### Algebra-Based Architecture

<define algebra-based architecture>

#### SystemT (Krishnamurthy et al, 2008)

Krishnamurthy et al proposed an Information Extraction system that takes advantage of the classical database ideas to overcome the limitations of developing and using grammar-based information extraction systems. The system was called SystemT. In their paper, Krishnamurthy et al provided the architecture that they used in making SystemT.

SystemT has two modules or in their case “environments” on which the information extraction is done namely the Development Environment and the Runtime Environment. In the Development Environment, constructing and refining the rules that will be used for the actual extraction is done repeatedly and after which; the rules are registered and specified in a language called the Annotation Query Language (AQL). It is in this environment where the rules are being compiled into algebraic expressions and where the results of executing the rules over a corpus of representative documents are being visualized. Once these results are deemed to be satisfactory, they can now be published into an annotator. The process of publishing the annotator includes the feeding of the AQL into the Optimizer (which compiles the rules into an algebraic expression) and the instantiation of corresponding physical operators by the Runtime Environment. On the other hand, in the Runtime Environment, the environment/module receives a continuous stream of input documents and then it annotates each of the documents and outputs them for further specific processing. Usually, the Runtime Environment is embedded in the processing pipeline of the information extraction system. Figure 3.1.1b illustrates the architecture that was used by SystemT.

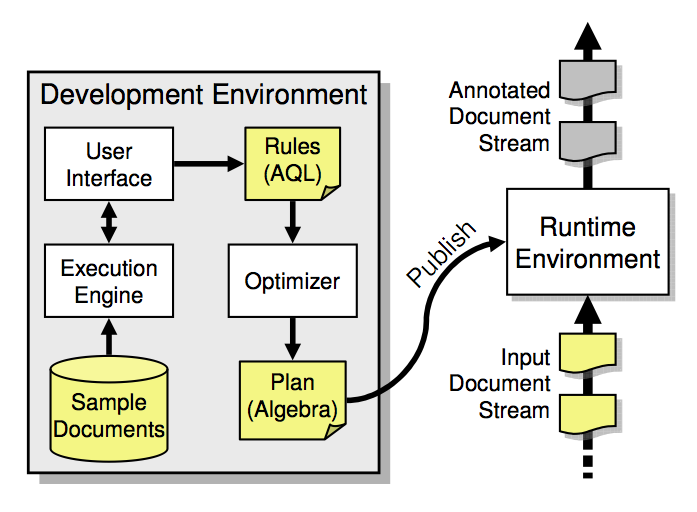


Figure 3.1.1b: The Architecture of SystemT

### Machine Learning-Based Information Extraction

Machine learning-based information extraction system uses machine learning algorithms to automatically learn rules based on the training dataset. It is more cost effective than manually creating the rules as machine learning-based information extraction only needs the domain knowledge. Some algorithms that can be used in information extraction are Decision Trees, Artificial Neural Networks (ANN), and Support Vector Machine (SVM) (Farkas, 2009).

#### TwitIE (Bontcheva et al, 2013)

Bontcheva and her team of researchers have proposed an information extraction system that is specifically targeted for extracting relevant information from texts coming from microblogs. Bontcheva et al made use of the GATE ANNIE architecture in developing the system and took advantage of some of its built-in tools to further streamline the process of information extraction. In their paper, they presented how they designed the architecture of TwitIE and how they used the existing tools from ANNIE.

ANNIE offers a wide array of information extraction tools like tokenizer, sentence splitter, POS tagger, gazetteer lists, finite state transducer (from GATE’s built-in regular expression over annotation language), orthomatcher and coreference resolver but in the case of TwitIE, Bontcheva et al only reused the sentence splitter and name gazetteer because the other components/tools have to be modified to fit the specifics of microblog texts like noisiness, brevity, idiosyncratic language and social context.

Overall, TwitIE has the following main components: Language Identifier, Tokenizer, Gazetteer, Sentence Splitter, Normalizer, POS Tagger, and the Named Entity Recognizer. For the Language Identifier, TwitIE made use of the TextCat language identification algorithm, which heavily relies on n-gram frequency models to identify languages. For the Tokenizer, TwitIE followed Ritter’s Tokenization Scheme to treat abbreviations and URL’s as one token and hashtags and mentions as two token. This scheme also features orthography and capitalization preservation. For the Gazetteer, TwitIE used the unmodified version from ANNIE, which compiles lists of entities into finite state machines that can match text tokens. For the Sentence Splitter, TwitIE still used the unmodified version from ANNIE, which is a cascade of finite-state transducers that segments text into sentences. For the Normalizer, TwitIE made use of a combination of a generic spelling-correction dictionary and a social media-specific dictionary. The list of variations is also dynamic by using heuristics to correct spellings. For the POS Tagger, TwitIE made use of the same technique used by a Stanford Tagger, which was trained on tweets that were tagged using the Penn TreeBank Tagset. The improved tagger also included tag labels to support retweets, mentions, URL’s, hashtags and user mentions. Lastly, for the Named Entity Recognizer, TwitIE made use of the existing ANNIE NER as its pattern and just added some features that would help it support recognition of entities in social media texts. Figure 3.1.1c illustrates the architecture used by TwitIE.

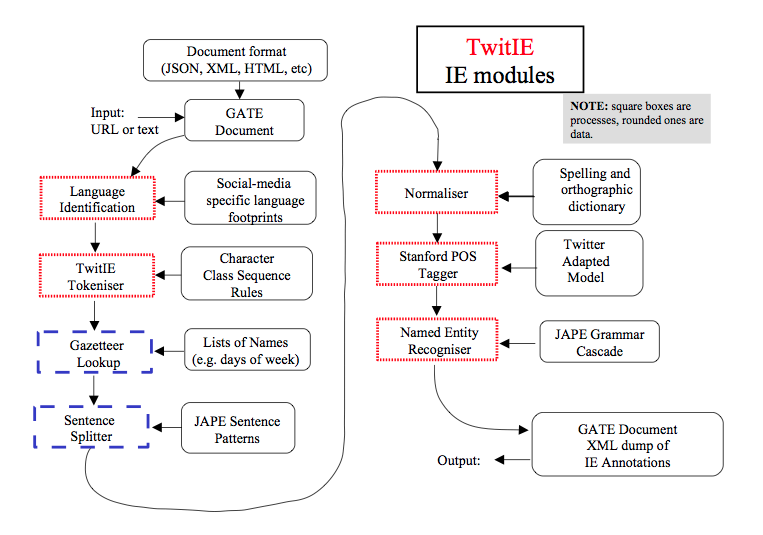


Figure 3.1.1c: The Architecture of TwitIE