Segmentation of On-line Job Advertisements

Kwok-Wai Cheung, King-Wai Fung, Kwok-Ching Tsui
Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
{william, kwfung, tsuikc}@comp.hkbu.edu.hk

ABSTRACT

To build a search engine with high precision and recall rates, more sophisticated techniques are inevitable for on-line document understanding. In this paper, a prototype for segmenting on-line job ads using the pattern matching approach is proposed. To evaluate its effectiveness, it has been used to extract company names and job titles from a set of real job ads with promising results.

Keywords

On-line document understanding, text segmentation, information extraction

1. INTRODUCTION

The Web has widely been used for posting various kind of information, where job advertisements (ads) is one of the examples. In order to facilitate information searching in a hugh on-line repository, manually created indices or full text retrieval techniques are commonly adopted. However, the former approach requires tremendous editorial effort while the latter one normally returns more information than needed. More sophisticated techniques are needed for on-line document understanding.

A job ad contains information like *company name*, *business nature*, *job title & job nature*, *job requirements*, *salary & benefits*, as well as *contact information*. The goal of job ads understanding is to correctly extract those information for supporting further analysis. Contrasting with other types of documents, job ads contain grammatical, telegraphic and ungrammatical text. Their writing style and paragraph formats are also much less formal. So, understanding such a mixture of semi-structured to free text with some special formatting styles turns out to be a non-trivial task. In general, the understanding process can be broken into two stages – text segmentation and information extraction [1]. As there does not exist a generic extraction approach which is optimal for all kinds of information, correctly segmenting the advertisement into conceptually coherent paragraphs and associating them to the various information types are crucial preprocessing steps.

There are generally two different approaches for text segmentation. The first one is based on detecting the contextual patterns of the beginning and ending of a coherent paragraph (also referred to as the beginning and ending delimiters), which assumes that the characteristics of the segments to be extracted can be identified without considering the paragraph contents at all. Good examples in a job ad include *company name* and *job title* where special layout formats are normally used for emphasizing them. The second approach for segmentation is based on the word statistics of the text contents so that conceptually coherent sentences can be grouped together. Good examples are *job nature* and *job requirement*. The two approaches are complementing to each other and both should be used in a complete segmentation system.

In this paper, we only concentrate on segmenting job ads using the pattern matching approach and describe the implementation details of a related system proposed for the segmentation. The system has been evaluated by applying it to extract company names and job titles from a set of real job ads obtained from a local on-line newspaper.

2. REPRESENTATION

Using the proposed system, segmentation is performed by specifying a segmentation template, which contains an ordered set of paragraph descriptors. Each descriptor contains a set of segmentation rules defining the possible pairs of beginning and ending delimiters of the target paragraph with the help of some specially designed tokens.

2.1 Token Classes

In our current implementation, eight different token classes are included (inspired from [2]), where each represents a particular text pattern.

Uppercase Strings: `[Upper,#num]`, e.g., "HONG KONG".

Capitalized Strings: `[Cap,#num]`, e.g., "Baptist University".

Lowercase Strings: `[Lower,#num]`, e.g., "computer science".

Numeric Strings: `[Number,#num]`, e.g., "2000".

Punctuation Symbols: `[Punct,#num]`, e.g., ",", ";", ...

HTML Tags: e.g., `["<HTML>"]`. (default case insensitive)

AsAppear Strings: e.g., `["ebXML"]`, `[","]`, `["\iProfessor"]`.
(`\i` means case insensitive)

Lexicon Lists: e.g., `[<Company>]`, `[<\iCompany>]`.
(to match strings found in an associated lexicon file)

where `#num` can be any integer greater than zero or set to "*" if the count cannot be pre-defined. Delimiters can either be included in or excluded from the finally matched pattern by using `"[..]"` and `"(..)"` respectively. Combinations of different brackets are also allowed.

2.2 Segmentation Rules

A segmentation rule consists of five parts — the beginning delimiter, the end delimiter, the maximum number of words allowed in between and two exclusion lists (one for completely matched patterns and one for partially matched patterns). The first two are expressed as disjunctions of tokens classes, separated by "|"s. For example, `[Upper,*] | [Cap,*] , [Lower,1]`. The use of distinction implies that the order of token matching within an expression has to be resolved. Different matching order can result in different segmentation results.

3. EXPERIMENTS

Our system prototype is written in Perl 5. In order to evaluate the system performance, we have created a dataset by retrieving job ads from a local on-line newspaper and preprocessing them so that irrelevant information like company logos, graphical advertisment banners, etc. are removed. A total of 848 job ads, which have been pre-classified into 22 categories, are solicited. 578 of them are found to be pre-formatted into exact three main blocks using the tag `<font>`, where the first block includes *company name* followed by *business nature*, the second one includes *job title*, *job nature* followed *job requirement*, and the third one includes *salary & benefits* followed by *contact information*. Some of them may have missing information but the order of the available ones remains unchanged. We call this subset of data *gd*. The remaining job ads do not have similar presentation order. We call this subset *bd*.

The system has been applied to filter out the company names and the job titles from the advertisements in the dataset using the segmentation template and the exclusion lists shown in Fig. 1. For the order of rule application, preliminary results show that greedy algorithms give unsatisfactory results. In our system, all the rules are applied and the one corresponding to the earliest appeared pattern wins. The segmentation accuracy results are tabulated in Table 1. As all the job titles in the dataset are consistently formatted using the bold tags, the segmentation accuracy is close to 100%. For company names, the situation is less straight-forward and the accuracy achieved so far is around 70%.

```
target=P1: {
CompanyName {
[Upper,1]|[Cap,1], [<Company>], 6, <\ilist1>,
<\ilist2>;
[Upper,*]|[Cap,*], [Cap,1]|[Upper,1], 1,
<\ilist1>, <\ilist2>;
[Upper,*]|[Cap,*], (Lower,1), 0, <\ilist1>,
<\ilist2>;
[Upper,*]|[Cap,*], (Punct,1), 0, <\ilist1>,
<\ilist2>;
} }
target=P2 {
JobTitle {
("<B>"), ("</B>");
} }
<list1>
"a", "an", "to", "in", ...
<list2>
"Admiralty", "Applications", "Are", ...
```

Figure 1: The segmentation template and exclusion lists used in the experiments.

4. FUTURE PLAN

4.1 Template Learning

While preliminary results are encouraging, we believe that further expanding the set of token classes and fine-tuning the templates can result in even higher accuracy. However, the manual effort required lowers the system portability. One of our future plans is to study how machine learning techniques, which have been widely used for wrapper induction [3], can be adopted for automatic template and exclusion list creation.

Company Name		Job Title	
gd	*bd*	*gd*	*bd*
74.6%	65.2%	100%	99.6%

Table 1: Segmentation results

4.2 The Use of WordNet and Domain-specific Ontology

Currently, the Lexicon Lists are created manually using some prior knowledge. To facilitate the process, on-line dictionaries, like WordNet, that contains semantics relationship between words can be used, especially the synonymy (words with similar meaning) and hypernymy (terms that subsume their subordinate terms in the hierarchy of semantics) [4]. Besides, specific industry jargons are sometimes used in the *job nature* and *requirement* of job ads. Additional domain-specific ontologies are generally required.

4.3 Towards A Job Matching Engine

Our system simply reveals a small part of our ultimate goal to build an on-line matchmaking system for matching job ads and applicants' resumes. For job ads, at least another segmentation module based on text contents as well as various information extraction modules tailor-made for each individual segment are required. On the other hand, similar segmentation and extraction modules should also be developed for the applicants' resumes.

5. CONCLUSION

In this paper, a preliminary prototype for on-line job ads segmentation implemented using the pattern matching approach is described. The proposed system has been evaluated using real job ads data collected from the Web with promising results. Future plans include template learning and the use of WordNet or domain-specific ontologies for further performance enhancement.

6. REFERENCES

[1] I. Muslea, Extraction patterns for information extraction tasks: A survey, In *Proceedings of AAAI-99 Workshop on Machine Learning for Information Extraction*, Orlando Florida, July, 1999.

[2] C.N. Hsu and M.T. Dung, Generating finite-state transducers for semi-structured data extraction from the Web, *Information Systems*, 23(8):521-538, 1998.

[3] N. Kushmerick, Wrapper induction: Efficiency and expressiveness, *Artificial Intelligence*, 118:15-68, 2000.

[4] J.Y. Chai and A.W. Biermann, A WordNet based rule generalization engine for meaning extraction, In *Lecture Notes in Artificial Intelligence (1325): Foundations of Intelligent Systems*, pp. 529-539, Springer-Verlag, 1997.