5 System Framework

5.1 System Architecture

In the last section system requirements are investigated according to case studies. As discussed above, some components are necessary for constructing a practice oriented information extraction system. In this section we introduce the total system architecture of our Information Extraction system CapturePlus. Particular components of the system and their functions are described briefly. The design of the system is practice oriented. That is, system is developed for practical problem solutions, which are analysed in the case studies (see Section 4). The system architecture is showed in Figure 13.

We partition the system into four components: Pre-processor, NE-Agents, TE-Agents and Output Generator.

![Figure 13: System Architecture of CapturePlus](image-url)
5.1.1 Pre-processing

In CapturPlus, the first step is pre-processing. The main goal of pre-processing is to enable a consistent and formatted input for later name extracting and template filling. In the phase pre-processing documents are segmented and indexed.

For normal documents, pre-processing indexes the documents at first. In this step an input document is decomposed at the word level. Some properties of the words (such as frequency of occurrence, similar words etc.) can be investigated. Such properties can be used for later processing to enhance the extracting performance.

An optional process is to segment documents. Documents can be segmented in different levels such as clauses (capitals), paragraphs and sentences. Later extracting can then be executed only within defined segments, so that precision and recall can be improved.

For HTML documents, documents are converted at first to well-formed XML format. After converting, documents have a well-formed hierarchical tree structure. With help of standard XML techniques (such as XSL-Patterns or XQL) contents can be located exactly. Therefore, pre-processing is necessary for later text zone extracting.

Actually, the pre-processing for normal document is not an imperative step. Later processing for normal documents need in some special cases the indexed or segmented documents for a quick processing. But the main tasks of Information Extraction are performed not in the phase pre-processing. For HTML-documents, however, the pre-processing must be called so that later processing can get the input documents with XML-format.

The pre-processed documents are fed into the following processor: NE-Agents.

5.1.2 NE-Agent

After pre-processing, the system does feature extracting, which is defined in MUC as name extracting (NE, Cunningham 1999). The component is named here as NE-Agent. The main task of NE-Agent is to extract the contents from the pre-processed documents using structured information and classical IE methods. Feature extracting is the most important step in the total IE system. As discussed in Section 4, for different document types, an IE system has to use special technique to solve different problems. Therefore, the solutions to
the part problems become variegated. Actually, to carry out feature extracting from different input document types and different application domains, all corresponding solutions have to be integrated in the NE-Agent. Hence, NE-Agent consists of different NE-components to work around possible coming problems in the practice.

Such components can either work independently according to the input document type or cooperate with each other, so that complicated tasks can be decomposed and solved stepwise. In the first case, components have a horizontal relationship. In the last case, on the other hand, vertical relationship between components is required too. The construction of NE-Agent is illustrated in Figure 14.

![Figure 14: Construction of NE-Agent](image)

There are three steps in feature extracting: text zone recognising, contents extracting and result merging. As described above, the task of text zone extracting is to detect text boundaries of underlying contents. Such text zones restrict extracting area for underlying contents. For different document types (free document vs. semi-structured or structured documents, normal
document vs. HTML) corresponding methods with respect to their document properties are to be developed for extracting their text zones. In Section 6 various detecting methods will be introduced in detail.

Components performing contents extracting are named contents extractors. Contents extractors use IE-techniques to extract exactly the underlying contents. Because contents have various syntactic and semantic properties, a set of extracting methods are implemented for defined content types. Each contents extractor is implemented for one or some defined problems. CapturePlus integrates not only typical extracting methods such as syntactic rules, simple NLP techniques, but also sophisticated techniques (e.g. hierarchical concept extracting etc.) In order to distribute tasks to corresponding contents extractors a NE-Scheduler is therefore necessary. NE-Scheduler distinguishes extracting tasks and then distributes them to suitable contents extractors. Such recognition and distribution can be hard coded, or automatically processed by NE-Scheduler. In the later case, the advanced NE-Scheduler can be considered as a dynamic, adaptive and intelligent NE-Scheduler.

The outputs of contents extractors are normally result lists, which consist of useful information such as position and length of contents, relevance of contents etc. Because each content extractor can work independently, the outputs of such extractors can contain result with overlapping, redundant or conflicts. Therefore, results must be merged and cleaned up.

Section 7 describes components in NE-Agent and their underlying techniques in detail. Also various learning techniques to create extracting rules automatically are described in that section.

5.1.3 TE-Agent

After feature extracting, TE-Agent (to do Template Filling, Cunningham 1999) receives input as a list of instances for all elements in a template. The main task of TE-Agent is to fill the extracted instances into the pre-defined template. Most recent information extraction systems do not have TE-component, they concentrate mainly on name extracting. The template filling is processed either for very primitive simple template, or as post-processing manually. There are at least three reason to show why a extra TE-Agent is needed.

First, we know the spatial relationship between template elements plays an important role in IE (see also Section 4). Such relationship information is sometimes very especially useful in template filling, if more instances of a defined template element are extracted (that means, that only one of instance is the right one, the other have to be dropped). Information about spatial relationship between contents can help the system to select correct instance
of extracted content. For instance, in a structured document CIA-Factbook (description see Section 4.1), we can find that content “Economic Description” occurs always after content “Geographic Description”. After the feature extracting, however, we get two result instances for content “Geographic Description”. If one content candidate for element “Geographic Description” is located before “Economic Description” and the other one after “Economic Description”, we can then easily drop the second instance of “Geographic Description” and set the first one in the template. This is only a simple example. Normally, dependence between elements is very complicated so that sophisticated model for representing such relationship has to be investigated. Hence, a stand-alone TE-Agent must be developed, so that spatial relationship can be applied for template filling.

Second, many documents in practical domains have multi-slots property. That is, such documents consist of more than one template instance. Hence, for these documents, template filling has to enable an instance duplicating, in order to fill all instances in templates. A typical example is product catalogues which consists of many product descriptions in the same product family. The template is at first defined for all possible product descriptions in the product family. At the beginning of extracting, only one empty template is given because the number of possible product descriptions is not clear. Nevertheless, the result after template filling must consist of all extracted and filled template instance for all product description. That means, that the template instance is duplicated. In the case of multi-slot documents, methods for detecting boundary of template instance and automatic duplicating a template on demand are to be implemented. Such methods do not belong to feature extracting and are therefore extra implemented in TE-Agent.

Furthermore, template elements can also have undefined cardinality (see case study 1, 3, Section 4). For example, in the CIA-Factbook, the neighbour countries vary depending on the actual country. For these complicated templates, both description of template and filling task become no more trivial. More detailed and systematic template description and also its corresponding filling techniques are to be developed.

Last but not least, automatically generated rules and patterns are normally learned through inductive learning algorithms. The learning algorithms typically require each rule to have a high precision and recall performance so only strong rules are generated. This leads to the general problem that the set of automatically generated rules and patterns for contents extracting filter too strongly – i.e. by enforcing early-on a high precision rate, the overall recall of the knowledge extraction system is impaired. Therefore, at least for the class of semi-structured or structured documents of interest, a more robust approach to information extraction is two-stage, with relatively crude NE Agent achieving high recall rates followed by a TE-Agent that restores the overall precision. One can argue that individual NE-Agent is unlikely in most cases to
receive enough input information from a document to be able to make a 
reliable decision (at least without resorting to a more profound Natural 
Language Processing analysis of the raw text) – the characteristics of a 
content instance are a combination of local syntactic clues (that can be 
captured easily in the NE-Agent) combined with the broader dependency that 
this content has with other contents within the document (as captured by the 
TE-Agent). Hence, the introduction of TE-Agent provides a stepwise 
refinement in information extraction: achieve high recall in the earlier phase 
(name extracting) and improve precision in the later phase (template filling).

Due to the discussion above, a TE-Agent has to be implemented as a part of 
a complete IE system. The main task in TE-Agent is to model the order 
relationship of elements in a template and guarantees a correct filling of 
extracted contents in template. Detailed discussion and description about 
implementation of TE-Agent are give in Section 8.

5.2 Search Profiles

Beside IE processing components there is another important component, 
which is used to store extracting rules and patterns and to describe the 
template. We name this component “Search Profile”. In CapturePlus, two 
types of Search Profile are defined, namely NE-Search Profile and TE-Search 
Profile. The first type of Search Profile defines complicated rules and pattern 
for NE-extracting. The second one describes the construction of template and 
the spatial relationship between template elements. In the Search Profile in 
CapturePlus, NE-Profile and TE-Profile are stored separately. The knowledge 
engineer can configure the empty template and assign corresponding 
extracting rules for each element in template. In this section we will discuss at 
first the advantages of Search Profile. Afterwards the idea to construct Search 
Profile is introduced.

5.2.1 Advantages of Search Profile

The initial notion to use Search Profile is to represent advanced extracting 
rules. Advantages of using Search Profile are many sided. In following we will 
discuss why Search Profile is needed for a complete IE system.

1. Patterns and rules are developed for various IE components. Formats 
of such patterns and rules are therefore total different. NE-Profile is 
therefore developed, so that it is able to support a consistent 
description for all patterns and rules.
2. Real IE tasks are hard to solve. In most cases one IE technique is not able to extract all contents correctly. A combination of various IE techniques is therefore necessary. Combination can be designed as sequential or parallel. In the first form, IE techniques are driven one after the other. Example of such combination is text zone detecting and the following feature extracting. The later form, on the other hand, combines various IE techniques with logical operations. For instance, we can define a complex extracting rule, which consists of some syntactic rules and NLP patterns. The logical operation for such rules and patterns is OR operation. That is, all contents, whatever is extracted by which rules or patterns, are considered as result instances. Analogy an AND-operation is also possible, i.e., contents are only accepted if they are result of all relevant NE-Agents. With help of Search Profile, we can define more complex combination of extracting rules rather than that only with single IE techniques.

3. Search Profile brings high reusability and flexibility. Defined Search Profiles can be directly used in other domain without any changing requirement. For instance, once we have a Search Profile for extracting date, we can use this Search Profile in any other domains directly. Defined Search Profiles can be exported or imported easily. This advantage provides a good flexibility and portability of system.

4. A generic form of Search Profile can enable future integration of new IE components into system easily. If each IE component has same format for extracting rules, the system can be configured as an open system, in order to use new IE techniques without many restructuring cost. Furthermore, controlling and scheduling total process of IE is also to be investigated, which can be defined with help of Search Profile.

As discussed above, using Search Profile can provide an advanced method for implementation of a complete IE system. Defined Search Profile can also be reused in other domains and therefore provides a good reusability and portability. Furthermore, a consistent description of extracting rules and patterns can make the integration of new IE techniques into system easily. Using Search Profile is one of major differences between CapturePlus and other IE systems.

5.2.2 Relationship between template and Search Profile

We distinguish Search Profile between NE-Profile and TE-Profile. The main difference between two profiles is that one is template dependence, while the other is not. NE-Profile defines extracting rules and patterns for feature extracting. The rules and patterns are template independent. That is, one defined NE-Profile can be applied for various domains and templates. In
contrast, a TE-Profile defines at first construction of a template and then describes spatial relationship of template elements. Such description is only valid for one defined template and therefore template dependent. Hence, NE-Profiles can be stored separately to defined template model and be used for other domains, while TE-Profile must be saved with template together. The relationship between Search Profiles and template model is showed in Figure 15. In Figure 15 we can see also, that a NE-Profile can belong to multiple template elements.

![Figure 15: Relationship between model and Search Profile](image)

In the configuration phase, each template element is assigned with a corresponding NE-Profile. For the total template, only one TE-Profile is assigned, which describes the construction of the template and the sequence relationships of template elements. In the extracting phase, NE-Agent goes through at first all elements in template, loads corresponding rules assigned to the element and extracts contents for each element. This process is repeated for each element one after the other. After extracted all contents in template, NE-Agent mergers all results in one list and set it as input for TE-Agent. Then, TE-Agent analyses extracted list according to the information provided by TE-Profile and performs then template filling.

5.2.3 Construction of NE-Search Profile

A NE-Profiles set consists of various defined NE-Profiles. Two types of NE-Profile are defined: Basic Profile and Complex Profile. A Complex Profile consists of a couple of Basic Profiles. A Basic profile consists of again a
couple of IE-techniques. In Basic Profile, various IE-techniques are combined with logical operation (such as OR and AND). Each IE-technique provides an extracting method and extracts the features exactly. In general an IE-technique has two parts: rules for Text Zone recognising and rules for underlying contents extracting. Text zone detecting, in the view of Search Profile, is implemented as constraints. Figure 16 shows the construction of NE-Profiles.

Figure 16: Construction of NE-Profile

To extract underlying content, in the design phase, each element in template is assigned to a Complex Profile, which is responsible for NE-task of underlying template element.

In the extracting phase, the NE-extracting for one complex profile is performed as following: at first, Complex Profile is load from NE-Profile base. From the Complex Profile, one Basic Profile is selected. The extracting of a Basic Profile is processed again in four steps:

- Step 1, within the Basic Profile, system loads each IE-technique.
- Step 2, before the underlying IE technique is called, text zones are extracted according to Text Zones constraints.
• Step 3, underlying IE techniques (such as syntactic rules, NLP modules etc.) are executed. Contents are extracted within defined text zones in step 2. Intermediate data of each IE-techniques, such as position and length information of the extracted instances, is stored by system. Steps 1 to 3 are operated until all IE-techniques in the Basic Profile are executed.

Finally, in step four, all results are combined with logical operator such as AND or OR. Normally, a Complex Profile contains more than one Basic Profile. The total NE extracting of one template element is repeated until all Basic Profiles are processed. Figure 17 shows the procedure for extracting a Complex Profile.

```
Load Complex Profile
   While not all basic profiles are processed
      Load one basic profile from complex profile
      While not all IE-techniques are executed
         Load one IE-technique from basic profile
         Extract Text Zones
         Execute underlying IE-technique within defined Text Zones
      End while
   End while
End Extracting
```

Figure 17: Procedure for extracting a complex profile

Having logical construction of NE-Profile, another requirement to Search Profile is to define a generic description, in order to provide an open environment to allow easy integration of new NE components in the future. Based on the structured construction of Complex Profile, an ideal language to describe the Search Profile is the Extensible Markup Language (XML). XML is initially designed as the universal format for structured documents and data on the Web. XML provides at first a possibility for description of structured information, which is suitable for the profile description.

Secondly, with help of XML-Schema new elements can be defined and then used in XML documents easily. This advantage is very useful for implementation of an open system and provides therefore good portability. A sample DTD of NE profile is showed in Figure 18. In this sample description, a Complex Profile consists of two elements: an element for profile name and a list of Basic Profiles. A Basic Profile contains a name descriptor, a definition to extracting space, which describes whether extracting is processed for all loaded documents or only for current one, a definition to logical operation (such as AND, OR, etc.) and a list of various NE-techniques. Each NE-
technique consists of two parts again: one is the constraints description, which is designed for detecting of text zones, the other is the description for underlying extracting rules, whose contents are dependent on the extracting method.

Figure 18: Sample DTD for description NE Profile with XML

5.2.4 Construction of TE-Search Profile

TE-Profile defines construction of empty template and consists of statistical sequence relationship for template elements. Such sequence information can be hard coded for structured documents or generated with statistical analysing of spatial relationship from given examples. A TE-Profile can be loaded only with template together. As same as NE-Profile, the construction of template is defined with XML-Schema. With help of XML-Schemas, complicated template model, which contains optional elements or elements with arbitrary cardinality, can be well defined. The spatial relationship is also stored in the XML-Schema based template definition, namely with separate field. A sample description of a CIA-Factbook is shown in Figure 19.
In the extracting phase, algorithms are used to perform correct template filling and duplicating when documents have multi-slots. More detailed description about the working of TE-Profile see Section 8.

5.2.5 Related Work

A major advantage of using Search Profiles is the portability. Once a NE-Search Profile is defined, it can be reused in another domain easily. Furthermore, as discussed above, separating the NE-Profile and TE-Profile is necessary. Basically, NE-Profile is focused on extracting single content instance, while TE-Profile helps to fill the extracted instance into a defined template. Knowledge engineers can develop the two different profiles independently. Some common NE-Profiles can be used in many document domains without modification (such as rules for extracting date, currency, number etc.) Hence, knowledge engineers can define a new TE-Profile and assign the pre-defined NE-Profile to each element. The flexible portability of the approach by using search profiles saves development time for creating a new template.
Unfortunately, some IE-systems, including most of MUC systems (Grishman 1996), do not have Search Profiles. Other systems have primitive Search Profile, which do not distinguish NE- and TE-profile. One reason why such systems did not develop complete Search Profiles but only primitive rules is, that some systems are initially designed without portability in aimed. For instance, some MUC systems are evaluated only with the data set defined by MUC. Such systems developed searching rules and hard-coded template for the MUC data set specifically. Hence, although such systems reported very high precision and recall results, they can not be used in the real life applications.

Some systems support portability. An example is Pinocchio (Ciravegna 1999). Pinocchio is a toolkit for developing and running Information Extraction applications (Ciravegna 1999). The system can be ported to new applications and domains in a few weeks by modifying declarative resources. The development configuration is specifically designed for helping in this process. Two types of resources are needed: static resources (the lexicon and the knowledge base) and dynamic resources (cascades of rules). The knowledge base defines the ontology for the application domain. For each word relevant for the domain the lexicon provides the mapping with the ontology and a description of the syntactic and semantic features (e.g. sub-categorization frame). The same formalism (and even the same primitives) is used for all the dynamic resources. This construction is an important advantage, allowing new applications to be developed by a single person. At the end of the application development process, resources are compiled into source code to be used by the delivery configuration. The development environment also allows to port the system to new languages. As mentioned, only declarative resources must be modified when changing domain or language.

In Pinocchio the Search Profiles are defined as two types of resources. Knowledge developer can edit and modify the resources with help of GUI tools. A shortcoming of porting methods in Pinocchio is that resources have to be compiled into the source code again (Ciravegna 1999), which prevents the portability of the system.

5.3 Learning Instruction

Hand-coding patterns and rules is time consuming and error-prone. The system provides various learning algorithms to semi-automated the construction of an IE System.

As mentioned in Section 5.1, to fulfil the different extracting requirements, a set of techniques of Information Extraction are used in CapturePlus. Some techniques are based on machine learning algorithms. Learning techniques in CapturePlus include: generation of XSL-patterns (wrapper induction, see
Section 7.1.2), inductive token-based rules (see Section 8), statistical keywords extraction (see Section 7.2.1) and statistical derivation of spatial constraints (see Section 9).

Generally we use the supervised Machine Learning (learning with given examples) technique: in the training phase, the input is some filled template examples, which are typically filled by a non-skilled manual worker. We use a GUI tool to create the examples interactively. With Drag & Drop operations user can fill the examples into predefined empty templates. The examples are defined by begin and end positions in the original document.

Training NE-Agents proceeds in a stepwise manner for each separate element/slot defined in a template. For each element the system gathers all corresponding examples and generates patterns and rules for this element. For the TE-Agent, the system needs to gather at first all examples and makes statistical analysing to get the relationship under the template elements.

After the training the results are saved in Profiles. For each trained template element, a Complex NE-Profile is created. All NE-Profiles are save in one NE-Profile base. Such NE-Profile base is actually independent on the given template example. The trained TE-Profile, on the other hand, is save together with the given template. Hence, NE-Profiles and TE-Profiles are saved separately. In the extracting phase the system can load the trained profile, extract values and fill them in template.

It is possible that NE-rules in different elements in one template are learned with different machine learning techniques. To enable that all elements in template are learned at one time in the training phase, we use a GUI dialog to let user configure the training. Figure 20 shows the GUI interface for learning.
Figure 20: GUI interface for learning

This section gives an overview and basic techniques of the whole information extraction system CapturePlus. Actually, the system has three main characteristics:

- Component based stepwise processing (clear separation between NE and TE-tasks)
- Search Profiles for complicated IE-processing and better portability
- Integrated learning environment for various machine-learning algorithms.

These three main characteristics enable CapturePlus as practical oriented complete IE-system rather than a labour system with single algorithm.
From the next section, we will describe each components in details. New approaches and algorithms are introduced. Deep analyses and discussions are made after evaluation with well-known data set.