A Server for Real-Time Event Tracking in News

Ralf D. Brown
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3890 USA
ralf@cs.cmu.edu

1. INTRODUCTION

As the flood of information continues to grow, it becomes ever more necessary to extract just the portion of the flow which is of interest to each user. The Topic Detection and Tracking (TDT) project [1, 3, 6, 5] addressed and continues to address this need, but has been of necessity applied in a batch-processing context on a static collection. What is required for topic detection and tracking to be of utility to end-users is a real-time system which operates on a live stream of information. This paper describes the extension and modification of a batch-oriented tracking system into a real-time server for event detection, event tracking, document summarization, and translation.

2. ARCHITECTURE

To allow sharing of resources such as the collection of news stories between multiple users, a client-server architecture is used. For added flexibility, not all functionality need be implemented in the central server; in addition to user-interface clients, several types of service-provider clients are supported. Service-provider clients initially connect to the server and authenticate themselves in the same manner as a user would, but then send additional commands to identify themselves as service providers and which service(s) they provide. It is also possible for a service-provider client to act as an interface to a news source such as a modem-based newswire service, extracting stories from the news source and adding them to a specified collection on the server. Such external interface programs will likely be the primary source of live data; the current prototype server by itself is only capable of retrieving specified web pages, either once on demand or at regular intervals.

For a multilingual context with high-volume news streams, one needs more than simple alerting – the system must also translate stories which are not in the user's language and generate summaries of sets of relevant stories. Since the language(s) of the news streams that will be used is not fixed before-hand, translation and summarization are handled by external processes. The server can load and/or establish network connections to one or more instances of our multi-engine machine translation system[1] [2, 4] and will invoke the proper instance when a client requests a translation from one language into another. Similarly, service-provider clients provide summarization services for specified languages, with the server routing the summarization request to the appropriate summarizer (or a very rudimentary language-independent summarizer built into the server if there is no service provider available for a particular language). Tracking requests are processed not only by the internal tracking engines, but are also passed to any external clients which have registered as trackers.

Since having a network interface implies that the server can be accessed from anywhere in the world, each user has an account with an associated set of privileges that can, for example, restrict a guest account to minimal read-only access without the ability to view the list of users currently logged in to the server.

Although the primary interaction between clients and the server consists of synchronous request-response pairs, notification of newly detected or tracked events occurs asynchronously on a separate network connection. Using a separate connection permits the notification to be broadcast to all interested clients even if a request-response interaction is currently in progress with a particular client, and allows a dedicated thread or process on the client side to monitor the real-time notifications. The main notifications are `NEWEVENT`, which indicates that some (unspecified) event differing from all other current news stories has occurred, and `TRACKED`, which indicates that the story discusses an event which was previously defined using one or more example instances and optionally some counterexamples. Secondary notifications are `SHUTDOWN` and passed-through requests for tracking, summarization, or document clustering.

Because the live data stream continues 24 hours per day but the client may not be logged in all the time, all notifications are permanently stored in a file, and clients may later request retrieval of the notifications that were missed while the client was not active. Thus, for example, a user can get a listing of all "interesting" stories received overnight when (s)he logs in each morning.

Figure 1 summarizes the various components of this distributed system. It shows the server communicating with

[1]The multi-engine translator currently supports translation between English and French, Spanish, German, Mandarin Chinese, Croatian, and Haitian Creole, with "toy" versions of Korean and Slovenian also available.

.

multiple tracking, detection, summarization, clustering, and translation servers (some on the same computer, some remote), as well as a web crawler and newswire retrieval engine for adding news stories to the server. Multiple users, each with a separate client program, access the server simultaneously.

3. IMPLEMENTATION

The first hurdle in implementing the TDT server was to create a real-time version of the topic tracking system. The pre-existing system had been batch-oriented because the definition of the TDT tracking task requires each event to be processed as though it were the only event, generating a separate output file for each event. For efficiency, the entire collection was loaded into memory and then multiple tracking passes (one per event) made over the collection. Fortunately, the individual tracking engines were structured with separate decision procedures and control structures to allow multi-engine combination, so only the control structure needed to be modified to create a tracking system which operates incrementally as news stories are loaded into the in-memory collection. To handle a live data stream – which, unlike a static collection, is potentially unbounded – stories which are too old to further affect the training phase of the tracking engines are removed from the in-memory collection after each addition of new stories.

Because the removal of documents from the in-memory collection would make older stories inaccessible to clients fairly quickly in a high-volume application, the documents which are removed from memory are stored on disk to allow retrieval by their document ID (which is provided in each notification message). At this time, the permanent repository is not indexed for retrieval by document content or metadata such as timestamp and language, but there is no inherent obstacle to adding such indexing.

Once the incremental version of the tracker was operational, a network interface was added. The network protocol for the server uses plain-text commands and result codes to facilitate debugging – one can simply "telnet" to the server and start entering commands (a simple command-line client which prompts for a command's arguments was also written). Commands to the server include user authentication, enabling/disabling asynchronous alerts, adding and removing documents from a collection, management of multiple collections, requests to track a particular event, lookup of documents by date/time or boolean query, translation, summarization, fetching of web pages, server statistics, and registration as a tracker, summarizer, new-event detector, or clustering engine.

The final step in producing the full system is the implementation and integration of various clients, which is currently in progress. A summarizer for English, Spanish, Mandarin, and Japanese has been adapted into a service-provider client for the TDT Server, as has a document-clustering program. A user-interface client is near completion, and a new-event detection client is planned. The central server is written in C++, and the various clients are implemented in C, C++, and Java.

4. REFERENCES

[1] J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic Detection and Tracking Pilot Study Final Report. In *Proceedings of the DARPA Broadcast News Transcription and Understranding Workshop*, Feb 1998.

[2] R. D. Brown. Example-Based Machine Translation in the PANGLOSS System. In *Proceedings of the Sixteenth International Conference on Computational Linguistics*, pages 169–174, Copenhagen, Denmark, 1996. `http://www.cs.cmu.edu/~ralf/papers.html`.

[3] J. Carbonell, Y. Yang, J. Lafferty, R. D.Brown, T. Pierce, and X. Liu. CMU report on TDT-2: Segmentation, Detection and Tracking. In *Proceedings of the DARPA Broadcast News Workshop*, pages 117–120, San Francisco, CA, 1999. Morgan Kaufmann Publishers, Inc.

[4] C. Hogan and R. E. Frederking. An Evaluation of the Multi-engine MT Architecture. In *Machine Translation and the Information Soup: Proceedings of the Third Conference of the Association for Machine Translation in the Americas (AMTA '98)*, volume 1529 of *Lecture Notes in Artificial Intelligence*, pages 113–123. Springer-Verlag, Berlin, October 1998.

[5] Y. Yang, T. Ault, T. Pierce, and C. W. Lattimer. Improving Text Categorization Methods for Event Tracking. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[6] Y. Yang, J. Carbonell, R. D. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning Approaches for Detecting and Tracking News Events. *IEEE Intelligent Systems*, 14(4):32–43, July/August 1999. Special Issue on Applications of Intelligent Information Retrieval.

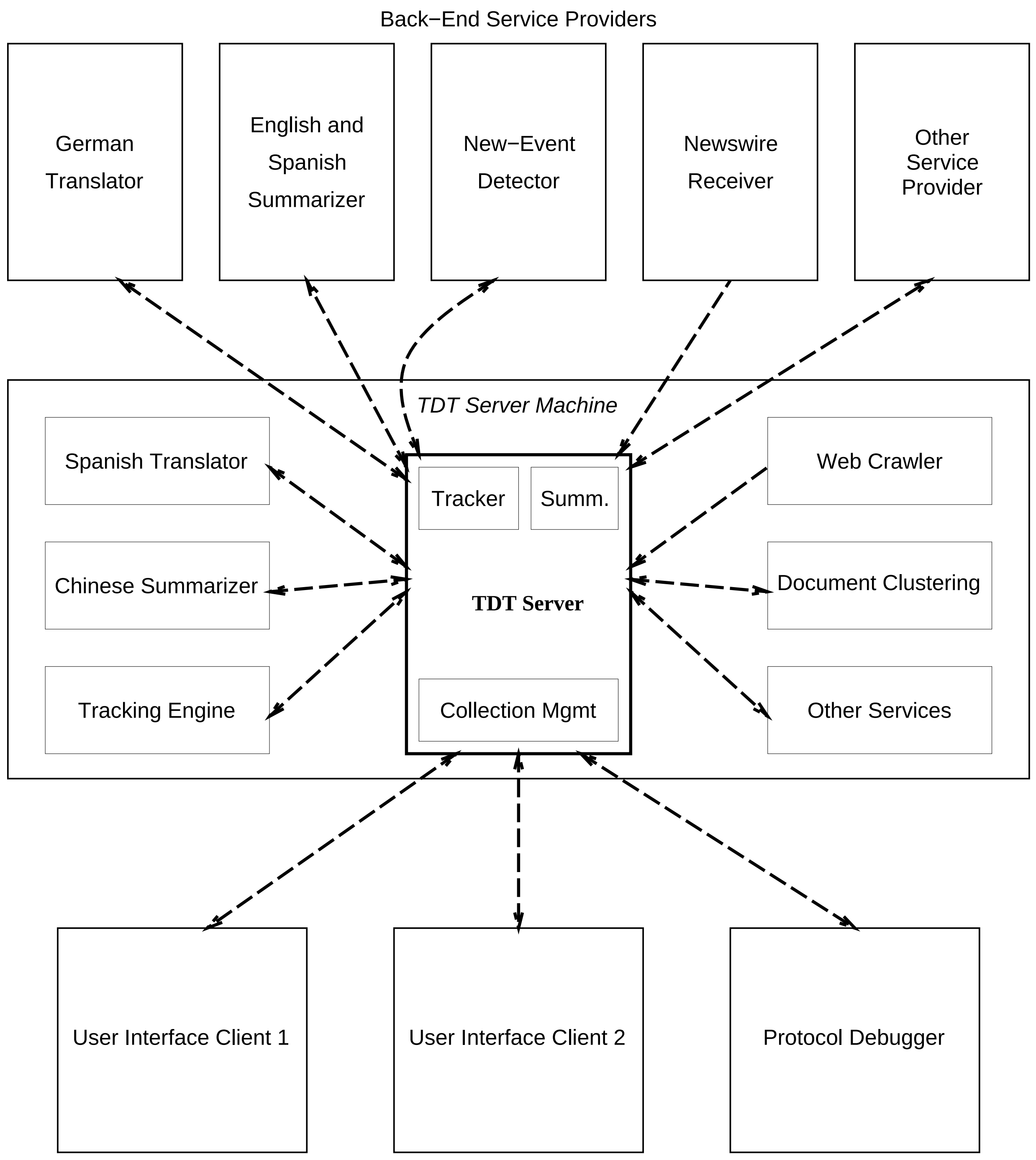

Figure 1: The Distributed TDT-Server Architecture