Natural Language Engineering **14** (1): 33–69. © 2006 Cambridge University Press
doi:10.1017/S1351324906004116 First published online 9 June 2006 Printed in the United Kingdom

InfoXtract: A customizable intermediate level information extraction engine

ROHINI K. SRIHARI
Janya Inc., 1408 Sweet Home Road, Amherst, NY 14228, USA
State University of New York at Buffalo
e-mail: rohini@janyainc.com

WEI LI, THOMAS CORNELL
Janya Inc., 1408 Sweet Home Road, Amherst, NY 14228, USA
e-mail: {wei cornell}@janyainc.com

CHENG NIU
Microsoft Research China, 5/F, Beijing Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing 100080, P.R.C.
e-mail: cniu@microsoft.com

(*Received 15 March 2004; revised 13 September 2005*)

Abstract

Information Extraction (IE) systems assist analysts to assimilate information from electronic documents. This paper focuses on IE tasks designed to support *information discovery* applications. Since information discovery implies examining large volumes of heterogeneous documents for situations that cannot be anticipated *a priori*, they require IE systems to have breadth as well as depth. This implies the need for a domain-independent IE system that can easily be customized for specific domains: end users must be given tools to customize the system on their own. It also implies the need for defining new *intermediate* level IE tasks that are richer than the subject-verb-object (SVO) triples produced by shallow systems, yet not as complex as the domain-specific scenarios defined by the Message Understanding Conference (MUC). This paper describes *InfoXtract*, a robust, scalable, intermediate-level IE engine that can be ported to various domains. It describes new IE tasks such as synthesis of entity profiles, and extraction of concept-based general events which represent realistic near-term goals focused on deriving useful, actionable information. Entity profiles consolidate information about a person/organization/location etc. within a document and across documents into a single template; this takes into account aliases and anaphoric references as well as key relationships and events pertaining to that entity. Concept-based events attempt to normalize information such as time expressions (e.g., yesterday) as well as ambiguous location references (e.g., Buffalo). These new tasks facilitate the correlation of output from an IE engine with structured data to enable text mining. InfoXtract's hybrid architecture comprised of grammatical processing and machine learning is described in detail. Benchmarking results for the core engine and applications utilizing the engine are presented.

1 Introduction

The last decade has seen great advances in the area of information extraction (IE). In the US, the DARPA sponsored Tipster Text Program (Grishman 1997), the Message

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Understanding Conferences (MUC) (Chinchor and Marsh 1998), DARPA's Evidence Extraction and Link Discovery and Translingual Information Detection, Extraction, and Summarization have been driving forces for developing this technology. The most successful IE task thus far has been Named Entity (NE) tagging. The state-of-the-art exemplified by systems such as *NetOwl* (Krupka and Hausman 1998), *IdentiFinder* (Miller, Michael, Fox, Ramshaw, Schwartz and Stone 1998) and InfoXtract (Srihari, Niu and Li 2000) has reached near human performance, with 90 per cent or above F-measure. On the other hand, the deep level MUC Scenario Template (ST) IE task is designed to extract detailed information for predefined event scenarios of interest. It involves filling slots of complex event templates, including causality and aftermath slots. It is generally felt that this task is too ambitious for deployable systems at present, except for very narrowly focused domains. This paper focuses on new intermediate level information extraction tasks that are defined and implemented in an IE engine, named *InfoXtract*. InfoXtract is a domain independent but portable information extraction engine that has been designed for information discovery applications (Srihari, Li, Niu and Cornell 2003).

Information Discovery (ID) is a term which has traditionally been used to describe efforts in data mining (Han 1999). The goal is to extract novel patterns of transactions which may reveal interesting trends. The key assumption is that the data is already in a structured form. Discovery in this paper is defined within the context of unstructured text documents; it is the ability to extract, normalize/disambiguate, merge and link entities, relationships, and events which provides significant support for ID applications. Furthermore, there is a need to accumulate information across documents about entities and events. Due to rapidly changing events in the real world, what is of no interest one day may be especially interesting the following day. Thus, information discovery applications demand *breadth* and *depth* in IE technology.

A variety of IE engines, reflecting various goals in terms of extraction as well as architectures are now available. Among these, the most widely used are the *GATE* system from the University of Sheffield (Cunningham, Maynard, Bontcheva, Tablan, Ursu, Dimitrov, Dowman, Aswani and Roberts 2003), the IE components from Clearforest (`www.clearforest.com`), SIFT from BBN (Miller *et al.* 1998), REES from SRA (Aone and Ramos-Santacruz 2000) and various tools provided by Inxight (`www.inxight.com`). Of these, the GATE system most closely resembles InfoXtract in terms of its goals as well as the architecture and customization tools. InfoXtract differentiates itself by (i) using a hybrid model that efficiently combines statistical and grammar-based approaches with some procedural modules (e.g., the Alias Sub-module and the Profile/Event Linking/Merging Module); and (ii) using an internal data structure known as a *token-list* that represents hierarchical linguistic structures and IE results from pipelined modules.

The work presented here focuses on a new intermediate level of information extraction which supports information discovery. Specifically, it defines new IE tasks such as *Entity Profile* (EP) extraction, which is designed to accumulate interesting information about an entity across documents as well as within a discourse (Li *et al.* 2003a). Furthermore, *Concept-based General Event* (CGE) is defined as a

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

domain-independent representation of event information that is useful and more tractable than MUC ST. InfoXtract represents a hybrid model for extracting both shallow and intermediate level IE: it exploits both statistical and grammar-based paradigms. A key feature is the ability to rapidly customize the IE engine for a specific domain and application. Information discovery applications are required to process an enormous volume of documents, and hence any IE engine must be able to scale up in terms of processing speed and robustness; the design and architecture of InfoXtract reflect this need.

In the remaining text, section 2 defines the new intermediate level IE tasks. Section 3 presents the hybrid IE model. Section 4 delves into the engineering architecture and implementation of InfoXtract. Section 5 presents extensions to InfoXtract to support corpus-level IE and section 6 discusses domain porting. Section 7 presents three applications which have exploited InfoXtract. Finally, section 8 compares this to related work and section 9 summarizes the advances towards intermediate-level IE reflected in the InfoXtract engine.

2 InfoXtract: Defining new IE tasks

InfoXtract (Li and Srihari 2003; Srihari *et al.* 2003) is a domain-independent and domain-portable, intermediate level IE engine. Figure 1 illustrates the overall architecture of the engine which will be explained in detail shortly. Although the idea of intermediate-level IE is not new, the specific definition of what constitutes intermediate-level IE and the implementation vary; this paper describes our approach. Intermediate-level IE is defined as concept-based IE: when filling values of extraction templates, the focus is on using conceptual values as opposed to strings of words extracted from text. Specifically, this leads to the definition of the following outputs of the InfoXtract engine:

- *Entity profiles:* information about the same entity is merged into a single profile. While NE provides very local information, an entity profile which consolidates all mentions of an entity in a document is much more useful. This includes document-level discourse processing, such as alias and co-reference resolution.
- *Concept-based general events:* extract generic events in a bottom-up fashion, as well as map them to specific event types in a top-down manner. The use of generic event slots such as *persons-involved*, *organizations-involved*, *location*, *date*, etc., are often sufficient for a user searching for key events. Event template slots are filled by entity profiles, rather than named entities. Furthermore, information is normalized wherever possible; this includes time and location normalization. Recent work has also focused on mapping key verbs into verb synonym sets reflecting the general meaning of the action word.

A description of the IE outputs from the InfoXtract engine in order of increasing sophistication is given below:

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

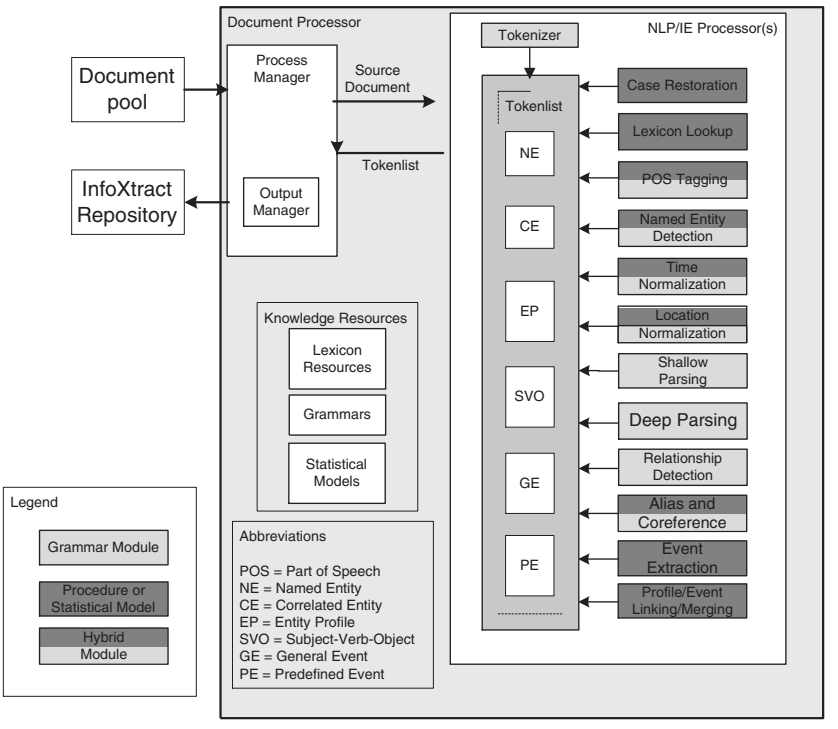


Fig. 1. InfoXtract engine architecture.

- **NE** Named Entity objects represent key items such as proper names of *person*, *organization*, *product*, *location*, *target*, contact information such as *address*, *email*, *phone number*, *URL*, time and numerical expressions such as *date*, *year* and various measurements *weight*, *money*, *percentage*, etc.
- **CE** Correlated Entity objects capture relationship mentions between entities such as the *affiliation* relationship between a person and his employer. The results will be consolidated into the information object Entity Profile (EP) based on co-reference and alias support.
- **EP** Entity Profiles are complex rich information objects that collect entity-centric information, in particular, all the CE relationships that a given entity is involved in and all the events this entity is involved in. This is achieved through document-internal fusion and cross-document fusion of related information based on support from co-reference, primarily based on alias association. Work is in progress to enhance the fusion by correlating the extracted information with information in a user-provided existing database.
- **GE** General Events are verb-centric information objects representing 'who did what to whom when and where' at the logical level. Concept-based GE (CGE) further requires that participants of events be filled by EPs instead of NEs and that other values of the GE slots (the action, time and location) be disambiguated and normalized (Li, Srihari, Niu and Li 2002).

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

- **PE** Predefined Events are domain specific or user-defined events of a specific event type, such as *Product Launch* and *Company Acquisition* in the business domain. They represent a simplified version of MUC ST. InfoXtract provides a toolkit that allows users to define and write their own PEs.

To engineer a complex system for real life applications, it is extremely important to design a common basis for all components. In the case of InfoXtract, this common basis is the data structure *Token List*. All the modules or submodules of the engine update this common data structure one way or another, limited to three basic types of actions, namely Tagging, Chunking and Linking. Therefore the underlying operation of the engine is very clearly defined.

The processing modules or submodules are of three basic types: (i) modules using finite state grammars, (ii) modules based on statistical models, and (iii) procedures directly implemented in C++. As a general rule, for phenomena with fairly clear boundaries or predictable patterns, a grammar approach is used which handles the sparse data problem well. For more complex and fuzzy problems, statistical learning-based techniques are used. For routine tasks such as preprocessing, lexical lookup, and linking and consolidating information, the modules are directly coded in C++, often with configuration parameters for flexibility.

With a common data structure and three available paradigms for implementation, a strictly modular approach to handling complicated language phenomena is pursued. The system consists of six major components; each component typically consists of several modules which are further divided into sub-modules. This results in a pipeline involving over 100 processing scans of the input; since the tokenizer converts the input text into a token list, the text is only processed once. In order to address the inherent error propagation issue of a pipeline architecture, a design philosophy for modular development is established such that each module is developed and tested independently, to the extent possible. In practice, this means that except for a small set of global features accessible by any module, developers rely mainly on internal macros or sub-modules. Furthermore, to facilitate the maintenance of a continually evolving system and to reduce side effects of system modifications, the majority of modules are based on lexicalist techniques. This includes lexical statistics, custom lexicons, as well as Expert Lexicons and Lexicon Grammars, to be described in detail shortly.

Figure 2 illustrates a portion of a text document that has been processed by InfoXtract. Figure 3 illustrates the results of InfoXtract processing in the form of XML. It shows partial output for named entities, profiles, predefined events and general events. Named entities such as organization can have sub-types such as company; location can have sub-types such as country. The relationship section depicts typical relationships such as affiliation and position, and also includes semantic parsing as well as co-reference links. The predefined event represents a business acquisition event where *Serono* is the acquiring company and *Geneva Biomedical Research Institute* the organization being taken over. Note that profiles for both the person *Ernesto Bertarelli* and the organization *Serono* are linked to events in which these entities play a role. Finally, the general event section

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Ernesto Bertarelli, CEO of Serono, explained how serendipity can play a role. In January 1998, Serono acquired Geneva Biomedical Research Institute from Glaxo Wellcome. The Serono Pharmaceutical Research Institute boasts more than 100 experienced scientists working on several state-of-the-art technologies in molecular biology and genomics. "We basically leapfrogged by five years our strategic intent to expand our portfolio in these areas," Bertarelli said.

A scientist whom Serono was sponsoring at the Weizmann Institute of Science in Israel (where the company has funded several research teams for more than 15 years) discovered an essential molecule in her intracellular transduction models. The company lacked the technology to develop that discovery and began searching for a partner to fill the gaps. The Weizmann Institute of Science led them to Signal, which is a part of the Salk Institute.

Fig. 2. Sample input text.

illustrates an event based on the logical S-V-O structure headed by the key verb *was sponsoring*. Note the more generic slot names used here, compared to those seen in predefined events. Using XML to represent InfoXtract output facilitates interfacing to a surrounding application, as well as the ability to support special configurations of the engine for different purposes by selective filtering of output.

The InfoXtract engine has been deployed both internally to support Cymfony's *Brand Dashboard*™ product and externally to third-party integrators for building IE applications in the intelligence domain.

2.1 Natural language processing modules

The linguistic modules serve as an underlying support system for different levels of IE. This support system involves major linguistic areas: orthography, morphology, syntax, semantics, and discourse. A brief description of the linguistic modules is given below.

- **Preprocessing** This component handles file format converting, text zoning and tokenization. The task of text zoning is to identify and distinguish metadata such as title, author, etc., from normal running text. The task of tokenization is to convert the incoming linear string of characters from the running text into a token list; this forms the basis for subsequent linguistic processing. An optional HMM-based *Case Restoration* module is called when processing caseless text (Niu, Li, Ding and Srihari 2004).
- **Word Analysis** This component includes word-level *orthographical analysis* (capitalization, symbol combination, etc.) and *morphological analysis* such as stemming. It also includes part-of-speech (POS) tagging which distinguishes, e.g. a noun from a verb based on contextual clues.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

```
⟨content⟩
  ⟨named_entities⟩
    ⟨named_entity type="person.person" id="544"⟩Ernesto Bertarelli⟨/named_entity⟩
    ⟨named_entity type="organization.organization" id="6"⟩
      ⟨named_entity subtype="organization.company" id="6"⟩Serono⟨/named_entity⟩
    ⟨named_entity type="location.location" id="93"/⟩
      ⟨named_entity subtype="location.country" id="93"⟩Israel⟨/named_entity⟩
    ...
  ⟨/named_entities⟩
  ⟨relations⟩
    ⟨relation type="affiliation_R" id="544" rid="6"⟩Ernesto Bertarelli -> Serono⟨/relation⟩
    ⟨relation type="locationOf_R id="661" rid="660"⟩in Israel -> at the Weizmann Institute
        of Science⟨/relation⟩
    ⟨relation type="subject_predicative_R" id="544" rid="4"⟩Ernesto Bertarelli -> CEO⟨/relation⟩
    ⟨relation type="verb_object_R" id="21" rid="545"⟩acquired -> Geneva Biomedical Research
        Institute⟨/relation⟩
    ⟨relation type="aliasLink_R" id="544" rid="414"⟩Ernesto Bertarelli -> Bertarelli⟨/relation⟩
    ⟨relation type="coreferenceLink_R" id="604" rid="6"⟩the company -> Serono⟨/relation⟩
    ...
  ⟨/relations⟩
  ⟨profiles⟩
    ⟨profile type="profile.person" id="726"⟩
      ⟨attribute type="profileName_R" id="544"⟩Ernesto Bertarelli⟨/attribute⟩
      ⟨attribute type="aliases_R" id="414"⟩Bertarelli⟨/attribute⟩
      ⟨attribute type="affiliation_R" id="6"⟩Serono⟨/attribute⟩
      ⟨attribute type="position_R" id="4"⟩CEO⟨/attribute⟩
      ⟨attribute type="eventsInvolved_R" id="714"/⟩
      ⟨attribute type="eventsInvolved_R" id="723"/⟩
    ⟨/profile⟩
    ⟨profile type="profile.organization" id="733"⟩
      ⟨attribute type="profileName_R" id="6"⟩Serono⟨/attribute⟩
      ⟨attribute type="focalEntity_R" id="736"⟩true⟨/attribute⟩
      ⟨attribute type="staff_R" id="544"⟩Ernesto Bertarelli⟨/attribute⟩
      ⟨attribute type="ceo_R" id="544"⟩Ernesto Bertarelli⟨/attribute⟩
      ⟨attribute type="acquisitions_R" id="545"⟩Geneva Biomedical Research Institute⟨/attribute⟩
      ⟨attribute type="eventsInvolved_R" id="711"/⟩
    ⟨/profile⟩
    ...
  ⟨/profiles⟩
  ⟨predefinedEvents⟩
    ⟨predefinedEvent type="business.AC_Acquisition" id="705"⟩
      ⟨attribute type="business.AC_Acquisition" id="21"⟩acquired⟨/attribute⟩
      ⟨attribute type="AC_acquiring_company" id="20"⟩Serono⟨/attribute⟩
      ⟨attribute type="AC_acquired_company" id="545"⟩Geneva Biomedical Research
          Institute⟨/attribute⟩
      ⟨attribute type="AC_date" id="654"⟩In January 1998⟨/attribute⟩
      ⟨attribute type="subsequentEvent_R" id="711"/⟩
      ⟨attribute type="AC_acquiring_company" id="733"/⟩
      ⟨attribute type="AC_acquired_company" id="798"/⟩
    ⟨/predefinedEvent⟩
    ...
  ⟨/predefinedEvents⟩
  ⟨generalEvents⟩
    ⟨generalEvent type="generalEvent" id="711"⟩
      ⟨attribute type="who" id="83"⟩Serono⟨/attribute⟩
      ⟨attribute type="keyVerb" id="570"⟩was sponsoring⟨/attribute⟩
      ⟨attribute type="whom.what" id="602"⟩a scientist⟨/attribute⟩
      ⟨attribute type="otherModifier" id="660"⟩at the Weizmann Institute of Science⟨/attribute⟩
      ⟨attribute type="precedingEvent_R" id="705"/⟩
    ⟨/generalEvent⟩
    ...
```

Fig. 3. Partial xml output produced by infoxtract on sample text of figure 2.

- **Phrase Analysis** This component, also called *shallow parsing*, undertakes basic *syntactic analysis* and establishes simple, un-embedded linguistic structures such as basic noun phrases (NP), adjective phrases (AP), verb groups (VG), and basic prepositional phrases (PP). This is a key linguistic module, providing the building blocks for subsequent dependency linkages between phrases.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

```
ambassador NN:  [profession, title, entity, life_form,
                 person, worker]
fled VBD:       [V2A, V2C, V6A, V13A, V14, leaving]
```

Fig. 4. Sample lexical entries.

- **Sentence Analysis** This component, also called semantic parsing, decodes underlying dependency trees that embody logical relationships such as V-S (verb-subject), V-O (verb-object), H-M (head-modifier). The InfoXtract semantic parser transforms various patterns, such as active patterns and passive patterns, into the same *logical form*, with the *argument structure* at its core. This involves a considerable amount of **semantic analysis**. The decoded structures are crucial for supporting structure-based grammar development and/or structure-based machine learning for relationship and event extraction.
- **Discourse Analysis** This component studies the structure across sentence boundaries. One key task for discourse analysis is to decode the co-reference (CO) links of pronouns (*he*, *she*, *it*, etc.) and other anaphora (*this company*, *that lady*) with the antecedent named entities. A special type of CO task is 'Alias Association' which will link *International Business Machine* with *IBM* and *Bill Clinton* with *William Clinton*. The results support information merging and consolidation for profiles and events.

2.1.1 Lexical resources

The InfoXtract engine uses various lexical resources including the following:

- General English dictionaries available in electronic form providing basic syntactic information. The Oxford Advanced Learners' Dictionary (OALD) is used extensively.
- Specialized glossaries for person names, location names, organization names, products, etc.
- Specialized semantic dictionaries reflecting words that denote *person*, *organization*, etc. For example, *doctor* corresponds to person, *church* corresponds to organization. Both WordNet as well as custom thesauri are used in InfoXtract.
- Statistical language models for Named Entity tagging and case restoration (retrainable for new domains)

Sample lexical entries of the major categories (verb VBD and noun NN) are shown in Figure 4. The OALD subcategorization features include [V2A, V2C, V6A, V13A, V14] for *fled*. The lexical semantic features (some derived from WordNet) include [profession, title, entity, life_form, person, worker] for *ambassador*, and [leaving] for *fled*.

InfoXtract exploits a large number of lexical resources. Three advantages are obtained by separating lexicon modules from general grammars: (i) high speed due to indexing-based lookup; (ii) sharing of lexical resources by multiple grammar modules; (iii) convenience in managing grammars and lexicons. InfoXtract uses

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

two approaches to represent lexical items. The first is a traditional feature-based grammatical/machine learning approach where lexical features are assigned to lexical entries that are subsequently used by the later modules. The second approach involves *expert lexicons*, which are designed for grammar lexicalization, to be discussed in the next section.

3 Hybrid technology

InfoXtract represents a hybrid model for IE since it combines both grammar formalisms as well as machine learning. Achieving the right balance of these two paradigms is a major design objective of InfoXtract. The core of the parsing and information extraction process in InfoXtract is organized as a pipeline of processing modules. All modules operate on a single in-memory data structure, called a *token list*. The early motivation for this work was the FASTUS system (Hobbs 1993) which used a cascaded set of finite state grammars to achieve syntactic analysis. This demonstrated the use of Finite State Transducer (FST) technology in Natural Language Processing (NLP) (Roche and Schabes 1997) beyond lexical look-up. It also highlighted the advantages of such a modular system.

3.1 Token list representation

A token list is essentially a sequence of tree structures, overlaid with a graph whose edges define relations that may be either grammatical or informational in nature. The nodes of these trees are called *tokens*. InfoXtract's typical mode of processing is to skim along the roots of the trees in the token list, building up structure 'strip-wise' by packaging sequences of top-level tokens into new higher-level tokens. As a result, even non-terminal nodes behave in the typical case as complex tokens, hence the name. There are several advantages to representing a marked up text using trees explicitly in the data structure, rather than implicitly using paired bracket symbols appearing as individual tokens in the token list. For one thing, it allows a richer organization of the information contained within the tree. For example, it is possible to construct direct links from a root node to its semantic head or its 'foot' (like the infinitive marker *to* in the verb group *to step* in Figure 5). It also makes it trivially easy for pattern matching rules to skip over an entire complex token with a single token wild-card pattern element. It is not necessary to iterate over the contents of a group until a closing bracket is met; indeed, it is not necessary ever to even look at the contents of a group. As a matter of notational equivalence, there is obviously no structure that can be represented in this way that cannot be represented by embedding start-tag and end-tag tokens in the token list, but from an engineering standpoint that approach can make complex pattern matching unnecessarily cumbersome. For example, an automaton that reads bracket symbols as elements of its input alphabet will have some trouble matching a pattern that specifies that the semantic head of a phrase should match X and it must contain material matching Y, somewhere. If it is not known in advance where in the phrase the head will appear, the automaton's read head will have to back up and restart in

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

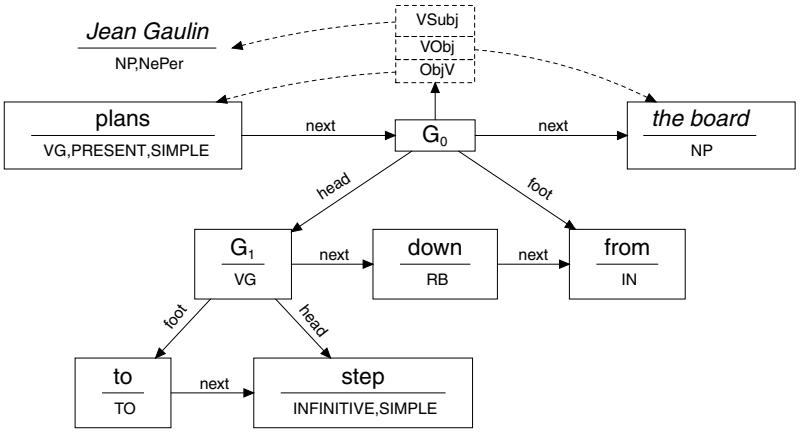


Fig. 5. Token list for sentence fragment.

order to guarantee finding both X and Y, or else all possible ordering permutations will have to be generated as paths through an automaton.

Figure 5 illustrates a portion of the token list on the sentence fragment *Jean Gaulin … plans to step down from the board*. Tokens (both leaf and group) can be marked with features (e.g., *NePer* for Person Named Entity, *VG* for Verb Group, and so on). They can be grouped into higher level token groups, like noun phrases (NP) and verb groups. The figure illustrates a sequence of three tokens, including one token group (G_0). Token groups inherit almost all of the features of their heads, so, for example, the group *to step down from* counts as a VG because its head (*to step*) is a VG. The top-level tokens and token groups may be assigned semantic roles with respect to the verbal elements. Also illustrated are various other structural links that can be traversed from a given node, as well as within a group. The dotted boxes in the upper part of the figure are not part of the 'official' token list; these reflect an additional layer of links representing relationships that have been extracted, in this case links representing argument and modifier roles within a proposition. For example, the 'secondary arc' labeled *VSubj* connects token G_0 to the token containing *Jean Gaulin*, which need not be adjacent to any of the illustrated tokens. In this particular example it is in fact immediately followed by *plans*, but the *VSubj* arc would still be used in this way in a more complex case such as *Jean Gaulin, CEO of Widgets Inc., plans.…* Later in this paper, the use of token lists as the basis for grammar customization will be discussed.

The processing modules that act on token lists can range from simple lexical lookup to the application of hand written grammars to statistical analysis based on machine learning all the way to arbitrary procedures written in C++. Despite the variety of implementation strategies available, InfoXtract processing modules are restricted in what they can do to the token list to actions of the following three types:

1. *Tagging*: assertion (and erasure) of token properties (features, normal forms, etc.)

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

2. *Chunking*: grouping token sequences into higher level constituent tokens.
3. *Linking*: connecting token pairs with a (directional) relational link.

Processing modules take a token list as input, and operate directly on it. Thus, processing modules are 'externally' very simple: they are all simple operators on token lists, and there are a fairly limited number of actions they can take. They differ only internally, in the resources they require and in their logic and inner workings. The configuration of the InfoXtract processing pipeline is controlled by a configuration file, which handles the pre-loading of required resources as well as ordering the application of modules to the token list.

3.2 Grammar formalism

Grammatical analysis of the input text makes use of a combination of phrase structure and relational approaches to grammar. Basically, early modules build up structure to a certain level (including relatively simple noun phrases, verb groups and prepositional phrases), after which further grammatical structure is represented by asserting relational links between tokens.

Our grammars are written in a formalism developed for our own use (which has evolved in some unexpected ways over time),[1] and also in a modified formalism developed for outside users, based on our in-house experiences. In both cases, the formalism mixes regular expressions with boolean expressions. Actions affecting the token list are implemented as side effects of pattern matching. So our processing modules do not resemble Finite State Transducers so much as the regular expression based pattern-action rules used in Awk or Lex. This is broadly similar to the way grammars are written in the JAPE formalism of the GATE system, based in its own turn on a formalism developed as part of the TIPSTER project (CPSL).

Grammars can contain non-recursive macros, which can take parameters. This means that some long-distance dependencies which are very awkward to represent directly in finite state automata can be represented very compactly in macro form. While this has the advantage of decreasing grammar sizes, it does increase the size of the resulting automata.[2] Grammars are compiled to a special type of finite

[1] The formalism was originally designed as a very standard sort of regular expression language over flat sequences of feature-annotated tokens. Extensions that our grammarians have requested over the years include things like the ability to match patterns against the internal structure of group tokens, the ability to traverse 'secondary' arcs as if they were primary structural arcs, and the ability to attach string-based 'normal forms' of various types to tokens.

[2] For example, consider a rule to chunk verb groups, assigning them the correct tense, aspectual and modal features. If the rule finds that the current token is a modal verb, it can call a macro to match the rest of the verb group with the feature *MODAL* as one argument. That macro may expand to a pattern that matches the token *have* and in turn calls another macro one of whose parameters is the concatenation of the features-to-be-assigned parameter (currently = *MODAL*) with *PERFECT*. A complete grammar for the core cases of English verb groups can be written in somewhat less than two pages of code in this way but the result is not easy to maintain: there is no such thing as a small change to such a compact grammar. Also, because information found at the left edge of the verb group must be remembered until the right edge of the group is found, the resulting automaton is very large, and contains a number of paths that will not ever match verb

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

automata. These *token list automata* can be thought of as an extension of *tree walking automata* (Aho and Ullman 1971; Engelfriet, Hoogeboom and Van Best 1999; Mönnich, Morawietz and Kepser 2001). These are linear tree automata, as opposed to standard finite state tree automata (Gécseg and Steinby 1997), which are more naturally thought of as running in parallel. The problem with linear automata on trees is that there can be a number of available 'next' nodes to move the read head to: *right sister*, *left sister*, *parent*, *first child*, etc. So the vocabulary of the automaton is increased to include not only symbols that might appear in the text (test instructions: "Does this token match *X*?") but also symbols that indicate where to move the read head (directive instructions: "Go to the next sister node," or "Go to the first child node."). We have extended the basic tree walking formalism in several directions. First, we extend the expressiveness of test instructions by allowing them to check features of the current node and to perform string matching against the semantic head of the current node (so that a syntactically complex constituent can be matched against a single word).[3] Second, we include symbols for action instructions, to implement side effects. For example, we have instructions for (a) marking a node with a feature, or erasing a feature from a node, (b) grouping a sequence of tokens into a new group token and (c) constructing secondary arcs linking a pair of nodes. Finally, we allow movement not only along the root sequence (string-automaton style) and branches of a tree (tree-walking style), but also along the terminal frontier of the tree and along relational links.[4]

These extensions to standard tree walking automata extend the power of that formalism tremendously, and could pose problems (including non-termination). However, the grammar formalisms that compile into these token list walking automata are restrictive, in the sense that there exist many token list transductions that are implementable as automata that are not implementable as grammars. Also the nature of the shallow parsing task itself is such that we only need to dip into the reserves of power that this representation affords us on relatively rare occasions. As a result, the automata that we actually plug into the InfoXtract NLP pipeline generally run very fast. Just over two thirds of InfoXtract's processing modules are automata, totalling about 1,014,000 transitions and 270,000 states, but all together they account for only one third of its processing time.

groups occurring in grammatical English prose text. It turns out that grammars written at a lower level of generality are easier to maintain and produce smaller automata.

[3] This 'expressiveness' does not actually count as an extension of the real power of a tree walking automaton. The recognition power of tree walking automata is not increased even if the tests they apply are arbitrary monadic second order definable predicates, which should be the case here.

[4] These latter changes do materially increase the power of the formalism over what classical tree walking automata can do. In fact, it seems likely that (with considerable effort) one could implement Turing Machines using this formalism. For example, there is no way to move backwards in our system. However, because it is possible to add arcs to the token list, it should be quite easy to ensure that every node comes with a (secondary) link to its left sister. Also it is not possible to erase nodes from the token list, but it is possible to mark them with features, so it should be possible to 'erase' a node by marking it with a feature *IGNORE*, and adding something like $IGNORE^*$ in every place where a directive instruction is used. We have not proved this conjecture, but it is extremely likely that it is true.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

```
[
    "appoint"
    ( // These macros are all conjunctions of
      // contextual conditions on "appoint"
      // and similar "executive change" verbs
        @V_new_executive
      | @V_per_position
      | @V_per_to_post
      | @V_per_to_run
      | @org_V_position
    )
    <ExecutiveChangeEvent>
];

// Some cases are better handled by
// token list regexps:
@regexp_V_to_be("appoint");
@regexp_org_do_X_and_V("appoint");
@regexp_person_appositives_was_V("appoint");

/*
  "ABC Company named Ian O'Reilly to be its CFO"
  @org, @per and @position are macros encapsulating
  primitive organization and person recognizers
  (based on NE tagging results and word lists)
*/
regexp_V_to_be(KeyWord) =
    [ $KeyWord      // e.g., "appoint", "name"
      LSubj: [@org]
      <ExecutiveChangeEvent>
    ]
    [ ~VG ]*        // skip non-verb group tokens
    [ "be" ]
    [ @position ]
;

V_per_to_run =
    LObj: [@per]
    (Mod|Comp): [ @run_verb  // word list
                  LObj: [@org]
                ]
;
```

Fig. 6. Sample grammar fragment used in InfoXtract.

Figure 6 illustrates a sample grammar fragment using one of our formalisms. Grammars are collections of *rules* and *macros*. A grammar as a whole is interpreted as the disjunction of the rules that it contains. A rule is defined as a regular expression over *token descriptions*. In event detection grammars (such as the grammar from which Figure 6 is drawn), very often the rules consist of only a single token description, describing the head verb token. The rule illustrated here is meant to find mentions of *Executive Change* events based on the word *appoint*. We have

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

provided the definitions of (some of) the macros whose expansions would be used in matching a string like *they have appointed John Doe to run Widgets, Inc.*, or *John Doe was appointed to run Widgets, Inc.*

Individual token descriptions are marked by enclosing them in square brackets. Token descriptions describe tokens as a Boolean combination of properties of the token itself (strings and features) together with properties of its 'grammatical neighborhood'. In the latter case, it can be a property of token X that it has (for example) a subject Y satisfying an embedded token description. The expansion of the macro `V_per_to_run` includes examples of such embedded token descriptions. For example, if the `V_per_to_run` branch of the disjunction in the appoint-rule is selected, then the top-level rule will only match a token if (a) it is headed by a form of the string *appoint*, (b) its logical object (which may be the surface subject if the clause is passive) is a person (as defined by the person recognizer macro `per`) and (c) it has either a modifier or a complement which is a '*run*-class verb', whose own logical object is recognized as an organization.

It is also possible to write regular expressions over token descriptions. So a single rule (like the expansion of the `regexp_V_to_be` macro, which is meant to match cases like *Widgets, Inc. has appointed John Doe to be its CXO*) can mix both 'grammatical' and 'sequential' notions of adjacency via a mix of attribute-value style notation (for grammatical adjacency) and regular expression style notation (for sequential adjacency). The 'attributes' in the attribute-value style constraints are actually implemented as directives in the token list walking automata, so token descriptions can include regular expressions over attribute symbols as well, as in the `V_per_to_run` macro, where we specify that the target token may be found by moving across either a *Mod* or a *Comp* link.

We have found this approach to grammars, based on token list automata, to be extremely flexible. During the initial stages of parsing, grammar rules are almost exclusively regular expressions over simple token descriptions (strings or combinations of lexical features). At the final stages of processing – where we put most event-detection grammars, for instance – sequential regular expressions are of more limited use and descriptions based on the grammatical functions that have been established by that point become more useful. But the transition from a string-based to a function-based notion of adjacency is quite smooth. This flexibility allows for very rapid development of grammars. In particular, it supports very well a generally data-driven approach to grammar development, as successive generalizations from initial examples. On the other hand, because we are doing strictly bottom up parsing, we lose the predictive power available in top-down phrase structure parsing or in mixed bottom-up/top-down approaches to parsing. Also, this is an approach which tends to emphasize precision over recall: it takes dedicated and sustained effort to achieve good recall numbers.

3.3 Semantic parsing

The semantic parser maps the surface structures into *logical forms* (semantic structures) via sentence/clause level analysis. This differs from shallow parsing

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

NP[Ernesto Bertarelli] , NP[CEO] PP[of Serono] , VG[explained] how NP[serendipity] VG[can play] NP[a role] . PP[In January 1998] , NP[Serono] VG[acquired] NP[Geneva Biomedical Research Institute] PP[from Glaxo Wellcome] .

Fig. 7. Sample shallow parsing results.

V-S : VG[explained], NP[Ernesto Bertarelli]
S-P : NP[Ernesto Bertarelli] , NP[CEO]
H-M : NP[CEO], PP[of Serono]
V-O : VG[explained], VG[can play]
V-S : VG[can play], NP[serendipity]
V-S : VG[can play], NP[a role]
H-M : VG[can play], how
V-T : VG[acquired], PP[In January 1998]
V-S : VG[acquired], NP[Serono]
V-O : VG[acquired], NP[Geneva Biomedical Research Institute]
H-M : VG[acquired], PP[from Glaxo Wellcome]

Fig. 8. Sample semantic parsing results.

which only identifies basic linguistic structures like NP, VG, etc. Traditional semantic parsers are based on high-level grammar formalisms like Context Free Grammars (CFG), but they are often difficult to maintain due to the lack of modularity. The system described here permits the cascading of the deep-level semantic parsing modules in a pipeline after shallow parsing. This makes grammar development much less complicated. Figures 7 and 8 illustrate the shallow parsing and semantic parsing results.

The core semantic parsing grammars consist of an elaborate VP (verb phrase) grammar, two SV (subject-verb) grammars and three verb-adjunct grammars. The logical relationships being captured form a verb-centric dependency tree (*verb* in *verb-centric* refers to a verb notion, including both verbs and deverbal nouns), consisting of links between: V-S (or VSubj, verb-subject), V-O (or VObj, verb-object), V-C (verb-complement), V-T (verb-time), V-L (verb-location), V-F (verb-frequency) and H-M (head-modifier) for other head-modifier relationships as well as the S-P (logical subject-predicative) link which captures the 'equivalence' relationship between two NPs, including appositive structures and copula structures. The rules for semantic parsing are fairly abstract. Most grammar rules check the category and sub-category (information like intransitive, transitive, di-transitive, etc.) of a verb (or a deverbal noun) instead of the word literal in order to build the semantic structures. On-line lexical resources, such as OALD, provide fairly reliable sub-categorization information for English verbs. Cymfony has used and enhanced OALD in the development of semantic parsing grammars.[5]

[5] For example, in addition to the subcategorization codes for English verbs available from OALD (V2A, V6A, etc.), we also added subcategorization codes for deverbal nouns (e.g.,

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

3.4 Expert lexicon

Recently, we have developed an extended regular expression based formalism named *Expert Lexicon* (Li, Zhang, Niu, Jiang and Srihari 2003), following the general trend of lexicalist approaches to NLP. An expert lexicon rule consists of both grammatical components as well as proximity-based keyword matching. All Expert Lexicon entries are indexed, similar to the case for the finite state tool in INTEX (Silberztein 1999). The pattern matching time is therefore reduced dramatically compared to a sequential finite state device. The formalism supports any number of expert lexicons being inserted at any levels of the pipeline processing, side by side with general grammars and statistical modules.

Every lexicon entry can trigger individual pattern matching rules. Most of these lexicalized pattern matching rules can, in principle, be equivalently coded using existing finite state grammar modules. But such grammars are difficult to maintain due to lack of good organization between general rules and lexicalized rules. Efficiency is also a problem due to the large branching factor of the finite state grammar.

The InfoXtract expert lexicon formalism has added power beyond the local pattern. An expert lexicon rule can check discourse constraints, such as co-occurrence of tokens at document, paragraph or sentence level or to check document level meta-data, such as statistical categories, zones, titles, source, author, etc. of the documents.

The Expert Lexicon is very effective in handling tasks such as phrasal verb identification and alias resolution. A phrasal verb lexicon handles problems like separable verbs such as 'put on the coat' vs. 'put the coat on' (Li *et al.* 2003b) and other individual or idiosyncratic linguistic phenomena. A special Alias Expert Lexicon is good at handling exceptional cases of general aliasing rules, to either link two strings with an *AliasLink* relationship or a *NotAliasLink* relationship. Expert lexicon has proven to be a very effective disambiguation tool used by our business team in specifying consumer brand lexicons. For example, in order to tag *747* as a brand name instead of a number, the word co-occurrence condition can be formulated as follows in the brand expert lexicon. The interpretation of this sample expert lexicon rule entry is: *Tag '747' as brand if 'Boeing' is also mentioned in the same paragraph.*

```
747:  CONDITION:  @cooccur(Boeing, p);
      ACTION:  \%assign_feature(NeBrand)
```

3.5 Machine learning

Machine learning (ML) has been used extensively in NLP. The most common use of ML in IE has been named entity tagging (Bikel, Schwartz and Weischedel 1999). ML

N2A, N6A). The InfoXtract semantic parser not only parses different syntactic *verb patterns* such as active vs. passive (e.g., *XYZ Inc. appointed John Smith as CEO* vs. *John Smith was appointed as CEO by XYZ Inc.*) into the same logical dependency links (V-S, V-O, V-C), but also decodes some logical dependency relationships for the nominal constructions headed by the corresponding deverbal noun (e.g., *XYZ Inc. appointment of John Smith as CEO*).

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

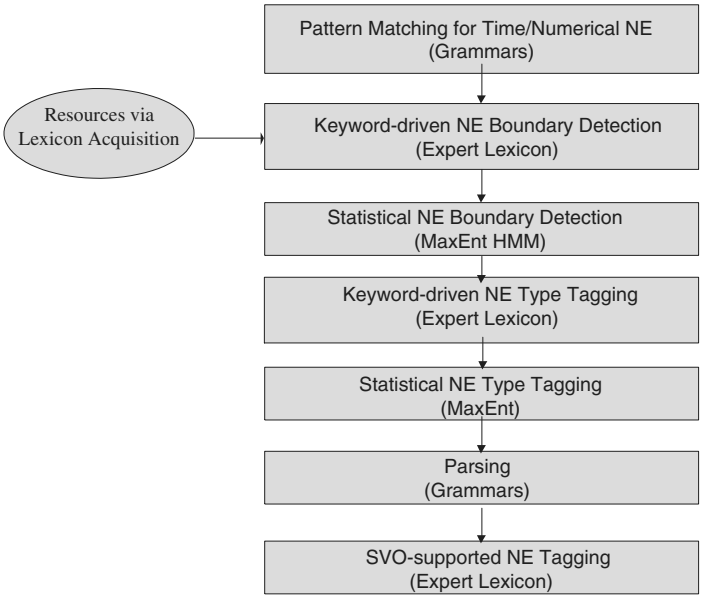


Fig. 9. Structure of hybrid NE tagger used in InfoXtract.

is typically useful in classification tasks, such as POS tagging, or topic categorization. In InfoXtract, ML techniques are used early in the cascade for the above tasks. ML is also used later in the cascade in tasks such as co-reference, verb-sense tagging (in progress) as well as text mining tasks such as profile and event clustering. In between, grammar modules are used to decode syntactic and semantic structures from text. Both supervised machine learning and unsupervised learning are used in InfoXtract. Supervised learning is used in hybrid modules such as NE (Srihari *et al.* 2000), NE Normalization (Li *et al.* 2002) and Co-reference. It is also used in the preprocessing module for orthographic case restoration of caseless input (Niu *et al.* 2004). Unsupervised learning involves acquisition of lexical knowledge and rules from a raw corpus. The former includes word clustering, automatic name glossary acquisition and thesaurus construction. The latter involves bootstrapped learning of NE and CE rules, similar to the techniques used in Riloff (1996). The results of unsupervised learning can be post-edited and added as additional resources for InfoXtract processing. A machine learning toolkit has been developed which is able to exploit InfoXtract-parsed corpora to produce rich feature representations useful in unsupervised learning.

The structure of the hybrid NE tagger system, consisting of both grammars and ML components, is shown in Figure 9. The Local Pattern Matching module contains simple grammar rules for time, date, percentage, and monetary expressions. These tags include the standard Message Understanding Conference (MUC) tags, as well as several other sub-categories defined by our organization such as *age*, *duration*, *measurement*, *address*, *email*, and *URL*. Subsequent modules are focused on tagging *location*, *person*, *organization*, *product* and *event names*. These rely on expert lexicons as well as a ML technique that combines the efficiency of HMMs with the increased

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Table 1. *Case restoration performance*

Case Restored	*P*	*R*	*F*
Overall	0.97	0.97	0.97
All-Lowercase	0.98	0.99	0.98
Non-Lowercase (not distinguishing i and ii)	0.92	0.88	0.90
(i) All-Uppercase	0.72	0.67	0.70
(ii) Mixed-case (not distinguishing iii and iv)	0.90	0.87	0.88
(iii) Initial Capitalization	0.90	0.87	0.88
(iv) Irregular Mixed-case	0.90	0.59	0.71

sophistication of maximum entropy (ME) techniques. ME learning permits diverse sources of information to be brought to bear in determining the best tag for a given string. NE tagging consists of two distinct subtasks in InfoXtract, namely NE boundary detection and NE type tagging. The former can directly use the lexical resources learned via lexicon acquisition. The decision on a final tag may be delayed till much later in the cascade after information from semantic parsing is available.

3.6 Benchmarking InfoXtract accuracy

This section shows performance benchmarking of the major components of InfoXtract from shallow processing to deep processing.[6] Wherever possible, we adopt community corpora, standards and scoring procedures.

3.6.1 Case restoration

Since much of the text used by the intelligence community is entirely in uppercase, it is necessary to adapt InfoXtract for better performance on this type of data. A case restoration technique was chosen to provide best performance as well as most flexibility. A corpus of 7.6 million words drawn from the general news domain is used in training the case restoration HMM (Niu, Ding and Srihari 2004). The irregular mixed-case word list is obtained from a general news corpus of 50 million words. A separate blind testing corpus of 1.2 million words drawn from the same domain is used for benchmarking. Table 1 shows that the overall F-measure is 97% (P for Precision, R for Recall and F for F-measure).

The score that is most important for NE is the F-measure (90%) of recognizing 'Non-Lowercase' words. This is because both 'All-Uppercase' (e.g. *IBM*) and 'Mixed-case' (e.g., *Microsoft*, *eBay*) are strong indicators for proper names while making the distinction between 'All-Uppercase' and 'Mixed-case' is both challenging and often unnecessary. The score of 90% is likely to be an underestimate: we found quite a number of 'Non-Lowercase' errors involve sentence-initial words due to the lack of a powerful sentence final punctuation detection module in the case restoration stage.

[6] Limited by space, the performance for Part-of-Speech tagging and Alias Co-reference (for person and organization NEs) are not shown here, but these two are very solid components, benchmarked consistently in the upper 90 (95%–99% F-score) in the general news domain.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Table 2. *NE Benchmarking on MUC-7 corpus*

NE Type	*P*	*R*	*F*
TIME	92%	91%	92%
DATE	92%	91%	92%
MONEY	100%	86%	93%
PERCENT	100%	95%	97%
LOCATION	96%	94%	95%
ORG	95%	95%	95%
PERSON	96%	93%	95%
Overall	95%	93%	94%

Table 3. *NE Benchmarking with or without Case*

	Normal case			*Case restored*		
NE Type	*P*	*R*	*F*	*P*	*R*	*F*
Overall	89.1%	89.7%	89.4%	86.8%	87.9%	87.3%
Degradation			2.3%	1.8%	2.1%	
Name NEs	88.4%	88.5%	88.5%	85.4%	86.0%	85.7%
LOCATION	85.7%	87.8%	86.7%	84.5%	87.7%	86.1%
ORGANIZATION	89.0%	87.7%	88.3%	84.4%	83.7%	84.1%
PERSON	92.3%	93.1%	92.7%	91.2%	91.5%	91.3%
Non-Name NEs	90.9%	93.5%	92.2%	90.8%	93.3%	92.0%
TIME	79.3%	83.0%	81.1%	78.4%	82.1%	80.2%
DATE	91.1%	93.2%	92.2%	91.0%	93.1%	92.0%
MONEY	81.7%	93.0%	87.0%	81.6%	92.7%	86.8%
PERCENT	98.8%	96.8%	97.8%	98.8%	96.8%	97.8%

3.6.2 NE tagging

NE remains the core foundation of an IE system; any effort in improving NE performance is very worthwhile. We have benchmarked our system on MUC-7 dry run data in a blind test; this data consists of 22,000 words and represents articles from *The New York Times*, as shown in Table 2.

In addition to the limited MUC corpus, we have also developed an annotated testing corpus in-house of 177,000 words in the general news domain. Using the same automatic scorer following MUC NE standards, we have benchmarked the NE performance on this corpus in its original mixed case text (Normal Case) and in case restored text (Case Restored), i.e. artificially removing the case information and then using the case restoration module before applying NE (Table 3). The performance benchmark on this dataset is not as good as that for the MUC testing corpus most probably due to the use of the MUC training corpus in our NE statistical tagger.

The overall F-measure for NE for the case restored corpus is 2.1% less than the performance of the system that takes the original case sensitive corpus as input. It is noteworthy that the 2.1% degradation due to the use of case restoration approach

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Table 4. *Shallow parsing benchmarking*

Category	VG	NP	AP
P	98.27%	98.72%	94.79%
R	99.14%	98.00%	83.49%

Table 5. *Parsing/CE benchmarking I*

	SVO		CE	
Case	Normal	Restored	Normal	Restored
P	89.50%	89.86%	96.0%	93.5%
R	81.67%	81.25%	82.8%	74.1%
F	85.41%	85.34%	88.9%	82.7%

is much lower than the degradation of 6.3% using the traditional NE re-training approach (Niu *et al.* 2004).

3.6.3 Parsing and CE

We have implemented an automatic scorer for benchmarking shallow parsing results. The benchmark indicates precision around 98% and recall ranging from 83.49% to 99.14% for shallow parsing, as shown in Table 4.[7] The MUC-7 dry run file was manually annotated by our linguist tester, who had not been involved in the development. This testing corpus was used to benchmark the shallow parsing tagging which includes VG (verb group), NP (noun phrase), and AP (adjective phrase). Shallow parsing results provide a solid foundation for semantic parsing and high-level IE such as CE, EP and Event Extraction.

In benchmarking SVO parsing and CE extraction, we have conducted experiments on two test corpora containing news articles regarding international terrorism. The first corpus (26.7 MB, 5,504 news articles) is drawn from Foreign Broadcast Information Service (FBIS) sources, with the date range Jan. 2000 through Jan. 2002. The second corpus (106 MB, 23,554 news articles) is drawn from North American (NA) news sources in 2002 (NY Times, CNN, etc.). From the InfoXtract-processed testing corpora, our testers who were not involved in the corresponding grammar development randomly picked 250 logical Subject-Verb-Object (SVO) structural links and 60 AFFILIATION and POSITION relationships for manually checking the performance of SVO parsing and CE extraction (Table 5).

In addition to our own testing corpora, we also used the MUC-7 dry run corpus in further benchmarking parsing and CE (Table 6) for normal, mixed-case

[7] The lower recall for AP is due to insufficient rule development; it is not because AP is more difficult to chunk than NP and VG. AP was added as a shallow parsing target later on, and at the time of benchmarking, the AP grammar only contained a few rules.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Table 6. *Parsing/CE benchmarking II*

Type	Example	P	R	F
Shallow parsing	NP[John Smith] , 46 , NP[CEO] PP[of Seattle - based XYZ Inc.] , VG[was sued] yesterday	95.3%	96.7%	96.0%
Semantic parsing	V-O: [was sued], [John Smith] H-M: [John Smith], 46 S-P: [John Smith], [CEO] H-M: [CEO], [of Seattle - based XYZ Inc.] H-M: [was sued], yesterday	83.3%	79.3%	81.3%
Affiliation	CE-Affiliation: [John Smith], [XYZ Inc.]	92.3%	53.3%	72.8%
Position	CE-Position: [John Smith], [CEO]	91.3%	70.0%	80.7%
Location-of	CE-Location-of: XYZ Inc., Seattle	83.3%	50.0%	66.7%
Descriptors	XYZ Inc. is a leader in TV industry. CE-Descriptors: [XYZ Inc.], [a leader]	58.3%	53.8%	56.1%
Others	Other CE relationships such as: CE-Age: [John Smith], 46	60.4%	48.7%	54.6%

input. This time, we have included shallow parsing and additional semantic parsing relations (Verb-Complement relation, Head-Modifier relation, Equivalence Relation and Conjunctive relation) as well as other CE relationships.

Generally speaking, the keyword-driven, multi-level rule system for CE is geared more to precision than recall. Our observation is that it is fairly easy to reach 85–90% precision and 50–70% recall for initial development once a domain is determined. To achieve between 50% and 70% for recall, it is basically the size of the grammars that matters. When more time is spent in the development of more pattern rules, the recall will gradually increase. But going beyond 70% recall is difficult. The majority of CE relationship instances are expressed with fixed or easily predictable patterns in a given domain while the remaining instances can take various forms in both vocabulary and structures. If we narrow the general news domain to a more specific domain, the recall is expected to rise.

3.6.4 Entity profile fusion

Since the corpus-level EP is a *fused* information object, there is no easy way of directly benchmarking it using traditional annotated corpus methods. The quality of EP is directly affected by the underlying support from Case Restoration (for FBIS), NE, Alias Co-reference and Parsing as well as CE. Due to the information redundancy in a large corpus, even with a modest recall at about 50%–70%, the EP extraction system demonstrates tremendous value in collecting entity-centric information which is otherwise scattered in different documents in the corpus. We attempted to measure the richness of information for the extracted EPs , an important indicator for their value in applications. The measurement was conducted on two corpora, illustrated in Table 7. It contains the following columns:

- Discourse EPs: number of document-level EPs;
- Corpus EPs: number of corpus-level EPs after fusion;

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Table 7. *EP Information statistics*

Corpus	EP Type	Discourse EPs	Corpus EPs	CE	Events	Message per EP	Top 10
FBIS	Person	15,666	9,571	12,864	11,718	2.57	109
(26.7MB)	Org	35,468	16,827	24,858	10,758	2.12	41
NA	Person	131,769	55,377	108,921	94,806	3.68	472
(106MB)	Org.	218,310	60,922	132,301	72,066	3.35	417

- CE: total number of *distinct* CE relationship messages;
- Events: total number of involved event mentions;
- Message/EP: average number of distinct messages per corpus EP;
- Top 10: average number of distinct messages per corpus EP for the top 10 information-rich EPs. This includes *Osama bin Laden*, *George W. Bush*, *Colin Powell*, *Saddam Hussein*, *Taliban*, *United Nations*, *Pentagon* for the NA corpus and *Kim Chong-il*, *Vladimir Putin*, *United Nations*, *European Union*, *Taliban* for the FBIS corpus.[8]

4 Engineering architecture

The InfoXtract engine has been developed as a modular, distributed application and is capable of processing up to 20 MB per hour on a single Pentium processor. The system has been tested on very large (>1 million) document collections. The architecture facilitates the incorporation of the engine into external applications requiring an IE subsystem. Requests to process documents can be submitted through a web interface, or via FTP. The results of processing a document can be returned in XML. Since various tools are available to automatically populate databases based on XML data models, the results are easily usable in web-enabled database applications. Configuration files enable the system to be used with different lexical/statistical/grammar resources, as well as with subsets of the available IE modules.

InfoXtract supports two modes of operation, active and passive. It can act as an active retriever of documents to process or act as a passive receiver of documents to process. When in active mode, InfoXtract is capable of retrieving documents via HTTP, FTP, or the local file system. When in passive mode, InfoXtract is capable of accepting documents via HTTP. Figure 10 illustrates a multiple processor configuration of InfoXtract focusing on the typical deployment of InfoXtract within an application. The arrows indicate data flow through the system. Ultimately, the Document Manager produces an XML document representing the analysis of the

[8] A distinct message is defined as a unique attribute-value pair in the fused EP template that consists of a list of pre-defined CE attributes (such as AFFILIATION, POSITION). Note that the fusion involves elimination of redundant messages, currently implemented by sub-string matching of the head units of the CE value phrases between the values of the same CE attribute.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

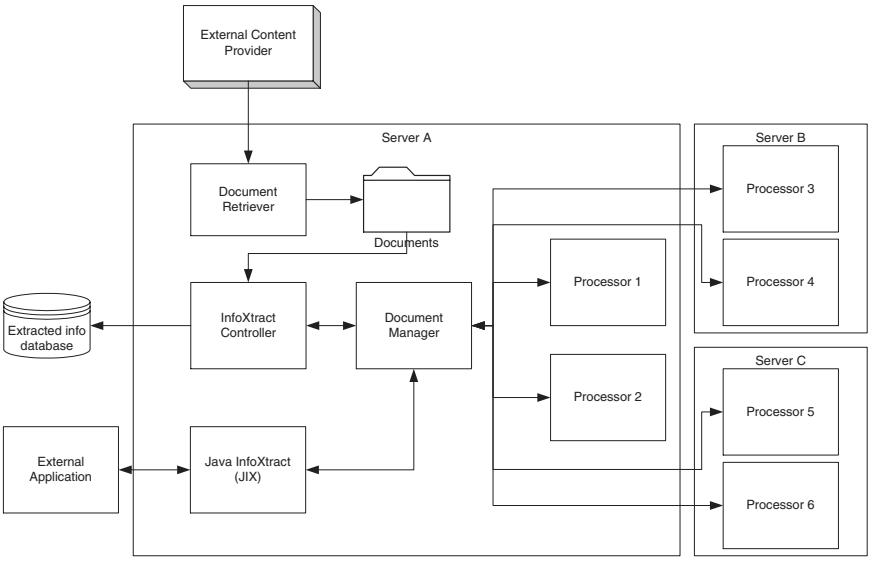


Fig. 10. High level architecture.

input text. This document is then transformed by the InfoXtract Controller (which knows about details of the document source and metadata encoding) into an XML document that can be parsed and loaded by a database manager.

The architecture facilitates scalability by supporting multiple, independent Processors. The Processors can be running on a single server (if multiple CPUs are available) and on multiple servers. The Document Manager distributes requests to process documents to all available Processors. Each component is an independent application. All direct inter-module communication is accomplished using the Common Object Request Broker Architecture (CORBA). CORBA provides a robust, programming language independent, and platform neutral mechanism for developing and deploying distributed applications. Processors can be added and removed without stopping the InfoXtract engine. All modules are self-registering and will announce their presence to other modules once they have completed initialization.

The Document Retriever module is only used in the active retriever mode. It is responsible for retrieving documents from a content provider and storing the documents for use by the InfoXtract Controller. The Document Retriever handles all interfacing with the content provider's retrieval process, including interface protocol (authentication, retrieve requests, etc.), throughput management, and document packaging. It is tested to be able to retrieve documents from content providers such as Northern Light, Factiva, and LexisNexis. Since the Document Retriever and the InfoXtract Controller do not communicate directly, it is possible to run the Document Retriever standalone and process all retrieved documents in a batch mode at a later time.

The InfoXtract Controller module is used only in the active retriever mode. It is responsible for retrieving documents to be processed, submitting documents for processing, storing extracted information, and system logging. The InfoXtract

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Controller is a multi-threaded application that is capable of submitting multiple simultaneous requests to the Document Manager. As processing results are returned, they are stored to a repository or database, an XML file, or both.

The Document Manager module is responsible for managing document submission to available Processors. As Processors are initialized, they register with the Document Manager. The Document Manager uses a round robin scheduling algorithm for sending documents to available Processors. A document queue is maintained with a size of four documents per Processor. The Processor module forms the core of the IE engine. InfoXtract utilizes a multi-level approach to NLP that has already been described. The JIX module is a web application that is responsible for accepting requests for documents to be processed. This module is only used in the passive mode. The document requests are received via the HTTP Post request. Processing results are returned in XML format via the HTTP Post response.

A typical configuration provides throughput of approximately 12,000 documents (avg. 10KB) per day. A smaller average document size will increase the document throughput. Increased throughput can be achieved by dedicating a CPU for each running Processor. Each Processor instance requires approximately 500 MB of RAM to run efficiently. Processing speed increases linearly with additional Processors/CPUs, and CPU speed. In the current state, with no speed optimization, using a bank of eight processors, it is able to process approximately 100,000 documents per day. Thus, InfoXtract is suitable for high volume deployments. The use of CORBA provides seamless inter-process and over-the-wire communication between modules. Computing resources can be dynamically assigned to handle increases in document volume. Low bandwidth installations can be run on servers with very modest specifications. High bandwidth installations can be set up by using several low cost servers rather than a single high cost server. There are bottlenecks at each end of the system. In the current architecture there is only a single Document Manager, which is responsible for distributing documents to Processors and packaging their output. So, at some point adding new Processors would no longer help. There is no logical necessity preventing us from multiplying the number of Document Managers, so the bottlenecks can be moved around, but there is obviously an inescapable bottleneck in any application that requires the integration of information from multiple documents. Ultimately, the ability of the database to integrate new information will induce a ceiling beyond which the addition of new Processors is of little practical use. Although there has been considerable work in the area of database optimization, a discussion of this is beyond the scope of this paper.

A standard document input model is used to develop effective preprocessing capabilities. Preprocessing adapts the engine to the source by presenting metadata and zoning information in a standardized format and performing restoration tasks (e.g. case restoration). Efforts are underway to configure the engine such that zone-specific processing controls are enabled. For example, zones identified as titles or subtitles can be tagged using different criteria than running text. The engine has been deployed on a variety of input formats including HUMINT documents (all uppercase), the Foreign Broadcast Information Services feed, live feeds from content

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

providers such as Factiva (Dow Jones/Reuters), LexisNexis, as well as web pages. A user-trainable, high-performance case restoration module (Niu *et al.* 2004) has been developed that transforms caseless input such as speech transcripts into mixed-case before being processed by the engine. The case restoration module eliminates the need for separate IE engines for handling caseless and casefull documents; this is easier and more cost effective to maintain.

4.1 Scalability

In this section, we address the scalability of the InfoXtract engine. Scalability has been defined in the software engineering discipline as follows: *the ease with which a system or component can be modified to fit the problem area* (Software Engineering Institute: `http://www.sei.cmu.edu/str/indexes/glossary/scalability.html`). The issue of volume is addressed in the context of this definition of scalability. There are several typical operational environments that InfoXtract could be deployed in, including DoD, other government agencies, as well as corporate settings. Typically, the documents are obtained through live news feeds, intelligence feeds, intranets, or from the web. Although it is theoretically possible to increase the number of servers to increase bandwidth, there is a practical limit on the number of servers that can be effectively maintained.

In the case of typical DoD installations, it is not uncommon to see a volume of documents exceeding 100,000 per day. Typically, these documents are first filtered by a message dissemination system; individual users or departments can customize their profiles such that only documents matching those profiles are sent through. This has the effect of greatly reducing the overall number of documents that an IE engine needs to process. Feedback from the intelligence community suggests that the value of an IE engine would be greatly enhanced if it could be used before any filtering were to take place, thus requiring the engine to process in excess of 100,000 documents per day. The advantage of this scenario lies in the message filtering system's ability to exploit IE output. In the case of business settings using live feeds, volume has thus far not been a problem. Here again, documents are first filtered using business criteria (such as mention of certain consumer brands, companies, etc.); the resulting volume of documents does not exceed more than a few hundred per day and has thus far not posed a challenge. Finally, the issue of using InfoXtract with web documents is addressed. Although it is not claimed that InfoXtract's indexing capabilities at present scale up to that of search engine indexes, the engine can still be used to process live web pages. This is accomplished by using InfoXtract after the result of a broad-based query, which typically results in several thousands of web pages per topic area. The resulting pages are then within the scope of InfoXtract's current bandwidth.

There are some major issues remaining with respect to the scalability of InfoXtract for very large volume, dynamically changing collections. These include (i) optimizing the speed of processing, (ii) the design of indexes to store and query InfoXtract output, and (iii) the ability to customize grammars without having to reprocess documents. In order for InfoXtract to come close to the speed of word-based

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

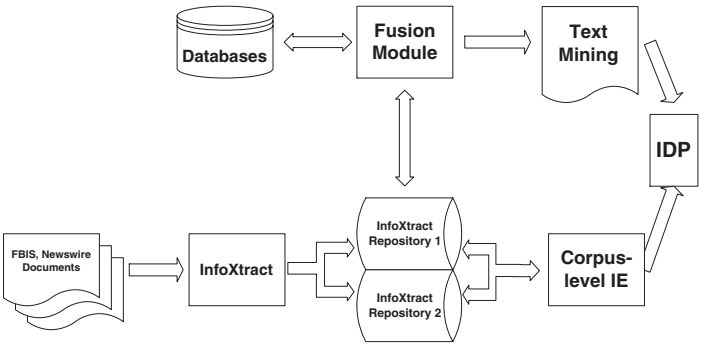


Fig. 11. Extensions to InfoXtract.

indexing used by search engines, it is necessary to look at the underlying grammar and machine-learning formalisms. More clever techniques of using indexed and/or compressed versions of the token list must be investigated. A Repository module has been designed for the purpose of corpus-level IE: this supports exact and inexact matching and incorporates grammatical as well as statistical similarity. It is described in more detail in the next section. The Repository module indexes the complete token list information: this provides the basis for adding domain-specific grammars without having to reprocess documents. Current efforts include the ability to adapt user-specified grammars to work on these archived token lists. The Repository module is also used in semi-automated domain porting, specifically in learning new NE glossaries and relationships.

5 Corpus-level IE

Efforts have extended IE from the document level to the corpus level. Although most IE systems perform corpus-level information consolidation at an application level, it is felt that much can be gained by doing this as an extended step in the IE engine. A *repository* has been developed for InfoXtract that is able to hold the results of processing an entire corpus. A proprietary indexing scheme for indexing token-list data has been developed that enables querying over both the linguistic structures as well as statistical similarity queries (e.g. the similarity between two documents or two entity profiles). The repository is used by a *fusion* module in order to generate cross-document entity profiles as well as for text mining operations. The results of the repository module can be subsequently fed into a relational database to support applications. This has the advantage of filtering much of the noise from the engine level and doing sophisticated information consolidation before populating a relational database. The architecture of these subsequent stages is shown in Figure 11 (directional arrow shows the data flow and bidirectional arrow stands for the access to storage data).

Information Extraction has two anchor points: (i) entity-centric information which leads to an EP; and (ii) action-centric information which leads to an event scenario. Compared with the fusion of extracted event instances into a cross-document

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Table 8. *Sample entity profile*

Name	Mohamed Atta
Aliases	Atta; Mohamed
Position	apparent mastermind; ringleader; engineer; leader
Age	33; 29; 33-year-old; 34-year-old
Where-from	United Arab Emirates; Spain; Hamburg; Egyptian; . . .
Modifiers	on the first plane; evasive; ready; in Spain; in seat 8D . . .
Descriptors	hijacker; al-Amir; purported ringleader; a square-jawed 33-year-old pilot; . . .
Association	bin Laden; Abdulaziz Alomari; Hani Hanjour; Madrid; American Media Inc.; . . .
Involved-events	move-events (2); accuse-events (9), convict-events (10), confess-events (2), arrest-events (3), rent-events (3), . . .

event scenario, cross-document EP fusion is a more tangible task, based mainly on resolving aliases. Even with modest recall in the underlying CE relationship extraction, the corpus-level EP demonstrates tremendous value in collecting information about an entity. This is shown in Table 8 for only part of the profile of 'Mohamed Atta' from one experiment based on a collection of news articles.

6 Domain porting

Considerable efforts have been made to keep the core engine as domain independent as possible; domain specialization or tuning happens with minimum change to the core engine, assisted by automatic or semi-automatic domain porting tools that have been developed. InfoXtract incorporates several distinct approaches in achieving domain portability: (i) the use of a standard document input model, pre-processors and configuration scripts in order to tailor input and output formats for a given application, (ii) the use of tools in order to customize lexicons and grammars, and (iii) unsupervised machine learning techniques for learning new named entities (e.g._weapons) and relationships based on sample seeds provided by a user.

6.1 Lexicon grammar development environment

A primary objective of the InfoXtract design has been rapid and easy domain porting by non-linguists (Li *et al.* 2003a). This has resulted in a development and customization environment known as the Lexicon Grammar Development Environment (LGDE). The LGDE permits users to modify named entity glossaries, alias lexicons and general-purpose lexicons. It also supports example-based grammar writing; users can find events of interest in sample documents, process these through InfoXtract and modify the constraints in the automatically generated rule templates for event detection. With some basic training, users can use the LGDE to customize InfoXtract for their applications. This facilitates customization of the system in user applications where access to the input data to InfoXtract is restricted. The grammar previously presented in Figure 6 is actually an example of a lexicon grammar. It shows that the domain specific event *executive change* grammar exploits the core syntactic and semantic parsing provided by InfoXtract. Figure 12 illustrates part of

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

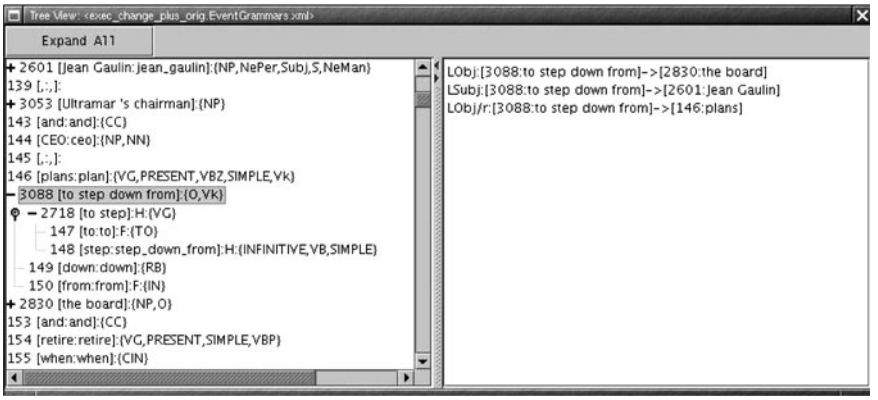


Fig. 12. Screen snapshot of LGDE.

the LGDE's visual environment, in particular the token list viewer. The process of customizing grammars for a specific domain typically involves: (i) finding documents and instances of the event in question and highlighting them, (ii) processing the documents/sentences through InfoXtract and generating initial grammars for these events, (iii) loosening or tightening the constraints on the initial grammar to increase precision, recall or both. InfoXtract and the LGDE are currently being used in a classified environment in order to customize event detection for intelligence purposes. It is important to point out that those customizing the engine are (and were) not part of the InfoXtract development team.

6.2 Automatic domain porting

In terms of automatic domain porting, (Riloff, Ellen and Jones 1999) have explored multi-level bootstrapping for learning IE lexicons/glossaries based on a limited number of seeds. As a domain porting technique for InfoXtract, a new two-phase bootstrapping approach to IE has been explored (Niu *et al.* 2003a). The objective is to automatically learn new entities and new relationships that can be defined by the user. This approach only requires a large raw corpus in the target domain plus a few IE seeds from the user to guide the learning. The weakly supervised learning is performed using the repository module that contains the InfoXtract-parsed corpus. This approach combines the strengths of parsing-based symbolic rule learning and the high performance, linear-string-based Hidden Markov Model (HMM) to automatically derive a customized IE capability with balanced precision and recall. The key idea is to apply precision-oriented symbolic rules learned in the first stage to a large corpus in order to construct an automatically tagged training corpus. This training corpus is then used to train an HMM to boost the recall. The experiments conducted in named entity (NE) tagging (Niu *et al.* 2003a) and relationship extraction (Niu, Li, Srihari and Crist 2003b) show a performance close to the performance of supervised learning systems. The use of this approach in event bootstrapping is now in progress.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

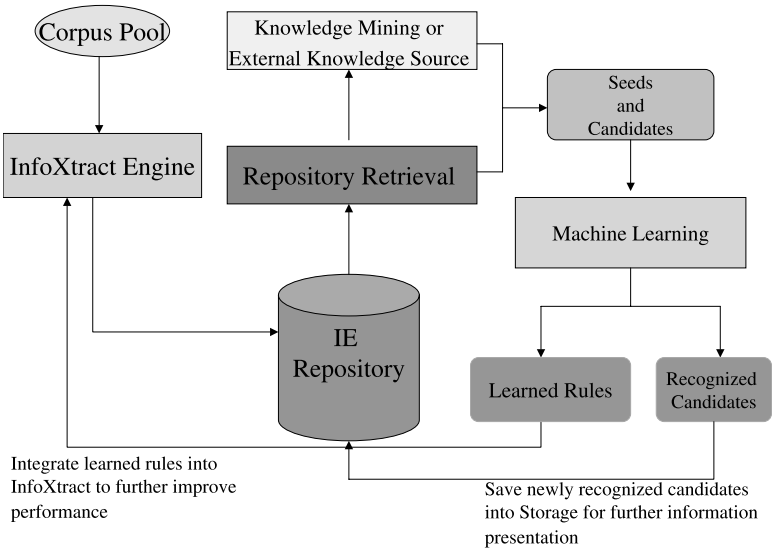


Fig. 13. IE bootstrapping architecture.

Figure 13 shows the overall design of the IE bootstrapping framework. The system contains four key components: (i) InfoXtract core engine, (ii) IE Repository, (iii) machine learning module, and (iv) knowledge mining module. The foundation for bootstrapping is InfoXtract which is also the target object for customization. The IE bootstrapping relies heavily on the process of knowledge mining, including IE rule learning and lexical knowledge acquisition. The repository module links knowledge mining modules with the core InfoXtract engine. This defines a process for discovering domain dependent knowledge by applying domain-independent NLP and IE to a raw training corpus in the target domain. The mined knowledge will be fed back into the core engine to support the adaptation of the engine to the new domain, thus creating a domain dependent version of the engine.

More specifically, this new bootstrapping approach uses successive learners to overcome the error propagation problem faced by traditional iterative learning. The first learner makes use of the structural evidence to learn high precision patterns. Although the first learner typically suffers from the recall problem, we can apply the learned rules to a huge parsed corpus. In other words, the availability of an almost unlimited raw corpus compensates for the modest recall. As a result, large quantities of IE instances can still be tagged to form a sizable automatically constructed training corpus. This automatically annotated IE corpus can then be used to train the second HMM-based learner to boost the recall by exploiting rich surface patterns.

7 Applications

The InfoXtract engine has been used in several applications, including question-answering, an Information Discovery Portal (IDP) and the Brand Dashboard system (`www.cymFony.com`).

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

7.1 InfoXtract-supported question answering

An effective question-answering system (Srihari and Li 2000) has been developed that incorporates multiple levels of information extraction (NE, CE, SVO) in extracting answers from unstructured text documents. Most QA systems are focused on sentence-level answer generation. Such systems are based on information retrieval (IR) techniques such as passage retrieval in conjunction with shallow IE techniques such as named entity tagging. Sentence-level answers may be sufficient in many applications focused on reducing information overload: instead of a list of URLs provided by search engines which need further perusal by the user, sentences containing potential answers are extracted and presented to the user. However, if the goal is precise answers consisting of a phrase, or even a single word or number, new techniques for QA must be developed. Specifically, there is a need to use more advanced IE and NLP in the answer generation process. The QA system based on InfoXtract represents a technique whereby multiple levels of IE are utilized in an attempt to generate the most precise answer possible, backing off to coarser levels where necessary.[9] In particular, generic grammatical relationships are exploited in the absence of information about specific relationships between entities.

7.2 Intelligence Discovery Portal (IDP)

The IDP is an application built on top of InfoXtract designed to help intelligence analysts in the task of situation awareness, namely the ability to identify significant patterns of events from unstructured text documents. The IDP supports both the traditional top-down methods of browsing through large volumes of information as well as novel, data-driven browsing. A sample user interface is shown in Figure 14. Users may select 'watch lists' of entities (people, organizations, targets, etc.) that they are interested in monitoring. Users may also customize the sources of information they are interested in processing. Top-down methods include topic-centric browsing whereby documents are classified by topics of interest. IE-based browsing techniques include entity-centric and event-centric browsing. Entity-centric browsing permits users to track key entities (people, organizations, targets) of interest and monitor

[9] There are four levels of granularity in this system. Level 1 is the precise answer (typically a name or a short phrase). This happens in two cases: first, where the question asks for a relationship known to the system and that relationship has been extracted in our repository. For example, to answer the question *Who is John Smith's employer*, if InfoXtract has extracted *XYZ Inc.* as the CE affiliation for *John Smith*, then this answer will be returned. The second case is a full match between the core semantic parse trees of the question and a potential snippet. For example, the parse tree of the question *When was John Smith hired* matches the parse tree of the snippet *XYZ Inc. hired John Smith in 2001*. The decoded asking point in the question parse tree matches the time PP *in 2001*, which is then returned as the answer. Level 2 provides clause-level answers. This happens when the entity asking point type-matches at least one NE in the most probable candidate clause-level snippet decided by the answer rank module. Level 3 provides paragraph-level answrs. This level is desgined to answer special types of questions such as some *wh*y-questions and *how*-questions (e.g. *How to make chocolate cookies?*) which typically require longer snippets and contexts for answers. The fourth level involves a question which cannot be interpreted, meaning that no asking point or question type can be determined. The system defaults to traditional search engine output, namely URLs.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

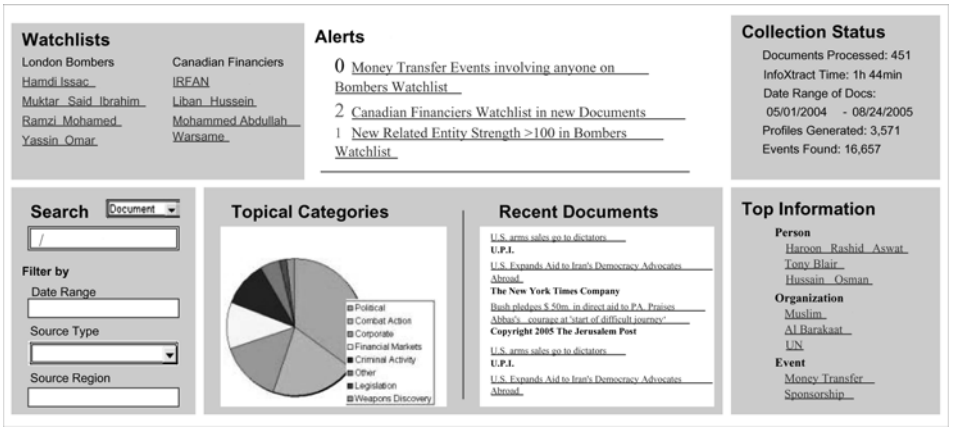


Fig. 14. Information Discovery Portal.

information pertaining to them. Event-centric browsing focuses on significant actions including money movement and people movement events. Visualization of extracted information is a key component of the IDP. InfoXtract's ability to generate linked entity-event profiles allows a user to visualize an entity, its attributes and its relation to other entities and events. Starting from an entity (or event), event or relationship chains can be traversed to explore related items. Timelines facilitate visualization of information in the temporal axis.

Recent efforts have included the integration of InfoXtract with visualization tools such as the Web-based Timeline Analysis System (WebTAS) (`www.webtas.com`). The IDP reflects the ability for users to select events of interest and automatically export them to WebTAS for visualization. Efforts are underway to integrate higher-level event scenario analysis tools such as the Terrorist Modus Operandi Detection System (TMODS) (`www.21technologies.com`) into the IDP.

7.3 Brand Dashboard

Brand Dashboard is a commercial application for marketing and public relations organizations to measure and assess media perception of consumer brands. The InfoXtract engine is used to analyze several thousand electronic sources of information provided by various content aggregators (Factiva, LexisNexis, etc.) The engine is focused on tagging and generating brand profiles that also capture salient information such as the descriptive phrases used in describing brands (e.g. *cost-saving*, *non-habit forming*) as well as user-configurable specific messages that companies are trying to promote and track (*safe and reliable*, *industry leader*, etc.) The output from the engine is fed into a database-driven web application which then produces report cards for brands containing quantitative metrics pertaining to brand perception, as well as qualitative information describing characteristics. A sample screenshot from Brand Dashboard is presented in Figure 15. It depicts a report card for a particular brand, highlighting brand strength as well as highlighting metrics that have changed the most in the last time period. The 'buzz box' on the right hand side illustrates

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

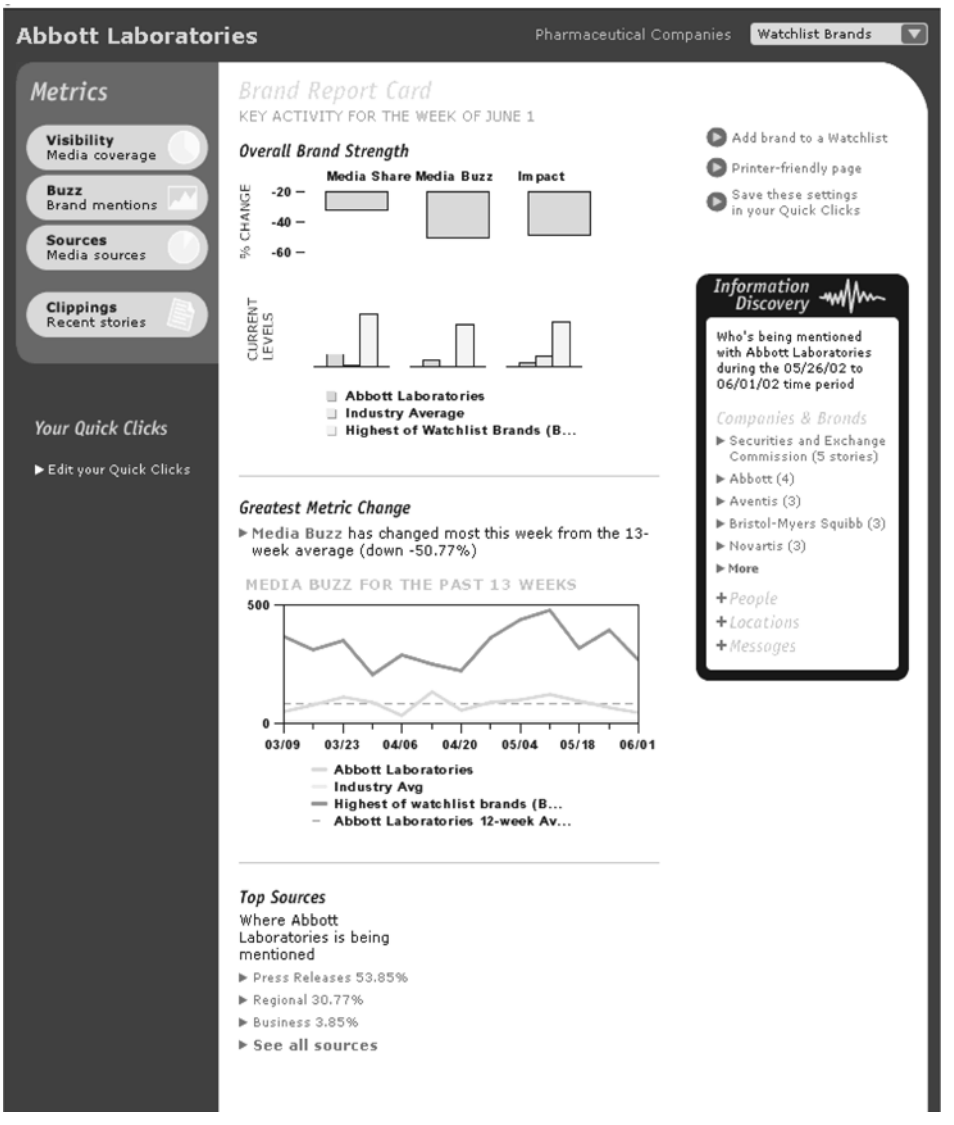


Fig. 15. Report card from Brand Dashboard.

companies and brands, people, analysts, and messages most frequently associated with the brand in question.

8 Related work

There are many aspects in which to discuss work related to the IE system presented here, including sophistication of IE output, accuracy, and NLP techniques used. Figure 16 illustrates the various levels of information extraction, the capabilities associated with each level, along with the linguistic challenges in achieving these

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

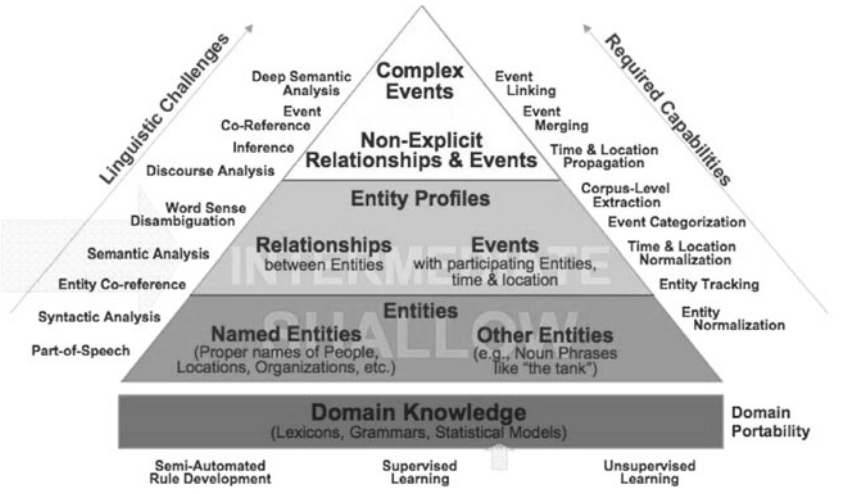


Fig. 16. Levels of information extraction, and required capabilities and NLP challenges.

capabilities.[10] As the diagram illustrates, three levels of IE, shallow, intermediate and deep levels can be defined reflecting increasing sophistication. These roughly correspond to local (sentence-level), document-level and corpus-level text processing. This section cannot survey all the NLP techniques mentioned in the diagram; the focus is on complete IE systems incorporating advanced techniques.

Much of the early work in IE focused on shallow information extraction, consisting primarily of named entity taggers, and shallow parsers. Nymble (Bikel *et al.* 1999) and NetOwl (Krupka and Hausman 1998) are among the most widely used NE taggers reflecting a statistical and rule-based approach respectively. Several IE systems such as LasIE (Humphreys, Gaizauskas, Azzam, Huyck, Mitchell and Cunningham 1998) and SIFT (Miller *et al.* 1998) have achieved capabilities required of intermediate-level IE such as relationship and event detection. LasIE incorporates co-reference leading to template element filling as well as scenario template filling; the latter involves event co-reference as well as entity co-reference. One focus of the corpus-level IE efforts is cross-document entity profile construction (Bagga and Baldwin 1998).

Temporal and spatial normalization have also been the focus of much work. *HLT/NAACL 2003 Workshop on Analysis of Geographic References* (Kornai and Sundheim 2003) presents several efforts focused on disambiguating geographical references. Recent efforts have utilized the world-wide web as a source of training documents in order to perform such disambiguation. Temporal normalization, in particular the task of time-stamping events (Pustejovsky, Castaño, Ingria, Sauri, Gaizauskas and Setzer 2003), has also been researched. The MiTAP system

[10] Figure from Pine, Srihari and Li (2003) Situation Awareness: Text Extraction. *Briefing by the Air Force Research Laboratory information Directorate (AFRL/IF) to the AF Scientific Advisory Board (SAB) 2003.*

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

(Damianos, Wohlever, Kozierok and Ponte 2003) reflects many of these capabilities in one system.

Automated domain porting for IE systems is essential to achieve better performance. Machine learning techniques have primarily focused on supervised learning techniques. While these produce good results, they suffer from the typical problems associated with supervised approaches, namely the need to annotate data. For NE tagging, even with the use of graphical annotation tools, this can be an onerous task. To overcome this, weakly supervised methods, primarily based on bootstrapping, have been proposed. Riloff (2003) has extensively investigated such techniques in NE tagging, relationship and event detection. The Proteus project[11] developed at NYU has also used bootstrapping techniques.

Finally, benchmarking of IE systems has also evolved, from the early MUC standards to the current ACE standards (ACE 2004) which extend the task to generic entities (nominals), as well as entity tracking. Good performance on the generic entity tagging task requires sophisticated co-reference. Relationships and events remain a standard by which IE systems are benchmarked, but even here there have been changes. Whereas the MUC standards emphasized complex scenario template filling as a goal, the focus is shifting to benchmarking more practical tasks which are necessary along the path to deep extraction.

9 Summary and future work

This paper has described the motivation behind InfoXtract, a domain independent, portable, intermediate-level IE engine. InfoXtract focuses on intermediate-level extraction, which is tractable and provides usable information such as entity profiles and concept-based general events. Applications have been built based on this output. Although these information objects reflect sophisticated NLP abilities, they provide the focus for further enhancements to tasks such as co-reference resolution, word sense disambiguation, etc. They also provide the basis for further work on corpus-level information extraction where the focus is on generating cross-document entity profiles and event linking and co-reference. Domain customization using LGDE exploits the domain independent processing fully; new event types are seen as a remapping of the generic events and their slots. Furthermore, it permits non-linguists to attempt the domain customization process in a more intuitive manner. New formalisms such as the expert lexicon permit users flexible options for specifying context necessary for disambiguation. The architecture of the engine, both from an algorithmic perspective and software engineering perspective, has been discussed. The token list structure has been designed for effective, efficient and modular pipeline processing. Benchmarks have demonstrated state-of-the-art performance of the InfoXtract engine. InfoXtract was designed to be a scalable engine capable of being used in diverse applications.

There are several directions in which the engine could be, and is being, improved. An active area of research is time stamping of events and relationships; an effort is

[11] http://nlp.cs.nyu.edu/projects/index.shtml

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

underway to automatically extract temporal relationship information about events mentioned in texts, as well as to track shifts in the current reference time in a text (so that, for example, the time denoted by *a year later* may depend on where it appears in a given text). This in turn will lead to better linking and merging of information as well as greatly enhancing the value of the output. We believe that this will require some ability of our system to represent aspects of narrative and rhetorical structure however. Another major initiative is enhancing corpus-level IE; a key factor here is the ability to correlate InfoXtract output with structured databases. Support for more intuitive domain customization tools, especially the semi-automatic learning tools, is also a major focus. Engineering issues include the support for more diverse input formats and genres (e.g. e-mail) as well as exploiting metadata in the extraction tasks. Support for languages other than English remains a goal.

Finally, InfoXtract can be seen as a means to an end, the end being text mining and true information discovery. In order to support these goals, it will be necessary to integrate InfoXtract with traditional information retrieval tools such as search engines, topic categorizers and filters, as well as question answering systems. One of the grand challenges is to develop hybrid indexing and retrieval systems which seamlessly integrate the above functionalities. Although IE technology has reached the stage of being useful, it needs to evolve to a 'self-learning' point where given a representative raw corpus from a target domain, it will be automatically adapted to the domain.

Acknowledgments

This work was supported in part by SBIR grants F30602-01-C-0035, F30602-03-C-0156 and F30602-02-C-0057 from the Air Force Research Laboratory (AFRL/IFEA). The authors wish to thank Carrie Pine of AFRL for reviewing and supporting this work. The authors were previously with Cymfony, Inc., and gratefully acknowledge the support of the Cymfony R&D, Engineering and Testing teams who have contributed to the refinement and success of InfoXtract over the years.

References

ACE 2004. `http://www.nist.gov/speech/tests/ace/index.htm`.

Aho, A. V. and Ullman, J. D. (1971) Translations on a context-free grammar. *Information and Control* **19**(5): 439–475.

Aone, A. and Ramos-Santacruz, M. (2000) REES: A Large-Scale Relation and Event Extraction System. *Proceedings of ANLP-NAACL 2000*. Seattle, WA. `http://acl.ldc.upenn.edu/A/A00/A00-1011.pdf`.

Bagga, A. and Baldwin, B. (1998) Entity-based cross-document coreferencing using the vector space model. *Proceedings of COLING-ACL'98*, pp. 79–85. Montreal, Canada.

Bikel, D. M., Schwartz, R. and Weischedel, R. M. (1999) An algorithm that learns what's in a name. *Machine Learning* **34**: 211–231.

Chinchor, N. and Marsh, E. (1998) MUC-7 information extraction task definition (version 5.1). *Proceedings of MUC-7*.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C., Dimitrov, M., Dowman, M., Aswani, N. and Roberts, I. (2003) *Developing Language Processing Components with GATE: A User Guide*.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Damianos, L., Wohlever, S., Kozierok, R. and Ponte, J. (2003) MiTAP: A case study of integrated knowledge discovery tools. *Proceedings of the Thirty-Sixth Annual Hawaii International Conference on Systems Sciences (HICSS-36)*. Big Island, Hawaii.

Engelfriet, J., Hoogeboom, H. J. and Van Best, J.-P. (1999) Trips on trees. *Acta Cybernetica* **14**(1): 51–64.

Gécseg, F. and Steinby, M. (1997) Tree languages. In: G. Rozenberg and A. Salomaa (Eds.), *Handbook of Formal Languages: Beyond Words*. (Vol. 3, pp. 1–68). Springer.

Grishman, R. (1997) *TIPSTER Architecture Design Document Version 2.3*.

Han, J. (1999) Data Mining. In: J. U. a. P. Dasgupta (editor), *Encyclopedia of Distributed Computing*. Kluwer Academic.

Hobbs, J. R. (1993) FASTUS: A system for extracting information from text. *Proceedings of the DARPA workshop on Human Language Technology*, pp. 133–137. Princeton, NJ.

Humphreys, K., Gaizauskas, R., Azzam, S., Huyck, C., Mitchell, B. and Cunningham, H. (1998) University of Sheffield: Description of the LaSIE-II System as used for MUC-7. *Proceedings of the Seventh Message Understanding Conference*, pp. 84–89.

Kornai, A. and Sundheim, B. (editors) (2003) *Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References*. Edmonton, Alberta, Canada.

Krupka, G. R. and Hausman, K. (1998) IsoQuest Inc.: Description of the NetOwl (TM) Extractor System as Used for MUC-7. *Proceedings of MUC-7*.

Li, W. and Srihari, R. (2000) Flexible Information Extraction Learning Algorithm. Phase 1 Final Technical Report AFRL-IF-RS-TR-2000-26. Rome, NY: Air Force Research Laboratory.

Li, W. and Srihari, R. K. (2003) Intermediate-Level Event Extraction for Temporal and Spatial Analysis and Visualization. Phase 1 Final Technical Report AFRL-IF-RS-TR-2002-245. Rome, NY: Air Force Research Laboratory, Information Directorate.

Li, H., Srihari, R., Niu, C. and Li, W. (2002) Location normalization for information extraction. *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002)*. Taipei, Taiwan.

Li, W., Srihari, R., Niu, C. and Li, X. (2003a) Entity profile extraction from large corpora. *Proceedings of Pacific Association for Computational Linguistics 2003 (PACLING '03)*, pp. 295–304. Halifax, Nova Scotia, Canada.

Li, W., Zhang, X., Niu, C., Jiang, Y. and Srihari, R. (2003) An expert lexicon approach to identifying English phrasal verbs. *Proceedings of the Association for Computational Linguistics (ACL 2003)*, pp. 513–520. Sapporo, Japan.

Miller, S., Michael, C., Fox, H., Ramshaw, L., Schwartz, R. and Stone, R. (1998) Algorithms that learn to extract information; BBN: Description of the SIFT System as Used for MUC-7. *Proceedings of MUC-7*.

Mönnich, U., Morawietz, F. and Kepser, S. (2001) A regular query for context-sensitive relations. In: P. B. S. Bird and M. Liberman (editors), *IRCS Workshop Linguistic Databases 2001*, pp. 187–195.

Niu, C., Li, W., Ding, J. and Srihari, R. K. (2003a) A bootstrapping approach to named entity classification using successive learners. *Proceedings of the Association for Computational Linguistics (ACL)*, pp. 335–342.

Niu, C., Li, W., Srihari, R. K. and Crist, L. (2003b) Bootstrapping a Hidden Markov Model for relationship extraction using multi-level contexts. *Proceedings of Pacific Association for Computational Linguistics 2003 (PACLING '03)*, pp. 305–314.

Niu, C., Li, W., Ding, J. and Srihari, R. K. (2004) Orthographic case restoration using supervised learning without manual annotation. *International Journal of Artificial Intelligence Tools* 141–156.

Pustejovsky, J., Castaño, J., Ingria, R., Sauri, R., Gaizauskas, R. and Setzer, A. (2003) TimeML: Robust specification of event and temporal expressions in text. *New Directions in Question Answering 2003*, pp. 28–34.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press

Riloff, E. (1996) Automatically generating extraction patterns from untagged text. *Proceedings of AAAI-96*, pp. 1044–1049.

Riloff, E. (2003) From manual knowledge engineering to bootstrapping: Progress in information extraction and NLP. *ICCBR 2003*, p. 4.

Riloff, E. and Jones, R. (1999) Learning dictionaries for information extraction by multi-level boot-strapping. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pp. 1044–1049. AAI Press/MIT Press.

Roche, E. and Schabes, Y. (1997) *Finite-State Language Processing*. MIT Press.

Silberztein, M. (1999) INTEX: a Finite State Transducer toolbox. *Theoretical Computer Science* **231**(1). Elsevier Science.

Srihari, R. and Li, W. (2000) A question answering system supported by information extraction. *Proceedings of ANLP 2000*, pp. 166–172. Seattle, WA.

Srihari, R., Niu, C. and Li, W. (2000) A hybrid approach for named entity and sub-type tagging. *Proceedings of ANLP 2000*, pp. 247–254. Seattle, WA.

Srihari, R. K., Li, W., Niu, C. and Cornell, T. (2003) InfoXtract: A customizable intermediate level information extraction engine. *Proceedings of NAACL '03 Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*, pp. 52–59. Edmonton, Alberta, Canada.

https://doi.org/10.1017/S1351324906004116 Published online by Cambridge University Press