Infrastructure and Systems for Adaptive Speech and Text Analytics

S. Caskey, U. Chaudhari, E. Epstein, R. Florian, J. S. McCarley,
M. Omar, G. Ramaswamy, S. Roukos, T. Ward
IBM T. J. Watson Research Center
PO Box 218
Yorktown Heights, NY, 10598, USA
roukos@us.ibm.com

Keywords: Search and Retrieval, Information Extraction and Link Analysis, Foreign Language Processing, Knowledge Discovery and Dissemination

Abstract

We describe an infrastructure for integrating Speech and Text Analytics using the Unstructured Information Management Architecture (UIMA) framework. UIMA is a framework for composing analysis engines to create rich annotations and analyses of multimedia data. Our analysis engines include Speaker Detection (SD) and Speech-To-Text (STT) for audio inputs, and Entity Detection and Recognition (EDR) and Statistical Machine Translation for text inputs. All engines handle both Arabic and English content. We illustrate the architecture by describing two systems we have implemented: a cross-lingual search enabling the use of English queries to find relevant Arabic documents and an adaptive cross-modal distillation system capable of correlating information across audio and text channels.

The semantic annotations produced by the above analysis engines enable “semantic search” and enable cross-modal distillation across data streams (different modalities and languages) such as tracking and summarizing an entity’s statements on a topic, whether in spoken or written form.

1 Introduction

Our group is famous for stating that “there is (sic) no better data than more data.” In this spirit, today’s torrent of multimedia and multilingual data (radio and TV broadcasts, internet radio, newswire, blogs, other web data, etc.) is simultaneously an unprecedented opportunity and a systems engineering challenge. We simplify the challenge of developing advanced and scalable speech and text analytics systems by using a software architecture and framework (software libraries) called UIMA* that streamline the composition of analytics engines. UIMA provides a common analysis structure (CAS) that all analytics engines share to fetch their inputs and to record their outputs. Sample CAS entries in an audio processing solution would include the waveform, transcribed words with time spans produced by a speech-to-text engine, and speaker names with time spans from a speaker detection engine. The CAS data model is strongly typed: no annotator can consume or produce metadata incompatible with the type system. The type system and the input-output specifications of annotators enable streamlined composition of annotators to produce rich analytics systems. We describe two applications to illustrate the breadth of the issues addressed by UIMA and our analytics:

1) Cross-lingual search: This system enables an English speaker to use English queries to retrieve relevant documents from a collection of Arabic documents. Given an English query, the system uses a statistical translation dictionary to find a set of relevant Arabic documents. Each document in the result list is summarized by the English translations of the important entities mentioned in that document in addition to crawl date and source information. To support this functionality, the document ingestion pipeline consists of a set of UIMAfied annotators described in more detail below. They include 1) an entity detection and recognition analytic to identify mentions of entities such as people, organizations, and locations in the Arabic source documents and 2) a name translation analytic to translate or transliterate the spotted names. At query time, the system also provides on demand machine translation to produce a “gist” translation of the document suitable for extracting the information of interest.

* Unstructured Information Management Architecture
Cross-modal distillation: This system uses light-weight text analytics to identify quotes in text news source such as “Iraqi President Ghazi Al-Yawer said ‘It’s only complete nonsense to ask the troops to leave in this chaos and this vacuum of power’,” and then tracks down the primary source by finding in audio channels a snippet where President Al-Yawer is speaking the quotation of interest to confirm the quote’s veracity. To optimize computational resources, we employ hierarchical filtering: inexpensive speaker detection can be run on a compressed audio stream (such mp3, G723, etc.) to identify putative segments of Al-Yawer speaking; then depending on availability of computing resources a fraction of these segments are decompressed and a more accurate and resource-intensive speaker detection system is run to identify those segments with a higher likelihood of Al-Yawer speaking; then for the highest scoring segments a speech-to-text system is run to recognize the words which will then be scanned for approximate matches to the quotation of interest. The tuning of the various filtering thresholds is the subject of further research.

2 UIMA

UIMA, the Unstructured Information Management Architecture, (Ferrucci and Lally 2004) offers a set of interfaces and components that facilitate the distillation of information from unstructured data like audio streams, natural language text, and video streams, in which the information content is not immediately machine-accessible but must be extracted via a sequence of analytics. In particular, UIMA has extensions to allow for ingestion of streamed audio data and processing of its analytically derived segments. UIMA provides a standardized framework to combine a set of analytics engines to generate rich analyses of input data in pursuit of relevant information. Analytics engines may be developed independently of one another, since by conforming to the UIMA specification they become pluggable components.

2.1 CAS

The CAS (Common Analysis Structure) is the UIMA object which stores all intermediate and final analyses, and thus serves as both the input and output of analytics engines. A consistent view of the CAS can be maintained across analytics engines written in C++ and Java and also by remote analytics engines connected over a computer network; this provides significant mix-and-match flexibility to system integrators.

The CAS has three primary components:

- First, it contains the artifact that is to be analyzed, either directly or via a URI reference. The artifact being analyzed is called the “subject of analysis.” In some complex applications, different analytics engines have been designed to work on different levels of representation, for example raw acoustic waveform or html-formatted text, and a higher level plain text view of the artifact. The CAS supports multiple secondary subjects of analysis to aid designers in composing analytics requiring different views into the artifact with different levels of resolution.
- Second, it stores the analytics-generated metadata (called annotations.) All annotations are instances of types from the UIMA type system.
- Finally, it maintains a repository of annotation indices which allow programmers to easily visit all annotations of a particular type (e.g. all locations) or supertype (e.g. all entities.)

The annotations can be quite rich. Figure 1 shows annotations produced by several annotators. At the lowest level is the subject of analysis, six words in this case. First, a syntactic chunker identifies various types of phrases (NP, VP, and PP.) Next, a set of entity mentions are detected (Person, Gov. Title, and Country.) Finally, a relation detector annotates a relation (Located-In) with two arguments. Each of the arguments is itself an annotation; this highlights the rich interplay possible between analytics engines. As indicated by the arrows, all annotations point to their constituent spans in the subject of analysis, in this case the original text.

2.2 Type system

All annotations stored in the CAS are compliant with the UIMA type system. The type system is hierarchical and consists of primitive types (e.g. Unicode string, integer, and float), complex (derived) types, and lists and arrays of these. The type system supports single inheritance which facilitates the inter-operation of analytics engines. Each application’s type system is specified in an xml configuration file loaded at initialization time. The UIMA Software Developers Kit provides a convenient Eclipse type system editor plug-in to free developers from the details of the xml format.

2.3 Collection Processing

UIMA also provides a collection processing framework in 3 parts. The collection reader interface defines a pull model for ingesting artifacts from varying sources (e.g. web crawlers, audio feeds, files in a directory.) The CAS consumer interface allows one or more components to tap into the final product of the analysis pipeline. Finally, the collection processing manager orchestrates the interactions between the collection reader, the analysis pipeline, and the CAS consumers.
3 Cross-Lingual Search

Figure 2 shows the ingestion pipeline for our cross-lingual search system. Crawled web pages are fed to a collection reader which passes them to the analytics pipeline. First, the fingerprint of a page is computed (for approximate duplicate detection), next its language and character encoding are detected and the “text” of the web page is extracted from the html markup by a detagger, then depending on the language the text is tokenized with the appropriate tokenizer or morphological analyzer, then mentions of names of entities are detected in the document, then the extracted names are translated and/or transliterated to English, and finally all the extracted metadata in the CAS is indexed by one CAS consumer and stored by another CAS consumer for later retrieval based on a user query. As part of the ingestion, the system may optionally identify segments of input that can used to adapt the models used by the various annotators.

As mentioned earlier, all our analysis engines create annotations that point to regions in the original artifact to enable us to map the analyses onto the original input consistent with the original layout to the extent possible. For example, in Figure 3 the region(s) of html corresponding to the text of each name have been wrapped with html code to highlight the Arabic name and to juxtapose the English gloss while respecting the left-to-right order of written English and the right-to-left order of Arabic.

3.1 Entity Detection and Recognition

The EDR system uses maximum entropy modeling technology to identity mentions of entities (e.g. Person, Organization, Location, etc.). The system is trainable from example data and is described in more detail in (Florian...
et al. 2004.) The UIMAfied engine identifies the span of the mentions in the original html page as can be seen in Figure 3. The IBM EDR toolkit has an annotation GUI tool to speed up the labeling of example data and test data with entities (and relations) of interest. It provides a utility to coordinate multiple human annotators to reduce the development effort in training a new system. It also provides model builder utilities to produce maximum entropy models for new domains and regression testing to measure development progress. Since the system is data driven, the system can be easily used to build EDR capability in multiple languages—systems have been built for Arabic, Chinese, English, Farsi, Spanish, and Italian, all using the same semantic types (about 50 entity types and about 50 relation types). The accuracy of the system is world class as demonstrated in the recent NIST sponsored Automatic Content Extraction (ACE) evaluation (Nov 2004).

3.2 Statistical Machine Translation
We have also UIMAfied the IBM Arabic-English statistical machine translation (SMT) system. This engine is used both for name translation and on demand document translation. All spotted mentions are translated at ingestion time. For those names that are new to the system, a transliteration system that uses letter and phonetic models creates a transliteration for the names. The Arabic-English SMT engine is also used in the cross-lingual search application to enable users to get "gist" translations for documents that they feel are potentially relevant to their query based on the (translated) entities present in the document, the document source, and its date. This two-step translation strategy—pretranslating entities for all documents and translating whole documents only on demand—circumvents the computational expense of generating millions of unused translations at ingestion time for large collections.

The statistical machine translation engine is trained from a corpus of parallel sentences for a particular language pair. The trainability enables the adaptation of the model to a domain by including translated sentence pairs from the new domain. The engine achieves state-of-the-art translation quality and is currently available for several language pairs (e.g. Arabic-English, Chinese-English, Spanish-English, and English-French.)

3.3 Cross-Lingual Search Engine
The system we have developed crawls about 20 Arabic news sites on a daily basis. Our current store has about 1 million Arabic ingested documents. The cross-lingual search engine has about 250k English word stems which are translated to a max of 5 Arabic stems with a trained probability distribution. The ranking of results uses a convolutional model as described in (Franz and McCarley 2002.) In the result page, we summarize documents by the translated top entities in the 3 categories of person, organization, and location/country. We also include the source information and the crawl date. The search engine produces its results in under 0.4 seconds. Users may view
the original version of interesting Arabic documents, a glossed version with the entities highlighted and translated as depicted in Figure 3, or as an on demand “gist” translation.

4 Cross-Modal Distillation

For processing a set of multimodal multilingual streams, we built a cross-modal distillation system that can efficiently correlate information across modalities (and potentially languages) by composing UIMAfied speech and text analytics. Figure 4 shows the cross-modal distillation system. Both http and compressed audio streams are analyzed. The http traffic consists of news sources. Audio traffic is compressed audio from news broadcast (e.g. the TDT2 corpus). Bold arrows in the figure show the creation and propagation of annotations by analysis engines such as speaker labels, recognized words, detected quotes, etc. The thin arrows back from the multimodal information broker represent feedback to the various annotators to either identify what entities to look for or change how much compute resources are consumed by an individual analytic. As an example of this feedback, in this case to enable adaptive usage of computing resources, is a control of the fraction of speaker segments that are passed on for forward processing (e.g. speech recognition) based on a confidence score. The components of the speech and text processing system are:

**Audio Stream Segmentation**: The audio segmenter is the first component to see the input audio streams. Its function is to partition the input into segments, which can be uniform, as with a fixed window size, or non-uniform and based on a variety of algorithms that decide whether or not a segment boundary occurs at any given point in time. The algorithms cover a range of complexity/accuracy tradeoffs (Omar et al. 2005.) The fixed partitioning essentially has no time cost associated and no attempt to accurately find segmentation points. The more complex algorithms, with high accuracy on a number of test sets, still operate an order of magnitude faster than real time.

**Speaker Recognition**: The speaker recognition engine (Chaudhari et al. 2001) produces a top-N list of hypothesized speakers, drawn from a speaker library, ranked according to the degree of confidence that the speaker is detected in the data. It updates the CAS by annotating the hypothesized speakers and confidence scores. The confidence for 2000 speakers can be measured 10 times faster than real-time. The speaker library can be extremely large, since the system has the property that a delta increment of compute time allows for approximately 2000 more speakers to be evaluated. Complexity/accuracy tradeoffs can be made by altering the model size and/or data sub-sampling.

For improved CPU utilization, one may build a speaker recognition system using the original compressed stream, then for a subset of segments with putative speaker hits, run decompression and use a more accurate speaker detection analytic as depicted in Figure 4. The speaker detection analytic (Ramawamy et al. 2003) uses the GMM-UBM (Gaussian Mixture Model - Universal Background Model) framework, wherein the deviation from a set of global, or “average,” statistics, comprising the UBM, is measured to produce a model for each target speaker of interest.

**Audio Data Conversion**: This element, not depicted in Figure 4, is general and serves as a transducer from available data types to required data types (e.g. sampling frequency conversion, decompression, etc.)
Speech To Text: The speech to text engine supports real-time Arabic and English LVCSR (Large Vocabulary Continuous Speech Recognition) over a wide range of environmental conditions (Chen et al. 2002.) For example, the engine enters the decoded sequence of words along with start and end times, but also creates a stream of text that can be processed by a downstream component such as the Named Entity (Mention) Detector. The engine has the capability to adapt recognition models to individual speakers. Furthermore complexity/accuracy tradeoffs can be made by altering search and model parameters.

Named Mention Detection: This component (also not depicted in Figure 4) is the same UIMAfied EDR component discussed earlier. This illustrates component re-use across a UIMAfied system. In this case, the EDR system annotates the text, derived from the STT engine in the scenario, to detect specific mentions of interest.

Multimodal Information Broker: This unit controls flow of data segments through the available analytics engines. It also has the ability to examine the contents of each CAS. This enables filtering of the data and dynamic sequencing of the engines. In the analogy with Unix pipes, this would be the entity that decides the order of programs in the pipeline. In the present context, one example is that of filtering. If speaker detection is done first, then only data with speakers of interest could be passed on to further analysis. The information broker may access additional UIMAfied analytics. For example, if an Arabic audio segment is recognized, the STT output can be sent to the statistical machine translation if an English gist is required by a user. Clearly, the composition of UIMAfied analytics can be quite rich and address a variety of needs that will be explored in future developments.

5 Conclusion
We have described a set of advanced speech and text analytics covering speaker detection, speech-to-text, entity detection and tracking, and statistical machine translation. These components represent the best that today’s technology can achieve for linguistic processing and are scalable for large scale audio and text streams. Using the UIMA framework, we have outlined some examples of advanced analytic functionality such as cross-lingual search (English queries to search foreign language documents) and cross-modal analytics with adaptive resource usage to optimize usage of available computing resources. The UIMA framework enables the incorporation of 3rd party analytics to augment the set of IBM analytics we described earlier.

6 References


\(^3\) The UIMA SDK is available for download at http://www.alphaworks.ibm.com/tech/uima