HUGHES TRAINABLE TEXT SKIMMER: MUC-3 TEST RESULTS AND ANALYSIS

Charles P. Dolan
Thomas V. Cuda
Seth R. Goldman
Alan M. Nakamura

Hughes Research Laboratories
3011 Malibu Canyon Road M/S RL96
Malibu, CA 90265

Test results

Figure 1 gives the official results for the Hughes Trainable Text Skimmer used for MUC3 (TTS-MUC3). TTS is a largely statistical system, using a K-Nearest Neighbor classifier with the output of a shallow parser as features. (See the System Summary section of this volume for a detailed description of TTS-MUC3). The performance, on a slot by slot basis is, therefore, what one might expect: the pure set fills such as "Incident Type" and "Category" have much better performance than the string fills such as "Human Target." In addition, we can see that "Incident Date" and "Incident Location," for which special code was written, have performance above that of the string fills.

SLOT	REC	PRE	OVG	FAL
template-id	74	53	47	
incident-date	41	55	1	
incident-type	46	63	0	1
category	55	55	21	27
indiv-perps	17	39	23	
org-perps	30	35	33	
perp-confidence	20	28	32	5
phys-target-ids	18	37	33	
phys-target-num	18	41	29	
phys-target-types	13	28	33	1
human-target-ids	25	17	65	
human-target-num	15	20	20	
human-target-types	31	20	65	10
target-nationality	0	*	*	0
instrument-types	0	0	50	0
incident-location	37	54	0	
phys-effects	19	35	50	2
human-effects	5	12	58	2
MATCHED ONLY	41	36	38	
MATCHED/MISSING	31	36	38	
ALL TEMPLATES	31	22	62	
SET FILLS ONLY	29	34	41	2

Figure 1: Official TST2 Score report

Distribution of labor

One calendar month and approximately three (3) person months were spent on MUC3. Before MUC3, we had constructed a text database facility and the pattern matcher used for

shallow parsing. Therefore much of the time for MUC3 was spend evaluating alternatives for the statistical engine. Approximately 45% of the time was spent developing code for ideas that were not used in the final system. Of the remaining time, 30% was spent developing code to extract and format information for MUC3 templates (including code to parse the templates of the DEV corpus), 15% was spent coding and tuning the K-Nearest Neighbor classifier, and 10% was spent creating phrasal patterns, either by hand or extracting them automatically from the templates for the DEV corpus.

Test settings

The test settings for TTS-MUC3 were tuned to maximize recall. This resulted in roughly equal recall and precision. Some results in the companion paper in the System Summary section of this volume indicate that we might tune TTS-MUC3 for higher precision at the expense of recall. However, we believe that there are enough different algorithms that might substantially improve the performance of TTS that evaluating such trade-offs is premature. For the official test, we used K=12 in the pattern classifier. The pattern classifier returns a set of hypotheses for various set and string fills. The hypotheses are returned with strengths between 0.0 and 1.0 which are then compared to a threshold; all the thresholds on the feature extraction were extremely low (e.g., 0.1).

Limiting factors

The limiting factor for the Hughes TTS-MUC3 system was time. The K-Nearest Neighbor classifier is surprisingly effective, but there are many variations that we did not have time to try. With a small amount of extra time we could make small improvements there. In addition, we suspect that our algorithm for grouping sentences into topics was responsible for many of our errors. However, improving this portion of the system will take much more time and, we believe, will require the addition of domain knowledge into the processing.

Training

The training regimen was extremely simple. A word frequency analysis was performed on the DEV corpus, and we selected those words that occurred between 10 and 105 times as our content bearing words, resulting in about 1000 such words. These words were then grouped by hand into approximately 400 conceptual classes. In addition, words were added to the lexicon for numbers, ordinals, roman numerals, compass points, etc. The lexicon and the DEV templates were used to drive the construction of phrases. Phrases were created from string fills by substituting conceptual classes for words. For example, `"SIX JESUITS"` would drive the creation of the phrase, `(:NUMBER-W :RELEGIOUS-ORDER-W)`. The type of the string fill served as the semantic feature for the phrase.

For some phrases there where conflicts, for example, many phrases that might be mapped to `:ACTIVE-MILITARY` as a human target, might also be mapped to `:STATE-SPONSORED-VIOLENCE-INDIV` as a perpetrating individual. For these phrases, the most frequent usage was chosen. After creating a large number of phrases automatically (approximately 1000), a set of hand constructed phrases was added to augment and repair that set (approximately 200).

All the stories in the DEV corpus were used to build the case memory, however, the number of cases per different "Type of Incident" was limited to 35. This means that once 35 cases of a particular incident type (i.e., Murder) had been seen, future cases of this type were ignored. This attempt to balance the training data was necessary because the number

of stories for each type of incident varied greatly. By restricting the maximum stories per topic, we tended to ignore many of the later stories in the training set.

Domain independent modules

All the modules in TTS-MUC3 are domain independent. However, all the modules except the date extraction module, require some amount of training. Besides the training described above, the location extraction module requires a location database, including what locations contain what other locations. The overhead for constructing such training sets and databases is quite large, but we feel that for applications of sufficient leverage, good user interface design will ease the burden of constructing the training set and reduce the time for deploying TTS in new domains. In addition, integration with on-line data sources such as map databases will eliminate the burden of creating special data files for natural language processing.