HUGHES TRAINABLE TEXT SKIMMER: DESCRIPTION OF THE TTS SYSTEM AS USED FOR MUC-3

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

OVERVIEW OF CAPABILITIES

The objective of the Hughes Trainable Text Skimmer (TTS) Project is to create text skimming software that: (1) can be easily re-configured for new applications, (2) improves its performance with use, and (3) is fast enough to process megabytes of text per day. The TTS-MUC3 system is our first full scale prototype.

TTS-MUC3 incorporates semi-automated lexicon generation and almost fully automated phrase pattern generation. Associative retrieval from a case memory provides raw data for computing set fills and string fills. TTS-MUC3's modular process model integrates the results of case memory retrieval over sentences from multiple stories, extracts the date and location of incidents, and computes cross-reference information for various slots.

SYSTEM COMPONENTS

The TTS-MUC3 system incorporates a number of different modules shown below:

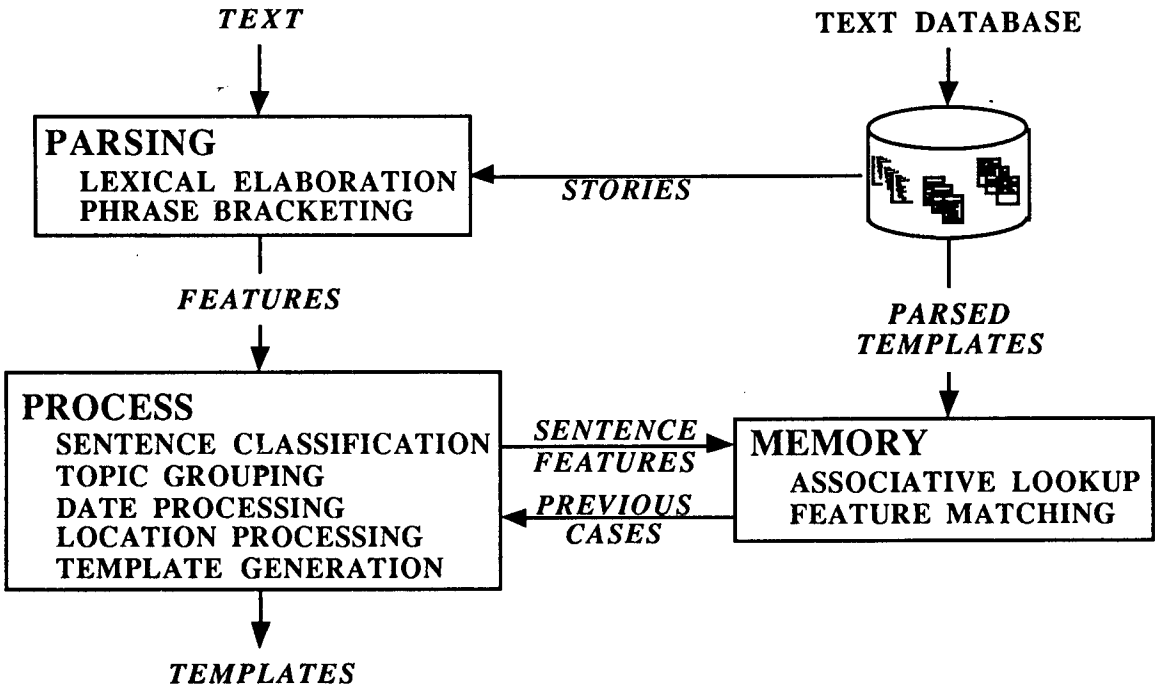


Figure 1: Hughes TTS System Block Diagram

Text Database

The text database provides the capability to retrieve a fragment of text from a large collection that may be spread over multiple disk files. In TTS-MUC3, the text database was used to store: (1) the database of training stories, (2) the database of testing stories, (3) the database of training templates for user browsing, and (4) the database of parsed templates for use by the associative case memory. Retrievals from the text database may return: (1) a raw text string (used for templates), (2) a recursive token structure with individual words at the leaf nodes (used for stories), or (3) an s-expression (used for parsed templates).

Phrasal Parser

The phrasal parser is a fast, shallow, conceptual parser. The parser accepts a token structure, a lexical hierarchy, and a phrase pattern set. The parser returns an ordered list of text features. A text feature includes: (1) a member of the concept hierarchy, (2) the string covered by the phrase, and (3) a recursive token structure spanning the tokens covered by the phrase.

Lexicon entries are created by adding word stems to a concept hierarchy as follows,

```
(ks:isa h-lex "PRIEST"      :religious-individual-w)
(ks:isa h-lex "MISSIONARY"  :religious-individual-w)
(ks:isa h-lex "CONFERENCE"  :conference-w)
(ks:isa h-lex "SUMMIT"      :conference-w)
(ks:isa h-lex "RECEPTION"   :conference-w)
```

Phrasal patterns may reference either elements of the concept hierarchy, or specific words:

```
(ph:defpattern (net ? h-con) (:determiner :small-number
                              :unidentified-w :human-group-w)
  :civilian)

(ph:defpattern (net ? h-con) (:civilian-w "FROM" :number-w
                              :spanish-name-w "AREA")
  :civilian)

(ph:defpattern (net ? h-con) (:public-w :communication-device-w
                              :building-w)
  :communications)
```

The features are extracted using a depth first search of the patterns, with a preference for patterns that have specific words over those the have only concept names and a preference for longer patterns.

Case Memory

The case memory takes an ordered list of text features and returns the K-nearest neighbors. For TTS-MUC3, K was 12 and the metric was the Euclidean distance in a binary vector space. The case memory also accepts a set of slots to fill (set fill and string fill). For each sentence, the case memory returns weighted suggestions for filling each of the requested slots. Case indices are kept in main memory. Parsed templates, used for computing slot fillers, are loaded as needed.

Process Model

The process model has four phases: (1) memory access, (2) topic grouping, (3) slot filling, and (4) template generation. There is also an initial training phase which initializes the case memory.

Training Phase

The training phase uses the provided templates to build up the phrase lexicon and the case memory. Phrases are generated from the fillers for the template slots. Cases are generated from the sentences that provided the fillers. The word lexicon is generated by performing a word frequency analysis on the raw text. For TTS-MUC3, all words that occurred between 10 and 105 times were included in the lexicon.

Memory Access

For each sentence of a story, the memory access phase queries the case memory to obtain suggestions for all slots. The resulting structure contains all the weighted suggestions and the source cases.

Topic grouping and relevance assessment

Topic grouping (analogous to discourse processing) is based on the *TYPE OF INCIDENT* slot. The weight for each type of incident is computed for every sentence. The weights are then passed through a competitive filter, resulting in binary signals. The competitive filter first normalizes the topic weights using a Gaussian mask on a sentence by sentence basis, then computes the best topic. A topic is a set of contiguous sentences with the same computed value for *TYPE OF INCIDENT*.

Figure 2 shows the inputs and outputs to the topic grouping process. Note that moderately high evidence of kidnapping throughout the story is suppressed in favor of the bombing interpretation, which turns out to be correct. This filter used is topic grouping is designed to pick out signals that are high but that "drop out" from time to time, as one can see in the smoothing over the arson signal.

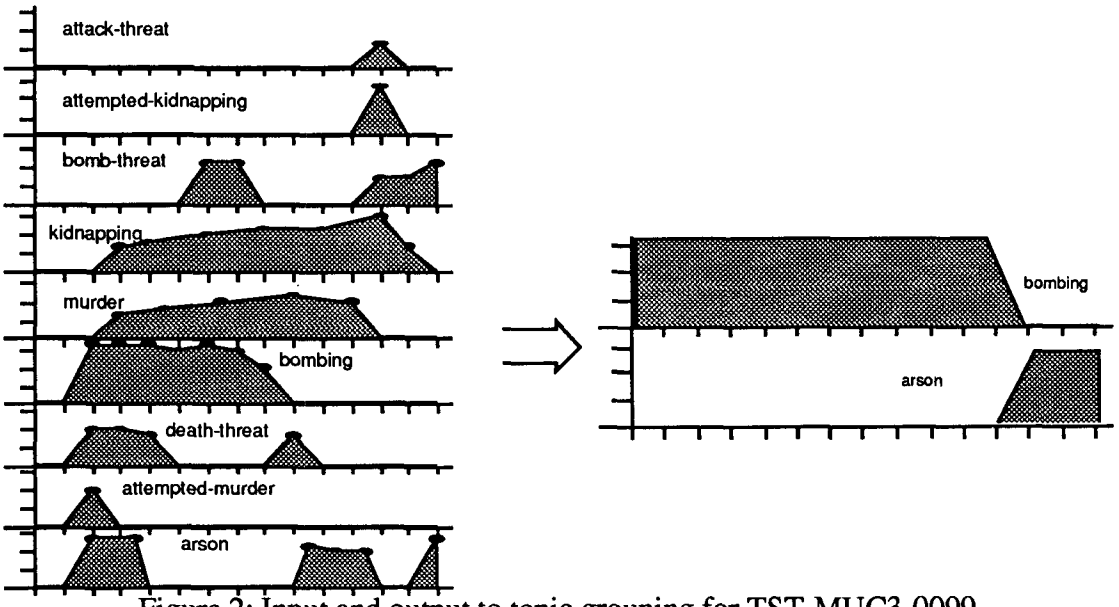


Figure 2: Input and output to topic grouping for TST-MUC3-0099

Slot filling

Slot filling consists of five parts: (1) pure set fills, (2) string fills, (3) cross-referenced slots, (4) date extraction, and (5) location extraction. The first three parts consider only relevant sentences. A relevant sentence shares the same topic with the previous sentence or contains no competing topic. There are two distinct types of processing for slot filling. Most slots are filled using hypotheses returned by the associative memory, two, date and location, are filled by domain specific procedures. Three types of slots are filled from the associative memory:

1. Set fills—Pure set fills are computed by averaging the weights over all sentences for a given topic and picking the highest score.

2. String fills—String fills are computed in a similar manner to set fills. The difference is that the suggestions returned by the case memory are subject to a threshold on the weights. For the official run of TTS-MUC3, the string fill threshold was set at 0.1.

3. Cross reference generation—Cross reference generation is performed by choosing the most likely tag (as suggested by the case-memory) for the sentence that contains the string fill.

Date processing

For date extraction, all sentences within a topic are scanned for absolute or relative date references. Absolute date references are combined into a range. Absolute dates are preferred over relative dates within a given sentence. Relative date references are interpreted with respect to either the current date specification for a story (if one has been found) or the story date line.

Location processing

For location extraction, all sentences within a topic are scanned for known location names. The resulting list of location names is then searched for a maximal, legal, location containment chain.

EXAMPLE RUN

For the first sentence in TST-MUC3-0099,

```
"[TEXT] POLICE HAVE REPORTED THAT TERRORISTS TONIGHT BOMBED
THE EMBASSIES OF THE PRC AND THE SOVIET UNION."
```

TTS-MUC3 extracts the following features,

```
((feature :police-w "POLICE" #<token:POLICE>)
 (feature :statement-w "REPORTED" #<token:REPORTED>)
 (feature :terrorist-act-indiv "TERRORISTS"
  #<token-terrorist-act-indiv>)
 (feature :time-of-day-w "TONIGHT" #<token:TONIGHT>)
 (feature :explosive-w "BOMBED" #<token:BOMBED>)
 (feature :embassy-w "EMBASSIES" #<token:EMBASSIES>)
 (feature :place-name "PRC" #<token-place-name>)
 (feature :place-name "SOVIET UNION" #<token-place-name>))
```

Based on semantic features such as :POLICE-W and :PLACE-NAME, the following template (along with approximately 11 others) is retrieved from memory.

```
0.  MESSAGE ID                          DEV-MUC3-0174 (BELLCORE)
1.  TEMPLATE ID                         1
2.  DATE OF INCIDENT                    16 APR 89
3.  TYPE OF INCIDENT                    ATTEMPTED BOMBING
4.  CATEGORY OF INCIDENT                TERRORIST ACT
5.  PERPETRATOR: ID OF INDIV(S)         "MIGUEL RODOLFO AGUILAR FLORES"
                                        "WOUNDED MAN"
6.  PERPETRATOR: ID OF ORG(S)           "HONDURAN LEFT"
7.  PERPETRATOR: CONFIDENCE             SUSPECTED OR ACCUSED BY GOVERNMENT:
                                           "HONDURAN LEFT"
8.  PHYSICAL TARGET: ID(S)              "U.S. EMBASSY WAREHOUSE"
9.  PHYSICAL TARGET: TOTAL NUM          1
10. PHYSICAL TARGET: TYPE(S)            DIPLOMAT OFFICE OR RESIDENCE:
                                           "U.S. EMBASSY WAREHOUSE"
11. HUMAN TARGET: ID(S)                 -
12. HUMAN TARGET: TOTAL NUM             -
13. HUMAN TARGET: TYPE(S)               -
14. TARGET: FOREIGN NATION(S)           UNITED STATES:
                                          "U.S. EMBASSY WAREHOUSE"
15. INSTRUMENT: TYPE(S)                 *
16. LOCATION OF INCIDENT                HONDURAS: TEGUCIGALPA (CITY)
17. EFFECT ON PHYSICAL TARGET(S)        NO DAMAGE: "U.S. EMBASSY WAREHOUSE"
18. EFFECT ON HUMAN TARGET(S)           -
```

Stored along with the story are the features extracted from DEV-MUC3-0174 that were used to index the template.

```
((feature (feature :police-w "POLICE")
 (feature :statement-w "BELIEVE")
 (feature :explosive-w "BOMB")
 (feature :depart-w "GOING")
 (feature :place-name "U.S.")
 (feature :embassy-w "EMBASSY")
 (feature :commercial-target-w "WAREHOUSE")
 (feature :human-individual-w "MEMBERS")
 (feature :terrorist-act-org "THE HONDURAN LEFT")
 (feature :civilian-w "PEOPLE")
 (feature :month-name-w "APRIL")
 (feature :favoring-w "FOR")
 (feature :place-name "UNITED STATES"))
```

Comparing the strings in the retrieved template with the strings for the indexing features, TTS-MUC3 looks for a feature in the new sentence that matches the features (FEATURE :EMBASSY-W "EMBASSY"). Using the semantic feature, :EMBASSY-W, TTS-MUC3 proposes (FEATURE :EMBASSY-W "EMBASSIES" #<TOKEN:EMBASSIES>), as a hypothesis for the physical target in the new story.

Processing proceeds in a like manner for the rest of the story to produce the following template,

```
0.  MESSAGE ID                          TST1-MUC3-0099
1.  TEMPLATE ID                         1
2.  DATE OF INCIDENT                    25 OCT 1989
3.  TYPE OF INCIDENT                    BOMBING
4.  CATEGORY OF INCIDENT                TERRORIST ACT
5.  PERPETRATOR: ID OF INDIV(S)         *
6.  PERPETRATOR: ID OF ORG(S)           "SHINING PATH"
7.  PERPETRATOR: CONFIDENCE             CLAIMED OR ADMITTED: "SHINING PATH"
8.  PHYSICAL TARGET: ID(S)              "PRC EMBASSY"
                                        "CAR"
                                        "VEHICLES"
                                        "USSR EMBASSY"
9.  PHYSICAL TARGET: TOTAL NUM          PLURAL
10. PHYSICAL TARGET: TYPE(S)            DIPLOMAT OFFICE OR RESIDENCE:
                                          "USSR EMBASSY"
                                        OTHER: "VEHICLES"
                                        OTHER: "CAR"
                                        DIPLOMAT OFFICE OR RESIDENCE:
                                          "PRC EMBASSY"
11. HUMAN TARGET: ID(S)                 *
12. HUMAN TARGET: TOTAL NUM             *
13. HUMAN TARGET: TYPE(S)               *
14. TARGET: FOREIGN NATION(S)           *
15. INSTRUMENT: TYPE(S)                 *
16. LOCATION OF INCIDENT                PERU: LIMA (DEPARTMENT)
17. EFFECT ON PHYSICAL TARGET(S)        SOME DAMAGE: "USSR EMBASSY"
                                        SOME DAMAGE: "VEHICLES"
                                        SOME DAMAGE: "CAR"
                                        SOME DAMAGE: "PRC EMBASSY"
18. EFFECT ON HUMAN TARGET(S)           *
```

TTS-MUC3 produces a reasonably good fill for this template. Three features are worth noting. First, the string fills "PRC EMBASSY" and "USSR EMBASSY" are extracted from sentences after the introductory sentence,

> "A CAR-BOMB EXPLODED IN FRONT OF THE PRC EMBASSY, WHICH IS IN THE LIMA RESIDENTIAL DISTRICT OF SAN ISIDRO. MEANWHILE, TWO BOMBS WERE THROWN AT A USSR EMBASSY VEHICLE THAT WAS PARKED IN FRONT OF THE EMBASSY LOCATED IN ORRANTIA DISTRICT, NEAR SAN ISIDRO."

The second feature worth noting is that "CAR" is picked up as a target, even though it is actually a part of the instrument "CAR-BOMB". The reason for this mistake is a deficiency in the phrases that pick out semantic features.

The third feature is that TTS-MUC3 produced only one template where there should have been two bombings. This merging of templates with the same incident type is an inevitable result of the topic grouping used in TTS-MUC3.

SENSITIVITY TO TRAINING SET

To test the sensitivity to different training sets, we loaded the associative memory with different templates from the development corpus. To show the difference in performance, Table 1 shows the overall recall and performance for the MATCH/MISSING row of the scoring, with various

portions of the training data loaded. Whenever a training set is loaded, the number of case with a given incident type is limited to prevent sampling bias. For Table 1 the maximum cases per topic is 10. Note that these training sets are much smaller than the full 1200 stories in the DEV corpus, and therefore the recall performance is substantially lower than the 31% achieved with the full training set on TST2.

Training Stories	Recall	Precision
1-100	20	43
101-175	16	40
476-550	22	50
551-625	16	38
626-700	16	32

Table 1: Recall and precision for various training sets with 10 cases per incident type

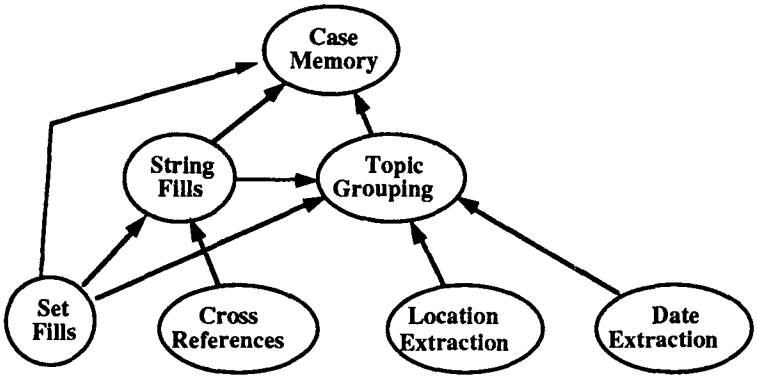
Training Stories	Recall	Precision
1-100	16	43
101-175	14	40
476-550	17	45
551-625	8	36
626-700	19	45

Table 2: Recall and precision for various training sets with 4 cases per incident type

Table 2 presents results similar to Table 1 but with a maximum of four (4) cases per topic. Intuitively, one would imagine that recall at least would fall drastically. Table 2 confirms that intuition as, for all but one training set, the recall drops when fewer cases per incident type are loaded. Both Tables 1 and 2 are the result of running the first 10 stories in the TST1 corpus through TTS-MUC3. The first ten stories contain two ARSON templates, and even after limiting the number of cases per topic to 10, ARSON still has fewer than half as many cases as the more common types: ATTACK, MUDER, BOMBING, and KIDNAPPING. However, when the number of cases per topic is limited to four (4), ARSON is perfectly balanced with the others. This under representation of ARSON in the training data may account for the anomaly between Tables 1 and 2 for stories 626-700.

SUMMARY

To understand the performance of TTS-MUC3, one should look at the the inter-dependence between the various processing modules. Figure 3 shows these dependencies. Each module points to the modules it depends on. Our contention is that improving a module will enable improvement of the behavior of its dependents.



For example, the case memory alone has recall and precision rates above 50%. Subsequent processing results in information loss that accounts for our final rates of 31% and 36%, respectively.

Figure 3: Module Dependency Graph

We believe that this ability to analyze, from a system wide perspective, where the errors occur is unique to TTS. From Figure 3, we can see that even a perfect case memory would not completely solve all performance problems, as every other component depends on topic grouping. Therefore we conclude that topic grouping is the system component where the most work is needed. We might also deduce that in topic group, we will find the largest leverage for adding knowledge to the processing. This conclusion concurs with conventional wisdom in natural language, that understanding text across sentence boundaries requires more knowledge that understanding within a sentence.