INFORMATION EXTRACTION RESEARCH AND APPLICATIONS: CURRENT PROGRESS AND FUTURE DIRECTIONS

Andrew Kehler, Jerry R. Hobbs, Douglas Appelt, John Bear, Matthew Caywood, David Israel, Megumi Kameyama, David Martin, and Claire Monteleoni

SRI International*

1 Introduction

Analysts face a daunting task: they must accurately analyze, categorize, and assimilate a large body of information from a variety of sources and for a variety of domains of interest. The complexity of the task necessitates a variety of information access and extraction tools which technology up to this point has not been able to provide. SRI's TIPSTER Phase III project has focused on two major obstacles to the development of such tools: inadequate degrees of accuracy and portability. We begin by providing an overview of SRI's information extraction (IE) system, FASTUS, and then describe our efforts in these two areas in turn. We then conclude with some thoughts concerning future directions.

2 Overview of FASTUS

FASTUS processes natural language and produces representations of the information relevant to a particular application, typically in the form of database templates. As an example, we consider the task specified for the Sixth Message Understanding Conference (MUC-6), which was, roughly speaking, to identify information in business news that describes executives moving in and out of high-level positions within companies (Appelt et al., 1995). When FASTUS encounters a passage such as example (1),

(1) John Smith, 47, was named president of ABC Corp. He replaces Mike Jones.

it should extract the information that *Mike Jones* is 'out' and *John Smith* is 'in' at the position of *president* of company *ABC Corp.*

FASTUS consists of three major components. The first is the *pattern recognition* module, which consists of a series of finite state transducers that recognize patterns in the text and create templates representing event and entity descriptions. Pattern recognition relies on a second component, the *coreference module*, which identifies the referents of a variety of types of referential expressions (e.g., pronouns, definite noun phrases). Finally, the *merger* unifies templates created from different phrases in the text that describe the same events.

We illustrate by walking through an analysis of passage (1). The input is initially processed by using the finite state transducers to recognize relevant patterns and annotate the text accordingly. First, one or more preprocessing phases recognize low-level patterns such as person names, organization names, and parts of speech.

> [John Smith]$_{PERS-NAME}$ [47]$_{NUM}$ [was]$_{AUX}$ [named]$_{V}$ [president]$_{N}$ [of]$_{P}$ [ABC Corp]$_{ORG-NAME}$

The parsing phase identifies very local syntactic constituents, such as noun groups and verb groups; no attachment of ambiguous modifiers is attempted.

> [John Smith]$_{PERS-NAME}$ [47]$_{NUM}$ [was named]$_{VG}$ [president]$_{NG}$ [of]$_{P}$ [ABC Corp]$_{ORG-NAME}$

The combiner phase pieces together slightly larger constituents when it can be done reliably.

> [John Smith, 47]$_{PERS-NG}$ [was named]$_{VG}$ [president of ABC Corp.]$_{POS-NG}$

Finally, the domain phase applies domain-dependent patterns to the sentence to identify clause-level states and events. In this case, the entire sentence will match such a pattern.

> [John Smith, 47, was named president of ABC Corp]$_{DOMAIN-EVENT}$

Recognizing a pattern in the domain phase typically causes one or more template objects to be created. In light of the MUC-6 task specification, we

*Artificial Intelligence Center, 333 Ravenswood Avenue, Menlo Park, CA 94025, `kehler@ai.sri.com`

defined *transition* templates that track movements in and out of positions at companies; a person's leaving a job is represented by the start state of a transition, whereas a person's taking a job is represented by the end state. Therefore, the person, company, and position in the first sentence of (1) are represented in an end state since Smith is taking the described position. To facilitate certain types of inferencing during the merging phase, we also posit that someone, at this time unknown, is most likely leaving the position, this being represented in the transition's start state as shown in Figure 1.

$$\begin{bmatrix} \textsc{start} & \begin{bmatrix} \textsc{Person} & \text{---} \\ \textsc{Position} & \textsc{president} \\ \textsc{Organization} & \text{ABC }\textsc{Corp.} \end{bmatrix} \\ \textsc{end} & \begin{bmatrix} \textsc{Person} & \textsc{John Smith} \\ \textsc{Position} & \textsc{president} \\ \textsc{Organization} & \text{ABC }\textsc{Corp.} \end{bmatrix} \end{bmatrix}$$

Figure 1: Template Generated from *John Smith was named president of ABC Corp.*

The second sentence in the passage, *He replaces Mike Jones*, is then analyzed by the pattern matching phases, the details of which we omit. During this analysis, the coreference module identifies John Smith as the referent of "he". Having recognized a domain-level pattern, all that is known is that there is a start state involving the person Mike Jones and an end state involving the person John Smith, represented by the template shown in Figure 2.

$$\begin{bmatrix} \textsc{start} & \begin{bmatrix} \textsc{Person} & \textsc{Mike Jones} \\ \textsc{Position} & \text{---} \\ \textsc{Organization} & \text{---} \end{bmatrix} \\ \textsc{end} & \begin{bmatrix} \textsc{Person} & \textsc{John Smith} \\ \textsc{Position} & \text{---} \\ \textsc{Organization} & \text{---} \end{bmatrix} \end{bmatrix}$$

Figure 2: Template Generated from *He replaces Mike Jones.*

As they stand, of course, these two templates do not appropriately summarize the information in the text; there is a discourse-level relationship between the two that must be captured. This is the job of the merging component. When a new template is created, the merger attempts to unify it with templates that precede it. In this case, the template shown in Figure 2 should be unified with the template shown in Figure 1 to produce the template shown in Figure 3, which will lead to the correct output.

$$\begin{bmatrix} \textsc{start} & \begin{bmatrix} \textsc{Person} & \textsc{Mike Jones} \\ \textsc{Position} & \textsc{president} \\ \textsc{Organization} & \text{ABC }\textsc{Corp.} \end{bmatrix} \\ \textsc{end} & \begin{bmatrix} \textsc{Person} & \textsc{John Smith} \\ \textsc{Position} & \textsc{president} \\ \textsc{Organization} & \text{ABC }\textsc{Corp.} \end{bmatrix} \end{bmatrix}$$

Figure 3: A Successful Merge

3 Focus on Accuracy

The first major obstacle to the broad deployment of IE technology we address is the inadequate level of accuracy of existing systems. We have sought to push the accuracy of each of the three major modules of FASTUS in our TIPSTER effort.

3.1 A Lattice-Based System for Pattern Recognition

One of the main reasons for the success of FASTUS is that it bypasses much of the complex linguistic processing characteristic of previous systems. Processing decisions are made using local rather than global evidence, minimizing the risk that correct analyses get lost in a sea of incorrect ones. For instance, at each phase in the pattern recognition component, only the analysis deemed to be the best is passed to the next phase. Unfortunately, while this strategy has proved advantageous in general, in many cases it leads to premature processing decisions based on too little information.

For instance, for the following example,

(2) The committee heads announced the appointment of John Smith as CEO.

the parser phase of FASTUS will correctly mark "the committee heads" as a noun group. This decision is made because the noun usage of "head" is more common than the verb usage in this domain, and because of a "greedy" preference for longer constituents. Using the same heuristics for example (3),

(3) The committee heads Viacom's CEO recruitment efforts.

the system will generate the same analysis for "the committee heads". Since "heads" is actually used as a main verb in this example (with "the committee" as its subject), the parser's incorrect choice will result in there being no domain-phase analysis for the sentence.

The trick, then, is to try to improve the scope (and thus, the accuracy) of the current mechanisms, without adopting the inadequacies of previous frameworks that FASTUS was designed to improve upon. To do this, we implemented a lattice-based version of FASTUS. In keeping with the finite state paradigm, the pattern recognition phases perform transductions over compact lattice representations of the input, passing such representations between phases. With a lattice representation, there are as many analyses for a string as there are paths through it, yet processing remains efficient. The processing of examples (2) and (3) will result in a lattice with both possible analyses for "the committee heads". In example (2), the successful match will result from matching a path in which "the committee heads" is analyzed as a noun group, whereas in (3), the successful domain-phase match will result from a path in which "the committee" is analyzed as a noun group and "heads" is analyzed as a verb group.

Although compactly represented, the numerous analyses that can result from lattice-based processing still require some methods for pruning and path selection. To date, we have implemented and evaluated a variety of strategies. Thus far, the results of these experiments, as measured by F-score on the MUC-6 task, have been somewhat mixed. A typical experiment will yield about one point of gain in F-score; as expected, recall generally climbs with a smaller sacrifice in precision. We plan to do further experimentation in the future.

3.2 Improvements to Coreference Resolution

We have implemented various high-precision and largely domain-independent incremental extensions to the coreference resolution module.

Delayed Resolution in the Lattice System The implementation of the lattice-based system opened up the possibility of addressing several coreference issues that could not be cleanly addressed within the nonlattice system. The first is a catch-22 which results from a need to perform coreference resolution both before and after the domain phase level of analysis. (Recall that coreference resolution comes before the domain phase.) We illustrate with example (4).

(4) Analysts have been expecting IBM to announce some changes. In fact, today they named John Smith as president.

Let us assume, plausibly enough, that the domain phase contains a pattern of the following sort, which will match the second sentence if the referent of "they" is a company (in this case IBM).

Event := *Company* named *Person* as *Position*

Coreference resolution must necessarily apply before the domain phase, since the pattern interpreter needs to know whether the denotation of the subject (the referent of "they") is a company. Unfortunately, in this particular case the coreference module is likely to choose "analysts" as the referent, since it occupies the subject position of the preceding clause, which usually indicates a higher degree of salience than the object position that "IBM" occupies. Intuitively, however, just the fact that the system has the aforementioned pattern suggests that one would expect a company to be situated at that point in the context of the clause. Thus, there is reason to want coreference to apply after it has access to that pattern, that is, after domain-phase processing.

A similar problem occurs with respect to intrasentential coreference constraints. Consider the sentence

(5) John Smith removed him from the CEO post.

Intrasentential constraints, dictated by the syntactic structure of the sentence, tell us that "him" cannot refer to *John Smith*. However, only the domain phase has a notion of sentence-level syntax, so the system has no way of knowing of the applicability of this constraint given that the coreference module operates before this phase.

The lattice-based system provides a way to incorporate and preserve ambiguities through the domain phase, and thus offers an opportunity to address these problems. Instead of selecting only the most preferred referent for a referential expression, the coreference module takes the set of alternatives and writes arcs for each onto the lattice in place of the referential phrase, including relative levels of preference. This lattice then serves as input to the domain phase, as before, at which point the above constraints can be enforced. In the case of example (4), for instance, the path in which "they" is rewritten as "analysts" will not result in a successful match, whereas the path in which it is rewritten as "IBM" will. Alternatively, if both potential referents were company names, then the one that the coreference module considers to be most preferred will be selected.

Contributions of the coreference and lattice components were independently measured on an earlier baseline system. We observed that both components increase the recall and precision independently, with the recall (error reduction of 8% to 10%) much more affected than the precision (error reduction of 1% to 2%). The coreference-lattice combination leads to an even greater increase in the recall (13% error reduction) but less impact on the precision (less than 1% error reduction). After this evaluation, we realized that the logic of the coreference-lattice integration was incomplete, because of certain destructive operations that should not be maintained in a strategy in which alternatives are preserved. We expect an even greater performance impact when integration is completed.

Extensions to Coverage We also implemented extensions to the coverage of the coreference module.

First, we implemented a module for resolving implicit arguments for certain relational nouns. Relational nouns are those whose denotations are determined in association to a possessor entity. In the business-political domain, position nouns such as CEO and vice president are relational, associated with (sometimes implicit) organizations at which these positions exist. As a first step, we added a mechanism for resolving implicit organizations for position expressions similar to those for pronoun resolution. This addition increased the IE recall by almost a point (0.85%), a nontrivial gain for a change of relatively limited scope.

We also added a resolution routine for definite temporal expressions. Indexicals such as "today", "next week", "last Monday", and "10 years ago" are resolved with respect to the document date. Partial temporal expressions such as "Friday" and "the 23rd" are resolved with respect to the combination of the closest verb tense and the salient date in the global or local context. The globally salient date is the document date, whereas the locally salient date is the most recent date mentioned in the text. The performance of the date resolution routine was evaluated with eight training articles containing a total of 53 definite date expressions. Among the currently intended coverage of 43 expressions, 37 were correctly resolved. We can interpret it as having 69.8% recall (37/53) and 86.0% precision (37/43).

Fragment Analysis After we observed the effectiveness of implicit argument resolution as described above, we added a domain-specific treatment of what we call *fragment analysis*. FASTUS often finds fragments of domain patterns in texts because of insufficient domain coverage—an inevitable limitation, given the ability for natural language to express the same content in many different, often unpredictable surface realizations. Consider the following example.

(6) John Doe, who is known for his "my way or the highway" management style, but who nonetheless receives rave reviews from industry insiders, even his enemies, was named president of IBM.

In this case, FASTUS is likely to match the fragment "was named president of IBM," outputting a transition with a position and organization. Unfortunately, given the intervening material between this fragment and the subject, it will also most likely fail to link the transition to the incoming person, John Doe. The fragment analysis code corrects this by inspecting each transition created for a sentence, and, assuming that a substantial but incomplete template is found, attempts to locate candidates from the surrounding discourse context to replace the empty slots. The overall effect, specifically of making partial domain event templates more complete, is similar to that of the merging phase. The difference is that while merging combines two or more partially filled domain events, missing argument resolution fills empty slots of each domain event with recently mentioned entities even if they are not associated with extracted events. We compared the effects of fragment analysis and merging on the overall score using the 100 message MUC-6 training set. The result is shown in Table 1; fragment analysis alone performed better than merging alone, with the two together performing the best.

An Analysis of WordNet Sanda Harabagiu, a former post-doctoral fellow at SRI, performed an analysis of how WordNet might be used to improve coreference resolution, particularly by exploiting hypernym and synonym information. Using the MUC-6 coreference training messages as her corpus, she found that 60% of the coreference examples fall into categories in which WordNet is of no potential use: Cases of identity between strings (e.g., "a company...the company") comprised 42.3% of the examples, and cases in which coreference is indicated by syntactic configuration (e.g., appositives, as in "John Smith, president of Acme Widgets") comprised 18.27% of the examples.

Reference involving a synonym relation made up 8.33% of the examples. Of these, 3.1% were synonyms in WordNet, such as "bill" and "measure". However, 5.23% were not in WordNet. Some of these cases one could imagine being in such a knowledge source, such as "business" and "company"; it just so happens that they are not. On the other hand, there are also more difficult cases, such as "IBM" and "wounded computer giant", for which no knowledge base is likely to contain a relation.

Reference involving a hypernym relation made up

Merging	Fragment Analysis	Precision	Recall	F-score
Off	Off	71	42	52.60
Off	On	68	48	56.10
On	Off	65	52	57.67
On	On	64	55	59.17

Table 1: Contributions of Fragment Analysis and Merging

11.0% of the cases. Of these, 3.7% were in WordNet, such as "quarter" and "period", and "chairman" and "officer". The other 7.3% were not in WordNet. Again, there were cases which one could imagine being there, such as "automaker" and "company". Others, however, such as "Clinton officials" and "Clinton camp", are not likely to be found in any such knowledge base.

The remaining cases were often more difficult; many involving metonymy.

3.3 Learning Merging Strategies

Early in the project, we performed an analysis of the errors FASTUS made on a subset of the MUC-6 development corpus. The majority of the errors indicted merging at least in part, suggesting that merging improvements had a potential for high payoff.

The existing FASTUS merging algorithm is quite simple – it attempts to merge newly created templates with previous ones, starting with the most recent. Templates are merged when they are unifiable in accordance with any prespecified constraints. Despite its simplicity, the algorithm has proven to be fairly successful. Nonetheless, it is quite possible that other merging strategies could yield better results.

There are two ways in which one might attempt to identify such strategies. First, one could perform data analyses to identify good merging principles, handcode them, and test the results. Alternatively, one could attempt to have merging strategies be acquired by the system automatically, using some training mechanism. We attempted both of these, which we discuss in turn.

Data Analyses and Experimentation The first action we took was to perform an extensive analysis of merging results. We developed detailed mechanisms for tracing merging behavior and distributed transcripts among several project participants. In analyzing these, we identified a variety of constraints which appeared to be extremely reliable, in particular, characteristics of templates that were almost always correlated with incorrect merges.

One by one, these constraints were implemented and tested. In each case, end-to-end performance on the scenario template task either remained the same or decreased slightly. In no case did we get a nontrivial increase in performance.

This was rather puzzling and frustrating, and highlighted some of the problems with handcoding system improvements. For one, the processes of data analysis, system coding, and testing are labor intensive. One cannot try all possible alternative sets of constraints one might consider, so one can never be sure that other, unattempted constraints would not have fared better. Second, it could be that we were being misguided by the relatively small data sets that we were analyzing by hand. Thus, we began considering other paradigms for identifying better merging strategies.

There were also other, longer-term considerations for moving away from handcoding merging improvements. For one, the optimal merging strategy is highly dependent on the quality of the input it receives, which is constantly evolving in any realistic development setting, thus requiring continual reexperimentation. Thus, changes that improve performance at one point in system development could potentially decrease performance at another time, or vice versa. Second, a general goal of IE research is to have systems that can be trained for new applications long after the system developers are involved, which precludes experimentation by hand.

These considerations motivate research to determine if merging strategies can be learned automatically. There are several different types of learning, including *supervised*, *unsupervised*, and an area in between which one might call *indirectly supervised*. We have performed experiments using all three types of technique, which we describe below.[1]

[1] The work reported on here, also discussed in Kehler (1998), concerns learning merging strategies in support of the *scenario template* task of MUC-6 as described in Section 2. While we are unaware of any other reported research on this task, other work has addressed other MUC-style tasks. For instance, Kehler (1997) describes a probabilistic approach to entity-level merging that outperforms several baseline metrics. Also, researchers at BBN (Ralph Weischedel, TIPSTER 18-month meeting)

Supervised Methods In our first set of experiments, we took the approach most commonly pursued in the computational linguistics literature, namely supervised learning. Supervised methods require a set of training data that the learning algorithm can consult in constructing its model. For our initial experiments, we ran the 100 MUC-6 training messages through FASTUS and wrote out feature signatures for the 534 merges that the system performed. The feature signatures were created by asking a set of 50 questions about the context in which the proposed merge is taking place, referencing the content of the two templates and/or the distance between the phrases from which each template was created. Some example questions are:

- SUBSUMED?: true if the contents of one template completely subsume the contents of the other.
- UNNAMED-REFERENCES?: true if either transition has a slot filled with an object lacking a proper name, e.g., "an employee" in the person slot. While these objects can merge with other (perhaps named) entities of the same type, in general they should not.
- LESS-THAN-700-CHARS?: true if the phrases from which the templates are created are less than 700 characters apart in the text.

After the feature signatures were written, we examined the texts and manually encoded a key for each.

We attempted two approaches to classifying merges using this corpus as training data. The first was to grow a classification tree in the style of Breiman et al. (1984). At each node, the algorithm asks each question and selects the one resulting in the purest split of the data. Entropy was used as the measure of node purity. In the second set of experiments, we used the approach to maximum entropy modeling described by Berger et al. (1996). The two possible values for each of the same 50 questions (i.e., yes or no) were paired with each of the two possible outcomes for merging (i.e., correct merge or not) to create a set of *feature functions*, or *features* for short, which were used in turn to define *constraints* on a probabilistic model. We used the learned maximum entropy model as a classifier by considering any merge with a probability strictly greater than 0.5 to be correct, and otherwise incorrect.

report on learned merging strategies achieving good performance on the less complex *template entity* and *template relation* tasks in MUC-7, although no comparison with a similar hand-coded system was provided.

Out of the available set of questions, each approach selects only those that are most informative for the classifier being developed. In the case of the decision tree, questions are selected based on how well they split the data. In the case of maximum entropy, the algorithm approximates the gain in the model's predictiveness that would result from imposing the constraints corresponding to each of the existing inactive features, and selects the one with the highest anticipated payoff. One potential advantage of maximum entropy is that it does not split data like a decision tree does, which may prove important as training sets will necessarily be limited in their size.

In our preliminary evaluations, we used two-thirds of our annotated corpus as a training set (356 examples), and the remaining one-third as a test set (178 examples). We ran experiments using three different such divisions, using each example twice in a training set and once in a test set. In each case the maximum entropy classifier chose features corresponding to either 6 or 7 of the available questions, whereas the decision tree classifier asked anywhere from 7 to 14 questions to get to the deepest leaf node. In each case there was considerable, but not total, overlap in the questions utilized. Adding the errors from the three evaluations together, the decision tree made 34 errors (out of a possible 534), in which 13 correct merges were classified as incorrect and 21 incorrect merges were classified as correct. The maximum entropy classifier made a total of 31 errors, in which 14 correct merges were classified as incorrect and 17 incorrect merges were classified as correct. This is compared to a total of 139 errors out of the 534 merges that the current merger made according to the annotations.

These results may appear to be positive, as it would seem that both methods found some reliable information on which to make classifications. However, our goal here was to improve end-to-end performance on the scenario template task, and thus we wanted to know how much of an impact these improved merging strategies have on that performance. Therefore, we replaced the existing FASTUS merging algorithm with two more discriminating mergers, each directed by one of our learned classifiers. The first version consulted the decision tree and merged only when the example was classified as correct. The second version did the same using the maximum entropy classifier. For these experiments, the two models were trained using the entire set of 534 examples.

As we were still experimenting at this point, we were not ready to perform an evaluation using our set of blind test messages. As an information gathering experiment, we applied FASTUS using the new mergers to the corpus of messages that produced the training data. We would of course expect these experiments to yield better results than when applied to unseen messages. Nonetheless, the results were humbling – both experiments failed to improve the performance of the overall system, and in fact degraded it slightly. Generally, a point of precision was gained at the expense of a point or two of recall.

Clearly, there is a rift between what one might consider to be good performance at discriminating correct and incorrect merges based on human judgments, and the effect these decisions have on overall performance. Because the baseline FASTUS algorithm merges too liberally, using the classifiers cause many of the incorrect merges that were previously performed to be blocked, at the expense of blocking a smaller number of correct merges. Thus, it is possible that the correct merges the system performs help its end-to-end performance much more than incorrect merges hurt it. For instance, it may be that correct merges often result in well-populated templates that have a marked impact on performance, whereas incorrect merges may often add only one incorrect slot to an otherwise correct template, or even result in templates that do not pass the threshold for extractability at all. In fact, in certain circumstances incorrect merges can actually help performance, if two incorrect templates that would produce incorrect end results are unified to become one.

In any case, it should be clear that improved performance on an isolated subcomponent of an IE system, as measured against human annotations for that subcomponent, does not necessarily translate to improved end-to-end system performance. Add this to the cost of creating this annotated data – which will continually become obsolete as the upstream FASTUS modules undergo development – and it becomes clear that we need to look to other methods for learning merging mechanisms.

Unsupervised Methods Naturally, the main alternatives to supervised methods are unsupervised methods. We consider replacing our merging algorithm with one that performs an unsupervised clustering of the templates and merges the templates in each cluster. Of course, we will not know *a priori* how many clusters there are, that is, how many templates we should be left with when we are finished. A method that does not require such knowledge is Hierarchical Agglomerative Clustering (HAC) (Duda and Hart, 1973; Everitt, 1980, inter alia).

The HAC algorithm is conceptually straightforward. Given a set of examples, the algorithm begins by assigning each to its own cluster. A predetermined similarity metric is then applied to each pairwise combination of clusters, and the most similar pair combined. The process is iterated until no pair of clusters have a similarity that exceeds a preset threshold.

Our application of clustering is somewhat different from many problems to which clustering has been applied. For one, our clusters will always have only one member, since templates are merged upon clustering. Issues with how to compute similarity between two nonsingleton sets of data points are therefore avoided. Furthermore, our notion of similarity is nonstandard. Usually, similar examples are distinct, but have properties that are "close" to each other in some space. Here, similarity is meant to measure the likelihood that the two templates are incomplete descriptions of the same complex of eventualities (i.e., the same transition), although the templates themselves may look very different.

We performed some informal experiments in which we intuited a similarity metric, assigning weights to a subset of the questions that we had defined for the supervised learning experiments. For instance, templates that were created from phrases close to each other in the text and that overlapped in content received high similarity, whereas those that were far apart and did not overlap received low similarity. Instead of merging incrementally as in the supervised learning experiments, pattern matching was first applied to the entire text, and the resulting templates were clustered and merged until no pair of templates passed a preset similarity threshold.

Running the system over the MUC-6 development set yielded results similar to our experiments using the supervised mergers. We did not find this to be particularly surprising; for instance, the mediocre results could be attributable to the similarity metrics not being very good.

We did not push this approach any further, because it is still lacking with respect to one of our goals for pursuing learning strategies. While it addresses the problem of requiring annotated training data, it does not address the fact that the optimal merging strategy is inherently dependent on its input. If we encode a similarity metric for clustering and keep it fixed, we are left with only a single degree of freedom – the similarity threshold at which to halt the clustering process. While this may yield some leverage (for instance, good input to the merger may call for a high threshold, whereas bad input may call for a lower threshold), it will certainly be too inflexible in the general case.

In sum, several factors could influence the likelihood of a potential merge within a particular application, and it therefore seems that something tied to the application needs to guide the learning process.

Indirectly Supervised Methods When developing an IE system, one typically encodes (or is given) a moderate-size set of end-to-end development keys for a set of sample messages. These keys need to be encoded only once. We did not use these keys for supervised learning because of the difficulties in aligning the inaccurate and incomplete intermediate templates produced by the system with the (normalized) end results. However, we can use the keys to evaluate the end results of the system, and attempt to tune a merging strategy based on these evaluations. After all, it is improved end-to-end performance that we are seeking in the first place.

Thus, we consider a form of what we are calling indirectly supervised learning. We use the HAC mechanism described in the previous section, but attempt to learn the similarity metric instead of stating it explicitly. The search through the space of possible similarity metrics will be driven by end-to-end performance on a set of training messages.

We start by defining a space of similarity metrics. In a preliminary experiment, we used 7 of the questions that were used in the supervised experiments, coupled with their negations, for a total of 14 questions. These questions are assigned weights, either positive or negative, that get incorporated into a similarity metric when the question is true of a potential merge. Let λ_i be the weights assigned to corresponding questions q_i, and let the function $f_{q_i}(t_1, t_2)$ be 1 if the question q_i is true of the templates t_1 and t_2, and 0 if not. Then the similarity $S(t_1, t_2)$ is given by

$$S(t_1, t_2) = \frac{e^{\sum_i f_{q_i}(t_1,t_2)*\lambda_i}}{e^{\sum_i f_{q_i}(t_1,t_2)*\lambda_i} + 1}$$

This function, which is adapted from the form of the probability model used in the maximum entropy framework, provides a similarity measure in terms of a probability.

We used an annealing strategy to tune the weights λ_i. The algorithm begins by processing the 100-message MUC-6 development set, usually with a randomly selected initial configuration that establishes a baseline F-score. The algorithm then iterates, selecting some of the questions at random (perhaps just one, perhaps all of them) and permuting their weights by a random amount, either positive or negative. The system is then rerun over the training set and the F-score measured. Any permutation resulting in an F-score that is strictly greater than the current baseline is adopted as the new baseline. To stay out of local maxima, a permutation leading to a decrease in performance may also be adopted. This is the annealing part – such negative permutations are accepted with a probability that is proportional to a steadily decreasing measure of 'temperature', and inversely proportional to the magnitude of the decrement in performance. Thus, permutations that decrease performance slightly in early stages of the search are likely to be adopted, whereas permutations that decrease performance either significantly or in later stages of the search are not.

The results of one of several experiments are shown in Figure 4. The search began with an initial similarity metric achieving an F-score of 58.83, and continued for 300 iterations. A low F-score of 57.70 was achieved early, in iteration 10. The best metrics considered yielded an F-score of 59.80.

Obviously, and somewhat surprisingly, this graph is practically flat. On one hand, it is unfortunate that there aren't higher high points: The learner was not able to leverage the available features to acquire a much better merging strategy than the one it started with. Perhaps even more surprising, however, is that there were also not lower low points – only iteration 10 achieved a score lower than 58. Because the learner was not given any bias with respect to the permutations it attempted, some of those it considered were intuitively poor (e.g., boosting the weight for phrases that are very far apart, lowering the weight for sparsely filled templates with no overlap). Thus, one might have expected certain of these to devastate performance, but none did. It seems that as long as a certain amount of merging is performed, it matters less which templates are actually merged, and in what order.

Conclusions and Future Directions In sum, the learned mechanisms were neither significantly better nor worse than a hand-coded merging strategy. The inability to outperform the existing strategy could be attributed to several facts. We suspect that a major problem is the lack of accessible, reliable, and informative indicators for merging decisions. Unlike lower-level problems in natural language processing (NLP) in which local information appears to bear highly on the outcome, including, for instance, part-of-speech tagging (Church, 1988; Brill, 1992, inter alia) and sense disambiguation (Yarowsky, 1994; Yarowsky, 1995, inter alia), none of the questions we have formulated appear to be particularly indicative of what effect a potential merge will have on system performance. This suggests that more research is needed to identify ways

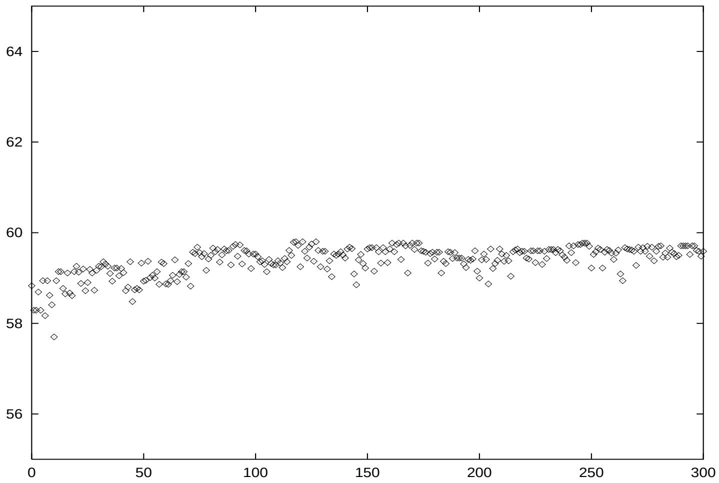

Figure 4: Results of a Learning Experiment

to access the necessary knowledge from independent sources such as existing knowledge bases, or by mining it from online text corpora using unsupervised or indirectly supervised learning techniques.

Furthermore, these experiments may be cause for concern about the nature of the scoring metric and procedure used in MUC-6. All of the merging strategies attempted, both hand-coded and automatically learned, performed similarly. This (rather unexpected) result would suggest that the scoring mechanisms be given a closer look, which we do in the following section.

3.4 Analysis of the Scoring System

As we have indicated, the lack of more significant progress in some of the foregoing efforts had us puzzled. Intuitively positive system changes were not showing much effect in terms of end-to-end performance, nor were certain intuitively negative changes. Of course, judgments of what constitute positive and negative changes are only as good as the scoring mechanism which is providing the feedback. As part of a related project at SRI, we began to find some more concrete evidence that at times this feedback has been misguiding our efforts. Incremental refinements in the system's output, ones that should yield superior results, nevertheless receive a lower score from the scoring mechanism.

The following text (WSJ article 870112-0001) provides an example illustrating this point:

(7) The board also named a three-man executive committee to perform the chief executive's role. The three members are Victor Steele, head of the company's beverage division; Brian Baldock, head of the leisure and health division; and Shaun Dowling, who runs industrial operations.

....

Further executive resignations or dismissals are widely expected. The positions of Olivier Roux, head of financial planning, and Thomas Ward, a U.S. attorney who is a close aide to Mr. Saunders, are "open to question," one Guinness source said.

FASTUS does poorly on this example, for understandable reasons. It did not produce any succession events for the first paragraph, because doing so would require resolving a variety of difficult linguistic issues lying beyond the depth of processing at which FASTUS operates. On the other hand, for reasons that won't be described in detail, the system generated a succession event from the second paragraph involving the position "head of financial planning", with four IN-AND-OUT templates involving Roux, Saunders, and two other people mentioned in the article.

While not much could be done for the first paragraph, we modified FASTUS so that it would not produce a template from the second paragraph. The change to the system performance on this message has to be positive: while we do not generate any additional correct information from the change, we eliminated four predications about an irrelevant position, three of which would be false even if one considered the position to be relevant. Other output for this text was not affected, so we would expect to observe the same recall (correct output was not changed), but notably higher precision from having eliminated the incorrect succession event, four incorrect IN-AND-OUTs, and two irrelevant PERSON templates.

In reality, this change resulted in a slight rise in precision (from 57 to 59) and a dramatic reduction in recall (from 50 to 33), causing the F-score *on this message* to plummet from 53.30 to 42.67. The reward for eliminating four irrelevant predications was a 20% drop in the score. This result is, to say the least, counterintuitive, and suggests serious problems in the ability of the scoring mechanism to provide adequate feedback.

We have several speculations regarding the causes of this behavior, but final conclusions await a more comprehensive study. It should be obvious in any case, however, that further progress in IE is crucially dependent on these issues being resolved. While this is true regardless of the approach one takes to system development, it is especially so if we want to move toward systems with rules and procedures that are learned automatically. Successful learning depends on the assumption that learned improvements are reflected in the evaluation function; if this is not the case then learning is all but hopeless. Thus, future research in IE must be coupled with research into evaluation strategies.

3.5 The Zipf Effect on Information Extraction Applications

A fundamental question with respect to IE applications is the nature of the Zipf curve relating pattern development to improved coverage. In a given application, there is usually a small set of patterns which will have broad applicability – that is, they are likely to match on many examples in any given set of unseen data. For instance, a MUC-6 pattern designed to match the sentence

(8) John Smith was appointed CEO of IBM.

will almost certainly match many other similar examples also. At the other end of the spectrum, there are many 'one-of-a-kind' examples in any given training corpus for which the corresponding pattern is unlikely to match many other examples. For instance, a pattern developed to handle the sentence

(9) John Smith and his associate, Roger Jones, the former of which will soon be on board at IBM and the latter of which will be heading to Apple, are in line to be CEO and chairman, respectively.

is unlikely to match other examples in any reasonably-sized corpus of unseen data. The big question, then, is at what point in development do the great majority of examples fall into the second class; at this point performance gains on training data do not transfer to gains on test data. It could very well be the case that after developing patterns to handle the examples in a moderately-sized training set – say 100 messages, as in the MUC-6 training corpora – one has reached the point of diminishing returns.

In support of a project related to TIPSTER, the Office of Research and Development provided us with an additional set (90 messages) of data with keys annotated in accordance with the MUC-6 task specification. This gave us an opportunity to see whether new improvements inspired by this data would transfer to the test data. The changes we implemented were all relatively minor. They included:

- Fixing a few problems in name recognition
- Adding a parser phase pattern
- Adding domain phase patterns for a few metaphorical expressions
- Eliminating a filter for irrelevant texts
- Fixing other minor bugs

These modifications caused our score on the new training data to increase from 46.4 to 52.1, which is not a surprising result. Given that the fixes were directed narrowly at specific examples in this set, we did not expect to see much of an improvement in either of the other data sets. Our suspicions were confirmed by results on the basic training data; our score on this set went from 58.6 to 59.6. Quite surprisingly, however, our score on the blind test set rose significantly, from 51.7 to 57.1 – an increase of over 10%.

Thus, necessarily adding a proviso about the adequacy of the evaluation metrics per the last section, we have a negative data point for the hypothesis that 100 training messages place us beyond the point of diminishing returns. The second set of messages apparently had considerable overlap with the test data in areas that did not overlap with the original training set.

4 Focus on Portability

A second major obstacle to the broad utilization of IE technology is the time and expertise needed to develop new systems. Users need to be able to develop extraction systems for new information needs rapidly and without the assistance of a system developer. We have been developing infrastructure, consisting of patterns, ontologies, and tools, which brings us closer to these capabilities.

4.1 Open Domain System

The majority of previously pursued IE tasks, including those in the MUC evaluations, have been centered on extracting information from a narrowly defined domain. Alternatively, one might imagine developing a system capable of extracting information about a significantly broader set of events that might potentially be of interest to an analyst. We call such a system an *open domain* application.

We are currently completing our implementation of an open domain system for business news. The system is built upon an infrastructure consisting of a broad set of patterns and ontologies. These patterns and ontologies will serve as a basis for the analyst to produce special-purpose IE systems (which we call FASTLETS) for specific information needs. Such FASTLETS could be used not only for database generation, but also to improve systems for document and subdocument retrieval and for task-driven summarization, among other applications.

The patterns and ontologies were developed from an in-depth analysis of the 150 most common verbs and nominalizations within a corpus of Wall Street Journal texts. A frequency analysis was performed to identify these verbs and nominalizations, and a list was generated of all the sentences in the corpus containing each. A chart was then constructed for each group, listing each verb and its role fillers (subject, object, prepositional objects). This gave rise to the patterns required to cover the examples, and the elements and organization of an ontology emerged. A few example patterns are shown below.

> *Person* analyzes { *Industry* | *Commodity* | *Financial-Instrument* }
>
> { *Company* | *Person* } controls *Company*
>
> { *Company* | *Country* } exports *Goods* to *Country*
>
> *Coperorg* invests *Money* in { *Financial-Instrument* | *Market* | *Country* | *Company* }

The italicized elements indicate concepts in the developed ontologies; for instance, *Coperorg* is a category subsuming several other concepts including *Person*, *Company*, and *Organization*.

Open domain patterns are integrated with the *compile-time transformation* component of FASTUS. This component is capable of taking a single pattern and specifying the different ways in which it can be expressed in English. Thus, the first pattern in the list above will not only match sentence (10),

(10) John Smith analyzed the automobile industry.

but it will also match examples such as (11) and (12).

(11) The automobile industry has been analyzed by John Smith.

(12) John Smith's analysis of the automobile industry...

The output of the open domain pattern set is a case-frame style template, marking roles and modifiers such as agent, patient, location, time, and purpose.

Open Domain and Rule Acquisition As we have indicated, one of the ways in which the open domain infrastructure can be used is as the basis for allowing end users to construct their own patterns tailored to their own information needs. The development process will be much like what expert developers do to build systems, except that there will be a richer set of tools for doing so. For instance, in our MUC-6 effort, we first outlined the events of interest, and then scanned training texts to determine the verbs and nominalizations that encoded those events. We then categorized them into classes of verbs with the same case frames, and wrote subject-verb-object patterns for each of the classes.

We are currently developing an interface that will allow end users to accomplish this. Analysts will select the open domain patterns that are relevant to their needs, and constrain their arguments in appropriate fashions. The system will support testing on existing corpora and provide assistance for further rule adaptation. The interface is being implemented in Java.

4.2 An Application: Using IE to Improve Document Retrieval

As we have mentioned previously, one of the possible uses for FASTLETS is to improve the quality of document retrieval (DR) results. We discuss some of our past and current work, as well as future plans.

Completed Experiments In work predating TIPSTER Phase III, a topic was chosen from the TREC-5 corpus which overlapped significantly with the MUC-6 management succession topic. SRI's MUC-6 system was used to reorder the retrieval results from the UMASS Inquery ad-hoc query system, based on the results of finite state pattern matching. This experiment produced a positive result, which, while far from being definitive, suggested that further investigation should be performed. Of course, the scenario that was being tested is not realistic, as such highly developed IE systems will not generally exist for most information needs. A more reasonable scenario would be one in which a rapidly developed FASTLET is used to perform such a task.

During TIPSTER Phase III, SRI teamed with GE R&D to participate in the TREC-6 evaluation. FASTLETS were developed for 23 of the 47 topics in the TREC-6 routing task, in which up to 4 hours per topic was spent reading a small set of relevant texts and writing a small number of grammar rules and lexical attributes. Each FASTLET was then run overnight on some additional training data, and another 1 to 2 hours was spent (on average) making any necessary adjustments, for a total of an average of 4 to 6 hours per topic. The majority of these FASTLET grammars were developed by a Stanford undergraduate, who has the characteristics one might expect the end user of such a system to have: he is smart and computer literate, but knows essentially nothing about NLP, linguistics, IE, and DR.

In the GE/SRI joint TREC-6 entry, the routing query version of GE's DR system was used to provide the top 2000 ranked documents for each topic. The FASTLETS were then used to rerank the list and produce the top 1000. The results were encouraging, albeit again not definitive. In abstract terms, the GE/SRI system improved on the results of the GE system alone for 16 topics, degraded them for 5 topics, and received the same results on 2 topics. Of the 16 topics in which the results were improved, in 2 cases the improvement was very significant, in 6 cases the improvement was significant, and in 8 cases the improvement was small and insignificant. For one topic, ours was the best performing system. Of the 5 cases in which the results were degraded, in 3 cases the decline was significant and in 2 cases it was very significant.

These results are encouraging in that they indicate that the FASTLET approach to improving DR may be feasible, considering that in at least some cases NLP techniques improved the results of an already competitive routing query system.

An Ongoing Study The results of the foregoing experiments are especially encouraging considering that they were achieved using a highly suboptimal overall architecture. The DR and IE systems were treated as black boxes: the DR system ranked documents using standard DR types of evidence (word frequency analysis), and then the FASTLETS reranked the documents based on pattern matching evidence, without considering (or even having access to) the DR evidence. All the FASTLETS had access to was the output ordering. In actuality, it is likely that both types of evidence are useful for relevance determination, and that the relative usefulness of each varies on a per-topic basis. What is needed is an architecture in which the DR and IE evidence is considered together, with a principled mechanism for selecting the most informative features for document relevance on a per-topic basis.

We are currently pursuing such an architecture, which, in addition to certain modifications to FASTUS, requires a research-level DR capability. We have implemented a variety of word collection and frequency analysis mechanisms which leverage the considerable tokenization and morphological analysis capabilities of FASTUS. We have also implemented several learning algorithms capable of incorporating and weighing heterogeneous types of evidence.

In order to speed up FASTUS processing on large data collections, we implemented a "trigger word" compiler for FASTUS grammars. The mechanism reads in a pattern set and generates a list of words required to match them. Any sentence that does not contain a word on the list can be ignored after early stages of processing.[2] Initial experiments have indicated a speed up of more than a factor of three.

Mechanisms for generating relevance features from FASTLET results are currently being implemented, in preparation for the learning experiments. We will report on the results of these experiments in a future forum.

5 Conclusions and Future Directions

We have summarized SRI's developments in addressing two major obstacles to the broad deployment of IE technology: accuracy and portability. The TIPSTER program has witnessed significant progress in both areas, and has perhaps witnessed even greater progress in our *understanding* of IE technology.

We believe that the current state of IE technology suggests two main directions for future work; directions which look to opposite directions of the research-to-applications spectrum. The first direction is to leverage the progress we have made to embed IE technology within applications in which it can be useful. Candidate applications include document retrieval, task-based summarization, task-based machine translation, cross-document and multimedia fusion, and trend analysis. Current progress

[2] Although it should be noted that every sentence needs to be processed up through the combiner phase if coreference is to work optimally, since referents for referential expressions can occur in otherwise irrelevant sentences. The degree to which ignoring this fact affects performance is an empirical question, which will be studied in future work.

prepares us well for such investigations, the critical question being whether current levels of accuracy are sufficient for success.

Our work has also suggested that if we are to achieve revolutionary (rather than merely evolutionary) improvements in the state-of-the-art, we also need to step back and focus on fundamental research. Current approaches are good at identifying the information that natural language "wears on its sleeve"; the remainder will require new and richer techniques. Basic research is necessary to guide the development of such mechanisms, and must be coupled with an investigation into evaluation mechanisms.

6 Acknowledgments

We thank Sanda Harabagiu and Jeffrey Petit for their contributions to this work. This research was supported by the Defense Advanced Research Projects Agency and the Office of Research and Development under the TIPSTER Phase III program. We also thank the Office of Research and Development for providing an additional set of annotated training data.

References

Appelt, Douglas E., Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Myers, and Mabry Tyson. 1995. SRI International FASTUS system MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, Maryland.

Berger, Adam, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Brill, Eric. 1992. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy.

Church, Kenneth W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143, Austin.

Duda, Richard O. and Peter E. Hart. 1973. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York.

Everitt, B. 1980. *Clustering Analysis*. John Wiley and Sons, New York.

Kehler, Andrew. 1997. Probabilistic coreference in information extraction. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP-97)*, pages 163–173, Providence, RI, August.

Kehler, Andrew. 1998. Learning embedded discourse mechanisms for information extraction. In *Proceedings of the AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*. AAAI, Stanford, CA.

Yarowsky, David. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)*, pages 89–95, Las Cruces, June.

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pages 189–196, Cambridge, MA, June.