription a symbol meaning "something outside the vocabulary". Recognizing that something is outside the vocabulary is not itself a reliable procedure, e.g., perhaps detecting 50% of such segments correctly with a 2% false alarm rate. However, even in the 50% correct detections, one still does not know the transcription; at present, a person would have to transcribe it. This seems far less promising than the first alternative above.

A third alternative would be to give a phonetic transcription of out-of-vocabulary items. While highly attractive, this seems like a long term research agenda.

LVSCR transcription places some demands on information extraction. First, the technology should allow for detection of important information even when fewer cues are present. In ST, one could produce a succession relation even if the person name or organization name is absent. Systems must rely far less on punctuation and on mixed case than the configurations in MUC-6. PLUM and other information extraction systems seem well poised to deal with such problems.

6. CONCLUSIONS

This paper briefly described a diverse collection of research activities at BBN on information extraction. We have concluded the following:

- Our information extraction engines for the MUC-6 Named Entity task and Template Element task employ no domain-specific information. We believe that development of other broadly

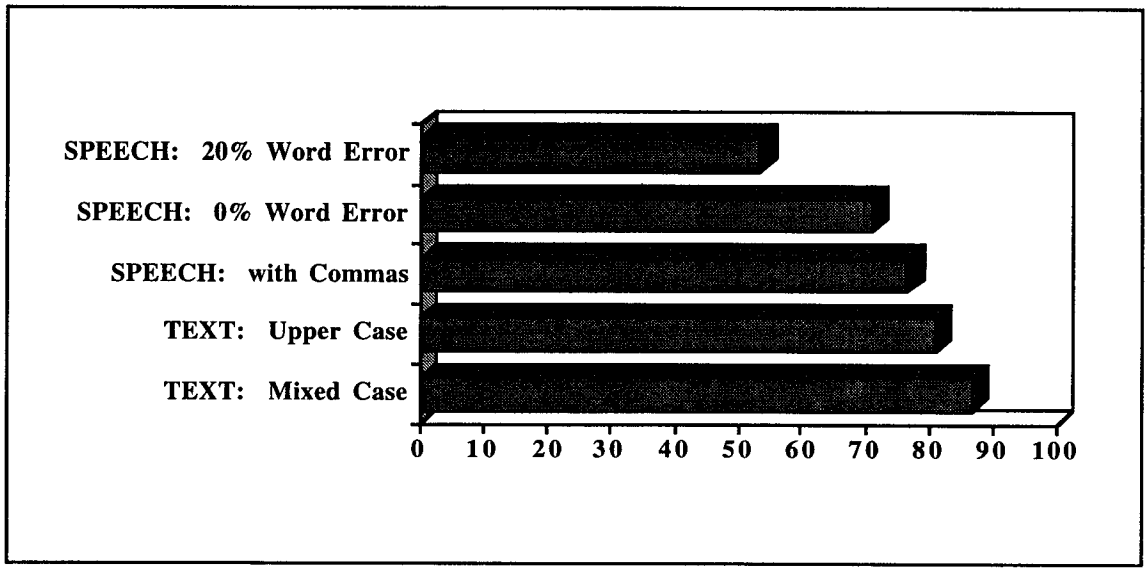

Figure 9: Challenge of Speech Input. *Several factors each cause the quality of information extracted to decrease on speech input.*

applicable information extraction functionality such as NE and TE will be a win, maximizing the value of defining reusable knowledge bases for information extraction.

- We developed a full parser of English, using a statistically learned decision procedure; SPATTER has achieved the highest scores yet report on parsing English text (Magerman, 1995). The fact that its recall and precision are both in the high 80s represents not just a quantitative improvement in parser performance, but also a qualitative improvement.
- The NLU Shell provides a way for non-programmers to build and maintain information extraction systems based on PLUM. The use of a GUI and a database in place of files of source code and data represents a fundamental advance in making natural language technology widely available. While the NLU Shell users need detailed familiarity with extracting formatted data from natural language, they do not need to be programmers.
- Compared to PLUM's previous performance in MUC-3, -4, and -5, our progress in MUC-6 was much more rapid and our official score was higher than in any previous template fill task. Furthermore, PLUM's performance was higher than in any of the previous full template MUC tasks.
- The template merging experiment provided a substantial range in recall versus precision, i.e., in undergeneration versus overgeneration. Nevertheless, we did not achieve a breakthrough in overall F-score, i.e., in ERR.
- Our preliminary results in learning the Named Entity Extraction task in English and Spanish are quite encouraging, since they are better than any previously reported scores for a learned system and since they are approaching the scores of the state-of-the-art for manually built, rule based systems.
- A preliminary experiment in information extraction from speech has shown that there are very significant challenges for TIPSTER text extraction technology, including the current 20-30% word error rate of transcription systems, the lack of punctuation within sentences, the lack of capitalization, and the error rate on names.

7. ACKNOWLEDGMENTS

The work reported here was supported in part by the Defense Advanced Research Projects Agency; technical agents for part of the work were Rome Laboratory under contract number F30602-95-C-0111 and Fort Huachucha under contract number DABT63-94-C-0062. Part of the work was supported by Rome Laboratory under contract number F30602-92-C-0078. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

Significant contributions to this paper were made by Richard Schwartz.

8. REFERENCES

Asadi, A. "Automatic Detection and Modeling of New Words in a Large-Vocabulary Continuous Speech Recognition System", Doctoral Thesis, Northeastern University, 1991.

Ayuso, D.M., Boisen, S., Fox, H., Ingria, R., and Weischedel, R. "BBN: Description of the PLUM System as Used for MUC-4", *MUC-4 Proceedings,* 1992.

Iwanska, et.al., "Computational Aspects of Discourse in the Context of MUC-3", *Proceedings of the Third Message Understanding Conference (MUC-3),* 1991.

Magerman, D., "Statistical Decision-Tree Modeling for Parsing", *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics,* 1995.

Matsukawa, T., "Hypothesizing Word Association from Untagged Text", *Proceedings of the ARPA Workshop on Human Language Technology*, 1993.

Matsukawa, T., Miller, S., and Weischedel, R. "Example-Based Correction of Word Segmentation and Part of Speech Labelling", *Proceedings of the ARPA Workshop on Human Language Technology*, 1993.

Weischedel, R., Ayuso, D.M., Boisen, S., Fox, H., Ingria, R., and Palmucci, J., "Partial Parsing, A Report on Work in Progress", *Proceedings of the Fourth ARPA Workshop on Speech and Natural Language,* 1991.

Weischedel, R., Ayuso, D.M., Bobrow, R., Boisen, S., Fox, H., Matsukawa, T., MacLaughlin, D., Papageorgiou, C., Sakai, T., Abe, J., Hoshi, H., Miyamoto, Y., and Miller, S., "BBN's PLUM Probabilistic Language Understanding System", *Proceedings of the TIPSTER Text Program (Phase I),* 1993.

Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L., and Palmucci, J. "Coping with Ambiguity and Unknown Words through Probabilistic Models", *Computational Linguistics (Special Issue on Using Large Corpora: II)* 19, 359-382, 1993.