4 Case-Studies for practical Information Extraction

This section describes case studies of Information Extraction in the real life applications. The scenarios are introduced for some real domains such as Product Catalogue, Travel Information, and Weather Forecast etc. Various properties of domains are analysed. These Case Studies point out practical requirements to an Information Extraction System and provide a guideline to design our Information Extraction system. From these case studies we can collect requirements for constructing an information extraction system for practical applications.

4.1 Case Study 1: Extracting Information from Structured Normal Documents

Extracting information from structured documents is one of the simplest tasks of IE. Example of such type of document is CIA factbook¹, which provides information about countries and political organisations. Figure 7 shows a simplified example of country description of CIA factbook.

![Figure 7: Example of CIA Factbook](http://www.cia.gov)

¹ CIA Factbook is downloadable from http://www.cia.gov
Information Extraction

The pages are structured only as formatted text, i.e. values with standard field names. Furthermore, all descriptions (such as geographical information, population, GPD value etc.) are in a strict order. That is, the sequence of descriptions is always the same. Normally, to extraction such information, the first step is to recognise text zones. In the structured documents, text zones can be easily identified. In this example, each text zone begins with specialised field names (such as “Location:”, “water:” etc.) and ends with new lines. The underlying contents are just the whole information within text zones. As soon as text zones are detected, NE (name extraction) is finished.

For structured documents, the property of strict order of contents plays an important role in IE. Information about the sequence (such as begin and end tokens, absolute or relative distance to previous content or strict position information) can help IE system to process high performance IE for structured documents. Hence, for structured documents, IE components for recognising text zones and components with respect to order and position of elements are important.

Although name extraction for structured documents is not a hard task, template filling is sometimes not a trivial task, when structured documents have complicated structure. For instance, in the CIA factbook, one filed describes the boundary countries. The number of boundary countries is not defined. That is, for this template element, the cardinality can be defined from at least one to many. Furthermore, some template elements can be optional (such as “introduction” in the factbook). For this case, template filling is no more trivial. A more sophisticated template model has to be investigated, in order to describe the template and to enable advanced template filling (see also Section 10).

4.2 Case Study 2: Extracting Information from Structured HTML Documents

Web pages, which are generated automatically by robots or by SQL extracting from database, are normally structured documents. In many cases, useful information is represented in web sites with defined forms such as tables and lists. An example of such documents is CNN Weather report. A sample page is showed in Figure 8.

\[2\] CNN Weather report is located at http://www.cnn.com/WEATHER
As same as structured normal documents, structured HTML documents are constructed with pre-defined structure. For instance, the CNN weather reports is generated by routine automatically, the weather information (such as temperature, five days forecast etc.) is at first fetched from database and then filled in a pre-defined HTML template.

Similar as structured documents, it is important to detect text zone. Compared with normal documents, text zones are not defined by field names but bounded by HTML tag pairs. If we analyse the relationship between HTML-tags, we can see that tag pairs, which define text zones, are bounded by other tag pairs again. That is, the tags are structured hierarchically. For example, in the CNN weather pages, the content city name has a boundary defined by tag pair “<span>” and “</span>”, while the tag pair is in the area that defined by tag pair “<td>” and “</td>”. Such containment relationship can be bottom up investigated until the area as whole document (normally defined by “<html>” and “</html>”) is reached. Actually, HTML documents have implicit tree-style hierarchical structure: if we follow the tags from the root “<HTML>” to the underlying content, we can get a path of tags (such as “html/body/table/td/...”). For structured documents, such hierarchical relationship is strictly defined. That is, same content has always the same path of tags in all documents of one domain (but different contents can also have the same path of tags). From this observation, the task to locate wanted text zones becoming the task to detect the hierarchical relationship of HTML-tags.

In the case of CNN weather report, as soon as the text zones of interesting contents are recognised, explicit extracting of the underlying contents is no
more necessary, because the contents defined by text zones are just that to be extracted.

In the practice, it is not unusual that different contents have same hierarchical structure relationship. For instance, the contents of temperature forecast about coming five days have same hierarchical property (same path). Although with help of hierarchical information, we can distinguish the contents for today’s temperature and those for the coming days, we are not able to determine the difference under the five coming days. In this case, the strict sequence property can help to distinguish the contents. Therefore, for structured HTML documents, IE components for detecting hierarchical relationship and components with respect of sequence property have to be developed and combined.

4.3 Case Study 3: Semi-Structured Normal Documents

This gene of documents is one of the most interesting areas in practical IE. Product catalogues, Newsgroup announcement, monthly or daily financial reports, error reports via email etc. are all belong to this gene.

To analyse the features of semi-structured documents, we use the example of Job Advertisements. The documents are set of emails that are posted into the newsgroup by the companies offering job opportunity. Because each job description is written by individual offer, the formats of such descriptions are various. That it, such documents do not have a strict structure. However, all descriptions contain similar information such as job title, salary, city, required skills etc. Figure 9 is an example of such job description. Although there is no strict sequence order under contents in semi-structured documents, nevertheless, some spatial relationships between contents can be found. For instance, in the job description, the position of information about sending date is always after that of sender address. The loose spatial relationship between template elements is a key property of semi-structured documents.
From: spectrum@onramp.net
Newsgroups: austin.jobs
Subject: US-TX-Austin - Mainframe (Sr.) Programmer Analysts - COBOL, CICS, DB2 - $43K to $65K
Date: Sat, 23 Aug 97 09:55:30
Organization: OnRamp Technologies, Inc.; ISP
Message-ID: <NEWTNews.872348139.12359.consults@ws-n>

US-TX-Austin - Mainframe (Sr.) Programmer Analysts - COBOL, CICS, DB2 - $43K to $65K

Multiple positions are open now. Major corporations need experienced mainframe programmers with COBOL and CICS and VSAM or DB2 or IMS. You must be able to work independently, take the initiative, and be flexible. Excellent growth opportunities for those hard workers who excel. You must speak and write English fluently. You must be a US citizen, a Permanent Resident, and meet all job requirements.

Figure 9: Example of Job Advertisement

Extracting information from semi-structured documents is much more difficult than the same task for structured one.

To perform a high quality extracting result, the total task must be processed stepwise. In order to explain the extracting clearly, we use the job description given above as example.

In the first step, text zones have to be recognised. Different to structured documents, there are no significant boundaries (such as name fields) for the underlying contents. Normally, underlying contents can not be directly extracted after detecting of text zones because text zones can only be detected roughly. For instance, in the job descriptions, we can identify the text zones for sending date, sender address and message ID without problem, because such information is located in the header of emails. On the other hand, however, text zones for the descriptions about job title, salary, required skills of employers etc. can not be delimited clearly. Nevertheless, we know such information is written in the body of emails. That is, text zones of such information are defined as body of posted emails. That implies, that contents such as “salary”, “required skill” etc. appear always after contents “sender” and “sending date”.

In the second step, we have to extract the underlying contents within the detected text zones. Assumed the defined template consists of some elements such as Job Title, City, State, Salary, required computer language, Required Years of Experience, etc., we try to analyse each individual content.
Sending date, message ID, and sender address have significant syntactic features and therefore can be extracted by using syntactic rules easily. Job city, state, required computer languages, company names etc are written with natural grammar. To extract such information, NLP techniques can be used to process POS-tagging and then NLP pattern matching. Another possible solution to extract such contents is to use background knowledge. For instance, the background knowledge base can define all possible computer languages (such as “C”, “C++”, “Basic” etc.) and their variants with help of thesaurus. Using this method, extracting processing becomes a task of searching predefined string in documents. Such background knowledge can be created either manually or from given examples.

Besides the two types of contents discussed above, there is third type of contents, which has significant syntactic property though, but can not be distinguished from other contents if only such syntactic property is used. For instance, the content “Required Years of Experience” is written with digit number, which can be defined by regular expression easily. But, using regular expression alone all digit are extracted and therefore affects poor precision and recall. To ensure the extracted digits are just the wanted, we have to use context surrounding the underlying digits. For instance, extracting rule for content “salary” can consist of two parts: the pre-token indicates the currency (such as “$”) and the regular expression for underlying digits. Another way to extract such contents is to compensate the error after extracting: using the greedy sequence relationship under contents the right contents can be selected with their probability. That is, with help of property of order some false extracted digits can be eliminated because of their false positions.

Hence, after analysing a sample domain in the gene of semi-structured documents, following IE components are necessary: components for recognising text zones, components for extracting various contents and components for post-processing according to greedy sequence of spatial relationship under contents.

### 4.4 Case Study 4: Semi-Structured HTML Documents

In this case study, we want to extract information from online travel catalogue. The example data is from web page Last Minute (http://www.lastminute.com). The web pages provide world wide travelling opportunities at the last minute. A sample page of such offers is shown in Figure 10.

The steps of extracting contents from semi-structured HTML documents are as similar to those for semi-structured normal documents (see Section 4.3). As same as structured HTML documents, hierarchical tag structures are used here for extracting text zones too. However, in this case, the hierarchical relationship can be ambiguous, because the contents are not order in strict
sequence. For instance, in the last minute domain, the information about transportation is various: some travels are provided by plane, the other by train or by bus. Each transportation possibility describes then different detail information. For instance, for train, depart and arrive dates are given. For airline, additional information such as airline company is also listed. Furthermore, there can be more than one date possibility for travelling, as shown in the sample page.

For structured HTML document, some contents are already available after detecting of text zones. In this type of documents, however, information is only a part of detected text zone. Therefore, later processing has to be called after text zones detecting, in order to perform exact extracting. Due to the ambiguity of text zones information, different contents may have same extracting rules for text zones. That means, later processing must identify such ambiguous text zones for defined contents. If the underlying contents are short (such as carrier, Inbound or Outbound Date etc.), syntactic rules or background knowledge can distinguish difference of contents. For instance, syntactic rules can define the date format, while background knowledge can enumerate names of all possible carriers. If the contents are long (e.g. a short description about the destination information), syntactic rules are no longer suitable. NLP techniques, on the other hand, can be used here. Unlike normal documents, POS-tagging for the whole documents is not possible because the HTML documents consist of many non-grammatical tag strings. Hence, NLP can only be used within the detected text zones. Because building a NLP system is in generally expensive, we can also use significant keywords to distinguish the underlying description from other short texts. Similar as normal semi-structured documents, the greedy ordering property is also useful for selection of correct contents when ambiguity occurs.

Like structured documents, some contents can have non-constant cardinality. For instance, one hotel accommodation can contain different rooms with different facilities. Compared to structured documents, template filling is here much more difficult. Because the content instances do not appear always in the same order, assigning right instance to right template becomes very difficult. Such documents are the most common input for many application domains. Hence, sophisticated modelling and filling algorithms for semi-structured documents become a crucial requirement for practical information extraction system.
In this section we have to face some hard tasks of IE occurring in real applications.

One of problems is variegated description of information. For semi-structured and free documents, texts are normally written manually rather than generated by robots or SQL statements. This fact causes a variegated description of information. For instance, we want to extract distance information about hiking from documents. The possible descriptions about this information are given in Figure 11.

In the example we see the variation either in distance value (such as digit number “2”, number presented as word “three”, description “dozens of” etc.) or in distance unit (such as km, hours, days etc.). Simple syntactic rules or primitive POS-tagging are not able to solve this problem. Using basic IE techniques lots of rules have to be developed to respect all possible variations. Another possible solution to reduce the variation of descriptions is pre-processing documents and replacing variegated information with formatted string. For example, pre-processing can replace all unit descriptions (such as km, hours etc) with an unique token “$UNIT$”. After pre-processing, number of variants can be considerably reduced.
• The most popular hike on the continent is the 33km (20mi) Inca Trail, west of Cuzco.
• You can hike to the beach in three hours.
• The hike to the top of Mt Finkol takes 8 to 10 hours
• There are dozens of day hikes and multi-day hikes throughout the island.

**Figure 11: Example for distance information about hiking**

More complicated situation is the requirement of deep “understanding” of documents. This “understanding” is not in syntactic or semantic, but in pragmatic level. An example is given here for financial analyse reports. We consider at first the judgement information about current financial situation. In following some sentences are given as financial judgement for a company in Figure 12. If we range the financial situation in some stages such as “very good”, “good”, “bad”, etc., how can we determine the objective judgement from the documents? Actually, such goal is similar to the task ST (Scenario Template production) defined by MUC.

• The current financial situation is very encouraging.
• The loss for the nine months amounted to £4.7 million and we expect a loss for the year to reach some £6.5 million.
• Cash in the balance sheet stood at £5.2 million at end this month, which will be enough for continuing operations until the end of this year, when more financing will be needed.
• Investor brings cash to the company that will be sufficient for operations for the next two years, taking out what we considered the most important and immediate risk related to investing in the company, that of lack of financial resources to continue development.
• We expect the upcoming quarters to exhibit sequential revenue growth in the 40-50% range.

**Figure 12: Example for financial analyse reports**

How to extract the judgement of current financial situation is especially difficult. First, there are no clear objective judgements such as good, bad etc. in texts. Second, the descriptions about the financial situations vary largely. Simple IE techniques are not able to “extract” or “understand” judgements in the texts. This example reaches almost the hardest task in IE, namely not only information extracting, but also information understanding. There is in general no complete solution for such hard problems. However, for defined domain some special techniques can be used. For instance, in the domain financial reports, text zones can be extracted as sentences, which contain significant
keywords or keywords terms such as “cash”, “financial situation”, “revenue” etc. After detecting text zones, complicated NLP patterns can be used to identify the key information in the sentence. The key information is e.g. “encouraging”, “growth”, “loss”, “sufficient” etc. Ranking of the financial situation can be implemented as relevance weighting. Each key information is featured with a relevance score. The total score represents the degree of ranking. Actually, this method is a mix between IE and document classification.

### 4.6 A short conclusion

In this section we investigated some case studies for information extraction in real applications. The cases mentioned here are typical tasks occurring in the practical applications tackled by such companies as Whizbang Labs and SemanticEdge. With analysing of the case studies and investigating theirs solutions, system requirements are developed. These system requirements are the guideline for developing an information extraction system for practical applications.

The case studies showed that some components are necessary for information extraction from different document types. To construct an IE system, there are at least two components for the minimal requirement: one is for name extracting, which extracts single content instances, and the other is for template filling, which defines empty template and fills the extracted instances into the right place in the template. Additional components can be pre-processing component, which pre-processes input documents for easier extracting later.

Within Name Extracting (NE) component, tasks can be stepwise processed by sub-components again. As analysed above, two sub-components within NE-component must be developed: one for detecting text zones, the other for extracting underlying contents within the text zones. Optionally an additional result-merging component can be used, in order to post-process the content instances extracted by content extracting component.

Furthermore, learning component can provide learning environment and enable that some extracting rules and template relationships can be generated automatically. Output component can be configured for various output information and format.

In the next section a system framework will be introduced. Such a information extraction system is designed for the practical situations, which are discussed in this section.