Classified Advertisement Search Agent (CASA):
A Knowledge-Based Information Agent for
Searching Semi-Structured Text

Xiaoying (Sharon) Gao and Leon Sterling
Technical Report 98/1

Intelligent Agent Laboratory
Department of Computer Science
The University of Melbourne
Parkville Victoria 3052, Australia

February 1998
Abstract

Information agents are increasingly being used for efficient information retrieval from the Internet. A domain specific agent, Classified Advertisement Search Agent (CASA), has been built for searching real estate advertisements obtained from multiple WWW sites to help users find accommodation. CASA is a knowledge-based system with three main features that distinguish it from other information agents. The first feature is the use of knowledge units (concepts) rather than keywords as the basis for matching. The knowledge unit representation and the text parsing algorithm are particularly useful for information extraction from semi-structured texts such as Web pages generated from online databases. The second feature is the integration of knowledge acquisition with the retrieval process. When CASA executes on the Internet, it can extract suburb names and price statistics to enlarge its knowledge base. The third feature is the information retrieval cycle where feedback from the user and the search result are used to adjust the query before restarting the search.

Keywords: Information Agents; Knowledge-Based Information Retrieval; Semi-Structured Text; Classified Advertisement Search; Real Estate Advertisement Search.

1 Introduction

The amount of information on the Internet is increasing dramatically, making efficient information retrieval very difficult. Information agents use artificial intelligence technology to support intelligent web access. One main group of information agents is browsing assistant agents or interface agents, such as Letizia [Lieberman 1995] and WebWatcher [Armstrong et al. 1995], which learn from users’ behavior or feedback, detect users’ interests and assist browsing. Another major agent group is search assistant agents that search the Web automatically and carry out users’ specific information retrieval tasks. Some examples are BargainFinder [Krulwich 1996], FAQ Finder [Burke et al. 1997] and CiFi [Loke et al. 1996].

Our research focuses on building a knowledge-based search assistant agent for one specific domain: online real estate advertisement search. We have infused our agent with knowledge about the real estate domain in order to address several problems evident in existing generic tools for online search. Currently, online classified advertisement search has the following problems:
Since classified advertisement Web services are distributed and dynamic, users have to browse each website individually, and they have to do this repeatedly to keep track of the frequently updated information.

Most of the classified advertisement search engines on the Web are based on keyword searching and suffer from low precision. The most common problem is that keyword searching fails when a fragment (or paragraph) consists of a set of advertisements for different real estate properties. When a user specifies a suburb name and a price, the suburb name matches that of one property but the price matches that of another. In addition, keyword searching alone does not handle abbreviations, synonyms, different price units.

None of the advertisement search engines can learn (or remember) anything from the advertisements to improve their search performance.

Users often suffer from receiving almost no advertisements or far too many advertisements (called the “too many” or “too few” problem). Users have to either relax or restrict the query, and redo the search many times.

The search results are usually not ranked and users may have difficulty in choosing among them.

Few advertisement search engines can use users’ feedback to modify and guide their future search.

Our research aims to build an online Classified Advertisement Search Agent (CASA) to solve these problems and automate the advertisement search process. This paper presents our work on using a knowledge-based approach to build an autonomous agent that reads online real estate advertisements to help its users find accommodation.

The paper is organized as follows. Section 2 outlines the agent specification. Section 3 presents an overview of the agent’s architecture and design. CASA’s three main features, namely, search based on partial understanding, knowledge acquisition and the retrieval cycle, are described in Section 4, 5, 6 respectively. The effectiveness of CASA is evaluated in Section 7. Section 8 presents related work, which is followed by conclusions and future work in Section 9.

2 Agent Specification

The task, input, output and main features of CASA are specified as follows:

- Task: online real estate advertisement search
- Input: two kinds of input are acceptable:
  - Query in “controlled” natural language. The query has to follow some format restrictions. For example, the suburb name has to be in capital letters.
  - Feedback in the form of a sample advertisement and instructions for modification. For example, “I like the following advertisement (attach the advertisement). It is even better if it is cheaper (or bigger, in another suburb)”
- Output: ranked real estate advertisements. If the query is refined, the refined query is also given as output.
- Features:
  - CASA is able to
    - Search multiple Web sites and integrate search results.
    - Search offline or online. The offline search automatically starts, runs periodically (e.g. once a day) and sends the results to users via email.
    - Support search based on partial understanding rather than on keyword matching.
♦ Acquire knowledge from the online advertisements, in particular, to extract suburb names and price statistics.
♦ Automatically refine the initial query according to users’ feedback and search results.
♦ Output a reasonable number of advertisements (e.g. between 1 to 10).

3 Agent Design

Our main argument is that knowledge plays a central role in advertisement search. For example, to enable an accurate search based on partial understanding, the agent needs knowledge to recognize real estate advertisements and to extract the suburb, price, size, type and features related to furnishing, transport, bond. The matching between the advertisements and a user’s requirements is not a simple string matching because the requirements are not equally important (e.g. suburb, size and price are more important than parking or furnishing). The matcher needs to know the differences and adjust its performance accordingly. The advertisement ranking is not just a calculation of the proportion of keywords found. It should reflect the attractiveness of a property. For example, a property with convenient features, such as close proximity to public transport or shops, should be highly ranked. To do this, the agent needs knowledge. Consequently, we used a knowledge-based agent control strategy for our agent.

In order to achieve searching based on partial understanding, we built a text parser that uses information extraction technology to extract knowledge units (concepts) from both the query and the documents. Then, future retrieval is based on knowledge unit matching rather than key word matching. The knowledge unit representation and the text parsing algorithm is presented in Section 4.

From our preliminary research [Gao and Sterling 1997], we found that we needed a suburb database to improve text parsing accuracy and price statistics to improve the search performance (e.g. to guide price refinement). A static database and handcrafted price statistics cannot cope with the dynamic environment. We decided to implement a learning machine to extract the suburb names and price statistics from online advertisements. The knowledge we used and the learning results, are presented in Section 5.

To address the “too many” and “too few” problem, and make use of users’ feedback, we viewed information retrieval as a cycle instead of a one way process. In one cycle, the agent takes a user’s query, gets the results, adjusts the query according to the search results and the user’s feedback, and restarts the search. The information retrieval cycle carries out the information retrieval task in 6 steps: query parsing, multiple Web site accessing, document parsing, matching, ranking and query refining. These steps are outlined in more detail in Section 6.

The agent architecture is shown in Figure 1. CASA is a knowledge-based system. Apart from the knowledge base, it has three main parts: the text parser, the learning machine and the retrieval cycle.

CASA is implemented in Eclipse Prolog as Eclipse has the HTTP library which provides basic functions for developing WWW applications [Bonnet et al. 1996].

4 Search Based on Partial Understanding

Traditional keyword search based Information Retrieval (IR) methods, which treat text as bags of unordered words, have two major problems. One is that one word may have many meanings and
this reduces the precision rate. The other is that many different words may have the same meaning and this results in lower recall rate. Ideally, IR should be based on concept (meaning) matching instead of keyword matching. However, current Natural Language Processing (NLP) technology is not mature enough to support natural language understanding.

Our main idea is to use Information Extraction (IE) technology, which is more limited than “full text understanding”, to extract knowledge units (concepts) from the text and to use the knowledge units for further information retrieval. We call word groups or phrases with an independent and specific meaning knowledge unit. For example, “$200 per week” is a knowledge unit price with the value 200. In the search process, both the query and the documents are parsed to obtain a set of knowledge units. CASA uses the knowledge units as the basis for matching instead of keywords. This process is shown in Figure 2.

In this domain, the main knowledge units are suburb, price, size, type, furniture, transport, facility, bond, available time, common words and abbreviations. Small knowledge units can be clustered to form larger knowledge units such as real estate ad, paragraph and document.

The main challenge is to extract knowledge units from text. Related previous work dates back to Schank’s primitive-act frame system in the 1970s [Winston 1984]. However, our work differs from others in that it introduces the knowledge unit representation using three kinds of constraints and the parsing algorithm consisting of three steps. Our method not only concerns the text but also the HTML tags and other structures. The method is particularly good at parsing semi-structured text in HTML such as web pages generated by Web services.
4.1 Knowledge Unit Representation

We aim to find a way to represent every knowledge unit, that is, to find a way to match a group of words (including their synonyms and abbreviations) to a certain concept. If we could represent knowledge units as a set of rules and facts, then we could use the rules and facts in the parsing algorithm to extract knowledge units from online advertisements.

Towards this aim, we view text in three perspectives: structure, length and content, each providing a kind of constraint on the text that gives a clue to its meaning (i.e. its matching concept). Structure is the boundary constraint for a group of words, which is usually expressed with HTML tags, character template, punctuation and special characters. Length is the number of characters in all the words. Content is the words, or their synonyms and abbreviations, which represent word meaning, word order, and relations between the words. An example of the structure, length and content of the text “<B> Hello World </B>” is given in Figure 3.

Text “<B> Hello World </B>”
Structure
Begin with HTML tag “<B>”
End with HTML tag “</B>”
Character template
“clll cllll”
(‘c’ represents capital letter;
‘l’ represents lower case letter;
‘n’ represents number)
Length 11
Content “hello world”

Figure 3 An example of the structure, length and content of text

For most knowledge units, one kind of constraint is sufficient. For example, knowledge units such as price, size, type can be defined by their content constraints. Other knowledge units such as paragraph can be defined by their structure constraints (e.g. paragraph begins with tag <p> and ends with tag</p>). Some knowledge units need all three constraints. Smaller knowledge units can be used to define larger knowledge units.

4.1.1 Content Constraints Representation

Definite Clauses Grammar (DCG) rules [Sterling and Shapiro 1994] are used to represent content. Here is one example of representing the knowledge unit price in DCG rules:

```
price(X) --> [$], number(X1), timeunit(N),{X is X1*N}.
number(X) --> [X], {integer(X)}.
timeunit(1) --> [perweek]; [pw]; [pwk].
timeunit(7/30) --> [permonth]; [pcm]; [p,c,m].
timeunit(1) --> [].
/* The knowledge represented:
There are three components in price: $, an integer number and a time unit.
There are two kinds of time units: per week and per month
Synonyms and abbreviations for per week and per month
The translation rate between the two is 1:7/30.
The default time unit is per week. */
```
DCG rules are good at representing ordered and continuous content. To represent free order content with discontinuous constituents, a set of predicates such as include_free_order/1, include_any_of/1, exclude/1 are created. One example is:

content(real_estate_ad, include_free_order([ku(suburb), ku(price), ku(size), ku(type)])).
/* A real estate ad consists of knowledge units suburb, price, size and type. The four knowledge units may come in any order and any extra information may appear among them */

4.1.2 Structure Constraints Representation

A set of predicates such as begin_with/1, begin_after/1, end_with/1, end_before/1 are created to describe the structure constraints. HTML tags (tags/1), character template (c_t/1), and knowledge units (k_u/1) can be used to specify the structure.

structure(paragraph, end_with([tags("<p"), tags("<hr"), tags("</p")])).
/* A paragraph ends with one of the paragraph tags or line tag */

structure(suburb, begin_with([c_t("ccc")])).
structure(suburb, end_before([c_t("*l"), tags("<")])).
/* A suburb name consists of upper case letters */

4.1.3 Length Constraints Representation

Two predicates max_length/1, min_length/1 are created to represent length constraints.

length(suburb, max_length(20)).
/* A suburb name has less than 20 characters */

4.2 Text parsing

All documents are fragmented to paragraphs. For every paragraph, basic knowledge units such as suburb, price, size and type are extracted. Then the basic knowledge units are clustered to groups to form the knowledge unit real estate ad.

For every knowledge unit, the extraction consists of three major steps: structure parsing, length parsing and content parsing. For knowledge units represented with three constraints, the process is to first locate the required text and extract a fragment according to the structure constraints. Then the length of the fragment is tested and its content checked. The parsing algorithm is given below. If other knowledge units are used in structure or content constraints, this algorithm is recursively called.

a) If the knowledge unit has structure constraints, the structure parser is started. Structure parser scans the text and extracts a fragment according to the structure constraints.

• If the structure constraint is specified using a character template (e.g. the structure constraint of suburb), then the text string is parsed character by character into a structure string consisting only of tags, character labels (“c” for capital letters, “l” for lower case
letters, “n” for numbers), punctuation and special characters. The text string and the structure string have the same length.

- Set the begin and the end pointer by checking the structure constraints on either the structure string (for character template parsing only) or the text string.
- Use the begin and the end pointer to extract a fragment from the text string. The structure string is only used to set the pointers. The output text is always obtained from the text string.

b) If the knowledge unit has length constraints, the length parser is triggered to check maximum and minimum length.

c) If the knowledge unit has content constraints, the text is parsed to a word list that only consists of words (without tags and punctuation). Then one of the two content parsers is triggered. One is a top down parser for knowledge units represented in DCG rules. The other is a parser for free order content with discontinuous constituents which parses every constituents in turn skipping over unrecognized words using the rule:

\[
\text{unrecognized(AnyWord)} \rightarrow [\text{Anyword}].
\]

d) After the three steps, the knowledge unit is extracted and stored into the database.

5 Knowledge acquisition

It is a very challenging task to enable information agents to not only search through the documents but also acquire knowledge from the documents to improve their future performance. For this particular domain, two learning efforts can dramatically improve agent performance.

One is to build a suburb database. Real estate advertisements usually start with a suburb name and the advertisements of different real estate properties are usually separated by a suburb name. So, accurately identifying and locating the suburb name is absolutely crucial for real estate advertisements parsing. A static database would be limited in its geographical information. It would be much better if the agent can read through the advertisements and build a suburb database automatically.

The other learning task is to get price statistics information such as the average price for a property of a certain size. The average price can be used as a default price, to specify users’ vague price requirements and to refine prices.

The learning is based on the following knowledge:
- There is at least one suburb name in a real estate advertisement.
- A suburb name usually consists of only upper case letters.
- A suburb name is usually less than 20 characters long.
- A suburb name usually does not contain common words or abbreviations.
- A suburb name is usually followed by price or size.
- If price and size appears one after the other (there is no other price and size in between but there may be other words) in one advertisement, then they belong to the same property, and so, can be used to calculate the average price.

Currently, CASA has built a database with about 180 suburbs and a periodically updated price table. A sample price table based on one day’s online advertisements is:
average_price(example(22), price(100.681816), size(1),_,_).
average_price(example(63), price(141.0952), size(2),_,_).
average_price(example(84), price(198.880951), size(3),_,_).
average_price(example(7), price(202.857147), size(4),_,_).
average_price(example(2), price(300.0), size(5),_,_).
average_price(example(0), price(0), size(6),_,_).

/* The fact “average_price(example(X), price(P), size(S),_,_)” means
that: based on X examples, the average price for S bedroom properties is $P per week.
The two unconstrained variables are reserved for other price related parameters such as suburb
and type. */

In the future, more complex learning tasks will be explored. For example, the price table shows
two facts: most properties have 2 bedrooms or 3 bedrooms and the average price of 3 bedroom
properties is very close to that of 4 bedroom ones. The strategies to extract these facts will be
studied in the future.

6 The Retrieval Cycle

We view the retrieval process as a cycle in which automatic query refinement is introduced to
address the “too many” and “too few” problem. The cycle consists of six major steps: query
parsing, multiple Web sites accessing, document parsing, matching, ranking and query
refinement. The cycle is show in Figure 4.

![Figure 4 The retrieval cycle](image)

Query parsing and document parsing have been discussed in Section 4. In this particular domain,
the other four steps have their own characteristics that differ from that of traditional methods.
They depend heavily on knowledge. The details are given below.

6.1 Multiple Web Sites Accessing

CASA uses the Eclipse HTTP library [Eclipse ] to access Web services. The interface of
advertisement search engines on the Web is usually a menu-based HTML form. Below, we show
how CASA queries these search engines.
To access an advertisement search engine, the following information needs to be explicitly represented:

- URL address (e.g. http://www.newsclassifieds.com.au/cgi-bin/nc21wrapper.pl)
- The access method: “GET” or “POST”
- Input field names (e.g. ns_collection, state, pub, cat, price, bedrooms, ns_query, ns_max_records)
- The information for every input field (e.g. state):
  - query prompt (e.g. “STATE=”)
  - the default value (e.g. “ALL”)
  - the Input Value List (e.g. [“South Australia”, “Western Australia”, “Northern Territory”, “Queensland”, “New South Wales”, “Victoria”, “Tasmania”, “Papua New Guinea”])
  - the Value Interior Representation List (e.g. [“sa”, “wa”, “nt”, “qld”, “nsw”, “vic”, “tas”, “png”]).

CASA needs to map a user’s query to different query strings to query different advertisement search engines. The algorithm is as follows:

- Get the query prompt for an input field (e.g. “STATE=”)
- Get the value for this input field,
  - If the input field is a “hidden” input field, then use the default value.
  - If there is no related user query for the input field, then use the default value.
  - If there is user query related to the input field, but it does not match any member of the Input Value List, then use the default value.
  - If there is related user input and it matches one of the values in Input Value List, then find the corresponding value in the Value Interior Representation List. For example, if the user’s input is “Victoria”, then the value is “vic”.
  - If there is related user input and either the Input Value List or the Value Interior Representation List is empty, then use the input as the value. For example, for the “text” input field, use the input text as the value.
- Attach the query prompt to the value and form the query of one input field, such as “STATE=vic”.
- Connect the string for every input field using “&” to form the query string for the search engine.

Currently, only three types of input are used as default. They are “Victoria” as the state, “real estate ad” as the collection and “renting” as the category. The other input, such as price and size, is not used because we would like CASA to be as general as possible, being able to utilize different advertisement search engines, even those that do not support querying by price and size.

CASA currently monitors two web site. One is Newsclassifieds [Newsclassifieds 1997] that contains advertisements from 27 Australian newspapers. The other is Fairfax market [Fairfax 1997] that collects advertisements from 6 top Australian newspapers. The two sites are accessed alternately. If the server is busy, CASA will wait and try again. If the server is still busy after a certain number of tries, it gives up.

6.2 Matching

After the query parsing and document parsing stages, both the query and the document are parsed to extract knowledge units. The knowledge units in a document are divided into groups, each group (which we call ad data) representing an individual advertisement. The matching compares the query knowledge units expressing the user’s requirements with every group of document knowledge units.
Generally speaking, there are two kinds of results when two sets of data are compared, “yes” for a match and “no” for no match. However, in this particular domain, the knowledge units are not all equally important, as reflected in the way the advertisements are presented. The knowledge units *suburb, price, size* and *type* are very important and most advertisements contain this information. The knowledge units *transport, furnishing, bond, parking* are less important and hence, are less likely to appear in advertisements. Some features such as *kitchen facility* and *bathroom facility* are very common and so rarely appear in advertisements. If an advertisement satisfies the important requirements and does not conflict with the less important or default requirements, the advertisement should be added to the result list.

For this particular domain, we made two changes. Firstly, we introduced the concept “weight” for knowledge units to represent their importance. The more important the knowledge unit is, the higher its weight. Currently, the weight for every knowledge unit is hardcoded into the knowledge base. Besides matching, the weights are used for ranking and query refinement (see Section 6.3, 6.4). Secondly, we added a new comparison result “not given”. Now, given a group of document knowledge units (ad data), for each query knowledge unit that represents a requirement, there are three types of comparison results:

- **Yes**: the ad data do contain this query knowledge unit and match the requirement. The matcher records the requirement and checks the next requirement.
- **No**: the ad data contain this query knowledge unit but do not match the requirement. The matcher skips this advertisement and checks the next advertisement.
- **Not given**: there are no related knowledge units found.

If the query knowledge unit has a high weight (e.g. *suburb, price, size, type*), the matcher skips this advertisement and checks the next advertisement.

If the query knowledge unit has a low weight (e.g. *transport, parking*), which means the information is less likely to be provided in the advertisements, the matcher ignores this requirement and checks the next requirement.

### 6.3 Ranking

Traditionally, result ranking is based on how closely the requirements and the results match. For example, the results can be ranked according to the proportion of keywords found (or knowledge units found in our case). However, this kind of ranking does not reflect how attractive a property is. Ideally, the ranking should correspond to a human view on how attractive a property is. For example, a place that is cheaper, bigger and has more convenient features (heuristically, a longer advertisement has more convenient features such as close to public transport, off street parking) is viewed as nicer and should be more highly ranked.

The ranking takes place in three steps. For every advertisement:

- Calculate the score for every knowledge unit, including *suburb, price, size, type, transport, facility, available time* etc. For example, the bigger, the cheaper a property is, the higher the score. The score is a value between zero and one. If the knowledge unit is not found, the score is zero (which means that a property with more features, usually a long advertisement, will be more highly ranked).
- Calculate the overall score as the sum of the single score of every knowledge unit multiplied by its weight.
- Rank the results based on the overall score.

### 6.4 Query refinement
The query can be refined according to users’ feedback and search results.

To refine a query based on users’ feedback, the following information is hardcoded:

- Modification of the value of knowledge units corresponds to a human view of what is cheaper and bigger.
- Suburb neighborhood, for example, Carlton is adjacent to Parkville.

To refine query according to search results, there are two options:

- If too many advertisements are found, then restrict the query. For example, the price range is narrowed and more requirements are added.
- If too few advertisements are found, then relax the query. For example, low weight requirements are deleted, the price is modified using the average price (e.g. the average price replaces the query price, or the query price is brought closer to the average price), and a suburb name is replaced by the name of a nearby suburb.

7 Evaluation

CASA has fulfilled or partially fulfilled most of the criteria in Etzioni’s definition for software agents [Etzioni and Weld 1995], such as autonomy, temporal continuity, communication ability and adaptability. One extra feature CASA has is knowledge acquisition. The features it does not have are mobility and personality.

Currently, there is no standard or well-developed method for information agent evaluation. Basically, an information agent is retrieving Hypertext information “intelligently”. However, there is no standard for Hypertext information retrieval either. Traditional information retrieval (IR) evaluation standard using precision and recall are difficult to adapt [Agosti and Smeaton 1996], especially when there is no obvious method to calculate recall. Due to these reasons, our evaluation is not complete. Some results are given below.

We evaluated our text parser and suburb learner using traditional IR evaluation standards. We tested these two components on a static collection of Web pages and calculated precision and recall by comparing CASA’s responses with manual parsing and learning results.

The text parser was tested on six Web pages downloaded from two Web sites. The six pages consist of “Houses to let” advertisements from Victoria. The first three pages are from Newsclassifieds and most of the advertisements are from the Leader Newspaper Group. The other three are from Fairfax Market and all advertisements are from The Age. The parsing results on four major knowledge units (suburb, size, price and type) are shown in Table 1. It can be seen that the overall precision is 96% and the recall is 78%.

The main problem is that the text parser has low precision and especially low recall for suburb parsing. The main reason for this is that the text parser is not sufficiently flexible to parse varying advertisement structures. For example, it fails when an advertisement begins with the full property address instead of a suburb name, and when the suburb is presented in lower case letters. Another problem is that the text parser gets confused when the alternatives of a knowledge unit are given in a single advertisement, for example, phrases such as “2/3 br”, “$400-$450 per week”, and “house, unit style” are not parsed correctly.
Table 1 Text parsing results

The suburb learner was tested on data from four days. Each day’s data consists of about 220 paragraphs from 7 Web pages on one site (Newsclassified). Most of the advertisements are from the Leader Newspaper Group. The recall on one set of data was calculated based on manual learning results. Since the manual recall checking was very time consuming, only precision was calculated for the other three sets of data by comparing CASA’s learning results with a local suburb list. The result is shown in Table 2. The recall on one set of data is 73%. The overall precision is over 86%.

Data Set | The number of paragraphs* | Precision** | Recall** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>229</td>
<td>105/120 = 88%</td>
<td>105/145 = 73%</td>
</tr>
<tr>
<td>2</td>
<td>224</td>
<td>97/113 = 86%</td>
<td>----</td>
</tr>
<tr>
<td>3</td>
<td>224</td>
<td>102/108 = 94%</td>
<td>----</td>
</tr>
<tr>
<td>4</td>
<td>213</td>
<td>82/93 = 88%</td>
<td>----</td>
</tr>
</tbody>
</table>

*The number of paragraphs is given to show the size of the data. There is at least one real estate ad in one paragraph.

**Precision=\(N_{Correct}/N_{Response}\)
Recall = \(N_{Correct}/N_{Key}\)
in which \(N_{Response}\) is the number of suburb names returned, \(N_{Key}\) is the total number of suburb names in the set of data and \(N_{Correct}\) is the number of correct suburb names returned.

Table 2 Suburb name learning results

One main problem is that the suburb learner fails to recognise the suburbs beginning with abbreviations, such as “ST KILDA” and “MT EVELYN”. Another problem is that some common words such as “INSPECT TODAY” are taken as suburb names. These two problems might be solved by adding heuristic knowledge to the knowledge base. One hard problem for both text parsing and suburb learning is handling varying advertisement structures.

To evaluate CASA’s retrieval performance, a comparison between CASA and the search engine of Newsclassifieds is shown in Table 3. CASA’s search was restricted in this site and its query
refining function was not used. The search scope for both systems was limited to online advertisements from “Victoria” state and in the “Houses to let” category on one day (21/11/97). The data was obtained by running 17 queries on both systems. Since a relatively small amount of advertisements were retrieved for each query (about 0-10), instead of the average precision, the sum of the numbers of advertisements retrieved, the numbers of correct advertisements and mistakes were calculated.

<table>
<thead>
<tr>
<th>System</th>
<th>Sum of NResponse</th>
<th>Sum of NCorrect</th>
<th>Sum of NMistake</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newsclassifieds</td>
<td>186</td>
<td>51</td>
<td>135</td>
</tr>
<tr>
<td>CASA</td>
<td>44</td>
<td>40</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3 Comparison between Newsclassifieds search engine and CASA

The results show that CASA makes much fewer mistakes than the Newsclassifieds search engine. The main reason is that our search, based on knowledge units matching, is particularly good at separating advertisements so that CASA shows a big advantage especially when there is a set of advertisements for different properties within a single paragraph. One main problem of the Newsclassifieds search engine is it performs very poorly on prices. CASA’s impressive accuracy proves that our search strategy based on partial understanding has been successful in this particular domain.

In order to evaluate our approach of viewing IR as a cycle, two versions of CASA, one with query refinement and one without, were set up to run once a day to carry out a specific user query. They both worked offline and did not accept users’ feedback. The query refinement in the second version was conducted according to the search results. Thirty days (from 21/09/97 to 21/10/97) data were recorded. The version without query refinement reported only one advertisement in the first 28 days (from 21/9/97 to 18/10/97) and after that it reported “nothing found”. It suffered from the typical “too few” problem. The version with query refinement reported not only the advertisement, which is the same as the one that the first version reported, but also seven extra advertisements. Among the seven extra advertisements, three did not match the lowest price requirement, one did not match the lowest price and type requirements, and three did not match the size, price and type requirements. These extra advertisements do give the user more choices. For example, the first three advertisements actually report properties that are cheaper than the user expected so that they might be better choices. A comparison of the two versions shows that the one with query refinement gives more information and suffers less from the “too many” and “too few” problem.

It should be pointed out that our test was limited in scope and coverage. Further evaluation on more sites with a larger variety of advertisements will be conducted in the future.

8 Related work

The closest work to CASA is an agent called RentMe [Burke et al. 1997], which is a knowledge-based system that browses real estate advertisements on the Internet. It differs from CASA in that it is a browsing assistant agent that uses case based reasoning to detect users’ interests, whereas CASA is a rule-based and goal-driven search agent.

Regarding the application domain, the other two projects on online classified advertisement search are Ineeda [Ineeda 1997], which is a project by Australian Newsclassifieds, and Cybergold [CyberGold 1996], which is an American classified advertisements service. They both aim to
filter and deliver advertisements according to users’ profiles. Unfortunately, not much information is available on their Web pages.

There has been research on knowledge-based information agents. Some examples are FAQ Finder [Burke et al. 1997], Internet Fish [LaMacchia 1996] and CiFi [Loke et al. 1996]. Apart from a different application domain, CASA differs from these agents in that it uses information extraction technology to enable search based on partial understanding, it has knowledge acquisition ability, and it views information retrieval as a cycle.

Our idea and method for parsing semi-structured text and extracting knowledge units are related to research on Information Extraction (IE) [SCIE-97 1997] and Message Understanding (MU) [MUC 1993]. With respect to knowledge representation based on DCG rules, similar work is found where DCGs are used to represent biological descriptions [Taylor 1995].

Two agents which have knowledge acquisition ability are ILA (Internet Learning Agent) [Perkowitz and Etzioni 1995] and Shopbot [Perkowitz et al. 1997]. Both are agents that learn from Internet documents based on a small amount of domain knowledge. However their learning methods and aims are quite different from our work. ILA uses a context-free algorithm to translate information sources into its own internal concepts. Shopbot learns how to shop at online stores by generalizing example queries. CASA learns knowledge units, such as suburb names, and statistical information, such as average prices, using information extraction technology.

9 Conclusions and Future Work

This paper has presented a knowledge-based information agent CASA that searches online real estate advertisements from the Internet help users to find accommodation. CASA automatically searches multiple Web sites and can work periodically. In the search process, both query and document are parsed through a text parser, which extracts knowledge units from text, then the extracted knowledge units are used as the basis for further information retrieval. CASA has shown better performance than the advertisement search engine at Newsclassifieds. CASA does not only search real estate advertisements but is able to learn suburb names with a precision of over 86% and calculate average prices for properties of certain sizes. CASA carries out the retrieval task in a cycle and is able to refine query according to the search results and user’s feedback so that it suffers less from the “too many” and “too few” problem.

This research automates the real estate advertisement search task and has proved that knowledge-based approach has been successful in this domain. The methods we used are particularly useful for monitoring Internet online services that provide semi-structured information. We believe that knowledge-based agents have great potential for efficient information access over the Internet.

Future work is needed:
• To extend the diversity of Web sites and advertisements being searched. Currently, CASA monitors two Web sites and most of the advertisements are from two newspapers. More Web sites will be added and strategies for choosing Web sites will be explored.
• To detect new Web sites and keep track the new and continuously updating advertisement search engines. It is a big challenge to keep the agent updated with the highly dynamic Web environment and to develop flexible and robust algorithms for accessing advertisement search engines.
• To build user models and use them to guide advertisement retrieval. Users have different search criteria and they should be allowed to express this by assigning different weights to
their requirements. In addition, search results can be shared by users who have similar interests.

- To develop new methods for agent evaluation. Because there is no standard database for testing, conducting a manual check of precision and recall is very time consuming. Also, for information agents, standards for evaluating hypertext information retrieval performance and testing their “intelligence level” are needed.
- To test knowledge-based architecture, knowledge representation and algorithms in other domains such as car advertisements, job advertisements and other Web services presenting semi-structured text. Also, more complex learning methods needs to be explored, which may help to build the knowledge base, thereby reducing the effort required in handcrafting knowledge.

Acknowledgements

We would like to thank Liz Sonenberg and Dominique Estival for their helpful comments and suggestions.

References


