A Streamlined Approach for Tabular Information Extraction

H. L. Wang
Department of Computer Science,
National Tsing Hua University
wyvern@cs.nthu.edu.tw

W. L. Hsu
Institute of Information Science,
Academia Sinica.
hsu@iis.sinica.edu.tw

Y. S. Chen
Institute of Information Science,
Academia Sinica.
ysc@iis.sinica.edu.tw

T. L. Lau
Institute of Information Science,
Academia Sinica.
tllau@iis.sinica.edu.tw

C. H. Tang
Department of Computer Science,
National Tsing Hua University
nicholas@rtlab.cs.nthu.edu.tw

H. M. Yeh
Institute of Information Science,
Academia Sinica.
hmyeh@robot.csie.ntu.edu.tw

W. K. Shih
Department of Computer Science,
National Tsing Hua University
wshih@cs.nthu.edu.tw

Abstract

In this paper, we propose a streamlined approach for extracting information from tables in HTML format. Our approach is based on a set of semantic templates associated with the knowledge representation maps. We apply an abstract model on the templates to support the extraction of tabular logical structure in different stages. Our abstract model includes category identification, reading path construction, and record collection. In this model, we use an abstract table to separate the logical structure from the physical layout. For each table, we try to extract the abstract table from its physical layout. Our approach has three stages. In the first stage, we use semantic tagging templates to identify all possible categories of the cells in the table. In the second stage, we construct the reading path by a cell linking algorithm. In the final stage, we perform the reverse traversal on the reading paths to extract and collect records from this table. We have implemented a prototype of tabular logical structure extraction system in MS-Windows environment. The prototype system provides an interface by which users can input a table in HTML format. Our system also has an interface to output the abstract table of the input table. We have done some experiments on several tables with distinct layout styles by using our system. Our experimental results show that our prototype system can extract the logical structure of these tables with high precision and recall rate.

1. Introduction

A table is a structured text format that is heavily used in several forms of media: such as newspapers, books, web pages, electronic text files and so on. A well-designed table presents important information with well legibility and accuracy. Readers can obtain information rapidly without making error from a well-designed table. However, extracting information from a table by computer only is not easy. Xinxin Wang and Derick Wood [1] investigated the issues of tabular composition including the abstract model, layout specification, editing and formatting. They developed a useful abstract model that can be applied to various kinds of tables with labels and entries. Shona Douglas et al. [2] also presented an approach to extract the logical structure from a table with plain text format. In their approach, they first recognize the canonical layout of a table. After that, they apply a series of transformations to the canonical layout. In [3], they presented a different approach. In their approach, they compute the cell cohesion factors and use the cohesion factors and domain templates

together to match and identify cell areas with the same domain. Then, they apply a greedy algorithm to tile the matched areas into a complete table.

In this paper, we propose a new approach for extracting information from table. Our approach is based on the extensive use of semantic templates. In addition, we also associate the semantic templates with the knowledge representation maps and the rules of cell linking. Since a general solution is quite unlikely, we will focus on tables which can be represented by the abstract model developed in [1] with limited layout characteristics.

In this paper, we apply a general model to support different stages of tabular information extraction including category identification, reading path construction, and record collection. Our general model uses an abstract table to separate the logical structure and the physical layout that consists of tabular topology. For each table that can be generated from an abstract table, our works is to extract the abstract table from a physical layout. For the cell in a table, we use semantic templates to identify the possible categories of the cell. Rules of cell linking are applied to the table *from top to bottom and from left to right to link* related cells together. By the way, we can find the reading paths to reach the entry cells. The cells with no links to other cells are called the entry cells. Finally, the abstract table is built from the reading paths and the information of this table is extracted from the abstract table. Based on this framework, we implement a prototype of our tabular information extraction system in MS-Windows environment. This prototype system provides an interface by which users can input a table in HTML format. Our system also has an interface that can output the abstract table. We have done some experiments on several tables with distinct layout styles. Our experimental results show that the prototype system can extract the logical structure of these tables with high precision and recall rate.

2. Modeling

Usually, the content of a table describes several different categories. These categories may have hierarchical label structure. The structure and contents of a table also describes the logical relationship between labels and entries. If we consider the function of a table, it provides readers a decision structure for searching an entry value through a sequence of labels selected from each category, and comparing these selected entries with different labels that are also selected from the same category. For example, in the table presented in Figure 1, the entries that we are interested are the numbers. There are three major categories of labels in this table. These categories are Year {1997, 1998}, Term {Winter, Spring, Fall}, and Mark {Assignment, Exam, Grade}. In the category of Mark, each label is by itself the category of its sublabels. For example, Assignment is the category of {Ass1, Ass2, Ass3} and Exam is the category of {Midterm, Final}. The label selection for searching the entry with value 60 is (Year.1997, Term.Spring, Mark.Assignment.Ass2)

Term	Assignment			Exam		Grade
	Ass1	Ass2	Ass3	Midterm	Final	
1997						
Winter	70	100	75	70	90	80.67
Spring	75	60	65	80	80	74.67
Fall	80	85	70	75	80	77.83
1998						
Winter	75	60	90	90	85	82.5
Spring	80	75	60	85	80	78.17
Fall	80	90	80	75	80	79.83

Figure 1 a typical table

A table may have two or more entry types. For example, there are two kinds of entry values in the table presented in figure 2, they are prices and beds.

Room Class	Price				Beds
	One Day	4 Months	Half Year	One Year	
Economy	$3,400	$84,000 x3	$75,000 x6	$66,000 x12	King x 1
Standard	$4,800	$120,000 x3	$108,000 x6	$93,000 x12	King x 1
Luxury	$6,400	$156,000 x3	$138,000 x6	$120,000 x12	King x 1 Queen x 2

Figure 2 a table with multiple entry types.

2.1 Basic Abstract Model

For convenience, we apply the abstract model developed in [1]. The abstract table can be described by the following notation [1]:

label, any string of characters and symbols, including the empty string.

labeled set, a set together with a label, can be specified as an ordered pair (*label*, *set*). (1991, ∅) and (Grade, {50, 60}) are two possible examples.

labeled domain, a labeled domain D is a labeled set (l, s) in which *label* (D) is the label part l and *set* (D) is the set part s. *set* (D) is either empty or a set of labeled domains $\{D_1,\ldots,D_n\}$ with distinct labels. A labeled domain is a hierarchical label structure of categories.

label sequence, a sequence of labels to identify a labeled subdomain. If a label sequence l identifies a labeled domain which contains a set of labeled domains $\{(l_1, s_1),\ldots, (l_r, s_r)\}$, then for $1 \le i \le r$ the $l.l_i$ is a label sequence that identifies the labeled subdomain (l_i, s_i).

frontier label sequence, a label sequence that identifies a labeled domain with an empty set. $fr(D)$ denotes the set of all frontier label sequences of a labeled domain D. For a set C of labeled domains, $fr(C) = \{fr(D) \mid D \in C\}$.

Unordered Cartesian Product, given $n \ge 1$ disjoint sets $A_1,\ldots,A_n$ their unordered Cartesian product $A_1 \otimes \otimes A_n$ is a set A such that each element of A is a set that contains exactly one element form each of the sets A_i ($1 \le i \le n$). Given a set C of $n \ge 1$ disjoint labeled domains $D_1,\ldots, D_n$, $\otimes fr(C)$ denotes $fr(D_1) \otimes\ldots\otimes fr(D_n)$.

abstract table, an abstract table $T = (C, \delta)$, where $C = \{D_i \mid 1 \le i \le n\}$ and δ is a map from $\otimes fr(C)$ to the universe of possible entry values. For example, δ (Year.1997, Term.Spring, Mark.Assignment.Ass2) = 60.

reading path, given an abstract table $T = (C, \delta)$, a reading path is an element of $\otimes fr(C)$.

2.2 Extension

The mapping from $\otimes fr(C)$ to entries might not be one to one. For example, in Figure 3, δ (Company.AA, Departing.City) = {TPE, AUS}. If we do not keep the order of mapping, we might get an ambiguous result while processing the following query: what is the departing city of aircraft 744 of company AA? In figure 3, it is obvious that this abstract model can not justify why these entries are put at the same row. We need to enhance the abstract model to overcome this problem. Our solution is that we merge all the mappings that map reading path to entries of the same row into a set. Therefore, the definition of an abstract table need to be extended to include the concept of a set of records $\{R_i \mid 1 \le i \le n\}$, where a record R_i is a set of mappings $\{\delta_j \mid 1 \le j \le m\}$ that map reading paths to entries of the same row.

Company	Departing		Arriving		Aircraft
	City	Date & Time	City	Date & Time	
AA	TPE	08/02 06:15pm	LAX	08/02 03:20pm	744
AA	AUS	08/02 05:15pm	LAX	08/02 10:44pm	S80

Figure 3 a table with multiple choices (in this table, the departing cities of the airplanes in company AA have two choices and the aircraft of the airplanes in company AA also have two choices.)

3. Knowledge Representation Base

Our Knowledge Representation Base is a base of conceptually knowledge representation. This base is constructed by a set of knowledge representation (KR) maps. A **KR Map** denotes the membership of a major **concept**. The membership of a major concept is represented by a tree structure of concepts. For example, an 'Airline Schedule' consists of information of 'Flight', 'Departing', 'Arriving', 'Aircraft', and so on. The information of 'Departing' and 'Arriving' in an 'Airline Schedule' consists of 'City', 'Date', and 'Time'. The KR Map denotes the **term representation** of each member concept by a set of **tagging rules**. For example, the instances of the term representation of 'City' are 'City' in English and ' ~ ' in Chinese. Beside the terminology

representation of a concept, the KR Map denotes the **instance** of each concept by a set of tagging rules. For example, an instance of a concept 'City' could be 'Taipei', 'TPE', and ' '. The KR base also provides tagging rules the necessary semantic patterns.

4. Information Extraction

To extract the abstract table from a table, our approach consists of three steps: In the first step, we examine every cell and identify all possible categories of labels or types of entries in each cell. In the second step, we construct the reading paths by finding all possible links between cells. In the final step, we partition all reading paths into several groups and construct the records.

4.1 Category Identification

In the first step, we need to identify the content of a cell to determine if it is a label or an entry. After that, we identify all possible categories and types for labels and entry cells respectively. To achieve the goal, the **information tagging** technique proposed in our previous paper [4] can be used.

The technique of information tagging can identify the possible meaning represented in an input text. We denote the **concept** as the identity of the meaning (for example, Year is the concept of 1999). Given a text fragment T and a set S of concepts $\{C_i \mid 1 \le i \le m\}$, the processing of information tagging will generate a set Γ of tags $\{\tau_j \mid 1 \le j \le n, n \le m\}$. A tag can be specified as an ordered pair (C, t) in which $C \in S$ and t is the text generated by reformatting the text T with one **tagging rule** associated with one concept C. In addition, for any two tags τ_i, $\tau_j \in \Gamma$, $C_i \neq C_j$. If the number of tags in Γ is larger than 1, we say that the meaning of the text T is ambiguous.

For example, if we apply the tagging technique to text '1367', the set of concepts {Year, Tel-ExtNum} and two tags--(Year, 1367) and (Tel-ExtNum, 1367)--will be generated,

A tagging rule consists of two templates: a **semantic pattern-matching template** and a **tag building template**. A semantic pattern-matching template is defined by a sequence of concatenated pattern-matching components. When a template is matched, each component in it should be mapped to a text fragment that satisfies the specified pattern of the component. In the original text, the strings that are mapped by all components are concatenated together and referred as the matched **text space**. In a template, the types of patterns used for matching are keywords, semantic categories, typing styles, the repetition of a pattern, a set of patterns, and the concatenation of some patterns. We can specify the limitation of matched text length in a pattern.

A tag building template has two components: tag header and tag body. The tag header defines the concept of the string matched. The tag body defines the order of concatenated strings. These strings are selected from the text that is matched by the components of semantic pattern-matching template or is assigned by the template creator.

A semantic category is a set of keywords that has the same meaning in some common situations. A category can be a subset of another category. A keyword might belong to more than one category because one word could have many different meanings in human language. In runtime, the size of a category can increase dynamically by recursively inserting the text part of tags into the categories referred by the concept part of these tags.

The information tagging system has two major components, there are semantic matcher and tag builder. For each input string, we use the semantic matcher to find the semantic pattern-matching templates that can match the string. The templates might match two or more overlapped text spaces. Some of the matched templates will be ignored if their text

space is covered by another template's text space. The other parts of the templates and the mapping that is associated with their components will be passed to the tag builder.

The tag builder creates the tag building templates according to the given semantic pattern-matching template. For a tag building template, the tag builder builds the text part of tag directly from the definition of tag body and assigns the concept defined in tag header to the concept part of the tag.

A semantic pattern-matching template can match keywords in several different forms. For example, it can match a simple keyword, a series of keywords, a set of keywords that has common meaning in some situations, and a string composed by the above patterns under specific length and repetition restriction. Because of this, the semantic pattern-matching template can be used to handle many different forms of concept representation.

In some aspects, the concept can be treated as a special kind of category. The strings within the same category have the same meaning in arranging some ideas. For example, the common meaning of 'Assignment', 'Examination', and 'Grade' is that they are some kinds of 'Mark', and the common meaning of 'Ass1', 'Ass2', and 'Ass3' is that they are an instance of 'Assignment'. In this example, the concept of a set of text also defines the category of the same set of text. Hence, we use the information tagging technique to identify the possible categories.

4.2 Reading path construction

When we read the content of a table from top to bottom, the path of our reading usually is a spanning tree of the table cells. Based on this observation, we can link cells together in the top-to-bottom direction. When we traverse the tables, the following rules need to be followed:

(1) One cell can be linked from topside,

(2) In the same column, cells only links to cells with equivalent category,

(3) The categories of the source cell and the target cell should be different.

(4) The cells crossed by the linkage should have different category to the source cell.

To process the cell linking, we start from the topmost cells and go downward to the bottom. For cells at the same column, we compare them row by row, trying to find possible links between the tags of the source cell and the tags of the target cell. When one or more links are found, we divide the links we find into several groups. Two links will be put into the same group if their source cells have the same category and their target cells also have the same category. At first, only the links in the first group will be selected. Links that are not selected will be pushed into a stack associated with the source cell of this column. One source cell can link to many target cells of this column as long as the previous rules are followed. After that we shift the source cell to one of the target cells and continue the same process in BFS order until all cells in the same column are processed. Since the link indicates the category we should select for the target cell, other tags in the target cell will be invisible in the next process. If no links were found when doing the cell comparison, the process need to be rolled back to the source cell with multiple links, and from that cell we select the next link to restart the process. If all links are examined, the process will be rolled back again. The process stops if all links of all topmost cells are finished.

Col.	Reading Paths in column[1]
1	Term{1997{Winter,Spring,Fall},1998{Winter,Spring,Fall}}
2	Assignment{Ass1 {1997 {70, 75, 80}, 1998 {75, 80, 80}}}
3	Assignment{Ass2 {1997 {100, 60, 85}, 1998 {60, 75, 90}}}
4	Assignment{Ass3 {1997 {75, 65, 70}, 1998 {90, 60, 80}}}
5	Exam {Midterm {1997 {70, 80, 75}, 1998 {90, 85, 75}}}
6	Exam {Final {1997 {90, 80, 80}, 1998 {85, 80, 80}}}
7	Grade{1997{80.67,74.67,77.83},1998{82.5,78,17,79,83}}}

Figure 4 the linking in each column.

[1] the denotation is *source* {*target*$_1$,...,*target*$_n$}

The same process is then applied to the same table again in the left-to-right direction by replacing column by row, row by column, top by left, and bottom by right respectively.

4.3 Record Collection

In this stage, we partition reading paths into several groups and build a record for each group. A terminal cell is a cell that does not have link to other cells. For each terminal cell, if we traverse the links backward, we will find a reading path for each column. Reading paths with terminal cells at the same row will be put into the same group.

Raw Record: {(Term.Winter, Year.1997), (Mark.Assignment.Ass1.70, Year.1997), (Mark.Assignment.Ass2.100, Year.1997), (Mark.Assignment.Ass3.75, Year.1997), (Mark.Exam.MidTerm.70, Year.1997), (Mark.Exam.Final.90, Year.1997), (Mark.Grade.'80.67', Year.1997)}
Condensed Record: {Term.Winter, Year.1997, Mark. Assignment.Ass1.70, Mark.Assignment.Ass2.100, Mark.Assignment.Ass3.75, Mark.Exam.MidTerm.70, Mark.Exam.Final.90, Mark.Grade.'80.67'}

Figure 5 the raw record and the condensed record

For each reading path, we collect all tags in this path and build a sub record. For example, the sub record of the reading path 'Term-1997-Winter' is (Term.Winter, Year.1997), and the sub record of the reading path 'Grade-1998-82.5' is (Mark.Grade.'82.5', Year.1998). To construct the record in the abstract table, we first construct a raw record from all sub records that are created from all reading paths of the same group. After that, we condense the raw record by deleting all duplicated label sequences in the sub records. In figure 5, we show a raw record and a condensed record as an example.

5. Experiments

To verify the effectiveness of our approach, we perform some experiments on the real data gathered from the www. After doing some survey, we found that the domain: 'Consultation Schedule' is one of the domains in which there are many difficult-to-handle tables. In our experiment, we collected 26 tables from the www and all of them are from the same domain as 'Consultation Schedule'. For each tabular consultation schedule, we perform 10 different queries listed in figure 6. The map information we used in our experiment is presented in figure 7. This map has all necessary category information demanded by all three steps.

Figure 8 presents the result of our experiments. Total 497 queries are made on 26 extracted tables. We got 1463 answers and among them 1352 answers are correct. From human operating on all queries, total 1353 correct answers reported. Our experiment shows that this approach get 92.41% high precision and 99.93% high recall. According to this high precision and recall rate, we may conclude that our approach can provide a general and useful solution for tabular information extraction.

Next, we examine the knowledge features of our test cases. In these 26 tables, 8 of them contain the features of department, date, service time, consulting room, and doctor. Their standard form is illustrated in figure 9. There are 13 tables contain features in a single department only. This kind of tables is illustrated in figure 10. The remaining tables contain only two or three features.

We examine the layout features for all tables. Basic layout features of tables are hierarchical labels and empty cells. Special layout features of tables are multiple tables extended in horizontal or vertical directions, multiple data in single cell, extra description cells, label omitted, footnotes, repetition reduction, and heterogeneous data in single cell. Our examination shows that some features are extremely difficult to handle. Those features include extra description cells, footnotes, and repetition reduction.

The extra description cells might contain strings that also appeared in regular cells. In the step of category identification, this condition will result in the creation of extra tags. Hence, in the step of reading path construction,

these extra tags might create unreasonable links.

In the current stage, we put our focus on the extraction of major information, the feature of footnote is unprocessed in our experiment, and we found that it would not affect the evaluation result.

Our approach does not solve the condition of repetition reduction since the reduction will result in the tags missing in the step of category identification. The lost tags also result in links missing in the step of reading path construction and the information lost is impossible to recover.

In the step of reading path construction, only links in the same group will be selected. Therefore, we can access the feature of multiple data in the same category. However, the feature of heterogeneous data in single cell becomes inaccessible under this approach. Since related information is often put together in a table, a category in the KR Map can be represented as a complex category combined from several categories. With this enhancement, we can transfer a combination of a set of heterogeneous data into a single data.

Figure 11 shows a table in which the span of a cell is divided and words are separated. Figure 12 shows a table in which the reasoning of words is in vertical direction. Our approach can not extract information correctly in these two cases.

ID	Query Input	Query Output
1	[Date]+[Department]+ [Service Time]	[Doctor]
2	[Date]+[Service Time]	[Doctor]
3	[Doctor]	[Date]+[Service Time]
4	[Date]+[Department]	[Doctor]
5	[Date]	[Doctor]
6	[Department]+[Service Time]	[Date]+[Doctor]
7	[Doctor]+[Service Time]	[Date]
8	[Department]	[Doctor]
9	[Date]+[Department]	[Doctor]+[Service Time]
10	[Doctor]	[Date]

Figure 6 queries to extract information from tables.

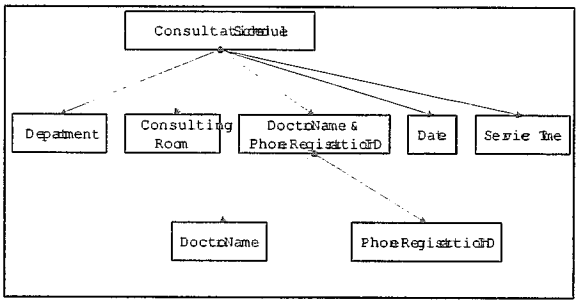

Figure 7 the KR Map of consultation schedule.

# Tables	# Queries	Total Answers	Correct Answers
26	497	1463	1352
Total Correct Items		Precision (%)	Recall (%)
1353		92.41	99.93

Figure 8 experimental result.

6. Conclusion and future work

In this paper, we develop a streamlined approach to extract the information from a table. Our approach is based on semantic templates that are associated with the knowledge representation maps. Given a map, our approach is a general and useful solution for extracting the information from tables in the same domain described by the map.

To extract information from the table, an abstract table is extracted from a physical layout with the support from an abstract model. According to the physical layout of tables, our experiment shows that our approach can access hierarchical labels, empty cells, multiple tables, multiple data in single cell, heterogeneous data in single cell, and label omitted tables well. Our experiment also shows our approach has difficulty to process extra description cells, footnotes, and repetition reduction.

Tabular information extraction is an important problem because there are many useful data and information represented in tabular form. If we know how to extract the information from tables, we can use the same technique to extract the information from tables in the web pages and save extracted information into database. A more general solution will help the computer to do the information collection more automatically. It can reduce the cost of human works on the web information collection.

In this paper, we have done some experiments on tables of a single domain. Our experience from these experiments can be applied to tables of other domains. We plan to do more experiments and more study on tables of other domains.

Reference

[1] Xinxin Wang. Tabular Abstraction, Editing, and Formatting. Ph. D. dissertation, University of Waterloo, Ontario, Canada, 1996.

[2] Shona Douglas, Matthew Hurst, and David Quinn. Using natural language processing for identifying and interpreting tables in plain text. Construction Industry Specification Analysis and Understanding System (CISAU) Project Np: IED4/1/5818, December 1994.

[3] Matthew Hurst and Shona Douglas. Layout and Language: Preliminary investigations in recognizing the structure of tables. Proceedings of ICDAR'97, August 18-20, 1997 in Ulm, Germany.

[4] Hui-Lung Wang, Wei-Kuan Shih, Chunnan Hsu, Yi-Shiou Chen, Yu-Lin Wang, Wen-Lian Hsu. Personal Navigating Agent. Proceedings of Third International Conference on Autonomous Agent (Agent 99), May 1-5, 1999 in Seattle, Washington, USA.

[5] T. J. Biggerstaff, D. M. Endres, and I. R. Forman. TABLE: Object oriented editing of complex structures. In Proceeding of the 7th International Conference on Software Engineering, pages 334-345, 1984.

[6] J. P. Cameron. A cognitive model for tabular editing. Technical Report OSU-CISRC-6/89-TR 26, The Ohio State University, Columbus, OH, June 1989.

[7] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. Advances in Knowledge Discovery and Data Mining. AAAI Press/The MIT Press, 1995.

Appendix

	:	# Z +		# Z 2		# Z ?	
		B	@	B	@	B	@
	1	« £ g 10201	— 20201	X " 10201	8 0 20201	^ 10201	« ı 20201
	2	« ı 10202	« ' 20202	10202	8 ‚ 20202	e 10202	20202
	3	´ £ 10203	& $ 20203	" fi 10203	˜ ` 20203	— 10203	H 20203
< -	1	11101		11101			21101

Figure 9 a regular style of consultation schedule.

B						
: ˜	# Z +	# Z 2	# Z ?	# Z	# Z v	# Z
1	« £ g 10201	X " 10201	^ 10201	« £ g 10201	|) 10201	˜ % 10201
2	« ı 10202	10202	e 10202	˜ % 10202	10202	10202
3	´ £ 10203	" fi 10203	— 10203	´ £ 10203	« ı 10203	10203
@						
: ˜	# Z +	# Z 2	# Z ?	# Z	# Z v	# Z
1	— 20201	8 0 20201	« ı 20201	e 20201	— 20201	
2	« ' 20202	8 ‚ 20202	20202	& $ 20202	20202	
3	& $ 20203	˜ ` 20203	H 20203	" fi 20203	˜ ` 20203	

Figure 10 a regular style of consultation schedule for single department.

˜ ˜			1	2	3	5
: t ˜			112	118	117	121
			0	0	0	0
	+		fi Q&	0	« ‡	0
		¡	0	0	0	{)
			fi Q&	0	s -	{)
	2		M U x	X W	0	0
		¡	0	0	« ‡	0
			0	0	« ‡	0
	?		0	X W	0	( )
		¡	˜ " |	0	s -	0

Figure 11 an irregular case our approach failed to extract.

	# Z +		# Z 2		# Z ?		# Z	
	B	@	B	@	B	@	B	@
+ s + :	... n		« n s	n œ	˜ › Ł x		3 Ł	˜ › Ł x
C ß ø t		«				«		
p :				n œ				C › ß x

Figure 12 case of words reasoning in the vertical direction.