VIETNAM NATIONAL UNIVERSITY, HANOI
COLLEGE OF TECHNOLOGY

TRAN NAM KHANH

SOME STUDIES ON A PROBABILISTIC FRAMEWORK
FOR FINDING OBJECT-ORIENTED INFORMATION
IN UNSTRUCTURED DATA

UNDERGRADUATE THESIS

Major: Information Technology

HANOI - 2009
TRAN NAM KHANH

SOME STUDIES ON A PROBABILISTIC FRAMEWORK FOR FINDING OBJECT-ORIENTED INFORMATION IN UNSTRUCTURED DATA

UNDERGRADUATE THESIS

Major: Information Technology

Supervisor: Assoc. Prof. Dr. Ha Quang Thuy
Co-supervisor: MSc. Nguyen Thu Trang

HANOI - 2009
ABSTRACT

With the rise of the Internet, there is more and more information available on the web. Among this, there is a lot of structured data embedded within web pages such as “an apartment with location, property type, price, bedrooms, bathrooms, area, direction”, etc...

However, there lacks an efficient method to retrieval those information. Therefore, in the two recent years, object search has been proposed and interested in as search method for domain-specific Internet application. To deal with the problem, some approaches have also researched such as Information Extraction, Text Information Retrieval []. Yet, these approaches have faced with the challenges about scalability and adaptability.

The thesis studies a novel machine learning framework to solve the object search problem and evaluate this approach to a Vietnamese domain - real estate. It shows a significant improvement in accuracy over the current retrieval method - the Mean Average Precision and Mean Reciprocal Rank of the approach is much better than those of baseline one, retrieve objects effectively and adapt to new domain easily. By developing from the idea, we also propose a method to generate snippet which helps users to identify the information they need without referring to document text. This method is also implemented and integrated successfully into object search systems.
ACKNOWLEDGMENTS

Conducting this first thesis has taught me a lot about beginning scientific research. Not only the knowledge, more importantly, it has encouraged me to step forward on this challenging area.

Firstly, I would like give my deepest thank to my research advisor, Prof. Dr. Ha Quang Thuy, who offers me an endless inspiration in scientific research, leading me to this research area. It is one of my biggest opportunities which have directed me to this way in higher education.

I would like to give my gratitude to MSc. Nguyen Thu Trang who has instructed me carefully and enthusiastically. She has given to me many advices and comments. This work can not be possible without her support.

I also want to thank Mr. Kim Cuong Pham, University of Illinois at Urbana-Champaign, who lets me a big opportunity work together with him for this work. He has encourages me a lot to finish this thesis.

Many thanks also go to all members of seminar group “data mining” who gave me motivation and pleasure during the time.

Finally, from bottom of my heart, I would specially like to say thanks to my family, my parents, my sister and all my friends.
TABLE OF CONTENTS

Introduction ...................................................................................................................1

Chapter 1. Object Search..............................................................................................3
  1.1 Web-page Search ...............................................................................................3
    1.1.1 Problem definitions .....................................................................................3
    1.1.2 Architecture of search engine ......................................................................4
    1.1.3 Disadvantages .............................................................................................6
  1.2 Object-level search .............................................................................................6
    1.2.1 Two motivating scenarios ...........................................................................6
    1.2.2 Challenges ...................................................................................................8
  1.3 Main contribution ...............................................................................................8
  1.4 Chapter summary ...............................................................................................9

Chapter 2. Current state of the previous work.........................................................10
  2.1 Information Extraction Systems ......................................................................10
    2.1.1 System architecture ...................................................................................10
    2.1.2 Disadvantages ...........................................................................................12
  2.2 Text Information Retrieval Systems ................................................................12
    2.2.1 Methodology .............................................................................................12
    2.2.2 Disadvantages ...........................................................................................13
  2.3 A probabilistic framework for finding object-oriented information in unstructured data .......................................................................................................13
    2.3.1 Problem definitions ...................................................................................13
    2.3.2 The probabilistic framework .....................................................................14
    2.3.3 Object search architecture .........................................................................17
  2.4 Chapter summary .............................................................................................20

Chapter 3. Feature-based snippet generation...........................................................21
  3.1 Problem statement ............................................................................................21
  3.2 Previous work ..................................................................................................22
  3.3 Feature-based snippet generation .....................................................................23
  3.4 Chapter summary .............................................................................................25
Chapter 4. Adapting object search to Vietnamese real estate domain..............26
4.1 An overview ........................................................................................................26
4.2 A special domain - real estate .........................................................................27
4.3 Adapting probabilistic framework in Vietnamese real estate domain ............29
  4.3.1 Real estate domain features ..................................................................29
  4.3.2 Learning with Logistic Regression .......................................................31
4.4 Chapter summary .............................................................................................31

Chapter 5. Experiment ............................................................................................32
5.1 Resources .........................................................................................................32
  5.1.1 Experimental Data ..................................................................................32
  5.1.2 Experimental Tools ...............................................................................33
  5.1.3 Prototype System ....................................................................................33
5.2 Results and evaluation .....................................................................................33
5.3 Discussion .........................................................................................................36
5.4 Chapter summary .............................................................................................37

Chapter 6. Conclusions ...........................................................................................38
6.1 Achievements and Remaining Issues ...............................................................38
6.2 Future Work ......................................................................................................38
LIST OF FIGURES

Figure 1. Web page graph ................................................................. 3
Figure 2. Example of web-page search ........................................... 4
Figure 3. General Architecture of Search Engine ............................ 5
Figure 4. Professor homepage search ........................................... 7
Figure 5. Real estate search ............................................................ 7
Figure 7. Examples of customizing Google Search engine ............ 12
Figure 8: Feature Execution on Inverted List ................................. 17
Figure 9. Object Search Architecture ........................................... 18
Figure 10. Examples of snippet ..................................................... 21
Figure 11. Feature-based snippet framework ............................... 23
Figure 12. Example of feature-based snippet ............................... 25
Figure 13. Some search engines in Vietnam ................................. 26
Figure 14. Two example websites about real estate ...................... 27
Figure 15. Search interface on real estate websites ...................... 28
Figure 16. Apartment search of Cazoodle ................................. 28
Figure 17. Camera product search ............................................. 29
Figure 18. Precision for Real Estate Search Engine .................. 35
Figure 19. Average Precision of comparison between BM25 and OS 36
LIST OF TABLES

Table 1. Web pages search problem ................................................................. 4
Table 2. Object search problem definition ...................................................... 13
Table 3. List of Operators and their functionality ......................................... 16
Table 4. List of features used in real estate domain in Vietnamese .............. 30
Table 5. Testing data for real estate domain .................................................. 32
Table 6. Real estate queries for testing .......................................................... 34
Table 7. Comparison MAP and MRR of BM25 and OS .............................. 35
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>IE</td>
<td>Information Extraction</td>
</tr>
<tr>
<td>IR</td>
<td>Information Retrieval</td>
</tr>
<tr>
<td>MAP</td>
<td>Mean Average Precision</td>
</tr>
<tr>
<td>MRR</td>
<td>Mean Reciprocal Rank</td>
</tr>
<tr>
<td>OS</td>
<td>Object Search</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
</tbody>
</table>
Introduction

The Internet has become important in daily life and as a result, Internet search has never played a more significant role. It is crucial for Internet users to obtain the desired information in an efficient and direct manner.

Currently, there is a lot of information available in structured format on the web. For example, an apartment on real estate website usually has its structured information such as location, number of bedrooms, price and area. A professor homepage usually contains information about his education, email, department and the university. These are examples of structured information that is exuberant on the web. From the object oriented perspective, considering each of above domains as a class of objects, a webpage containing detailed structured information as an object with its attributes. The problem of finding structured information on the web becomes object retrieval problem. Unfortunately, the current information retrieval approaches can not handle object search effectively.

Therefore, in recent two years, the problem is being interested by many scientists and researchers [7][13][14][20][27] They have proposed some approaches of overcoming the shortcoming of this current search engine for finding object on the web.

The thesis presents an investigation into the problem of searching for object, plausible solutions related to the problem. In particular, the main objectives of the thesis are:

- To give insight into object search problem, its motivation, some well-known object search systems and define the challenges which are required for these systems.

- To investigate the plausible solutions with literature techniques which have been published recently to solve the problem, especially study in-detail a novel machine learning framework [13].

- To propose a new approach to generate snippet for object search engine.

- To adapt object search to Vietnamese Real Estate domain and evaluate the performance of the approach through a number of experiments.

Roadmap: The organization of this thesis is follow
Chapter 1 provides a general overview of object search, its motivation comparing to the current search engine through some examples. This chapter then describes the challenges which they had faced with.

Chapter 2 presents the current state of previous work of searching for object with focus on the probabilistic framework for finding object-oriented information in unstructured data. This chapter also gives their advantages and shortcoming in solving object search problem.

Chapter 3 introduces our general framework for generating snippet based on feature language, index and document, then explains main advantages of the framework.

Chapter 4 investigates the object search problem in Vietnam. We first review the structure information on the web in Vietnam with focus on Real Estate domain. We then describe our adapting the probabilistic framework to Vietnamese Real Estate domain.

Chapter 5 presents our experiments on real estate domain to evaluate the performance of the probabilistic framework and discuss the results.

Chapter 6 sums up the main contribution, achievements, remaining issues and future work.
Chapter 1. Object Search

Current web search engines essentially conduct document-level ranking and retrieval. However, structured information about real-world objects embedded in static web pages and online databases exists in huge amounts. Typical objects are products, people, papers, organizations, and the like. Document-level information retrieval can unfortunately lead to highly inaccurate relevance ranking in answering object-oriented queries.

This chapter gives an insight into document-level information retrieval (web-page search), its shortcoming, as a result, motivating to object-level search. In the second section, we focus on object search, its concepts and some examples of real-world. We then give the challenges to the research community in the field and some conclusions.

1.1 Web-page Search

1.1.1 Problem definitions

The Internet can be considered a collection of web pages $P$, with link structure included in the web-page document. Thus, we have that $P = \{d_1, d_2, \ldots, d_n\}$ where $d_i$ is a web-page document.

The query $Q$ is a set of keywords which describe what the user wants to find out. Hence, we have $Q = \{k_1, k_2, \ldots, k_m\}$ where $k_j$ is a single keyword.

The output for web-page search approach is a list of web pages that contains query keywords ordered by the rank of the page. The rank typically expresses the quality of the web page related to the query. We assume that the result $R = \{p_1, p_2, \ldots, p_k\}$ where $p_i$ is a returned web page.

![Web page graph](image-url)
Therefore, the user should go through each page for determining whether the page contains information that he needs or not. To sum up, we model the web-page search problem as the table 1.

Table 1. Web pages search problem

| Given: A collection P of web pages with link structure |
| Input: Keywords query Q = \{k_1, k_2, \ldots, k_m\} |
| Output: Ranked list of pages R |

The figure 2 shows an example of the web-page search with document-level information retrieval approach on Google search engine.

![Figure 2. Example of web-page search](image)

1.1.2 Architecture of search engine

The general architecture of a web retrieval system (usually called Search Engine) is shown in the figure 3 [23]. The architecture contains all the major elements of a traditional retrieval system. There are also, in addition to these elements, two more components. One is the World Wide Web itself. The other is the Crawler which is a module that crawls web pages from the Web.
Each module in architecture of search engine has its own role.

- **Crawler module**: Walking on the Web, from page to page, download them and send them to the Repository.
- **Repository**: Storing the Web pages downloaded by Crawler module.
- **Indexing module**: The Web pages from Repository are processed by the programs of the Indexing module (HTML tags are filtered, terms are extracted, etc.)
- **Indexes**: This component of the search engine is logically organized as an inverted file structure.
- **Query module**: It reads in what the user has typed into the query line and analyzes and transforms it into an appropriate format.
- **Ranking module**: The pages sent by the Query module are ranked (sorted in descending order) according to a similarity score. It is presented to the user on the computer screen in the form of a list of URLs together with a snippet.
1.1.3 Disadvantages

First, from page view of the Web, it is obvious that it is very hard for users to directly describe what they want. They have to formulate their needs indirectly as keyword queries, often in a non-trivial and non-intuitive way with a hope to get “relevant pages” that may or may not contain target objects [20].

Second, users can not directly get what they want. The search engine only return a list of pages related to query ordered by ranking. Therefore, they have to scrutinize them to find out which pages they need. When the users have to examine each page for determine whether this page is their need, they will not feel comfortable.

1.2 Object-level search

As mentioned above, the good search engine has to be easy to use, however return what user want to get. Currently, Google search engine is the most popular to users in search technology. However, it also has some constraints for finding information about objects in some specific domains like person, product, etc…

In two recent years, many scientists have researched and proposed approaches to deal with the object search problem [7][13][14][20][27]. The section focuses on studying this problem: motivation, basic concepts, and challenges.

1.2.1 Two motivating scenarios

• Professor home page search

In this scenario, Ruby wants to look for the homepage of professors who are teaching at Illinois University and working in “databases” area. Firstly, she goes to Google and types “professor Illinois database”. However, Google returned her with list of pages related to the query. Some are homepages, some are publications and some are just news. She may have to look through each page to find out which pages she needs. Moreover, some professors in “biology” may be ranked higher than some “databases” professors and some professor’s homepages are ranked lower than some news article about themselves. All things make Ruby confused and turned to object search engine.

The system lets her enter the information into necessary field while leaving other field such as “name” blank. As soon as, Ruby hits “Search” button, the system returns the list of homepages ranked by the relevance to her query. She realized the top ranked result satisfies all of her constraints. Therefore, Ruby can have some ideas about returned objects without opening the links.
• **Real estate search**

In this scenario, Lien is looking for an apartment to buy. She wants an apartment in Ba Dinh, Hanoi, used area from 100 m² to 500 m² and price not over 1 billion VND. It is very difficult to find an apartment which satisfies these constraints with current search engine: Google, Yahoo. Therefore, she will turn to object search engine with hope to find a satisfied one.

Figure 5 provides an interface example for the problem of searching for an apartment.
1.2.2 Challenges

For object search problem, there are some requirements for a large-scale object-level vertical search engine.

- **Reliability**

  High quality structured data is necessary to generate direct and aggregate answers. If the underlying data are not reliable, then the users may prefer sifting the web pages to find answers rather than trust the noisy direct answers returned by an object-level vertical search engine [27].

- **Ranking Accuracy**

  With billions of potential answers to a query, an optimal ranking mechanism is critical for locating relevant object information from web pages [27].

- **Scalability**

  The size of the web gives rise to the requirement of scalability. If the size of the web is small, one can use above solutions. The large volume of web pages on the web makes the problem challenging. Furthermore, the information on the web is also changing such as price, etc...[13]

- **Adaptability**

  There is no standard on how websites have to be, except the HTML standard. In addition, many new websites are added and old ones are deleted every day. Thus, if a system cannot adapt to change, it might get obsolete and not usable at all [13].

1.3 Main contribution

Bearing in mind the importance of searching information on the Web, studies have shown that current search engine is not suitable for finding object in a specific domain on the Internet. It is necessary to build an object search engine to deal with the problem.

The thesis investigated the object search problem and some plausible solutions in which we focus on a probabilistic framework for finding object-oriented information in unstructured data [13].

To deal with this problem more efficient, we have proposed an approach for generating snippet for this system using feature language, index-based and document-based. We also adapt the probabilistic framework to Vietnamese Real Estate domain and have a satisfactory result.
1.4 Chapter summary

This chapter brought an overview of web-page problem and its disadvantages, as a result, motivating into object search problem in general and some specific domains in particular. After introducing some examples of searching for object which let users turn to object search engine, we then introduced the challenges which current approaches need to overcome in section 1.2.2. We then summarize our main contribution throughout this thesis.
Chapter 2. Current state of the previous work

We have introduced about the object search problem which have been interested in by many scientists. In this chapter, we discuss plausible solutions, which have been proposed recently with focus on the novel machine learning framework to solve the problem.

2.1 Information Extraction Systems

One of the first solutions in object search problem is based on Information Extraction System. After fetching web data related to the targeted objects within a specific vertical domain, a specific entity extractor is built to extract objects from web data. At the same time, information about the same object is aggregated from multiple different data resources. Once object are extracted and aggregated, they are put into the object warehouses and vertical search engines can be constructed based-on the object-warehouses [27]. Two famous search engines have built related to this approach: Scientific search engine - Libra (http://libra.msra.cn), Product search engine - Window Live Product Search (http://products.live.com). In Vietnam, Cazoodle company, which professor Kevin Chuan Chang has supported, is also developing under the approach (http://cazoodle.com).

2.1.1 System architecture

2.1.1.1 Object-level Information Extraction

The task of an object extractor is to extract metadata about a given type of objects from every web page containing this type of objects. For example, for each crawled product page, the system extracts name, image, price and description of each product.

However, how to extract object information from web pages generated by many different templates is non-trivial. One possible solution is that we first distinguish web pages generated by different templates, and then build an extractor for each template (template-dependent). Yet, this one is not realizable. Zaiqing Nie has proposed template-independent metadata extraction techniques [26][27] for the same type of objects by extending the linear-chain Conditional Random Fields (CRFs).
2.1.1.2 Object Aggregator

Each extracted web object need to be mapped to a real world object and stored into a web data warehouse. Hence, the object aggregator needs to integrate information about the same object and disambiguate different objects.

![System architecture of Object Search based on IE](image)

2.1.1.3 Object retrieval

After information extraction and integration, the system should provide retrieval mechanism to satisfy user’s information needs. Basically, the retrieval should be conducted at the object level, which means that the extracted objects should be indexed and ranked against user queries.

To be more efficient in returning result, the system should have a more powerful ranking model than current technologies. Zaiqing Nie has proposed the PopRank model [28], a method to measure the popularity of web objects in an object graph.
2.1.2 Disadvantages

As discussed above, one of obvious advantages is that once object information is extracted and stored in warehouse, it can be retrieved effectively by a SQL query or some new technologies.

However, to extract object from web pages it is usually labor intensive and expensive techniques. Therefore, it is not only difficult to scale to the size of the web, but also not adaptable because of different formats. Moreover, whenever new websites are presented in totally new format, it is impossible to extract objects without writing new IE module.

2.2 Text Information Retrieval Systems

2.2.1 Methodology

Another method for solving object search problem is that we can adapt existing text search engines like Google, Yahoo, Live Search. Almost of current search engines provide for users a function called advanced search which let them find out information that they need more exactly.

We can customize search engine in many ways for targeting domain. For example, one can restrict the list of returned sites such as “.edu” sites to search for professor homepages. Another way is to add some keywords, such as “real estate, price” to original queries to “bias” the search result toward real estate search.

![Figure 7. Examples of customizing Google Search engine](image)
2.2.2 Disadvantages

The advantage of using this approach is scalability because indexing text is very fast. In addition, text can be retrieved using inverted indices efficiently. Therefore, text retrieval systems scale well with the size of the web.

However, these approaches are not adaptable. In the above examples, the restriction sites or “bias” keywords must be input manually. Each domain has its “bias” keywords and in many cases, such customizations are not enough to target to the domain. Therefore, it is hard to adapt to the new domain or changes on the web.

2.3 A probabilistic framework for finding object-oriented information in unstructured data

Two above solutions can be plausible for solving object search problem. Yet, the Information Extraction based solution has low scalability and low adaptability while Text Information Retrieval based solution has high scalability but low adaptability. As a result, another approach has been proposed called probabilistic framework for finding object-oriented information in unstructured data which is presented in [13].

2.3.1 Problem definitions

**Definition 1:** An object is defined by 3 tuples of length n, where n is the number of attributes, N, V, T. N = (α₁, α₂.. αₙ) are the names of attributes. V = (β₁, β₂.. βₙ) are the attribute values. T = (µ₁, µ₂.. µₙ) are the types that each attribute value can take in which µᵢ often is of \{number, text\}.

**Example 1:** “An apartment in Hanoi with used area 100m², 2 bedrooms, 2 bathrooms, East direction, 500 million VND” is defined as N = (location, types, area, bedrooms, bathrooms, direction, price) and V = (‘Hanoi’, ‘apartment’, 100, 2, 2, ‘East’, 500) and T = (text, text, number, number, number, text, number).

**Definition 2:** An object query is defined by a conjunction of n attribute constraint Q = (c₁ ^ c₂ ^ … ^ cₙ). Some constraints would be constant 1 when the user does not care about the attributes. Each constraint depends on the type of attribute the object has. A numeric attribute can have a range constraint and a text attribute can be either a term or a phrase.

**Example 2:** An object query for “an apartment in Cau Giay at least 100 m² and at most 1 billion VND” is defined as Q = (loca=Cau giay ^ type=apartment ^ price<= 1 billion VND ^ 1 ^ 1 ^ areas>100 ^ 1). The query means the user does not care about “bedrooms”, “bathrooms”, “direction”.
Another way of looking at our object search problem from the traditional database perspective is to support the select query for objects on the web.

Table 2. Object search problem definition

| Given: Index of the web W, An object Domain D^n |
|---|---|
| Input: Object query (Q = c_1 \land c_2 \land \ldots \land c_n) |
| Output: Ranked list of pages in W |

To sum up, we imagine object search problem as advanced retrieval database.

```
SELECT web_pages
FROM the_web
WHERE Q = c_1 \land c_2 \land \ldots \land c_n is true
ORDER BY probability_of_relevance
```

2.3.2 The probabilistic framework

- **Object Ranking**

Instead of extracting object from web pages, the system returns a ranked list of web pages that contain object users are looking for. In this framework, ranking is based on the probability of relevance of a given object query and a document \( P(relevant \mid object\_query, document) \). Assuming that object query is a conjunction of several constraints for each attributes of object and these constraints are independent, the probability of the whole query can be computed from the probability of individual constraint.

\[
P (q) = P (c_1 \land c_2 \land \ldots \land c_n) \\
= P (c_1) P (c_2) \ldots P (c_n) \tag{1}
\]

To calculate the individual probability \( P(c_i) \), the approach uses machine learning to estimate \( P_{ml}(s \mid x_i) \) where \( x_i=x_{i1}, x_{i2}, \ldots, x_{ik} \) is the relevance features between constraint \( c_i \) and the document.

\[
P (c_i) = P (c_i \mid correct) \times P (correct) + P (c_i \mid incorrect) \times P (incorrect).
\]

\[
= P_{ml} (s \mid x_i) \times (1-\varepsilon) + 0.5 \times \varepsilon. \tag{2}
\]

\( \varepsilon \) is an error of machine learning algorithm. If machine learning is wrong, the best guess for \( P(c_i) \) is 0.5.
• **Learning with logistic regression**

The next task of the framework is how to calculate $P_m(s|x_i)$ by machine learning. To do this, the approach uses Logistic Regression [21] because it not only learns a linear classifier but also has a probabilistic interpretation of the result.

Logistic Regression is an approach to learning functions of the form $f: X \rightarrow Y$, or $P(Y \mid X)$ in the case where $Y$ is discrete-valued, and $X = <X_1 \ldots X_n>$ is any vector containing discrete or continuous variables. In this framework, $X$ is the feature vector derived from a document with respected to a constraint in user object query. $X$ contains both discrete values, such as whether there is a term ‘xyz’, and continuous values, such as normalized TF score. $Y$ is a Boolean variable corresponding to whether the document satisfies the constraint or not.

Logistic Regression assumes a parametric form for the distribution $P(Y\mid X)$, then directly estimates its parameters from the training data. The parametric model assumed by Logistic Regression in the case where $Y$ is Boolean is

$$P(Y = 1 \mid X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

and

$$P(Y = 0 \mid X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

The above probability is used for the outcome (whether a document satisfies a constraint) given the input (a feature vector derived from the document and the constraint).

• **High level feature formulation**

Another important part of this system is how to formulate $k$-feature vectors $x_i = x_i^1 x_i^2 \ldots x_i^k$ from the constraint $c_i$ and a document. To carry out this, a list desired features is defined [13].

- Regular expression matching features (REMF)

Because a lot of entities such as phone number (e.g: +84984 340 709), areas (e.g: 100m2)… can be represented by regular expression, the features “where such regular expression existed” should be used.

- Constraint satisfaction features (CSF)
Since the object queries contain constraints on each attribute value, it is desired to have features expressing whether the value found in a document is satisfied by the constraints.

- **Relational constraint satisfaction features (RCSF)**

  This type of feature specifies the relational constraints such as “proximity”, “right before/after”…between the two features.

- **Aggregate document features (ADF)**

  All of the above features are binary. This feature shows the way to aggregate them for a document. For instance, count how many CSF in a document, relevant scores of document and query such as TF-IDF, etc…

  - **Feature language**

    All features are executed based on inverted index. Therefore, the system gives a language called the feature language to provide capability of executing efficiently on the inverted index. The feature language is a simple tree notation that specifies a feature exactly the way it is executed in inverted index. Each feature has a syntax:

    \[ \text{Feature} = \text{OperatorName} ( \text{child}_1, \text{child}_2, \ldots, \text{child}_n). \]

    Each child is an inverted list and the OperatorName specifies how the children are merged together. The child of a feature node can either be another feature node or a literal (text or number). The feature query, which consists of many features, forms a forest.

Table 3. List of Operators and their functionality

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Leaf Node Operators</strong></td>
<td></td>
</tr>
<tr>
<td>Token(tok)</td>
<td>Inverted list for term ( tok ) in Body field</td>
</tr>
<tr>
<td>HTMLTitle(tok)</td>
<td>Inverted list for term ( tok ) in Title field</td>
</tr>
<tr>
<td>Number_body(C)</td>
<td>Inverted list for numbers filtered by constraint ( C )</td>
</tr>
<tr>
<td><strong>Merging operators</strong></td>
<td></td>
</tr>
<tr>
<td>And(A,B,C,…)</td>
<td>Merge-join child lists by docid</td>
</tr>
<tr>
<td>Or(A,B,C…)</td>
<td>Merge-join child lists</td>
</tr>
<tr>
<td>Phrase(A,B,C…)</td>
<td>Merge-join child lists as consecutive phrase</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Proximity(A,B,l,u)</td>
<td>Merge lists A and B and join them on “position distance [l,u]”</td>
</tr>
<tr>
<td><strong>Arithmetic Operators</strong></td>
<td></td>
</tr>
<tr>
<td>TF(A)</td>
<td>Inverted term frequency A</td>
</tr>
</tbody>
</table>

The system is constructed with a “parameterized form” called **macros** to denote a value from user object query. The macros are replaced by the value of user object query at runtime. Thus, a feature “TF(HTMLTitle(LOCA))” would mean the TF score of LOCA macro in Title. In addition, we can also express the regular expression with above features. For examples, “areas 100 m2” can be re-written as “Phrase(Number_body(_range(100,500)), Token(m2)))” meaning “an integer in range [100,500], followed by ‘m2’.

- **Feature execution**

Each node in the feature tree corresponds to an inverted list. Inverted list in parent nodes are the result of combining their children’s inverted list. Because all inverted lists are ordered by documents’ id, they can be joined efficiently without materialization.

![Feature Execution on Inverted List](image)

**Figure 8: Feature Execution on Inverted List**

### 2.3.3 Object search architecture

The general architecture* of object search based on the probabilistic framework is described in [14]. The system consists of several modules which are divided into two parts: *domain-independent* and *domain-dependent* modules.

- **Domain independent modules**
These modules can be adaptable to new domain without modifying or a little. They used some tools and well-known techniques for constructing.

**Crawler**

The crawler is a standard web crawler as described in [16]. In addition to the general crawler, we have several focused crawlers to collect pages from some targeted websites.

*This was done by DAIS Laboratory working in collaboration with SISLAB, VNU.

**Indexer**

Lucene is used to index basic feature from web pages so that the indexer can captures the fundamental elements of web pages content while allowing efficient query processing. The inverted terms include tokenized strings and number. Some attributes with each positing of these terms are also stored. This allows fast look up for queries such as a number between 100 and 300 in the body of web pages.

Moreover, it is considered that terms in different parts of a web page form different features. For examples, a word “chung ĉut” in Title of a page is different from that in body.

**Query Processor**

The query processor processes a given unstructured feature query and returns the list of web pages containing one or more features in the query. The unstructured feature query is a list of encoded features that can be efficiently answered using inverted index. The query processor reads the features from the query, maps them into an efficient query plan and executes it on the inverted index.
- **Domain dependent modules**

**Query Translator**

The goal of query translator is to translate a user object query defined into a ranking function that ranks web pages in our index. The ranking function is a weighted combination of the mentioned unstructured features. It is calculated as a product of the probabilities that each constraint in the object query satisfied by the document.

\[
\text{Score} (Q, d) = \prod_{i=0}^{n} \text{Prob}(\text{Field} | c_i, d)
\]

\[
= \prod_{i=0}^{n} f \left( \sum_{j=1}^{w_j} w_j \times F_j \right)
\]

The query translator sends an unstructured feature query to the query processor described above, aggregates the score for each returned web pages and finally returns the top ranked web pages to user.

**Annotator**

Annotator lets us tag web pages with the ground truths (object attributes) about the object it contains or “none”, meaning that the web page contains...
no object. The ground truths are used to train and evaluate Query Translator. By annotating important web pages only, the system reduces the developer’s effort to train the Translator Learner.

**Query Translator Learner**

Finally, the Query Translator Learner learns a ranking function that is used by a Query Translator for a particular domain. The ranking function involves the set of features $\mathbf{F}^j$ and the corresponding weights $w^j$. Given a set of features, we generate supervised training examples from Annotator’s ground truths. We use Logistic Regression to compute the set of weights that minimize training data classification error.

**2.4 Chapter summary**

This chapter gave an investigation into two plausible solutions of object search problem which are Information Extraction Systems and Text Information Retrieval Systems. Each solution based on its approach with different advantages, however, they also have some shortcomings. Information Extraction based solution has low scalability and low adaptability while Text Information Retrieval based solution has high scalability but low adaptability.

In the third section, the thesis studied in-detail the probabilistic framework for finding object-oriented information in unstructured data. It based on the domain-dependent features and machine learning for ranking object related to user’s query. To estimate the relevant of the feature and a document, the framework used Logistic Regression approach with high level features formulation and execution.

The last section described the general object search architecture based on the probabilistic framework. The architecture illustrated the capability of adapting this approach in a new domain.
Chapter 3. Feature-based snippet generation

The usual way of interacting with an IR system is to enter a specific information need expressed as a query. As a result, the system will provide a ranked list of retrieved documents. For each of these documents, the user will be able to see the title and a few sentences from the document. These few sentences are called a *snippet* or *excerpt* [8]. The presentation of query biased document snippets as part of results pages presented by search engines has become an expectation of search engine users. In this chapter, we investigate some previous methods to generate query-based snippet, then have proposed another approach for snippet generation problem based on feature language.

3.1 Problem statement

Each result in search results returned by search engine typically contains the title, the URL of the actual document and few sentences from document. These few sentences are called a *snippet* or *excerpt*. They play a key role of helping the user decide which of the retrieved document are more likely to convey his information need. Ideally, it should be possible to make this decision without having to refer to the full document text.

![Snippet](image)

Figure 10. Examples of snippet

Snippets are short fragments of text extracted from the document content. They may be static - query-independent (containing the first 50 words of document or the content of its description metadata) or *query-based*. A query-based snippet is one selectively extracted on the basis of its relation to the searcher’s query and now state of the art in text search [8].
3.2 Previous work

A variety of methods for generating query-based snippet have been proposed and implemented. The methods differ in which of the potentially many snippets are extracted, and in how exactly documents are represented so that snippets can be extracted quickly. However, they divided into two main approaches: document-based and index-based.

For document-based snippet generation, it follows two-step approach:

- For given query, compute the ids of the top ranking documents.
- For each of the top-ranked documents, fetch the document text and extract a selection of segments best matching the given query.

However, for combined proximity query such as 2..5 mps, the document-based approach has a trouble because only the segments from the document, where query words occur close each other, are displayed. In addition, for semantic query with several non-literal matches of query words, this approach can not able to identify matching segments to generate snippet. The early version of Lucene highlighter is one example of this approach.

For index-based snippet generation, it goes in four steps:

- For given query, compute the ids of the top ranking documents.
- For each query word, compute the matching positions of that word in each of the top-ranking documents.
- For each of the top-ranked documents, given the list of computed matching positions and given a pre-computed segmentation of the document, compute a list of positions to be output.
- Given the list of computed positions, produce the actual text to be displayed to the user.

Based on matching positions of query word and the top-ranking document, this method lets us be able to generate snippet for combine proximity query even semantic query. However, the requirement of this approach is to pre-compute segmentations of the document and cache them together index. Additionally, it may also face with the problem when snippet is a combination of two or more segments.
3.3 Feature-based snippet generation

Developing from the idea of the index-based approach and the feature language defined in probabilistic framework. We have proposed another approach called feature-based for generating snippet in object search system. The feature-index based snippet generation has been followed in four steps:

- (S0) For a given object query, compute the ids of the top-ranking documents. This is like index-based.

- (S1) For each constraint in object query and feature language, compute the matching positions in each of the top-ranking documents.

- (S2) For each of the top-ranked documents, given the list of matching positions computed in S1, computed a list of positions to be output

- (S3) From the cached document content, extract the text correlative to the position

![Feature-based snippet framework](image)

Figure 11. Feature-based snippet framework

To do our approach efficiently, we use Lucene to index basic features from web pages to create a positional inverted index. This simply means that each inverted list must store, for each document, the positions corresponding to term appears. For such an index, an inverted list conceptually looks as follows:
In the step S1, given a constraint in object query and feature query, for example in camera product domain, ZOOM has to be in range (2, 10) and feature query for this constraint is “Proximity(Number_body(ZOOM), Token(zoom), -5, 5”). From this constraint and feature query, we compute matching list based on positional inverted index. By substituting macro ZOOM into the feature query, we obtain “Proximity(Number_body(_range(2, 10)), Token(zoom), -5, 5)”. To get the position list related to this feature query, we compute the positions of each child “Number_body(_range(2, 10))” and “Token(zoom)” which is easily executed on positional inverted index and then merge two position lists into result list by constraint “window -5 5”.

In the step S2, after getting position lists of constraints, we have to decide which combination of positions from them into result list. For example, in camera product domain, the position list of ZOOM constraint consists of 29, 40, 90 while the position list of NAME constraint includes 25, 30. It is heuristic that the result list should be a combination of 25 and 29 because the constraints are usually close each other. It is optimal problem.

In the step S3, from cached document we extract the text correlative to positions computed in S2.

The feature-based snippet generation inherits the advantage of index-based approach which executes efficiently on positional inverted index and carries out good result for combined proximity query even semantic query. Furthermore, by using feature query for each constraint in object query makes this approach generate a more accurate snippet than previous ones.

The figure [12] shows feature-based snippet on object search engine in camera product domain with object query “NAME = Sony” “MEPIX = in_range(4, 20)” and “ZOOM = in_range(5, 20)”.

<table>
<thead>
<tr>
<th>docids</th>
<th>D2</th>
<th>D7</th>
<th>D9</th>
<th>D29</th>
<th>D79</th>
</tr>
</thead>
<tbody>
<tr>
<td>positions</td>
<td>9</td>
<td>19</td>
<td>23</td>
<td>29</td>
<td>99</td>
</tr>
<tr>
<td>word ids</td>
<td>W9</td>
<td>W9</td>
<td>W9</td>
<td>W9</td>
<td>W9</td>
</tr>
</tbody>
</table>
This chapter introduced snippet generation problem for object search systems. Two major approaches, document-based snippet and index-based snippet, were discussed in the second section. Studies showed that index-based snippet is more efficient than document-based one in generating with combined proximity query and semantic query.

Based on the idea of using feature query in the probabilistic framework discussed in chapter 2, we proposed another approach to generate snippet called feature-based. Through number of experiments, it indicates that this approach is suitable and effective for object search system.
Chapter 4. Adapting object search to Vietnamese real estate domain

We have introduced about object search on structured data on the web and some plausible solutions for the problem. In this chapter, we will provide an overview of object search in Vietnam and investigate structured data in Vietnamese websites on some domains. Finally, we show the potential of applying the object search and adapt this approach to Vietnamese real estate domain.

4.1 An overview

As search engine is a very important tool for finding information on the web, Vietnam companies have been constructed their own search engine such as www.xalo.vn, www.socbay.com, etc...

![Figure 13. Some search engines in Vietnam](image)

Each company achieved initial success on Vietnamese language. However, Google search engine still main one of searching for information on the web in Vietnam. Therefore, some companies only focus on specific domain such as music, real estate and product, etc... These famous search engines of this type in Vietnam consist of baamboo.com (for music domain), cazoodle.com (for product domain). It is obvious that these companies understand the potential of finding information in specific domain which is related to object search. This way will be better than focusing on general search engine to compete with Google (very strong search engine).

Furthermore, nowadays, Vietnamese websites have better design with many structured data in them. More specific, in real estate domain, a lot of web pages provide a good structure for finding information according to object search approaches.
For example, the figure 12 shows two popular websites about real estate in Vietnam:


Figure 14. Two example websites about real estate

### 4.2 A special domain - real estate

In this section, we show more details on real estate domain: structured websites, search engines and express the reason why we apply the probabilistic framework for object search problem on this domain in Vietnamese language.

In recent years, real estate is becoming hot problem in Vietnam. More and more people want to buy a new house or rent an apartment for times. Each person wants the one which satisfy their constraints such as: location, areas, price, etc… Thus, the requirement for finding information about real estate becomes very necessarily.

Additionally, there are more and more websites about real estate such as www.metvuong.com, www.batdongsan.com, www.nhadat24h.com, etc…as shown above figure 12. These websites have a structured for describing information about apartment: location, areas, bedrooms, bathrooms, price, etc…even though they have different ways of constructing.

Therefore, these websites also provide their search tool on this data. However, all of them have a database for storing all information about the apartments, so that they simply use SQL query to retrieval information. Therefore, the returned results are quite precise. The problem here is that if we want to scale more and more data from the Internet or compare data from many pages from the same object (apartment), object search engine is well suitable for this.
In Vietnam, Cazoodle company, which professor Kevin Chuan Chang from University of Illinois at Urbana-Champaign support, is constructing an object search engine under Information Extrainutions approach. They develop on domains which have good structured data such as product (camera, laptop) and real estate. However, they only apply on English language still not Vietnamese.

Figure 15. Search interface on real estate websites

In Vietnam, Cazoodle company, which professor Kevin Chuan Chang from University of Illinois at Urbana-Champaign support, is constructing an object search engine under Information Extrainutions approach. They develop on domains which have good structured data such as product (camera, laptop) and real estate. However, they only apply on English language still not Vietnamese.

Figure 16. Apartment search of Cazoodle

After we examine carefully about real estate domain in Vietnam and initial success of the approach on English domains* (professor homepages, product search), we decided to apply our approach on real estate in Vietnam. The below figure 17 shows the result of applying our methods on Camera product search in English.

*This work was done together with DAIS Laboratory, University of Illinois at Urbana-Champaign.
4.3 Adapting probabilistic framework in Vietnamese real estate domain

The probabilistic framework has implemented in some English domains: professor homepages, camera product, laptop product. From satisfactory results, we want to adapt this approach in Vietnamese real estate domain.

4.3.1 Real estate domain features

Firstly, we have to define features for the domain to support modules in the systems. To do this work, for each constraint, we use desired features described in section 2.3.2. For example, we consider a constraint in real estate domain for the areas field “areas at least 100m²”. In this case, what features should we use, to differentiate relevant from non relevant pages. Firstly, we can use a CSF feature “whether there is a number in the page greater than or equal to 100”. However, there are many pages that satisfy the CSF feature but they are not related to areas. We can also add a REMF feature “there is a term ‘m²’ in the document”. But still, there might be a page about areas 50m² but has also number more than or equal to 100 that appear randomly. Hence, we need to add a RCSF feature that specifies the constraints on proximity between the above two features such as “the number that is >= 100 must appear close term ‘m²’, more specifically, in a window of (-3, 3) distance apart”.

Through investigating real estate web sites in Vietnam, we define 8 constraints for this domain: LOCATION, TYPE, PRICE, BEDROOMS, BATHROOMS, AREAS, DIRECTION. We used a total 20 features as shown in table 5. Those features are divided into 8 subsets corresponding to 8 constraints used for real estate object query.
Table 4. List of features used in real estate domain in Vietnamese

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>For domain constraint</strong></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Has term ‘bán’</td>
</tr>
<tr>
<td>2</td>
<td>Has term ‘nhà’</td>
</tr>
<tr>
<td>3</td>
<td>Has term ‘bđs’</td>
</tr>
<tr>
<td>4</td>
<td>Has phrase ‘diện tích’</td>
</tr>
<tr>
<td>5</td>
<td>Has phrase ‘phòng ngủ’</td>
</tr>
<tr>
<td>6</td>
<td>Has phrase ‘phòng tắm’</td>
</tr>
<tr>
<td>7</td>
<td>Has term ‘giá’</td>
</tr>
<tr>
<td>8</td>
<td>Has phrase ‘đồng sàn’</td>
</tr>
<tr>
<td><strong>For location constraint</strong></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TF of the LOCA constraint in title text</td>
</tr>
<tr>
<td>10</td>
<td>TF of the LOCA constraint in body text</td>
</tr>
<tr>
<td><strong>For type constraint</strong></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Whether or not the TYPE constraint near the term ‘kiêu, loại, bđs’ in window [-5, 5]</td>
</tr>
<tr>
<td>12</td>
<td>Whether or not the phrase TYPE constraint in body text</td>
</tr>
<tr>
<td><strong>For price constraint</strong></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Whether or not the number PRICE constraint near the term ‘giá’ in window [-5, 5]</td>
</tr>
<tr>
<td>14</td>
<td>Whether or not the number PRICE constraint near the term ‘vnd, vnd, đ, d’ in window [-5, 5]</td>
</tr>
<tr>
<td><strong>For bedroom constraint</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Whether or not the number BEDS constraint near the term ‘ngủ’ in window [-5, 5]</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>For bathroom constraint</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Whether or not the number BATHS constraint near the term ‘tắm’ in window [-5, 5]</td>
</tr>
<tr>
<td>17</td>
<td>Whether or not the number BATHS constraint near the phrase ‘vệ sinh’ in window [-5, 5]</td>
</tr>
<tr>
<td>For area constraint</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Whether or not the number AREAS constraint near the phrase ‘diện tích’ or the term ‘dt’ in window [-5, 5]</td>
</tr>
<tr>
<td>19</td>
<td>Whether or not the phrase of number AREAS constraint and term ‘m2’</td>
</tr>
<tr>
<td>For direction constraint</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Whether or not the number FRONT constraint near the term ‘hướng’ in window [-5, 5]</td>
</tr>
</tbody>
</table>

### 4.3.2 Learning with Logistic Regression

With the features shown in table 5, we use Weka machine learning toolkit to compute the weights $w_i$ for each feature by logistic regression. $X$ is generated from the feature list associated with a field and $Y$ is taken after annotating training data.

### 4.4 Chapter summary

This chapter brought an overview of the object search problem in Vietnam, as well as, some websites tried to solve the problem. Bearing in mind the potential of searching for object in Vietnamese domain and good result of implementing the probabilistic framework in English domain [14], we adapted the approach into Vietnamese real estate domain which has structured information.
Chapter 5. Experiment

This chapter brings in-detail description of the probabilistic framework on searching object-oriented on unstructured data of Vietnamese dataset of real estate domain. The section 5.1 presents resource (data, tools) for experiment. The section 5.2 shows the results and evaluation of these results of the experiment. Before summarizing this chapter, we give some discussion for the results.

5.1 Resources

5.1.1 Experimental Data


Table 5. Testing data for real estate domain

<table>
<thead>
<tr>
<th>Domain</th>
<th>Real Estate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Mixture of about 2200 pages from real estate object-oriented web pages and pages from random domains from WebBase</td>
</tr>
<tr>
<td>Document available for training</td>
<td>1000</td>
</tr>
<tr>
<td>Testing document</td>
<td>6000</td>
</tr>
</tbody>
</table>

We index these web pages into basic features (term-based and entity-based) by using Lucene. Term-based features are from text because texts are indexed by terms. The different part of a document form different features such as body, title. Entity-based features are from entities extracted by parsers at index time. In the experiment, we index “Number” feature as entity based. We use Lucene Payload to store feature attributes.

One of the problems when indexing Number is the format of the number in standard and Vietnamese format. For example, “123456.78” in standard format is “123,456.78” which in Vietnamese format is “123.456,78”. We use regular expression for dealing with the problem.
5.1.2 Experimental Tools

We use Lucene (v2.3) to store inverted index and MySQL (v5.1) store training data such as annotations. We use Weka machine learning toolkit to compute the weight $w_j$ for each feature and Apache Tomcat (v5.5) as our web server.

Especially, we use some previous source code of Mr Kim Cuong Pham who is studying at University of Illinois at Urbana-Champaign to build a running system on real estate domain by adding and changing some modules.

In addition, we have built a service called image server to extract images from object web pages based on heuristic and snippet modules for generating snippet of the system.

5.1.3 Prototype System

We run demo on a single PC with the following environment: Intel Pentium ® 2.4 GHz CPU, 512 RAM.

5.2 Results and evaluation

For evaluation, we collected a set of known web pages and tagged each of them with correct object information. We mixed these web pages with random web pages from Web-Based corpus to add more noise and evaluate the system at relatively large scale.

We measure the precision at different levels up to 50 top positions because of the difficulty to measure recall. It is obvious that there are satisfied web pages in the random pages which we add in and it is impossible to label all of them.

We use Average Precision (AP) estimate to evaluate the ranking function to compare our approach and BM25 [30]. Assume that we have 5 object $a, b, c, d, e$ in which $a, b, c$ is precise result and the final ranking of the ranking function is $c, a, d, b, e$. The AP is defined:

$$AP = \frac{\sum_{K=1}^{5} P@K \times I(K)}{\sum_{K=1}^{5} I(K)}$$

In which

$$P@K = \frac{\text{Match@K}}{K} \quad (\text{Match@K} = \text{number of precise object in first K position})$$

$$I(K) = 1 \text{ if object is in position K, inversely } I(K) = 0$$
With above example, $P@1 = 1/1$, $P@2 = 2/2$, $P@3 = 2/3$, $P@4 = 3/4$. Therefore

$$AP = \frac{\frac{1}{1} + \frac{2}{2}}{3} = 0.92$$

In addition, we also use Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) estimate for evaluating the system. MAP is the mean value of AP of m queries [1].

$$MAP = \frac{\sum_{i=1}^{m} AP_i}{m}$$

MRR is a statistic for evaluating any process that produces a list of possible responses to a query, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q [1].

$$MRR = \frac{1}{Q} \sum_{l=1}^{Q} \frac{1}{\text{rank}_l}$$

We tested the search engine with five different queries as shown in table 7. We only test the location in Hanoi, Ho Chi Minh city and the types of them are “chung cư” or “nhà phố” because those are the most popular in our corpus.

Table 6. Real estate queries for testing

<table>
<thead>
<tr>
<th>No</th>
<th>LOCA</th>
<th>TYPE</th>
<th>PRICE</th>
<th>BEDS</th>
<th>BATHS</th>
<th>AREAS</th>
<th>FRONT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Thanh Xuân</td>
<td>Nhà phố</td>
<td>---</td>
<td>[2, 4]</td>
<td>[2, 4]</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2</td>
<td>Hai Bà Trưng</td>
<td>Nhà phố</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>[50,200]</td>
<td>---</td>
</tr>
<tr>
<td>3</td>
<td>Hoàng mai</td>
<td>Nhà phố</td>
<td>[200 tr, 1 ti]</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>4</td>
<td>Q1</td>
<td>Nhà phố</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>[90, 200]</td>
<td>Đồng</td>
</tr>
<tr>
<td>5</td>
<td>Hoàng mai</td>
<td>Nhà phố</td>
<td>[700 tr, 1tì 500 tr]</td>
<td>[1, 5]</td>
<td>[1, 4]</td>
<td>[50,200]</td>
<td>---</td>
</tr>
</tbody>
</table>

The results of above five queries are shown in Figure 17.
Figure 18. Precision for Real Estate Search Engine

The result noted that the top 10 ones for the queries are very precise. However, the lower the rank is, the less precise the result is. When a web page satisfies all of the constraints but with low probability, it is very easy to be ranked lower than the web pages satisfies less the constraints but with high probability, for example query 1 and query 4. To solve the problem, we can use more training data to the system to rank unsatisfied constraint with very low probability and satisfied constraints with high probability, then the later case will be ranked lower than former case.

We also use average precision, MAP, MRR of top 20 results with five above queries to compare object search (OS) and Baseline - BM25.

Table 7. Comparison MAP and MRR of BM25 and OS

<table>
<thead>
<tr>
<th>Estimation</th>
<th>BM25</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>0.54</td>
<td>0.93</td>
</tr>
<tr>
<td>MRR</td>
<td>0.63</td>
<td>1</td>
</tr>
</tbody>
</table>
The result shows how good the probabilistic framework in comparing to baseline approach - BM25 in all tested queries.

5.3 Discussion

In our work, we have examined the precision of the system through some queries with about 6000 testing web pages. The figure 17 showed the precision up to top 50 results. It is realized that the top 10 results are very accurate. However, the precision of result was better at top 20 ones, from 70% up to 85% for query 1 and from 80% up to 95% for query 4. It is reasonable that some web pages satisfied less constraints but high probability would be ranked higher than web pages satisfied more constraints but lower probability. To solve this problem, we can use more training data for learning ranking function. In addition, the precision of top 50 results is not very good, 46% for query 4, 56% for query 3 and 66% for query 5, however, the user usually interests in top 10 or 20 results. Therefore, the top 10 result is more important.

To illustrate how good the probabilistic framework in real estate domain is, we used average precision, MAP, MRR at top 20 results to compare to baseline approach. As shown in figure 18, with five testing queries, the average precision of the probabilistic framework is much higher than baseline approach, increasing 29% for query 1, 59% for query 2, 55% for query 3, 27% for query 4 and 22% for the last query. With the queries having number constraints, the average precision of BM25 is truly lower than our applied framework like query 2 and query 3. From table 18 about
the MAP and MRR of BM25 and the probabilistic approach, we can infer that our applied approach is much better than BM25.

5.4 Chapter summary

This chapter has presented the experiment in real estate domain using 6179 web pages of targeted object pages and random web pages. We then carefully evaluate the precision and average precision of our applied approach through 5 queries. The result has pointed out that the probabilistic framework is a satisfactory approach than previous one.

We then make a comparison between baseline approach and our applied approach. As shown in figure 18, our applied one is much better than baseline especially in queries containing number range constraint.
Chapter 6. Conclusions

Object search is new and potential field for researchers to study and promised to be trend in the development of searching technology in Vietnam. Ranking the results returned from search engine is an important part of this system and has attracted a lot of controversies in information retrieval community lately. The main object of the thesis is implementing the approach based on machine learning for ranking and building the modules to support the system. In the chapter, we summarize and conclude our main contributions as well as the future work in this area.

6.1 Achievements and Remaining Issues

In this thesis, we have brought an overview of object search on the web and investigated some plausible solutions recently. We have also studies a novel machine learning framework, which overcomes the challenges about scalability and adaptability of the previous approach.

We have then adapted the probabilistic framework into a Vietnamese domain - real estate. In practical, the results increased of 39 percent of MAP estimate comparing to baseline approach (BM25). Through experiments, it also indicates that the approach retrieve objects effectively and adapt to new domain easily.

Furthermore, we also propose a method to generate snippet for object search system based on feature language, index, and cache document and integrated successfully into the system.

However, while implementing the framework on real estate domain, we have not yet given the best result for queries containing “PRICE” constraint because of abundant money units in Vietnamese such as “triệu”, “ti”, “vnd”, “liệu”… Furthermore, we have done experiment with quite small data - 6000 URLs - comparing a lot of web pages about real estate.

6.2 Future Work

One of the future works is solving the limit of current system about the queries containing “PRICE” constraint. To do it, we define more features for “PRICE” field such as “Proximity(Number_body(PRICE/1000), Token(triệu, -5, 5)”. As a result, we can query with many money unit as well as other fields.
We also want to examine more in details the performance of the system with larger data. Furthermore, to improve results returned from object search engine, the top 10 ones is better than top 20 ones, we are going to improve learning model for ranking function with more training data.

Moreover, we can obviously group results based on objects which map pages of the same object. It is very helpful for user to make comparison about the information of the object from different pages for their goal. Ideally, we hope to build an object search engine on multiple domains in Vietnam.
Vietnamese References


English References


annual international ACM SIGIR conference on Research and development in information retrieval.


