1 Ontology-Based Information Extraction (OBIE)

1.1 Basic Definitions

Here, we provide the definitions for an ontology and an OBIE system.

**Definition 1 Ontology**: An ontology $O$ is a quintuple, $O = (C, P, I, V, A)$ where $C$, $P$, $I$, $V$, and $A$ are the sets of classes, properties, individuals, property values and other axioms (such as constraints) respectively. These sets are defined as follows.

- $C = \{ c \mid c \text{ is a unary predicate} \}$
- $P = \{ p \mid p \text{ is a binary predicate} \}$
- $I = \{ c(i) \mid c \in C \land i \text{ is a ground term} \}$
- $V = \{ p(x, y) \mid p \in P \land x \text{ and } y \text{ are ground terms} \land (\exists c \in C \land c(x)) \}$
- $A = \{ a \mid a \text{ is an assertion} \}$

**Definition 2 Ontology-Based Information Extraction (OBIE) System**: An OBIE system is a system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and presents the output using ontologies.

Note that this definition encompasses single-ontology as well as multiple-ontology OBIE systems.

1.2 Z Specification

We can provide a definition for an OBIE systems using the Z notation, which is a widely used formal specification language. It provides a specification for an ontology and uses this specification in presenting a specification for an OBIE system. This Z specification for an OBIE system as well as the Z specifications provided in subsequent sections of this chapter are non-deterministic in that they do not specify what extractions are made by the system. However, these specifications capture the essential functionality of the OBIE systems.
1.2.1 Single-Ontology OBIE Systems

We begin by providing a specification for an OBIE system that uses only one ontology. This is done by providing formal specifications for the two operations of such a system named Populate and Construct.

\[
\]

\[
\begin{array}{|c|}
\hline
\text{Corpus} \\
\text{documents} : \mathbb{P} \text{Document} \\
\#\text{documents} > 1 \\
\hline
\end{array}
\]

\[
\text{Response} \triangleq \text{success | empty\_ontology}
\]

\[
\begin{array}{|c|}
\hline
\text{Ontology} \\
\text{classes} : \mathbb{P} \text{UnaryPredicate} \\
\text{properties} : \mathbb{P} \text{BinaryPredicate} \\
\text{individuals} : \mathbb{P} \text{AssertionsOnUnaryPredicate} \\
\text{values} : \mathbb{P} \text{ AssertionsOnBinaryPredicate} \\
\text{axioms} : \mathbb{P} \text{Assertion} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{InitOntology} \\
\text{classes}' \\
\text{properties}' \\
\text{individuals}' \\
\text{values}' \\
\text{axioms}' \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{InitCorpus} \\
\text{documents}' \\
\#\text{documents} > 1 \\
\hline
\end{array}
\]
PopulateOk
\[\begin{align*}
\Delta & \text{Ontology} \\
\Xi & \text{Corpus} \\
r! & : \text{Response} \\
i! & : \mathbb{P} \text{AssertionsOnUnaryPredicate} \\
v! & : \mathbb{P} \text{AssertionsOnBinaryPredicate} \\
\end{align*}\]
$(\#\text{classes} > 0) \lor (\#\text{properties} > 0)$
$\text{classes}' = \text{classes}$
$\text{properties}' = \text{properties}$
$\text{individuals}' = \text{individuals} \cup i!$
$\text{values}' = \text{values} \cup v!$
$\text{axioms}' = \text{axioms}$
$r! = \text{success}$

PopulateEmpty
\[\begin{align*}
\Xi & \text{Ontology} \\
\Xi & \text{Corpus} \\
r! & : \text{Response} \\
\end{align*}\]
$(\#\text{classes} = 0) \land (\#\text{properties} = 0)$
$r! = \text{empty_ontology}$

Populate $\triangleq$ PopulateOk $\lor$ PopulateEmpty

Construct
\[\begin{align*}
\Delta & \text{Ontology} \\
\Xi & \text{Corpus} \\
r! & : \text{Response} \\
c! & : \mathbb{P} \text{UnaryPredicate} \\
p! & : \mathbb{P} \text{BinaryPredicate} \\
a! & : \mathbb{P} \text{Assertion} \\
\end{align*}\]
$\text{classes}' = \text{classes} \cup c!$
$\text{properties}' = \text{properties} \cup p!$
$\text{individuals}' = \text{individuals}$
$\text{values}' = \text{values}$
$\text{axioms}' = \text{axioms} \cup a!$
$r! = \text{success}$
$(\#\text{classes}' > \#\text{classes}) \lor$
$(\#\text{properties}' > \#\text{properties})$
It should be noted that “ontology population” systems only have the Populate operation while “ontology construction” systems only have the Construct operation.

1.2.2 Multiple-Ontology OBIE Systems

In a multiple-ontology system, we have a set of ontologies. However, the same corpus will be used. These can be formally represented as follows.

We have a set of \( n \) ontologies \( O_1, O_2, ..., O_n \). Each ontology \( O_i (1 \leq i \leq n) \) is defined as follows.

\[
\text{Ontology}_i = \begin{cases} 
\text{classes}_i : \mathbb{P} \text{ UnaryPredicate} \\
\text{properties}_i : \mathbb{P} \text{ BinaryPredicate} \\
\text{individuals}_i : \mathbb{P} \text{ AssertionsOnUnaryPredicate} \\
\text{values}_i : \mathbb{P} \text{ AssertionsOnBinaryPredicate} \\
\text{axioms}_i : \mathbb{P} \text{ Assertion} 
\end{cases}
\]

There is only one Corpus schema as defined in section 1.2.1.

Each ontology schema and the corpus schema is initialized as in section 1.2.1. For each ontology \( O_i \) operations - Populate\(_i\), PopulateOk\(_i\), PopulateEmpty\(_i\), and Construct\(_i\) are defined following the specifications of these operations in section 1.2.1.

According to this Z specification, a multiple-ontology OBIE system can simply be a collection of single-ontology OBIE systems with no collaboration between different systems. However, in practice, mappings between concepts of different ontologies are used to facilitate the reuse of components and improve information extraction through this process. In addition, when ontologies specialize on sub-domains, performance can be improved by processing individuals documents using the ontology most suitable for them. Multiple-ontology OBIE systems that operate using these mechanisms meet the Z specification given above.

2 Specification of an OBIE System under the OBCIE Approach

We have designed a comprehensive component-based approach for information extraction named OBCIE (Ontology-Based Components for Information Extraction). In this section, we present formal specifications for an OBIE system that is implemented under this OBCIE approach.
2.1 Z Specification

Here, we refine the Z specification for a generic OBIE system given in section 1.2 into a specification of a system that follows the OBCIE (Ontology-Based Components for Information Extraction). This shows that the OBCIE approach results in functional OBIE systems.

2.1.1 Single-Ontology OBIE System

In a single-ontology OBIE system developed under the OBCIE approach, the Construct operation is unchanged while the PopulateOk operation is refined into a pipe of four separate operations as follows.

\[ \text{PopulateOk} \triangleq \text{Preprocess} >> \text{Extract} >> \text{Aggregate} \]

These operations can be specified as follows.

\[
\begin{array}{c}
\text{Preprocess} \\
\exists \text{Ontology} \\
\exists \text{Corpus} \\
\text{intermediate}! : \mathbb{P} \text{PreprocessedDocument}
\end{array}
\]

\[
(#\text{classes} > 0) \lor (#\text{properties} > 0) \\
\#\text{intermediate}! = \#\text{documents}
\]

\[
\begin{array}{c}
\text{Extract} \\
\exists \text{Ontology} \\
\exists \text{Corpus} \\
\text{intermediate}? : \mathbb{P} \text{PreprocessedDocument} \\
\text{ex}_\text{inds}! : \mathbb{P} \text{AssertionsOnUnaryPredicate} \\
\text{ex}_\text{values}! : \mathbb{P} \text{AssertionsOnBinaryPredicate}
\end{array}
\]

\[
(#\text{classes} > 0) \lor (#\text{properties} > 0) \\
\#\text{intermediate}? = \#\text{documents}
\]

The Aggregate operation is defined as follows.
Under the OBCIE approach, each operation in the refined schema for the
*Populate* operation is performed by a single component. As such there will be
components for preprocessing documents (*preprocessor* components), for mak-
ing extractions with respect to specific classes or properties (*information ex-
tractors*) and for combining the results produced by formatters and producing
an ontology as the output (*aggregator* components).

### 2.1.2 Multiple-Ontology OBIE Systems

In section 1.2.2, the Z specification for a generic single-ontology OBIE system
was extended to a multiple-ontology OBIE system that uses *n* ontologies by
having *n* number of ontology schemas and *n* set of operations for each operation
in the specification for a single-ontology OBIE system (*Populate, PopulateOk,
PopulateEmpty* and *Construct*). The same approach is applied here to extend
the Z specification for a single-ontology OBIE system that follows the OBCIE
approach into a specification of a system that uses multiple ontologies under
the OBCIE approach.

For each ontology *O* _i_ (1 ≤ *i* ≤ *n*), *PopulateOk* _i_ operation would be refined into
a set of operations as follows.

*PopulateOk* _i_ ≜ *Preprocess* _i_ >> *Extract* _i_ >> *Aggregate* _i_

The Z specifications of *Preprocess* _i_ , *Extract* _i_ and *Aggregate* _i_ follow the specifications given in section 2.1.1.

As in the case of the Z specification of a generic multiple-ontology OBIE
system, the Z specification of such a system that follows the OBCIE approach
does not specify how mappings between concepts are used to facilitate the reuse
of information extractors. As far as the Z specification is concerned, compo-
ments that make extractions with respect to different ontologies can be totally independent. However, systems that use mappings in this manner are consistent with the specification.