ALGORITHMS THAT LEARN TO EXTRACT INFORMATION
BBN: DESCRIPTION OF THE SIFT SYSTEM AS USED FOR MUC-7

Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, Ralph Weischedel, and the Annotation Group

BBN Technologies
70 Fawcett Street
Cambridge, MA 02138
weischedel@bbn.com

ABSTRACT

For MUC-7, BBN has for the first time fielded a fully-trained system for NE, TE, and TR; results are all the output of statistical language models trained on annotated data, rather than programs executing hand-written rules. Such trained systems have some significant advantages:

- They can be easily ported to new domains by simply annotating data with semantic answers.
- The complex interactions that make rule-based systems difficult to develop and maintain can here be learned automatically from the training data.

We believe that the results in this evaluation are evidence that such trained systems, even at their current level of development, can perform roughly on a par with rules hand-tailored by experts.

Since MUC-3, BBN has been steadily increasing the proportion of the information extraction process that is statistically trained. Already in MET-1, our name-finding results were the output of a fully statistical, HMM-based model, and that statistical Identifinder™ model was also used for the NE task in MUC-7. For the MUC-7 TE and TR tasks, BBN developed SIFT, a new model that represents a significant further step along this path, replacing PLUM, a system requiring handwritten patterns, with SIFT, a single integrated trained model.

SIFT: AN INTEGRATED TE/TR EXTRACTION SYSTEM

Introduction

At the sentence level, the SIFT system ("Statistics for Information From Text") employs a unified statistical process to map from words to semantic structures. That is, part-of-speech determination, name-finding, parsing, and relationship-finding all happen as part of the same process. This allows each element of the model to influence the others, and avoids the assembly-line trap of having to commit to a particular part-of-speech choice, say, early on in the process, when only local information is available to inform the choice.

The SIFT sentence-level model was trained from two sources:

- General knowledge of English sentence structure was learned from the Penn Treebank corpus of one million words of Wall Street Journal text.
- Specific knowledge about how the target entities and relations are expressed in English was learned from about 500 K words of on-domain text annotated with named entities, descriptors, and semantic relations.

For extraction in a new domain, the names and descriptors of relevant items (persons, organizations, locations, and artifacts) are marked, as well as the target relationships between them that are signaled syntactically. For example, in the phrase "GTE Corp. of Stamford", the annotation would record a

"location-of" connection between the company and the city. The model can thus learn the structures that are typically used in English to convey the target relationships.

While the bulk of the TE/TR task happens within the sentence-level decoder, some further processing was still required to produce TE and TR answer templates. After the names, descriptors, and local relationships had been extracted from the decoder output, a merging process had to be applied to link multiple occurrences of the same name or of alternative forms of the name from different sentences. A second, cross-sentence model was then invoked to try to identify non-local relationships that were missed by the decoder, as when the two entities do not occur in the same sentence. Finally, type and country information was filled in using heuristic tests and a gazetteer database, and output filters were applied to select which of the proposed internal structures should be included in the output. We are actively exploring ways to integrate these post-processing steps more closely with the main model, since an integrated statistical model is the only way to make every choice in a nuanced way, based on all the available information.

Sentence-Level Model

Figure 1 is a block diagram of the sentence-level model showing the main components and data paths. Two types of annotations are used to train the model: semantic annotations for learning about the target entities and relations, and syntactic annotations for learning about the general structure of English. From these annotations, the training program estimates the parameters of a unified statistical model that accounts for both syntax and semantics. Later, when presented with a new sentence, the search program explores the statistical model to find the most likely combined semantic and syntactic interpretation.

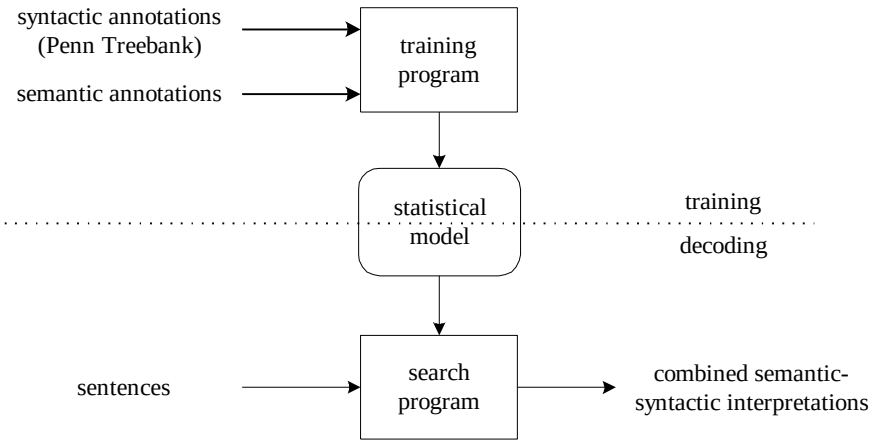

Figure 1: Block diagram of sentence-level model.

Training Data

Our source for syntactically annotated training data was the Penn Treebank (Marcus *et al*., 1993). Significantly, we do not require that syntactic annotations be from the same source, or cover the same domain, as the target task. For example, while the Penn Treebank consists of Wall Street Journal text, the target source for this evaluation was New York Times newswire. Similarly, although the Penn Treebank domain covers general and financial news, the target domain for this evaluation was space technology. The ability to use syntactic training from a different source and domain than the target is an important feature of our model.

Since the Penn Treebank serves as our syntactically annotated training corpus, we need only create a semantically annotated corpus. Stated generally, semantic annotations serve to denote the entities and relations of interest in the target domain. More specifically, entities are marked as either names or

descriptors, with co-reference between entities marked as well. Figure 2 shows a semantically annotated fragment of a typical sentence.

From only these simple semantic annotations, the system can be trained to work in a new domain. To train SIFT for MUC-7, we annotated approximately 500,000 words of New York Times newswire text, covering the domains of air disasters and space technology. (We have not yet run experiments to see how performance varies with more/less training data.)

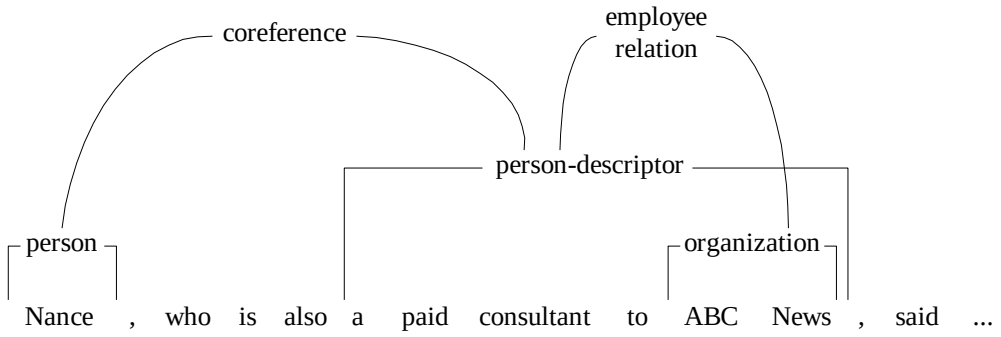

Figure 2: An example of semantic annotation.

Semantic/Syntactic Structure

While our semantic annotations are quite simple, the internal model of sentence structure is substantially more complicated, since it must account for syntactic structure as well as entities and semantic relations. Our underlying training algorithm requires examples of these internal structures in order to estimate the parameters of the unified semantic/syntactic model. However, we do not wish to incur the high cost of annotating parse trees. Instead, we use the following multi-step training procedure, exploiting the Penn Treebank:

1) Train the sentence-level model on the purely syntactic parse trees in the Treebank. Once this step is complete, the model will function as a state-of-the-art statistical parser.
2) For each sentence in the semantically annotated corpus:
    a) Apply the sentence level model to syntactically parse the sentence, constraining the model to produce only parses that are consistent with the semantic annotation.
    b) Augment the resulting parse tree to reflect semantic structure as well as syntactic structure.
3) Retrain the sentence-level model on the augmented parse trees produced in step 2. Once this step is complete, we have an integrated model of semantics and syntax.

Details of the statistical model will be discussed later. For now, we turn our attention to (a) constraining the decoder and (b) augmenting the parse trees with semantic structure.

Constraints are simply bracketing boundaries that may not be crossed by any parse constituent. There are two types of constraints: hard constraints that cannot be violated under any conditions, and soft constraints, that may be violated only if enforcing them would result in no plausible parse. All named entities and descriptors are treated as hard constraints; the model is prohibited from producing any structures that would break up these elements. In addition, we attempt to keep possible appositives together through soft constraints. Whenever there is a co-referential relation between two entities that are either adjacent or separated by only a comma, we posit an appositive and introduce a soft constraint to encourage the parser to keep the elements together.

Once a constrained parse is found, it must be augmented to reflect the semantic structure. Augmentation is a five step process.

1) Nodes are inserted into the parse tree to distinguish names and descriptors that are not bracketed in the parse. For example, the parser produces a single noun phrase with no internal structure for "Lt. Cmdr. David Edwin Lewis". Additional nodes must be inserted to distinguish the descriptor, "Lt. Cmdr.," and the name, "David Edwin Lewis."

2) Semantic labels are attached to all nodes that correspond to names or descriptors. These labels reflect the entity type, such as person, organization or location, as well as whether the node is a proper name or a descriptor.

3) For relations between entities, where one entity is not a syntactic modifier of the other, the lowermost parse node that spans both entities is identified. A semantic tag is then added to that node denoting the relationship. For example, in the sentence "Mary Fackler Schiavo is the inspector general of the U.S. Department of Transportation," a co-reference semantic label is added to the *S* node spanning the name, "Mary Fackler Schiavo," and the descriptor, "the inspector general of the U.S. Department of Transportation."

4) Nodes are inserted into the parse tree to distinguish the arguments to each relation. In cases where there is a relation between two entities, and one of the entities is a syntactic modifier of the other, the inserted node serves to indicate the relation as well as the argument. For example, in the phrase "Lt. Cmdr. David Edwin Lewis," a node is inserted to indicate that "Lt. Cmdr." is a descriptor for "David Edwin Lewis."

5) Whenever a relation involves an entity that is not a direct descendant of that relation in the parse tree, semantic *pointer labels* are attached to all of the intermediate nodes. These labels serve to form a continuous chain between the relation and its argument.

Figure 3 shows an augmented parse tree corresponding to the semantic annotation in Figure 2. Note that nodes with semantic labels ending in "-r" are MUC reportable names and descriptors.

Statistical Model

In SIFT's statistical model, augmented parse trees are generated according to a process similar to that described in Collins (1996, 1997). For each constituent, the head is generated first, followed by the modifiers, which are generated from the head outward. Head words, along with their part-of-speech tags and features, are generated for each modifier as soon as the modifier is created. Word features are introduced primarily to help with unknown words, as in Weischedel *et al.* (1993).

We illustrate the generation process by walking through a few of the steps of the parse shown in Figure 3. At each step in the process, a choice is made from a statistical distribution, with the probability of each possible selection dependent on particular features of previously-generated elements. We pick up the derivation just after the topmost *S* and its head word, *said*, have been produced. The next steps are to generate in order:

1. A head constituent for the *S*, in this case a *VP*.
2. Pre-modifier constituents for the *S*. In this case, there is only one: a *PER/NP*.
3. A head part-of-speech tag for the *PER/NP*, in this case *PER/NNP*.
4. A head word for the *PER/NP*, in this case *nance*.
5. Word features for the head word of the *PER/NP*, in this case *capitalized*.
6. A head constituent for the *PER/NP*, in this case a *PER-R/NP*.
7. Pre-modifier constituents for the *PER/NP*. In this case, there are none.
8. Post-modifier constituents for the *PER/NP*. First a comma, then an *SBAR* structure, and then a second comma are each generated in turn.

This generation process is continued until the entire tree has been produced.

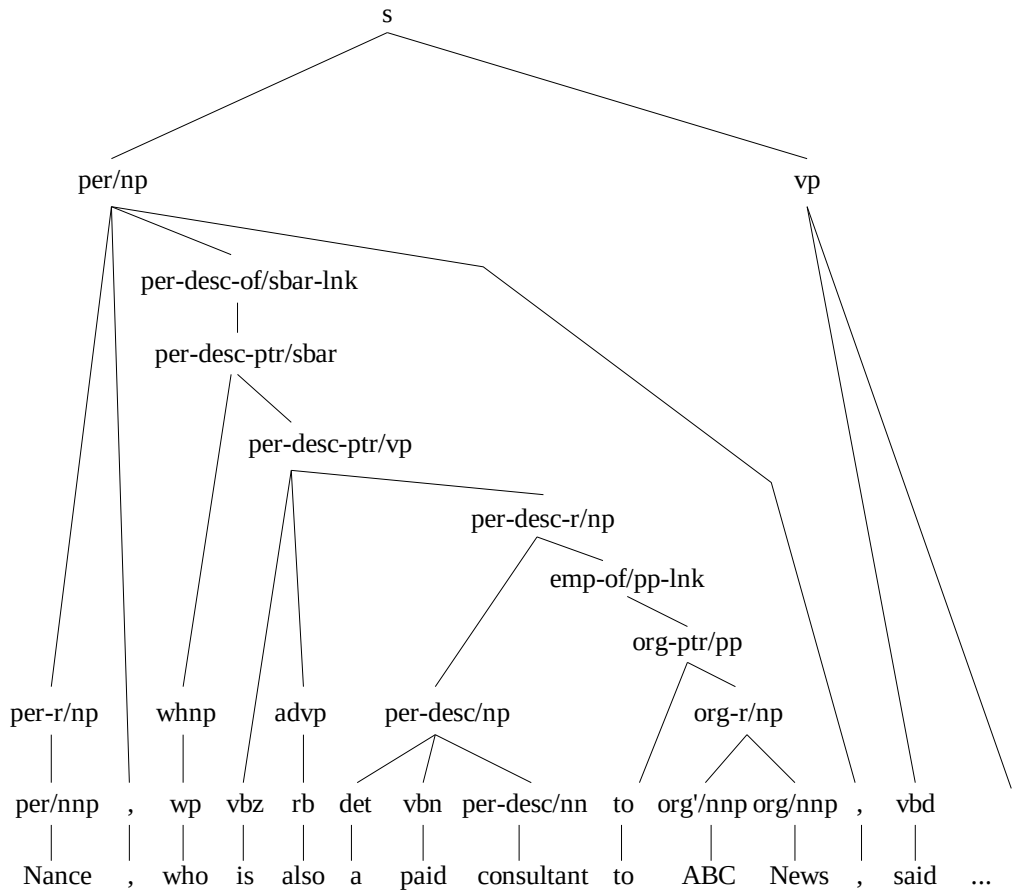

Figure 3: An augmented parse tree.

We now briefly summarize the probability structure of the model. The categories for head constituents, c_h, are predicted based solely on the category of the parent node, c_p:

$$P(c_h \mid c_p), \text{ e.g. } P(vp \mid s)$$

Modifier constituent cateogries, c_m, are predicted based on their parent node, c_p, the head constituent of their parent node, c_{hp}, the previously generated modifier, c_{m-1}, and the head word of their parent, w_p. Separate probabilities are maintained for left (pre) and right (post) modifiers:

$$P_L(c_m \mid c_p, c_{hp}, c_{m-1}, w_p), \text{ e.g. } P_L(per/np \mid s, vp, null, said)$$

$$P_R(c_m \mid c_p, c_{hp}, c_{m-1}, w_p), \text{ e.g. } P_R(null \mid s, vp, null, said)$$

Part-of-speech tags, t_m, for modifiers are predicted based on the modifier, c_m, the part-of-speech tag of the head word , t_h, and the head word itself, w_h:

$$P(t_m \mid c_m, t_h, w_h), \text{ e.g. } P(per/nnp \mid per/np, vbd, said)$$

Head words, w_m, for modifiers are predicted based on the modifier, c_m, the part-of-speech tag of the modifier word , t_m, the part-of-speech tag of the head word , t_h, and the head word itself, w_h:

$$P(w_m \mid c_m, t_{m,} t_h, w_h), \text{ e.g. } P(nance \mid per/np, per/nnp, vbd, said)$$

Finally, word features, f_m, for modifiers are predicted based on the modifier, c_m, the part-of-speech tag of the modifier word , t_m, the part-of-speech tag of the head word , t_h, the head word itself, w_h, and whether or not the modifier head word, w_m, is known or unknown.

$$P(f_m \mid c_m, t_{m,} t_h, w_h, known(w_m)), \text{ e.g. } P(cap \mid per / np, per / nnp, vbd, said, true)$$

The probability of a complete tree is the product of the probabilities of generating each element in the tree. If we generalize the tree components (constituent labels, words, tags, etc.) and treat them all as simply elements, *e*, and treat all the conditioning factors as the history, *h*, we can write:

$$P(tree) = \prod_{e \in tree} P(e \mid h)$$

Training the Model

Maximum likelihood estimates for all model probabilities are obtained by observing frequencies in the training corpus. However, because these estimates are too sparse to be relied upon, they must be smoothed by mixing in lower-dimensional estimates. We determine the mixture weights using the Witten-Bell smoothing method.

For modifier constituents, the mixture components are:

$$\begin{aligned} P'(c_m \mid c_p, c_{hp}, c_{m-1}, w_p) = {} & l_1 \, P(c_m \mid c_p, c_{hp}, c_{m-1}, w_p) \\ & + l_2 \, P(c_m \mid c_p, c_{hp}, c_{m-1}) \end{aligned}$$

For part-of-speech tags, the mixture components are:

$$\begin{aligned} P'(t_m \mid c_m, t_h, w_h) = {} & l_1 \, P(t_m \mid c_m, w_h) \\ & + l_2 \, P(t_m \mid c_m, t_h) \\ & + l_3 \, P(t_m \mid c_m) \end{aligned}$$

For head words, the mixture components are:

$$\begin{aligned} P'(w_m \mid c_m, t_m, t_h, w_h) = {} & l_1 \, P(w_m \mid c_m, t_m, w_h) \\ & + l_2 \, P(w_m \mid c_m, t_m, t_h) \\ & + l_3 \, P(w_m \mid c_m, t_m) \\ & + l_4 \, P(w_m \mid t_m) \end{aligned}$$

Finally, for word features, the mixture components are:

$$\begin{aligned} P'(f_m \mid c_m, t_m, t_h, w_h, known(w_m)) = {} & l_1 \, P(f_m \mid c_m, t_m, w_h, known(w_m)) \\ & + l_2 \, P(f_m \mid c_m, t_m, t_h, known(w_m)) \\ & + l_3 \, P(f_m \mid c_m, t_m, known(w_m)) \\ & + l_4 \, P(f_m \mid t_m, known(w_m)) \end{aligned}$$

Searching the Model

Given a sentence to be analyzed, the search program must find the most likely semantic and syntactic interpretation. More concretely, it must find the most likely augmented parse tree. Although mathematically the model predicts tree elements in a top-down fashion, we search the space bottom-up using a chart based search. The search is kept tractable through a combination of CKY-style dynamic programming and pruning of low probability elements.

Dynamic Programming: Whenever two or more constituents are equivalent relative to all possible later parsing decisions, we apply dynamic programming, keeping only the most likely constituent in the chart. Two constituents are considered equivalent if:

1. They have identical category labels.
2. Their head constituents have identical labels.
3. They have the same head word.
4. Their leftmost modifiers have identical labels.
5. Their rightmost modifiers have identical labels.

Pruning: Given multiple constituents that cover identical spans in the chart, only those constituents with probabilities within a threshold of the highest scoring constituent are maintained; all others are pruned. For purposes of pruning, and only for purposes of pruning, the prior probability of each constituent category is multiplied by the generative probability of that constituent (Goodman, 1997). We can think of this prior probability as an estimate of the probability of generating a subtree with the constituent category, starting at the topmost node. Thus, the scores used in pruning can be considered as the product of:

1. The probability of generating a constituent of the specified category, starting at the topmost node.
2. The probability of generating the structure beneath that constituent, having already generated a constituent of that category.

The outcome of the search process is a tree structure that encodes both the syntactic and semantic structure of the sentence, so that the TE entities and local TR relations can be directly extracted from these sentential trees.

Cross-Sentence Model

The cross-sentence model uses structural and contextual clues to hypothesize template relations between two elements that are not mentioned within the same sentence. Since 80-90% of the relations found in the answer keys connect two elements that are mentioned in the same sentence, the cross sentence model has a narrow target to shoot for. Very few of the pairs of entities seen in different sentences turn out to be actually related. This model uses features extracted from related pairs in training data to try to identify those cases.

It is a classifier model that considers all pairs of entities in a message whose types are compatible with a given relation; for example, a Person and an Organization would suggest a possible Employee_Of. For the three Muc-7 relations, it turned out to be somewhat advantageous to build in a functional constraint, so that the model would not consider, for example, a possible Employee_Of relation for a person already known from the sentence-level model to be employed elsewhere.

Given the measured features for a possible relation, the probability of a relation holding or not holding can be computed as follows:

$$p(rel \mid feats) = \frac{p(feats \mid rel)\,p(rel)}{p(feats)}$$

$$p(\sim rel \mid feats) = \frac{p(feats \mid \sim rel)\,p(\sim rel)}{p(feats)}$$

If the ratio of those two probabilities, computed as follows, is greater than 1, the model predicts a relation:

$$\frac{p(rel \mid feats)}{p(\sim rel \mid feats)} = \frac{p(feats \mid rel)\,p(rel)}{p(feats \mid \sim rel)\,p(\sim rel)}$$

We approximate this ratio by assuming feature independence and taking the product of the contributions for each feature.

$$\frac{p(rel \mid feats)}{p(\sim rel \mid feats)} \approx \frac{p(rel)\prod_i p(feat_i \mid rel)}{p(\sim rel)\prod_i p(feat_i \mid \sim rel)}$$

The cross-sentence feature model applies to entities found by the sentence-level model, which is run over all of the sentence-like portions of the text. An initial heuristic procedure checks for sections of the preamble or trailer that look like sentential material, that should be treated like the body text. There is also a separate handwritten procedure that searches the preamble text for any byline, and, if one is found, instantiates an appropriate employee relationship.

Model Features

Two classes of features were used in this model: structural features that reflect properties of the text surrounding references to the entities involved in the suggested relation, and content features based on the actual entities and relations encountered in the training data.

Structural Features

The structural features exploit simple characteristics of the text surrounding references to the possibly-related entities. The most powerful structural feature, not surprisingly, was distance, reflecting the fact that related elements tend to be mentioned in close proximity, even when they are not mentioned in the same sentence. Given a pair of entity references in the text, the distance between them was quantized into one of three possible values:

Code	*Distance Value*
0	Within the same sentence
1	Neighboring sentences
2	More remote than neighboring sentences

For each pair of possibly-related elements, the distance feature value was defined as the minimum distance between some reference in the text to the first element and some reference to the second.

A second structural feature grew out of the intuition that entities mentioned in the first sentence of an article often play a special topical role throughout the article. The “Topic Sentence” feature was defined to be true if some reference to one of the two entities involved in the suggested relation occurred in the first sentence of the text-field body of the article.

Other structural features that were considered but not implemented included the count of the number of references to each entity.

Content Features

While the structural features learn general facts about the patterns in which related references occur and the text that surrounds them, the content features learn about the actual names and descriptors of entities seen to be related in the training data. The three content features in current use test for a similar relationship in training by name or by descriptor or for a conflicting relationship in training by name.

The simplest content feature tests using names whether the entities in the proposed relationship have ever been seen before to be related. To test this feature, the model maintains a database of all the entities seen to be related in training, and of the names used to refer to them. The “by name” content feature is true if, for example, a person in some training message who shared at least one name string with the person in the

proposed relationship was employed in that training message by an organization that shared at least one name string with the organization in the proposed relationship.

A somewhat weaker feature makes the same kind of test for a previously seen relationship using descriptor strings. This feature fires when an entity that shares a descriptor string with the first argument of the suggested relation was related in training to an entity that shares a name with the second argument. Since titles like "General" count as descriptor strings, one effect of this feature is to increase the likelihood of generals being employed by armies. Observing such examples, but noting that the training didn't include all the reasonable combinations of titles and organizations, the training for this feature was seeded by adding a virtual message constructed from a list of such titles and organizations, so that any reasonable such pair would turn up in training.

The third content feature was a kind of inverse of the first "by name" feature which was true if some entity sharing a name with the first argument of the proposed relation was related to an entity that did *not* share a name with the second argument. Using Employee_Of again as an example, it is less likely (though still possible) that a person who was known in another message to be employed by a different organization should be reported here as employed by the suggested one.

Training

Given enough fully annotated data, with both sentence-level semantic annotation and message-level answer keys recorded along with the connections between them, training the features would be quite straightforward. For each possibly-related pair of entities mentioned in a document, one would just count up the 2x2 table showing how many of them exhibited the given structural feature and how many of them were actually related. The training issues that did arise stemmed from the limited supply of answer keys and that the keys were not connected to the sentence-level annotations.

The government training and dry run data provided 200 messages' worth of TE and TR answer keys. Those answer keys, however, contained strings without recording where in the text they were found. In order to train structural features from that data, we needed the locations of references within the text. A heuristic string matching process was used to make that connection, with a special check to ensure for names that the shorter version of a name did not match a string in the text that also matched a longer version of the same name.

Training the content features, on the other hand, did not require positional information about the references. The plain answer keys could be used in combination with a database of the name and descriptor strings for entities related in training to count up the feature probabilities for actually related and non-related pairs. The string database was collected first, and one-out training was then used, so that the rest of the training corpus provided the string database for training the feature counts on each particular message. The additional training data that was semantically annotated for training the sentence-level model but for which answer keys were not available could still also be used in building up the string database for the content features.

The probabilities based on the final feature counts were smoothed by mixing them with 0.01% of a uniform model.

Contribution of the Cross Sentence Model

When measured on 10 randomly-selected messages from the airplane crash training, the cross sentence model improved TR scores by 5 points. It proved a bit less effective on the 100 messages of the formal test set, improving scores there by only 2 points. (The F score on the formal test set with the cross-sentence model component disabled was 69.33%.)

TE/TR Results

The SIFT system worked by first applying the sentence-level model to each sentence in the message and then extracting entities, descriptors, and relations from the resulting trees, heuristically merging TE elements, applying the cross-sentence model to identify non-local relations, and finally filtering and formatting TE and TR templates for output. The system's score on the TE task was 83% recall with 84% precision, for an F of 83.49%. Its score on TR was 64% recall with 81% precision, for an F of 71.23%.

IDENTIFINDER™: A STATISTICAL NAME-FINDER

Overview of the IdentiFinder™ HMM Model

For the Named Entity task, we used the IdentiFinder™ trained named entity extraction system (Bikel, et. al., 1997), which utilizes an HMM to recognize the entities present in the text.

The HMM labels each word either with one of the desired classes (e.g., person, organization, etc.) or with the label NOT-A-NAME (to represent "none of the desired classes"). The states of the HMM fall into regions, one region for each desired class plus one for NOT-A-NAME. (See Figure 4.) The HMM thus has a model of each desired class and of the other text. Note that the implementation is not confined to the seven name classes used in the NE task; the particular classes to be recognized can be easily changed via a parameter.

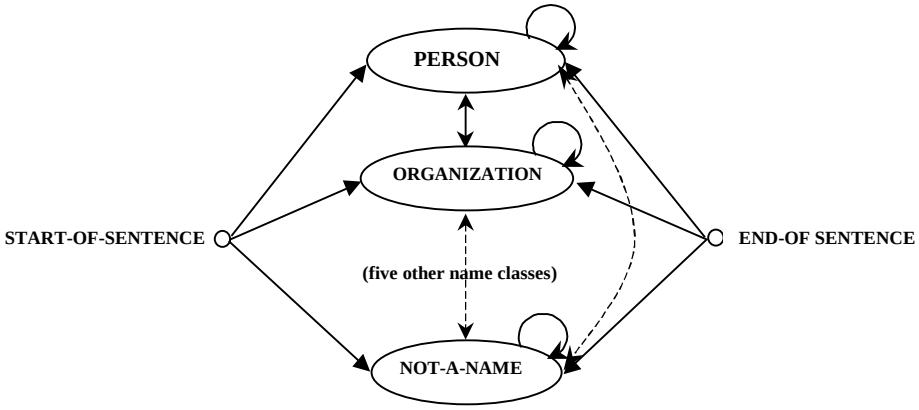

Figure 4: Pictorial representation of conceptual model

Within each of the regions, we use a statistical bigram language model, and emit exactly one word upon entering each state. Therefore, the number of states in each of the name-class regions is equal to the vocabulary size, $|V|$. Additionally, there are two special states, the START-OF-SENTENCE and END-OF-SENTENCE states. In addition to generating the word, states may also generate features of that word. Features used in the MUC-7 version of the system include several features pertaining to numeric expressions, capitalization, and membership in lists of important words (e.g. known corporate designators).

The generation of words and name-classes proceeds in the following steps:

1. Select a name-class *NC*, conditioning on the previous name-class and the previous word.
2. Generate the first word inside that name-class, conditioning on the current and previous name-classes.
3. Generate all subsequent words inside the current name-class, where each subsequent word is conditioned on its immediate predecessor.
4. If not at the end of a sentence, go to 1.

Whenever a person or organization name is recognized, the vocabulary of the system is dynamically updated to include possible aliases for that name. Using the Viterbi algorithm, we search the entire space of all possible name-class assignments, maximizing Pr(***W,F,NC***), the joint probability of words, features, and name classes.

This model allows each type of "name" to have its own language, with separate bigram probabilities for generating its words. This reflects our intuition that:

- *There is generally predictive internal evidence regarding the class of a desired entity.* Consider the following evidence: Organization names tend to be stereotypical for airlines, utilities, law firms, insurance companies, other corporations, and government organizations. Organizations tend to select names to suggest the purpose or type of the organization. For person names, first person names are stereotypical in many cultures; in Chinese, family names are stereotypical. In Chinese and Japanese, special characters are used to transliterate foreign names. Monetary amounts typically include a unit term, e.g., Taiwan dollars, yen, German marks, etc.
- *Local evidence often suggests the boundaries and class of one of the desired expressions.* Titles signal beginnings of person names. Closed class words, such as determiners, pronouns, and prepositions often signal a boundary. Corporate designators (Inc., Ltd., Corp., etc.) often end a corporation name.

While the number of word-states within each name-class is equal to $|V|$, this "interior" bigram language model is ergodic, *i.e.*, there is a non-zero probability associated with every one of the $|V|^2$ transitions. As a parameterized, trained model, for transitions that were never observed, the model "backs off" to a less-powerful model which allows for the possibility of unknown words.

Training

The model as used for the MUC-7 NE evaluation was trained on a total of approximately 790,000 words of NYT newswire data, annotated with approximately 65,500 named entities. In order to increase the size of our training set beyond the 90,000 words of training of airline crash documents provided by the Government, we selected additional training data from the North American News Text corpus. We annotated full articles before discovering a more effective annotation strategy. Since the test domain would be similar to the dry-run domain of air crashes, we used the University of Massachusetts INQUERY system to select 2000 articles which were similar to the 200 dry run training and test documents. About half of our training data consisted of full messages; this portion included the 200 messages provided by the Government as well as 319 messages from the 2000 retrieved by INQUERY. The second half of the data consisted of sample sentences selected from the remainder of the 2000 messages with the hope of increasing the variety of training data. This sampling strategy proved more effective than annotating full messages. Improvement in performance on the (dry run) airline crash test set is shown in Figure 6.

NE Results

Our F-measure for the official evaluation condition, 90.44, is shown as "Text Baseline" in Figure 5. In addition to the baseline condition, we performed some unofficial experiments to measure the accuracy of the system under more difficult conditions. Specifically, we evaluated the system on the test data modified to remove all case information ("Upper Case" in Figure 5), and also on the test data in SNOR (Speech Normalized Orthographic Representation) format ("SNOR" in Figure 5). By converting the text to all upper case characters, information useful for recognizing names in English is removed. Automatically transcribed speech, even with no recognition errors, is harder due to the lack of punctuation, spelling numbers out as words, and upper case in SNOR format.

The degradation in performance from mixed case to all upper case is somewhat greater than that previously observed in similar tests run on generic newswire data (about 2 points). One possible explanation is that

case information is more useful in instances where the test domain is different than the domain of the training set. The degradation from all upper case to SNOR is similar to that previously observed.

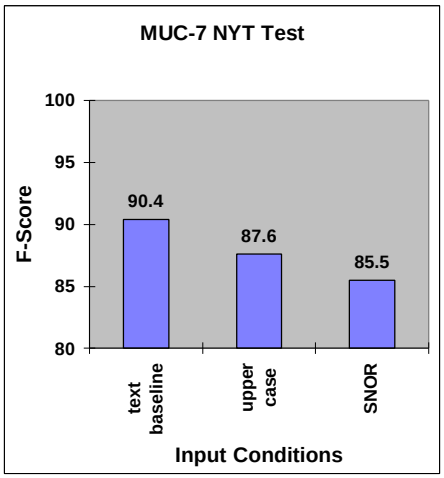

Figure 5: IdentiFinder™ Named Entity Results

We also measured the effect of the training set size on the performance of the system in the air crash domain of the dry run. As is to be expected, increasing the amount of training data results in improved system performance. Figure 6 shows an almost two point increase in F-measure as the training set size was doubled from 91,000 words to 176,000 words. However, the next doubling of the number of words in the training set only resulted in a one point increase in F-measure. This is most likely due to the fact that as training set size increases, the likelihood of seeing a unique name or construction decreases. Though performance might not have peaked, adding more training data will have a progressively smaller effect since the system will not be seeing many constructions which it has not already seen in previous training.

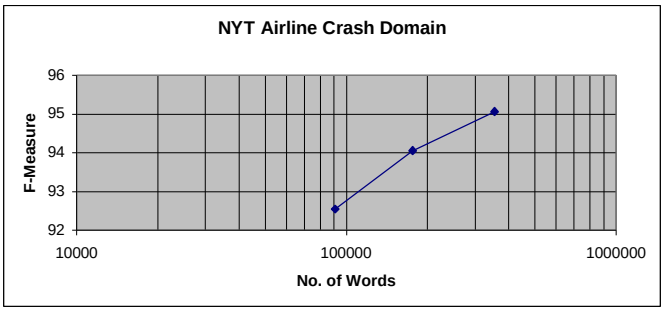

Figure 6: F-Measure Increases With Size of Training Set

SYSTEM WALKTHROUGHS

NE Walkthrough

BBN's Identifinder™ HMM-based approach to named entity recognition did well overall, and it scored 94% on the NE walkthrough article. Of the 7 errors, some can be related directly to choices made in marking our training data. For example, two cases were TV network names, which our annotators typically marked in training as organizations, but which the answer keys did not mark as such in the context where they occurred in the walkthrough article. One error can be attributed to the bigram nature of the current HMM model; while the phrase "Thursday morning" is to be tagged as a date and time, "early Thursday

morning" should instead be tagged as a single time, but the bigram model does not remember "early" when processing "morning". Two other errors were unfamiliar organization names (seen no more than twice in training data) that Identifinder™ guessed were persons, since that guess is more frequently correct in the absence of other clues.

TE and TR Walkthrough

In an integrated system of the sort we used for the TE and TR tasks, the main determinant of performance is the sentence-level model, and the semantic structures that it produces. Secondary but still significant effects on performance come from the post-processing steps that derive TE and TR output from the sentence-level decoder tree:

- Extracting elements and relations
- Merging TE elements
- Searching for additional relations with the cross-sentence model
- Filtering candidate entities and relations for output

This section will follow through selected portions of the walkthrough message, giving examples of the different effects that applied.

Example 1 from paragraph 16 shows a case where everything worked as planned.

```
(SINV
  ...
  (VP
    (VBD said))
  (PER/NP
    (PER/NPA
      (PER/NPP
        (NNP Eric)
        (NNP Stallmer)))
    (, ,)
    (PER-DESC-OF/NP-LINK
      (PER-DESC/NP-R
        (PER-DESC/NPA
          (NN spokesman))
        (ORG-OF/NP-PP-LINK
          (ORG-PTR/PP
            (IN for)
            (ORG/NP
              (ORG/NPA
                (DT the)
                (ORG/NPP
                  (NNP Space)
                  (NNP Transportation)
                  (NNP Association)))
              (LOC-OF/NP-PP-LINK
                (LOC-PTR/PP
                  (IN of)
                  (LOC-PTR/NPA
                    (LOC/NPP
                      (LOC/NPP
                        (NNP Arlington))
                      (, ,)
                      (LOC/NPP
                        (NNP Virginia)))))))))))
```

Example 1: From Walkthrough Article, Paragraph 16

Here the decoder correctly recognized a person name (PER/NPA) bound to a person descriptor (PER-DESC/NP-R). That descriptor contains an organization (ORG/NP) which in turn is linked to a location. The LINK and PTR nodes connect the descriptor with the person, the organization with the person descriptor (and thus indirectly with the person), and the location with the organization. In the post-processing, the person name is extracted, with the descriptor text is linked to it, the organization name is extracted, and the employment relationship noted. The organization is also linked to the nested location; of the two location elements in the LOC phrase, the first is taken as the LOCALE field filler, while the second is looked up in the gazetteer to identify a country in which the locale value is then looked up.

Example 2 from the last paragraph of the message shows the effect of a decoder error.

```
(ORG/NP
 (ORG/NPA
   (ORG/NPP
     (NNP Bloomberg)
     (NNP Information)
     (NNP Television)))
 (, ,)
 (ORG-DESC-OF/NP-LINK
   (ORG-DESC/NP-R
     (ORG-DESC/NPA
       (DT a)
       (NN unit))
     (PP
       (IN of)
       (ORG/NPA
         (ORG/NPP
           (NNP Bloomberg)
           (NNP L.P.))))))
 (, ,)
 (ORG-DESC-OF/NP-LINK
   (ORG-DESC/NP-R
     (ORG-DESC/NPA
       (DT the)
       (NN parent))
     (PP
       (IN of)
       (ORG/NPA
         (ORG/NPP
           (NNP Bloomberg)
           (NNP Business)
           (NNP News))))))
 (, ,))
```

Example 2: From Walkthrough Article, Paragraph 22

Here the sentence-level decoder linked both organization descriptors back to the top-level named organization, while the correct reading would have attached the second descriptor to the nested "Bloomberg L.P.". The post-processing also therefore links both descriptor phrases to "Bloomberg Information Television" internally. Only the longest descriptor, however, is actually output, which in this case results in output of only the mistaken value.

Not surprisingly, a number of the decoder errors that affected output stemmed from conjunctions. In paragraph 19, for example, the manufacturer organization name "Lockheed Space and Strategic Missiles" was incorrectly broken at the conjunction, causing the location relation with Bethesda to be missed.

The cross sentence model is the system component that tries to find further relations beyond those identified by the sentence-level model. In the walk-through article, that component did not happen to succeed in finding any such relations. Example 3 shows the sort of relation that we would like that model to be able to get. There the sentence-level decoder did link Rubenstein to the organization descriptor "company", but since that descriptor was never linked to "News Corporation", the employee relation was

missed. However, since News Corporation is mentioned both in that sentence and the following sentence, an improved cross sentence model would be one way of attacking such examples.

```
(PER-DESC/NP
  (PER-DESC/NP
    (PER-DESC/NPA-R
      (ORG-DESC-OF/NP-LINK
        (ORG-DESC/NP-R
          (NN company)))
      (NN spokesman))
    (PER-OF/NPA-LINK
      (PER-PTR/NPA
        (PER/NPP
          (NNP Howard)
          (NNP J.)
          (NNP Rubenstein)))))
```

Example 3: From Walkthrough Article, Paragraph 5

The last step in processing is the output filter, which heuristically determines whether a proposed constituent should be included in the output. Example 4 from paragraph 1 shows two examples where this filter overrode correct decoder structure.

```
(S
  (ART-DESC/NP-R
    (ART-DESC/NPA
      (DT A)
      (JJ Chinese)
      (NN rocket))
    (ART-PTR/VP
      (VBG carrying)
      (ART-DESC/NPA-R
        (DT an)
        (ORG/NPP
          (NNP Intelsat))
        (NN satellite))))
  (VP
    (VBD exploded)
  ...
```

Example 4: From Walkthrough Article, Paragraph 1

Here the decoder correctly identified both the artifact descriptors "A Chinese rocket" and "an Intelsat satellite", but the output filter chose not to include them. That choice was made because of frequent cases where an indefinite artifact descriptor not linked to any named artifact should not be output; an example is "the last rocket I'd recommend" in paragraph 16. But this example shows that this decision not to output such cases cost us some points.

CONCLUSIONS

BBN's results on the three tasks are summarized in the following table:

Task	*Recall*	*Precision*	*F-Score*
NE	89%	92%	90.44%
TE	83%	84%	83.49%
TR	64%	81%	71.23%

These results, close to those of the best system, demonstrate the power of trained systems for information extraction.

The NE result demonstrates the robustness of IdentiFinder™, the learning algorithm used for NE, to an unknown but similar domain. Further tests also showed its robustness to all upper case input, and input with no punctuation. Our future plans for IdentiFinder™ include

- evaluation in the broadcast news domain, which requires speech input in a much broader domain,
- applying IdentiFinder™ to unsegmented languages, and
- working on performance improvements and improvements in the training process.

The SIFT model, used for TE and TR, employs the Penn Treebank for syntactic information, and thus requires for its training data only the semantic annotation of entities, descriptors, and relationships. Its sentence-level model determines parts of speech, parses, finds names, and identifies semantic relationships in a single, integrated process, with a separate merging model then used to connect information between sentences.

Time was a limiting factor in SIFT's performance, since the decision to field an integrated, fully-trained model was made only in late January, so that SIFT first existed in end-to-end form only as of March 11. That left little time for experiments, or for addressing all issues, such as the handling of nationalities and unnamed TE entities. Given the early stage of development of the SIFT system, we believe that significant performance improvements are still possible. We are also interested in measuring performance as a function of training set size, and in applying SIFT to the broadcast news domain.

ACKNOWLEDGEMENTS

The work reported here was supported in part by the Defense Advanced Research Projects Agency. Technical agents for part of this work were Fort Huachucha and AFRL under contract numbers DABT63-94-C-0062, F30602-97-C-0096, and 4132-BBN-001. 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.

We appreciate the contributions of the Annotation Group at BBN: Ann Albrect, Elizabeth Arentzen, Rachel Bers, Ada Brunstein, Georgina Garcia, Maia Mesnil, and Hugh Walsh.

We thank Michael Collins of the University of Pennsylvania for his valuable suggestions.

REFERENCES

Bikel, Dan; S. Miller; R. Schwartz; and R. Weischedel. (1997) "NYMBLE: A High-Performance Leraning Name-finder." In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Association for Computational Linguistics, pp. 194-201.

Collins, Michael. (1996) "A New Statistical Parser Based on Bigram Lexical Dependencies." In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 184-191.

Collins, Michael. (1997) "Three Generative, Lexicalised Models for Statistical Parsing." In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 16-23.

Marcus, M.; B. Santorini; and M. Marcinkiewicz. (1993) "Building a Large Annotated Corpus of English: the Penn Treebank." *Computational Linguistics*, 19(2):313-330.

Goodman, Joshua. (1997) "Global Thresholding and Multiple-Pass Parsing." In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 11-25.

Weischedel, Ralph; Marie Meteer; Richard Schwartz; Lance Ramshaw; and Jeff Palmucci. (1993) “Coping with Ambiguity and Unknown Words through Probabilistic Models.” *Computational Linguistics*, 19(2):359-382.