Random Walk and Web Information Processing for Mobile Devices

Yin Xinyi

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the School of Computing

NATIONAL UNIVERSITY OF SINGAPORE

2006

Abstract

Random walk and web information processing for mobile devices

Yin Xinyi

Accessing web pages from a mobile device is becoming very valuable, especially for people constantly on the move. However, the small screen, limited memory, and the slow wireless connection make the surfing experience on mobile devices unacceptable to most people. In this thesis, we aim to solve three fundamental challenges in the mobile Internet: web page content ranking, web content classification, and web article summarization.

Firstly, most web pages are designed for computer screens which are usually 1024x768 pixels in size, much bigger than the common mobile device screens. It is very difficult to directly render content in a pleasant layout on such small screens of mobile devices. A method to rank content to allow optimization for small screens is necessary for a good viewing experience on the mobile device. Secondly, in one web page, there are often many different categories of content, which makes it hard for the user to find what he needs. A method of web content classification is needed to allow the mobile user to match his instant information needs. Thirdly, even after we have filtered out the useless content in a web page, the main article may still be too lengthy for the mobile device to display. A method of web content summarization is necessary to present the most relevant and important information to the mobile user.

In this thesis, we propose a new method to solve these three fundamental challenges. As a web page is too complex to analyze as a whole, we will first divide the entire web page into basic elements such as text blocks, pictures, etc. Next, based on the relationship between the elements, we will connect the elements with edges to make a graph. Finally, we will use random walk methods to provide solution for the three challenges.

The main contribution of this thesis is a graph and a random walk based framework for the Internet information process. It is shown to be very simple and effective. For example, our experiments of web page ranking show that from randomly selected websites, the system need only deliver 39% of the objects in a web page in order to fulfill 85% of a viewer's desired viewing content. In the experiments of web content classification, the system generates good performance with the F value for main content and advertisement (*A*) as high as 0.93 and 0.82 respectively. In the experiments of text summarization, with the use of the well-accepted dataset for single document summarization, the graph and random walking based text summarization system outperformed the results of all participants of the conference.

Contents

List of Figures iv
List of Tables v

Acknowledgments vi

Introduction 1

1.1 The Motivation 1

1.2 Overview of the Thesis 3
1.2.1 The Methodology 3
1.2.2 The Architecture 4
1.2.3 The Layout of the Thesis 5
1.2.4 Main Contributions 7

Background and Related Work 9

2.1 The graph 9

2.2 The Markov model 12
2.2.1 Markov process 12
2.2.2 Markov Chain 12

2.3 The random walk 15

2.4 Text Summarization 17
2.4.1 Summarization Systems 18

2.5 Related work 21
2.5.1 Web content optimization 21
2.5.2 Random walk 26
2.5.3 Text Summarization 28

Page Optimization with Random Walk 33

3.1 Introduction 33

3.2 Converting a web page into a graph 36
3.2.1 Basic elements 36
3.2.2 Graph in a web page 38
3.2.3 Random walk on the graph 43

3.3 Extracting and Optimizing 45
3.3.1 Extracting relevant elements 45
3.3.2 Optimizing for mobile device 47

3.4 Experiment and analysis 51

3.5 Conclusion 57

Content Classification with Random Walk 59

4.1 Introduction 59

4.2 Functional categories 61

4.3 Building category graphs 65
4.3.1 Category independent graph 66
4.3.2 Content (C) graph 68
4.3.3 Advertisement (A) graph 70
4.3.4 Relate (R) graph 72
4.3.5 Navigation and support (N) graph 73
4.3.6 Form (F) graph 76

4.4 Random walk on the graphs 77

4.5 Experiment result and analysis 78

4.6 Conclusion 83

Text Summarization with Random Walk 84

5.1 Introduction 84

5.2 The graphical models 87
5.2.1 The fully connected graph 87
5.2.2 The backward directed graphical model 89

5.3 The citation graph 91

5.4 Experiment result and analysis 94
5.4.1 The dataset and evaluation package 94
5.4.2 Fully connected graphical model 96
5.4.3 The backward graphical model 97
5.4.4 The citation model 98

5.5 Conclusion 100

Conclusion 101

Appendix 1. DUC and ROUGE bug analysis

Appendix 2. Stop word list

Reference

List of Figures

Figure 1.1: The theme development of the thesis

Figure 2.1: Directed and undirected Graph

Figure 2.2: Status transition in a Markov chain

Figure 3.1: The original website from the www.cnn.com and its corresponding elements structure detected by our algorithm.

Figure 3.2: The original HTML web page and its corresponding layout tree structure of the selected area.

Figure 3.3: The web content with the layout optimization for small screen device

Figure 3.4: Potential error that introduced in the data collection process

Figure 4.1: The original web page on the normal computer browser

Figure 4.2: The Content view on a mobile device.

Figure 4.3: The Related view and the Advertisement view of the original web page

Figure 4.4: The distribution of category element in our dataset

Figure 5.1: The fully connected graph model

Figure 5.1: The process of constructing the citation graph

Figure 5.2: Scheme of the growth of the backward graph model

Figure 5.3: The process of constructing the citation graph

Figure A.1: The extra words and the ROUGE value comparison

List of Tables

Table 3.1: The recall of different random and traction algorithm

Table 4.1: Experiment result with training set for all five category contents

Table 4.2: Experiment result with test set for all five category contents

Table 4.3: The WEKA result with test set for all five category contents

Table 5.1: The baseline performance for DUC 2001 and 2002 using ROUGE 1.5.5.

Table 5.2: Fully connected graph performance for DUC 2001 training set using ROUGE 1.5.5.

Table 5.3: The backward graph performance for DUC 2001 training set using ROUGE 1.5.5.

Table 5.4: The citation model performance on all dataset using ROUGE 1.5.5.

Table 5.5: The performance comparison of all system using ROUGE 1.5.5 on DUC 2002

Table A.1: The performance differentiation of all system using Rouge 1.2.2 and 1.5.5.

Acknowledgments

The last five years is a very important period in my life journey. I was so lucky to be offered a seat in one of the best university in the world, and I was so lucky to be able to work on the research topic that really fascinated me. More importantly, I am so lucky that I was given the support to conducted the research, and have made my small contribution to the human knowledge about the Internet for mobile device. I have learned and experienced so much that I will appreciate forever.

I want to thank my supervisor and mentor Prof. Lee Wee Sun, who has had great impact on me. I'd like to talk about three most important things that I have learnt from him. First, as a great researcher, Prof. Lee set a good example for me. He has great passion and serious attitude about research. Prof Lee believes that in research everything happens for a reason. Good experiment results are not enough; as researchers we must seek the reason behind the results. He insists that every claim or experiment must be verifiable and repeatable. This leaves a great impact on me in my future work. I will follow him to put seriousness, integrity, curiosity, rationale in everything I do. Secondly, as great teacher and research leader, Prof. Lee has a clear vision about strength of limitation of everyone. He helped me improve on my shortcoming, set the achievable objectives at each step and lit up the aspiration in me in my heart. As an engineering background student, he immediately arranged to have me take challenging computer science course in Singapore MIT Alliance; he encouraged me to read all the related fundamental theory in computer science, and encouraged me to aim at Rank 1 conferences. Without his guidance I could not have achieved so much. Thirdly and most importantly, Prof. Lee demonstrates the personality that as a person I want to be. He is very wise and knowledgeable. In the 4 years working under him I have at times been slow to understand theories, or proposed ideas that are obviously wrong. However, Prof. Lee is always and patiently guiding me. When I find my real passion in life he let me pursue my dream; when I encountered problems, he stood out and protected me. Everyone agrees that Prof. Lee Wee Sun is a very nice person. I have a deeper understanding of that, and I want to be like him.

I would like to thank Prof. KAN Min Yen. I have the honor to work under him on the graph based text summarization, which forms the content of Chapter 5. He guided me in different stage for the work and always was available when I needed his advice. He took effort in organizing a discussion group on graph theory, providing me with valuable comments and insight on the research direction. He helped me review this thesis and I thank you so much.

In our research, Prof. Mihalcea and Prof. Chin-Yew Lin provided great support, for that I really appreciate. I also want to thank Cheryl Cheng, Lei Lei, Tan Keshi, Huang Yicheng, Dell Zhang, and my external examiner; you are invaluable in providing comments and suggestions on this thesis, which ranged from linguistic to computational.

On the other side of my life, my family members never ceased to give me the most support in spite of being thousands of miles away. Even though they may know nothing about my research topic, they listen to my explanation of the topic and encourage me to pursue my dream. There are no words to thank them for that.

I consider myself to be a very lucky one. I found love in the early stage of my life, and I have devoted myself heart and soul into this career. I have friends, mentors, and family members to support me. I dedicate this thesis to them.

To Elina, Jiayou

To our parents

Chapter 1

Introduction

1.1 The Motivation

There is now a massive amount of information available in the World Wide Web (WWW). Most of this information, though, is available in a format suitable only for personal computer (PC). However, there are more mobile device users than PC users. It is important that the mobile device users possess the ability to conveniently access this ever-growing information in the web.

Web content is designed mainly for the desktop computer - a normal PC screen has a resolution of 1024*768 pixels, which displays many objects adequately on a single page; secondly, PCs are equipped with useful devices such as a mouse for the user to conveniently interact with any element in the web page; thirdly, PCs are usually connected to the Internet through an inexpensive and high capacity network, thus downloading a content intensive web page is rarely a problem.

In the past five years, mobile devices have become very popular. It is now possible to browse the web using personal digital assistants (PDA) such as a Palm or Pocket PC and even mobile phone. However, compared with the PC, these devices have great constraints for surfing the web. Firstly, the wireless bandwidth is very limited and expensive for the content-intensive web. Secondly, screen resolutions of mobile devices are usually very low (even high-end devices have only 240x320 pixels of resolution), which limits the amount of information that can be displayed in one screen. Thirdly, the limited memory capacity of mobile devices is often not able to hold even a single full-sized web page. Lastly, without convenient input devices, it is often a difficult task to interact with the element in the web page on a mobile device.

Researchers have put in a lot of effort trying to enable such devices to view the web content in a satisfactory manner. To make the mobile Internet acceptable for most people, we need to find solutions to the three fundamental challenges: web page content ranking, web content classification, and web content summarization. Firstly, most web pages are designed for the big computer screen and therefore we need to optimize the layout for small screen devices by ranking and filtering out unimportant content. Secondly, in one web page, there is too much different content which might overload mobile devices, therefore we need to develop web content classification methods that allows a user to quickly match his instant needs. Thirdly, after we have filtered out the useless content in a page, the main content may still be too lengthy for the mobile device; we need excellent text summarization techniques to solve this problem.

1.2 Overview of the Thesis

In this section, we will discuss the methodology of this thesis and the architecture of the proposed system. After that, we will discuss the thesis structure and our main contribution.

1.2.1 The Methodology

In this thesis, we will use the random walk and graph analysis for all the three fundamental challenges in the mobile Internet. We will convert all the problems, for example web page content ranking, web content classification and web content summarization into a graph model and using random walk model to solve it. The first step of our solution to each challenge is to build a graph, and the first step to build a graph is to identify nodes.

In the problem of web content optimization and classification, we notice the web page is too complex and too dynamic to analyze as a whole. However, there are no obvious "nodes" inside a web page, so we divide the whole web into basic elements, for example a text block or a picture, which are much easier to understand. In the task of web text summarization, we divide the text article into sentences, and use sentence as the "node" in the graph.

After we identify the nodes, the second step is to set up the relationship - the "link" - between the nodes and create the graph. Depending on the problem at hand, the link needs to capture the right features and relationship between the nodes. For example, in the text summarization task, the cosine similarity is a good candidate feature to build the link between sentences. For the web content optimization task, we will create directed links pointing towards important nodes.

After the graph is built, the random walk is performed on the graph. We will use the ranking of the elements to solve whatever task at hand, may it be web content ranking, web content classification or the text summarization.

1.2.2 The Architecture

The experience of browsing web pages from mobile devices can be quite unacceptable, as the normal web pages are designed for PCs, which may contain many content objects, require high screen resolution, and consume a large amount of bandwidth. One possible solution is to develop advanced filtering or optimization technology that runs on the mobile devices. The disadvantage is obvious: Firstly it can not solve the bandwidth and downloading time problem. Secondly, it puts very heavy computing burdens on the mobile device.

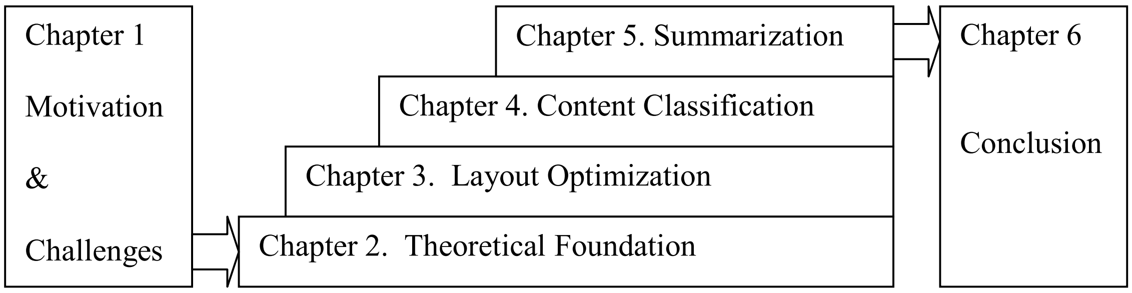
Our research is based on the thin-client computing concept. Instead of directly accessing the Internet, the mobile devices will access the proxy server. Upon receiving the request from mobile devices, the server will retrieve the HTML page and render the page in the memory. Based on the methods proposed later in this thesis, the proxy server may optimize, classify, filter or summarize the content in the webpage, and generate a new webpage that is optimized for the mobile device. The optimized webpage is then sent back to the mobile device to be viewed properly on the small screen.

In our experimental system, optimization of a web page can usually be done within 1 second on a normal Pentium IV computer. We envision that in the future, each person will have his own personal proxy for performing transformations and personalization at home. Since the desktop computer is connected to Internet via cable and only the optimized content is delivered wirelessly, adding the transformation part will not greatly decrease the performance for the user. The second advantage of the system is that user will retain full control of the content he will read on the mobile device. The displayed web page contains a subset of the content in original web page. This system is especially suitable and useful for surfing informative websites such as news sites on the mobile device, as it solves the three fundamental challenges we mentioned earlier, and it greatly improves the effectiveness and efficiency of the mobile Internet.

1.2.3 The Layout of the Thesis

In order to help the readers to better understand the theme development of the thesis, we provide a map as shown in Figure 1.1. A reader without background in this area may want to refer to Chapter 2 for the theoretical foundation and related work, which includes graph theory, Markov process, random walk and text summarization. This thesis focuses on three fundamental challenges of the Internet access on mobile devices. Chapter 3, Chapter 4 and Chapter 5 each target one challenge. They are based on the same theoretical foundation: the graph and random walk.

In Chapter 3, we present a system that provides automatic conversion of web content into an optimized form for mobile devices; it ranks the content by its importance and optimizes its layout for small screens. In Chapter 4, we focus on the second fundamental challenge: how to classify and extract certain type of content in a web page, and deliver only a subset to the mobile device to satisfy the user's particular information needs. In Chapter 5 is based on the assumption that the user wants to read the main content on web page on mobile devices, but the main content may still be too lengthy for the mobile user. We have developed a text summarization system to address this problem.



Chapter 1
Motivation
&
Challenges
Chapter 5. Summarization
Chapter 4. Content Classification
Chapter 3. Layout Optimization
Chapter 2. Theoretical Foundation
Chapter 6
Conclusion

Figure 1.1: The theme development of the thesis

As we can see, the thesis starts with an introduction chapter to discussion our motivation and the research challenges, and end with a conclusion. All the chapters are closely related to each other. From Chapter 2 to Chapter 5, each chapter serves as the foundation of the next chapter as well as the further development of previous one. They are consistent and at the same time independent, researchers can start reading from any chapter.

Chapters 3-5 are closely related and each earlier chapter serves as a building component of later chapter. For example, the optimization system we developed in Chapter 3 can work independently for mobile user. However, it can also be used as a layout optimization component in the system developed in Chapter 4. After the web content is classified into five categories, the web content within a category still requires ranking or optimization in order to be presented nicely on the small screens. In the same manner, the content classification system developed in Chapter 4 can be used as a building block for Chapter 5, as it can be used to extract the main article from the web page as text to be summarized.

1.2.4 Main Contributions

The main contribution of this thesis is a graph and a random walking based framework for Internet information process.

In this thesis, we propose a new framework to solve these three fundamental challenges. We first divide the whole web page into basic elements such as text blocks, pictures, and so on. Then, based on the relationship between the nodes, we connect the elements with edges to make them a graph. Finally, we use random walk to conquer the three challenges.

Our experiment shows that it is simple and effective to solve all the above mentioned three challenges within the graph random walk framework. For example, our experiments of web page optimization show that from randomly selected websites, the system need only deliver 39% of the objects in a web page in order to fulfill 85% of a viewer's desired viewing content. In the experiments of web content classification, the system generates good performance with the F value for main content and advertisement (*A*) as high as 0.93 and 0.82 respectively. In the experiments of text summarization, with the well-accepted dataset for single document summarization, the graph and random walking based text summarization system outperformed baselines of all reported results covered by our survey.

Chapter 2

Background and Related Work

In this chapter we will discuss the theoretical background of this thesis. As we are using random walk and graph as the theoretical foundation in our research, in 2.1 we will first introduce the basic graph theory. Before we introduce the random walk in 2.3, we will first introduce some fundamental concept of Markov chain and Markov process, as random walk can be thought of as Markov chain. In Section 2.4 we will give out background of text summarization, which is the major focus on Chapter 5 alone.

2.1 The graph

Graph theory is a very important theory in the computer science, as we can model many structures and practical problems as graphs to analyze it. For example, we can use the graph to represent a road network and analyze the traffic under different hypothetical conditions. Another example is the Internet; billions of web pages interconnected by links and thus forms probably the biggest and most dynamic graph of the world.

In the road traffic and the Internet example, the edges and nodes are real objects. However, in this thesis, we use the graph to model the problem where the edge and nodes are not obviously available. So we will first create both nodes and the links, then leverage on the graph theory to solve the problem.

According to Paul et al. [86], a graph is defined as an ordered pair *G:=(V, E)*. First, *V* is a set of elements that are nodes or vertices. Second, *E* is a set, whose elements are known as edges.

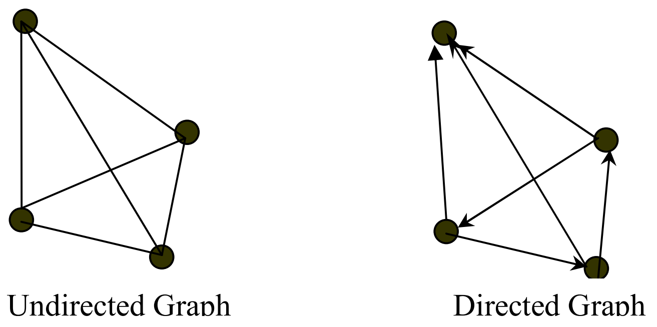


Figure 2.1: Directed and undirected Graph

Graphs can be classified as directed or undirected graphs. For example, cosine similarity graph of the text is not directional, while the graph of web pages is directed as all the links are directional. By definition, a directed graph *G* is an ordered pair *G:=(V, E)* that satisfies the two conditions. Firstly, *V* is a set of the nodes or vertices. Secondly, E is a set of ordered pairs of vertices or directed edges. An edge *E* = *(x, y)* is said to be directed from *x* to *y*, where *x* is the tail of e and *y* is the head of *E*. If the edge from node *A* to node *B* is considered to be the same as the one from *B* to *A*, the graph is undirected.

Random walk on a graph is a very important concept in this thesis - What is a "walk" on the graph? A walk on graph is an alternating sequence of vertices and edges, with each edge being incident to the vertices immediately preceding and succeeding it in the sequence. In Section 2.3 we will discuss random walk in detail.

2.2 The Markov model

Random walk can be looked as a Markov chain. Before we discuss the random walk, we will introduce the Markov process and Markov chain in this section

2.2.1 Markov process

According to Bergner [88], a Markov process is a stochastic process with Markov property: the conditional probability distribution of future states of the process, given the present state and all past states, depends only upon the present state and not on any past states. In another word, Markov process is memory-less, at any time, the status of next observation only depends probabilistically on the current status in sequence transitions.

The Markov process can be finite or infinite state space, continuous or discrete time horizons, and homogeneous (with constant one-step transition probabilities) or non-homogeneous (with time-varying one-step transition probabilities). In our research we are interested in one type of Markov process, called Markov Chain.

2.2.2 Markov Chain

The Markov chain is a discrete-time Markov process. It has three characters: Firstly, the process has finite states, which means at any given time the process will be in one of the N possible states, and N is a finite number. Secondly, the change of the status happens in a discrete time unit, it takes the same unit to switch from one state to another.

A Markov chain describes at successive times the states of a system. At these times the system may have transited to another or stay unchanged. A Markov chain is a sequence

$S_0, S_1, ..., S_n$ ... of random variables with the Markov property: the conditional probability distribution of the next future state S_{n+1} given the present and past states is a function of the present state S_n alone, The range of the variables, i.e., the set of their possible values, is called the state space, the value of S_n being the state of the process at time n.

We can visualize the Markov chain as a finite state machine.

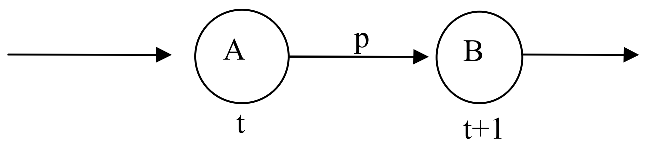


Figure 2.2: Status transition in a Markov chain

As shown in Figure 2.2, the system is at state *A* at time *t*, the probability *p* that it will move from state *A* to state *B* at time *t + 1* does not depend on *t*, and only depends on the current state *A*. A finite Markov chain can be characterized by a matrix of probabilities between states which never change. Such discrete finite Markov chains can also be represented as a directed graph. Most of the graphs that we will create in Chapters 3, 4 and 5 are directed graphs, where the transition probability distribution can be represented by a matrix, called the transition matrix, with the *(i, j)*'th element equal to

$$P_{ij} = \Pr(S_{t+1} = j \mid S_t = i) \qquad (1)$$

Inspired by (1), we are going to create a graph by linking up every pair of nodes in the graph with a probability distribution P_{ij} - given that we are at node *i* at time *t*, what is the probability to be at node *j* at time *t+1*. We can model a real problem into this Markov chain problem. A Markov chain is characterized by the conditional distribution which is called the transition probability of the process. This is sometimes called the "one-step" transition probability. The probability of a transition in two, three, or more steps is derived from the one-step transition probability and the Markov property. For a discrete state space, the *k*-step transition probability can be computed as the *k'th* power of the transition matrix. That is, if *P* is the one-step transition matrix, then P^k is the transition matrix for the *k*-step transition.

2.3 The random walk

In our research, we use random walk as a foundation to process the information on the web for mobile devices. The random walk theory is derived from the real world phenomenon. Researchers have used this theory to study, explain and simulate random events. For example, the random thermal perturbations in a liquid, known as the Brownian motion, are a random walk phenomenon. Web researchers also use random walk to approximate index quality. For the Web, a natural way to move between states is to follow a hyperlink from one page to another.

By definition, a random process consists of a sequence of discrete steps. Let $X = \{S_0, S_1, ..., S_N\}$ be a set of states. A random walk on X corresponds to a sequence of states. At each step, the walk switches from its current state to another or remains unchanged. If we put a walker onto one node in the graph, and let the walker take random direction based on probability to move from one node to another through the edges in the graph, it becomes a random walk on the graph. If we further assume the walker doesn't have any memory, in each step the walker will randomly pick an edge that links to other nodes and walk through with certain probability. In this way random walk satisfy all the property of a Markov chain, and we can use Markov chain property to analyze the random walk.

In our research, we are going to design graphs and random walk on it, one very important question we need to understand is whether the random walk will converge to the stationary distribution on the graph. If the random walk does not converge, we will not be able to generate any meaningful result out of the random walk, as at each step of the random walk, the graph will present a completely different status, and it will never end.

The random walk we perform on the graph is a Markov chain, so whether the random walk will converge depends on the Markov chain property. The Markov chain will converge if it is both “Irreducible” and “Aperiodic”. Irreducible means that every state is accessible from every other state. A process is periodic if there is at least one state to which the process will continually return with a fixed time period (greater than one). Aperiodic means that there is no such state. What properties of a graph that will determine the convergence of the random walk on it? A graph G is strongly connected if for every *u* and *v* in *V (G)* there exist paths in *G* from *u* to v and from *v* to *u*. The random walk on a directed graph *G* converges to a unique stationary distribution if *G* is strongly connected and is not periodic. One way to ensure the convergence to a stationary distribution is to include an additional source node that is connected to and from every other node in the graph. A Markov chain on such a graph is guaranteed to be both irreducible and aperiodic.

2.4 Text Summarization

A text summarization system will automatically process an article, and generate a shortened version of the original text as summary. The summary keeps as much useful information as possible, while keeping the length as short as possible.

Depending on how the summary is generated, the automatic summarization system can be classified into two categories: the abstraction summary and the extractive summary. In an extraction based system, original sentences are picked from the article to make up the summary. In an abstraction based system, summarization is formed by synthesizing new sentences representing the information in the documents. The quality and the readability of abstractive summary depend on the sentence synthesis.

If we compare the extractive and abstractive summarization system, we will find the extractive method is normally preferred. Today, we are still unable to generate sentences that are readable like the human language. Users may have difficulty in understanding the abstractive summary itself. However, for extractive summary, firstly, it presents the information as-is by the author, as any modification of the text would probably lead to something different. Secondly, extractive summaries are normally easier to understand. So we will choose extractive summary, especially for the mobile Internet. Mobile users would prefer to instantly understand the each sentence in the small screen.

2.4.1 Summarization Systems

The text summarization research started in the 1950's. Many different types of summarization systems have been built. In this section we are going to give an overview of three main different methods: Feature-based summary, machine learning-based methods and graph-based methods.

Feature-based Extractive Summary

The earlier extractive summarization system chooses summary sentences based on features. For example:

- **Cue words**: Those sentences with phrases like "conclusion", "significantly", "most importantly" are rewarded with higher weight.
- **Key words**: Sentences with statistically significant words are given higher score, the key words can be identified by high *TF-IDF* score
- **Title words**: Sentences containing non-trivial title words are considered important; title normally represents the main theme of the article.
- **Location**: For news article, normally the first few sentences or paragraph of an article is more important.

The key idea is to analyze the manually generated abstracts, and specify characteristics that we want in automatically generated summaries. Then we score the sentences based on the features and pick the top sentences. It is a simple and effective method; however, some times it might over-fit to a specific domain, and may not be a good choice for the open domain summarization problem like the whole Internet.

Machine Learning Methods

In the feature-based summarization system, we generate a feature vector for each sentence. Based on certain mathematical calculation or logical decision, it is either selected or not selected as summary. In this way, the machine learning framework is introduced to solve the extractive summarization problem.

Machine learning framework is introduced to the area of text summarization for two purposes. Firstly, as most of the existing summarization is domain specific, machine learning will make text summarizer adaptable to new domain. It adds intelligence and flexibility to traditional methods. Secondly, the machine learning method also allows us to create a summarization system based on existing sample summaries. The system will be able to generate summarization that will satisfy specific needs.

.

For machine learning methods, a set of training document and their extractive summaries is provided. For each sentence, we will generate a feature vector. Depending on whether the sentence is selected in the summary or, we will assign the vector a value *zero* or *one*. In this way, the summarization problem is converted to a typical binary classification problem: sentences are classified as summary sentences and non-summary sentences based on the feature vector. Classification methods can then be used to solve this problem.

Graph-based extractive summary

A text article can be represented as a graph in two simple steps. Firstly, we decompose the article into elements like sentences. Each sentence in the documents will be represented as one node in the final graph. Secondly, we will connect each two sentences with an edge. The weight of the edge is decided by the relationship between the sentences. For example, cosine similarity can be used to link up related sentences.

After graph representation is built for an article, we can use the graph technique to identify the central sentences in an article. The extractive summarization can be viewed as a process of choosing a subset of central nodes in the graph representation of an article. If we use the cosine similarity to build the graph, the centrality of a sentence is defined in terms of the centrality of the words it contains within the main topic. However, if the article has multiple threads or topics, its graph will be made up of a few well connected sub-graphs. For generic summaries, the central sentence of each sub-graph will be chosen as representative sentences.

2.5 Related work

Our research is related to three main research areas. Firstly, it is related to research in the web content processing and optimization for small screen devices. In Section 2.5.1, we will survey past research on web content optimization. Secondly, as described in the title of this thesis, we will cover the topic of the application of random walk and graph in Section 2.5.2. Thirdly, the challenge we are going to solve is part of a text summarization, and we will introduce the related work of the text summarization research in Section 2.5.3.

2.5.1 Web content optimization

People want to access the Internet from their small screen mobile devices. Optimizing web page layout and improving the surfing experience on the small screen devices has become a very important research topic. Many interesting systems have been proposed in this area - of all the papers, the Digestor system [9] is one of the first works that explicitly mentions device independence; it automatically transforms arbitrary documents from the web to display them appropriately on small screen mobile devices. The PowerBrowser [7] uses a proxy filter to modify HTML pages into a special format to improve information retrieval on PDA. Other papers [8, 10, 13, 16, 18] also provide similar systems for mobile Internet access.

There are many different angles that researchers have taken, and we have identified four main methods: web partitioning, web content ranking, web content transformation and web content classification. The four research directions actually interrelated to each other; in this section we will introduce the related work of each category.

2.5.1.1 Web content partitioning

In most cases, a single web page can be divided into many blocks. Each block contains different information. It brings great challenges to the information processing task on the web. Many researchers have proposed methods to analyze blocks. Yang et al. [2] proposed a novel approach to automatically analyze the semantic structure of HTML pages based on detecting visual similarities of content objects; Yu et al. [3] also proposed another vision based segmentation algorithm to detect the semantic content structure in a web page, and partition the web page to improve pseudo-relevance feedback in web information retrieval. The same group of researcher, Cai et al. [21], also proposed block level link analysis rather than the normal page level link analysis. The author is able to construct a semantic graph over the WWW such that each node exactly represents a single semantic topic. However, existing research on web partitioning aims to improve the performance of web information retrieval task, while our search study have a different goal, partitioning the web for the mobile device.

Besides the normal web information analysis tasks, page partitioning can also be used to facilitate the mobile Internet. Since mobile devices normally have smaller screens, segmentation of the web pages into blocks will be more suitable for mobile devices - many research efforts have been conducted in this direction. Kaasinen et al. [48] applied page partitioning to convert the web page to fit the 'cards' metaphor of mobile devices.

Under the same "divide and view" concept, the SmartView system [11] partitioned the HTML document content into logical sections that can be further selected by the user and viewed independently from the rest of the document. Gu et al. [4] also split the web into small and logically related units for the mobile device. The advantage of these methods is that it allows the user to randomly access any website and gives the user full control over the content to be displayed without predefining a "hot area". However, the method has its limitation, as it does not handle the situation when a logical section is much bigger than the screen size of the target device. Nie et al. [25] introduces PopRank, a domain-independent model to rank the objects within a specific domain. Other papers [5, 12, 13, 25] also provide similar web content partitioning. However most of existing research focus on the layout optimization while our system further consider the importance of each individual element in the web page.

2.5.1.2 Web content transformation

Beyond dividing web pages into block by visual clues, researchers further propose to analyze the blocks. Song et al. [28] proposed to rank the importance of the blocks in the web page. It extracts spatial features (position and size) as well as content features (number of image and links) of the blocks to form feature vectors. A machine learning algorithm is used to train for block importance. The "divide and rank" methodology is very similar to our research in Chapter 3, where we divide the web pages into basic elements, rather than "blocks", and use a graph and random walk method to rank and optimize the layout for mobile device. Firstly, both are trying to understand the role of each part after partitioning the page. Secondly, the two works solve the problem at a different granularity. The paper [28] has solved the problem from a block level; consequently, how to define the block becomes a very subjective problem that will affect the accuracy. In our work, we avoid manually picking features or defining rules for identifying blocks in web page. Our definition of the element is fixed at the level of indivisible elements.

The web content transmission can be personalized. For example, web Clipping [17], AvantGo [16], i-Mode [79] allows the user to select the favorite content channels for mobile device, and the information in each channel is specially prepared by limited web sites. However, the disadvantage of this method is very obvious, as it fails to provide a systematically way to automatically convert existing web page for mobile devices. In many cases, the mobile content can be manually generated separately but that limits the mobile user to surf only a small subset of the Internet. To overcome this limit, Bickmore et al. [45] provided the design of a system that re-renders web pages through a series of transformations, adapting the original web content for small screen devices.

2.5.1.3 Web content classification:

Web content classification and understanding can greatly improve the web surfing experience. For example, Billsus et al. [19] trained a Naive-Bayes classifier to recommend news stories to a user, using a Boolean feature vector representation of the candidate articles. Chen et al. [24] presented a function-based Object Model (FOM) that attempts to understand authors' intention by identifying object function instead of semantic understanding. This technique can also be used in the mobile access of the Internet. As we will explain in the Chapter 4, every element in a website serves as certain functions (for example main content or navigation links), which reflects the author's intention towards the purpose. Chen et al. [24] provided an automatic approach to detect the functional properties and category of object. As an example of the application they built a system for web content adaptation over Wireless Application Protocol (WAP) for mobile devices. However, there are limitations in this work. First, the selection of the functional categories like "Decoration Object", "Special Function Object" might not be directly meaningful for mobile devices. Secondly, its rule-based classification method may work well on some web sites, but may not be adaptable to a wide variety of web pages on the Internet. In our thesis, we define five basic functions and deployed a proxy-like system to classify the objects in the web page and generate a new content for the mobile devices.

The M-Links system proposed by Hilbert et al. [6] provided user interface that separates the integrated surfing activities on the computer into two modes: navigating and acting on web content. The acting on web content includes reading, printing etc., like in the system proposed in Chapter 4 where we divide a web page into five categories, and expect the user to have different action on each "view", to read the main content, to explore related links, to navigate, to submit forms, and avoid advertisements. The system gives the user a simpler surfing experience on the mobile device.

Shih et al. [26] proposed an interesting algorithm for automatic classification tasks such as content recommendation and advertisement blocking. The URL property and the visual placement on a referring page are used as features to train the machine learning algorithm to classify the elements in the web pages. An earlier paper of ours [33] also used machine learning methods to classify the elements. In Chapter 4, rather than machine learning, we use random walk models. The paper [14, 15] provides research in this area.

2.5.2 Random walk

The theoretical foundation of this thesis is random walk on graphs, which is a very active research area for the Internet as the whole web can be modeled as a graph. Brin and Page [1, 30] proposed the most successful ranking algorithm based on the random walk model for web. In this model, the whole web is treated as a graph of web pages connected by links. It assumes the Internet users start from a random web page, moving randomly from page to page by following the links. Each of the random walkers will follow the links until he gets bored. The probability of a user visiting a web page is proportional to the "PageRank", which can be calculated iteratively by

$$PR^t(i) = (1-d) + d \sum_{(j,i)\in E} PR^{t-1}(j) / C(j)$$

where $PR^t(i)$ is the "PageRank" of node *i* at time *t*, E is the set of edges, C(i) is the number of links going out of page *i* and *(1-d)* is the probability that the user will get bored and leave a certain web page back to the source.

The Google system [1] uses “PageRank” to describe the importance or quality of a single web page. We believe people read web page in a similar manner. The reader enters the page through a link and is drawn to elements that are related to the anchor text in the link and are located in central positions on the page. After reading an element, the reader moves on to another highly related element. Google returns the search result ranked by the page rank, while we rank the elements in a web page and return the top content for the mobile device. Deng et al. [21] proposed topic distillation, which is the process of finding authoritative web pages that are relevant to a given query. These pages are called the “hubs” by the author. This is fairly related to our research, as we are trying to find the “hub” of the topic within one single web page from an anchor text, using a similar algorithm.

Inspired by Google’s “PageRank”[1] and HITS (hubs and authorities)[56] algorithms for search, researchers have proposed structural re-ranking approach to many natural language processing (NLP) tasks. Because the obvious link doesn't exist in a typical NLP problem, it is created based on the relationship between elements. For example Kurland et al. [49] introduced “PageRank without link” concept, where the links are generated by exploiting asymmetric relationships between documents, which forms a graph, thus the centrality of a document is calculated and integrated into standard language model based retrieval. The paper [58, 59, 77] also provide research in this area.

Mihalcea et al. [22, 27, 37] proposed "Text Rank", a graph-based ranking model for natural language processing teaks. It is generalized into four steps: firstly, depending on the actual task, the right text units is chosen as vertices in the graph. For example for text summarization, the text unit is the sentence. Secondly, the relationship between the text units is set up as edges between vertices, for example, the cosine similarity between text units. Thirdly, graph-based ranking algorithm is introduced to rank the text units, for example, the random walk. Finally, based on the final score, certain text units are selected as the result. Based on the similar concept Erkan et al. [51] proposed "LexRank", another graph-based centrality scoring of sentences for summarization. It constructed the intra-sentence cosine similarity, then computed the centrality of the sentences in this similarity graphs. Erkan shows that "LexRank" with threshold method outperformed other systems participating in Document Understanding Conference (DUC) [34]. Florian [87] presents a set of discourse structure relations that are easy to code and to develop criteria for an appropriate data structure for representing these relations. Researchers have proven that the graph-based method can be very useful in the natural language processing tasks. In Chapter 5, we will propose a new citation model for text summarization which further improves the performance of text summarization.

2.5.3 Text Summarization

Automatic text summarization systems take an article as input, and produce a compressed version which provides useful information as a summary. It is a research topic in natural language processing that has been developed since the 1960's. Luhn [39], Edmundson[41] and Pollock et al. [42] proposed the classical approaches to text summarization. After more than 40 years' development, text summarization system has matured and has wide applications. For example, the government and hospital have huge amount of text to be process manually, the automatic text summarization can help to increase the efficiency.

Text summarization can be classified into two categories: The extractive summarization produces summaries by choosing sentences in the original article, while the abstractive summarization rephrases the information in the text. Early research on extractive summarization [39, 41, 80] is based on features of the sentences. For example position, the term frequency, key indicative phases or the Inverse Document Frequency (IDF). Researchers have proposed many advanced feature and techniques for summarization. For example, Barzilay et al. [60] and McKeown et al. [70] proposed using relations between sentences as features. Lin [68], Mani et al. [69], Radev et al. [73], Zha et al. [83] integrated machine learning into summarization as more features have been proposed and more training data is available. In Chapter 5, we will propose a graph and random walk based text summarization system. However, the first graph based summarization is introduced by Salton et al. [75], which uses degree centrality in single document text summarization. Other systems [22, 27, 37, 51, 83] also use the graph-based text summarization.

Summarization system has been proposed for mobile devices, since mobile devices normally have small screens which can display only small amounts of text in each screen. Researchers have proposed to use text summarization to improve the surfing experience of mobile device. Buyukkokten et al. [7] proposed the "PowerBrowser" system which is a proxy-based architecture that fetches web pages on the mobile devices' behalf and dynamically generates summary views. The summaries represent both the link structure and contents of web pages. Buyukkokten et al. [38] introduced a system for summarizing parts of web pages on mobile devices. In the system, a web page is broken into text units that can each be hidden, partially displayed, fully visible or summarized. Adam et al. [46] proposed "Ocelot", a system for summarizing web pages. The system synthesizes summaries rather than extracting representative sentences from text. All these research have demonstrated that summarization system can be very useful for the mobile device.

How do we evaluate the summarization systems? According to Goldstein et al. [44], human judges normally do not agree with each other if they are asked to pick sentences in the article as the summary, and there might be very little overlap of the sentences picked by different persons. From this, we can conclude that text summarization is very subjective task, and is difficult to evaluate. The commonly employed metric for extractive summaries is that proposed by Edmundson [41], automatically generated summaries are evaluated by computing the overlapping sentences with sentences picked by human judges as summary. Lin et al. [35] found that unigram co-occurrence statistics is a good automatic scoring metric, as it consistently correlated highly with human assessments. In our thesis, the evaluation is using the well-accepted evaluation toolkit ROUGE, which is a method based on n-gram statistics [35]. Two manually produced reference summaries are provided in Document Understanding Conference 2001 and 2002 [34] dataset, and used in the evaluation process.

Although text summarization has been studied for over 40 year, summarization for mobile Internet is still very challenging. Since the user might click on links to any article, the system needs to process the web page and generate a summary without pre-knowledge about the article. It must be flexible and adaptable to different type of content and information needs. Not all the automatic summarization methods are suitable for mobile Internet.

Firstly, the feature-based extractive summary is not a good option. It works for a specific document set where we can generalize common rules or features. In the Internet, articles are written by different authors and cover almost every topic. It will be almost impossible to find a feature that is well followed across all web content. The feature-based system tries to achieve the best over all performance for a group of articles, and it may not be optimal for a single document. In the mobile internet, we need a "best result" for each article.

Secondly, the machine learning approach is not a suitable method for mobile Internet summary because its performance depends mainly on the quality of the training set, as well as the consistency between the training set and real article. However, there are billions of articles on the Internet. It is almost impossible for us to collect a good training set for every type of article.

The graph-based method is the most suitable approach to summarize the Internet content form mobile devices. Firstly, the graph is generated to capture the relationship between the sentences rather than the actual features each sentence possesses. In this way, the graph based method is adaptive to articles that are written by different authors with various styles. It is an excellent tool to model the internal structure. It is able to achieve optimization for each article rather for the whole document set. Secondly, graph model can be build without a training document set. We can use it to model an unseen new article that the mobile users happen to click through. We can use graph-based methods to focus at the right level of granularity, and balance between the content coverage and the details.

Among all the different summarization methods, the graph-based method is most suitable for mobile Internet information access. In the coming chapters, we will focus on modeling the text with a graph model.

Chapter 3

Page Optimization with Random Walk

3.1 Introduction

In this chapter, we focus on using random walk to solve the first fundamental challenge in the mobile Internet: by using random walk on the elements in a web page, we can rank the elements and optimize the page layout for small screen device.

This chapter presents a system that automatic converts web content into a form that is optimized for mobile devices. Our approach is to extract only the important parts from a web page, and optimize the layout for delivery to the mobile device. Such a method saves not only download time but also the effort of scrolling on the small screen devices. Errors in extraction by the system can be corrected by allowing the user to request the whole page if they are not satisfied with the extracted content. If the extraction error can be kept at a minimal level, such a system will provide a more pleasant experience for surfing the web on a mobile device.

The basic technology behind the approach is a random walk based ranking algorithm for elements inside a web page. The idea is to use the ranking algorithm to first represent a web page as a graph model and then exploiting the graph structure to rank the elements. To build the graph, we first divide the page into inseparable basic elements. Based on the features like type, size, physical position shape, we give each basic element an initial rank value. We use weighted edges to represent relationships between two basic elements. The weights are a function of attributes of the two elements, such as word similarity and physical proximity of the elements within the page.

Our methods is based on our assumption how a person reads a web page. We assume the reader enters the page through a link and the reader's attention is drawn to elements that are related to the anchor text in the link and are located in central positions on the page. After reading an element, the reader's attention moves on to a highly related element. By modeling the strength of connections between elements according to their similarity, we are using a simplified model of the movement of the readers' attention on the web page. We then rank the elements according to the expected number of readers reading the particular element at any time. We believe this model reflects the actual behavior of most people's attention. The probability that our attention will stay at a give object in a web page at given moment is proportional to the importance of this objects. If we rank the objects and returns only the top objects for the small screen device, it is more likely we can satisfy the user's information needs. Based on the rankings, we select a rectangle covering all the important elements of the web page and transmit the content of the rectangle.

The graph model of a single web page is made up of hundreds of basic elements that are linked to each other in a very complex manner. It is different from the commonly used tree-based analysis of web pages. It is predominantly semantic-based instead of syntax-based. Such structure is similar to the whole Internet, which is also made up of many interrelated web pages.

We organize the chapter in the following way. In Section 3.2, we will discuss how to convert a HTML web page into graph. In Section 3.3, we will give the design of the extraction and ranking system. In Section 3.4 we will discuss the dataset, and describe the evaluation of the system. In Section 3.5, we will give our conclusion.

3.2 Converting a web page into a graph

3.2.1 Basic elements

To construct a graph from a web page, we first identify the nodes, which are the basic elements in the web page. We then specify the edges of the graph which encode the relationships between pairs of basic elements.

A web page is normally made up of hundreds of elements. We want to develop a consistent method for dividing a web page into basic elements. There are several requirements for the division process: firstly, the method should be adaptable to web page from different sites. Secondly, the granularity of the partition should be independent from the author, the reader and the content in the webpage, so that the data will present a consistent pattern.

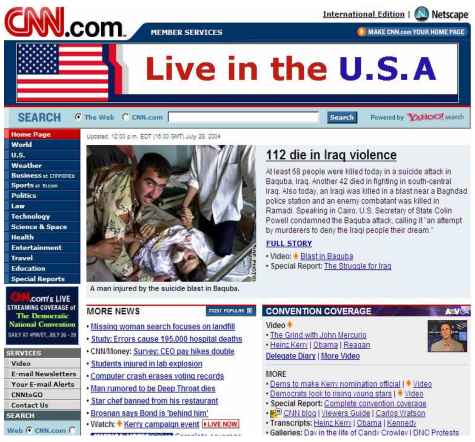
Researchers have proposed different methods to divide an HTML page into logical blocks. For example, Gu et al. [5] proposed a visual-based method to analyze the structure of a web page, and Yang et al. [2] provided a method to automatically understand the semantic structure of HTML pages based on detecting visual similarities of content objects. However, these methods are not suitable for our application because the granularity is subjective and affected by the semantic meaning of the content. We use the algorithm similar to the system [17] that uses the DOM interface provided by the web browser to divide the web page into non-overlapping visible elements.

In our system, we use simpler objects as nodes in the graph. DOM interface of the web browser provides the list of all elements in web page, for example, a picture, a link, a text graph. However we cannot directly use the DOM element as the node for our graph, because DOM will return overlapping elements. For example, a web page may contain a paragraph of text, and inside the text paragraph there is a link. The DOM interface will return two elements for this text paragraph: both the text paragraph and the link will be identified as DOM element separately and the physical location of the two elements overlaps. In our example we need all the non-overlapping visible elements in an HTML page. From bottom up we identify such nodes by using two simple rules:

1. A visible object like an image, link or text paragraph will be a basic element if it is not overlapping with another child or its parent node.
2. For overlapping objects, the minimal container of the two objects will be a potential element to be verified by the rule 1. The algorithm will seek from bottom up to locate the nearest common container, and the container will be treated as one node.

For example, a web page may contain many links that are not overlapping with each other. Each of the links will be treated as basic element. Another web page may have a text paragraph with a link. Here we have two overlapping objects. The bigger one, the text paragraph, will be chosen to be checked by rule 1. If the text paragraph is not overlapping other elements at a higher level, it will be chosen. Otherwise we will recursively search upward. In this manner, all the visible objects in a web page will be elements allocated to nodes in the graph.

Figure 3.1 shows how a CNN web page will be divided.



Updated: 12:00 p.m. EDT (16:00 GMT) July 28, 2004

112 die in Iraq violence

At least 68 people were killed today in a suicide attack in Baquba, Iraq. Another 42 died in fighting in south-central Iraq. Also today, an Iraqi was killed in a blast near a Baghdad police station and an enemy combatant was killed in Ramadi. Speaking in Cairo, U.S. Secretary of State Colin Powell condemned the Baquba attack, calling it "an attempt by murderers to deny the Iraqi people their dream."

FULL STORY

- Video: Blast in Baquba
- Special Report: The Struggle for Iraq

A man injured by the suicide blast in Baquba.

MORE NEWS

- Missing woman search focuses on landfill
- Study: Errors cause 195,000 hospital deaths
- CNN/Money: Survey: CEO pay hikes double
- Students injured in lab explosion
- Computer crash erases voting records
- Man rumored to be Deep Throat dies
- Star chef banned from his restaurant
- Brosnan says Bond is 'behind him'
- Watch: Kerry campaign event

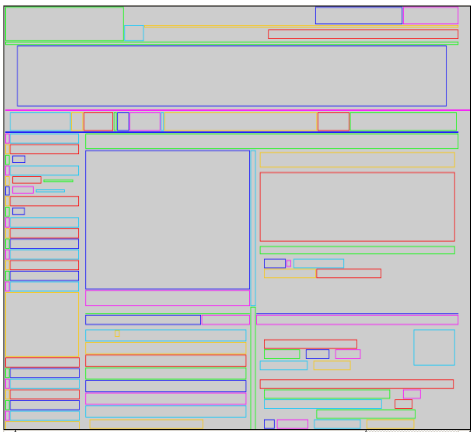
CONVENTION COVERAGE

Video
- The Grind with John Mercurio
- Heinz Kerry | Obama | Reagan

Delegate Diary | More Video

MORE
- Dems to make Kerry nomination official | Video
- Democrats look to rising young stars | Video
- Special Report: Complete convention coverage
- CNN blog | Viewers Guide | Carlos Watson
- Transcripts: Heinz Kerry | Obama | Kennedy
- Galleries: Day in the life of Candy Crowley | DNC Protests

Homepage of www.cnn.com.



Output from our algorithm.

Figure 3.1: The original website from the www.cnn.com and its corresponding elements structure detected by our algorithm. The original complex web page (left) is decomposed into non-overlapping small elements, which are represented by the rectangles in the right image.

3.2.2 Graph in a web page

Assume that we have N basic elements, and we will build a graph such that the sum of the weights coming out of each node is one. This allows us to use the weight on the edge as the probability of a reader going to the next element along that edge.

We first introduce an additional node *S* which can be considered as source of visitors to the web page. This node also serves as the sink where readers who stop reading at any particular element will go to.

Based on the features of a basic element, we will connect it to the source S with a weight that represents its contribution to the topic of a web page. We take the following features into consideration:

1. Size (*Ps*): An element with bigger size is more important than a smaller one. The contribution of size to the importance of element *i* can be calculated by

$$Ps(i) = Size(i) / \sum_{j=1}^{N} Size(j)$$

   The size is measured by the number of pixels.

2. Text length (*Pt*): Element with longer visible text has higher importance. The length is the number of visible words. For example, a link with longer anchor text will draw more attention. The contribution of text length can be calculated by:

$$Pt(i) = Length(i) / \sum_{j=1}^{N} Length(j)$$

3. Match (Pm): Visitors from the source S will pay more attention to the content that is similar to S. We calculate the cosine similarity between the visible text in the element and the anchor text of S. Through out this thesis we will use the cosine similarity definition from Sam [89]: cosine similarity is a common vector based similarity measure whereby the string is transformed into vector space so that the Euclidean cosine rule can be used to determine similarity.

$$Cos(q,r) = \sum_y q(y)r(y) / \sqrt{\sum_y q(y)^2 \sum_y r(y)^2}$$

We use a stop word list including non-informative words that are commonly used in the internet context such as “click” “next” “more” “read” and others.

4. Width/height ratio (*Pr*): The shape of an element reflects its importance. For example, for an image, a regular image is usually more important than an irregular image. We use the following formula to calculate the value for images:

$$\Pr(i) = \begin{cases} 1 & if\ 0.3 < Width(i) / Height(i) < 3 \\ 0 & otherwise \end{cases}$$

For different categories of elements we will use different formulas. For a text block we use a formula that favors higher Width/Height value:

$$\Pr(i) = \begin{cases} 1 & if\ Width(i) / Height(i) > 4 \\ 0.5 & otherwise \end{cases}$$

In both formulas the parameters are hand picked by the author and further adjusted through the experiment with the training set.

5. Physical offset (*Pp*): Physical position is calculated by pixels. This is actually a very important feature. Element closer to the center point is more important than those near the edge of the page. We calculate the position information from, first, the physical distance between the center of the element and the center of the screen and second its horizontal offset information. Let the screen center be (*Xc, Yc*) and the center of element *i* be (*Xi, Yi*). Then

$$dis(i) = 1 - \frac{\sqrt{(Xi - Xc)^2 + (Yi - Yc)^2}}{\sqrt{4Xc^2 + 4Yc^2}}$$

$$offsetX = 2 * Xi / ScreenWidt\ h$$

$$offsetX\ (i) = \begin{cases} OffsetX & if\ 0.3 < offsetX < 0.7 \\ 0 \quad else & \end{cases}$$

$$Pp(i) = offsetX\ (i) + dis(i)$$

In this formula, the factors for example, 0.3 and 0.7 are set empirically and adjusted through our experiment. We specially take *X* offset into consideration because normally a web designer will put the important content in the center of the screen, while both left and right sides are for irrelevant or less important content. However, we did not make use of the vertical offset, as we see that important content in a web page can be very long.

We normalize the distance with the diagonal of the whole HTML page. All the constant values in the formula are chosen according to an ordinary web page layout.

The weight from the source S to node i is calculated by the function $W(S,i)$ where

$$W(S,i) = I(S,i) / \sum_{j=1}^{N} I(S,j)$$

and

$$\mathrm{I(S,i)} = w_1 * \mathrm{Ps(i)} + w_2 * \mathrm{Pt(i)} + w_3 * \mathrm{Pm(i)} + w_4 * \mathrm{Pr(i)} + w_5 * \mathrm{Pp(i)}$$

The weight represents the probability that the reader's eye goes to each of the element as he enters the web page; it is obvious that not all the elements are equally likely to be viewed.

From each node i, we also set a weight *W(i, S)=β*, $0 \leq \beta \leq 1$ to indicate the probability that the person stops reading at node i and goes back to the source node S. The parameter β is a damping factor which can be set between 0 and 1. Later in our experiment with training set, we tried several different β, and eventually we fix β 0.25 in all our experiments.

As described earlier, the weight of the edge between any two basic elements is an evaluation of how likely the reader is to continue with the second element after reading the first. It is calculated using the following features:

1. Distance *Pd(i, j):* The physical distance (in pixels) of two elements in the layout of an html page.

$$Pd(i,j) = 1 - \frac{\sqrt{(Xi - Xj)^2 + (Yi - Yj)^2}}{\sqrt{4Xc^2 + 4Yc^2}}$$

2. Horizontal offset (*Ph*): set as one if two element's horizontal offset is the same, otherwise zero.
3. Neighborhood (*Pn*): Set as one if two elements are neighbors, otherwise zero.
4. Match (Pm): the cosine similarity between the visible texts in the two elements.
5. Width *(Pw)*: Set as one if two elements have the same width, otherwise zero.

The similarity *S(i,j)* between distinct elements *i* and *j* is calculated by the sum of the five features. For a node *i*, we have already used up a weight of β for the link back to the source. Of the remaining amount *(1-β),* we use a fraction α as the loop back to itself to indicate that the user continues reading on the element for a period. Hence $W(i,i)=(1-\beta)\alpha$. The weight from distinct nodes *i* to *j* is then calculated as

$$W(i,j) = (1-\beta)(1-\alpha)S(i,j) / \sum_{k=1}^{N} S(i,k).$$

3.2.3 Random walk on the graph

In Section 3.2.2, we have described the algorithm to convert any web page into a graph, with the nodes representing the basic elements, and edges representing the relationship between the basic elements. In this way, we convert an html web page into a structure that is similar to the whole Internet.

The most successful search engine is Google [1], which proposed the idea of "PageRank" to describe the importance or quality of a single webpage. In our research, we will borrow the idea of PageRank to calculate the importance and quality of each basic node in a web page.

We can similarly calculate a ranking that is proportional to the probability of a reader being at a node by using an iterative algorithm that does the following updates

$$R^t(i) = \sum_{j=1}^{N} W(j,i) R^{t-1}(j) + W(S,i) R^{t-1}(S)$$

and

$$R^t(S) = \beta \sum_{j=1}^{N} R^{t-1}(j),$$

where $R^t(i)$ is the ranking of node i at time t. The value $R^t(i)$ converges to a value that is proportional to the probability of being at the node (the first eigenvector of the transition matrix).

3.3 Extracting and Optimizing

3.3.1 Extracting relevant elements

The task of a search engine is to return the top results that match the search query. However, even though we have the ranking for each element in the web page, we cannot simply return the elements to the user by its rank. In a search engine, every individual webpage is independent and it does not matter if one web page is returned before or after another web page. But in our system, there are semantic and logical relationship between the elements and the order of relevant elements has to be returned as it appeared in the original web page. The user will feel unhappy if he gets an article extracted from a web page that looks nice but has the wrong ordering of the elements.

Our design goal is return to the user the most relevant content, which is those with the highest ranks, but still need to keep the original look and feel on the mobile device. We use a simple heuristic to retrieve complete article based on the ranking of the elements.

```
Select ()
{
    list.insert(topnode);
    T1= I(S,topnode);
    T2=R(topnode);
    while(node=list.getNext()!=NULL)
    {
       d=Distance(topNode,node);
       tw=f1(d)*T1;
       for(each ni W(node, ni)>tw)
       {
           tr=f2(d)*T2;
           if(R(ni)>tr)
             {
                 list.insert(ni);
             }
        }
     }
 }
```

As shown in the algorithm, firstly, we sort the nodes and pick the node with the highest rank. Then we set two thresholds based on the rank score of the top node. *T1* is a threshold on the edge weights, and *T2* is a threshold on the ranks.

We assume that the top node is the center topic or sentence of an article; our task is to traverse the graph from the top node following the links to reach all nodes that also belong to the main article. *T1* is used to set the minimal weight on the links that algorithm should follow, and *T2* is used to set the minimal rank that an element needs in order to be considered as relevant.

As the algorithm moves away from the top node (calculated by d=Distance(topNode, node), the number of links between topNode and node), we increase the threshold *tw* on the weight. Otherwise, the algorithm may traverse a weak link and reach to the center of an irrelevant part of the document. We use linear functions *f1(d) = 1+C1d* and *f2(d)=1-C2d*, which increases the *tw* on each link traversed, and decrease the threshold on rank tr to allow element with lower rank to join. We decrease the threshold on the rank because we believe that relevant nodes further away from the center node may have lower ranks.

After we obtain the list, we will put the element in its original position in the web page and find a minimal rectangle that covers all the nodes. The procedure guarantees the integrity of the original content.

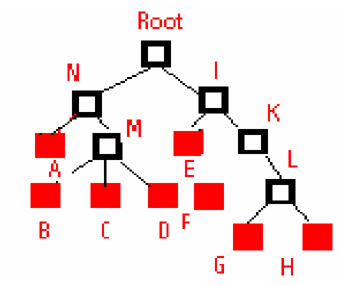
3.3.2 Optimizing for mobile device

In the previous section we obtained a rectangle within a web page that encloses the true article. The target rectangle may be larger than most mobile devices; we need to optimize the content and make sure it looks nice on the mobile device.

We have the following design goals:

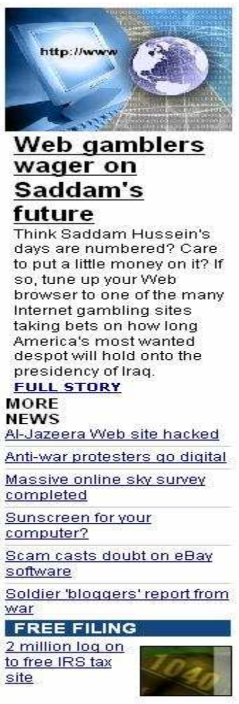
1. Minimize vertical scrolling action on small screen device and eliminate horizontal scrolling action.
2. Maximize the similarity between the layout of the optimized content and the original web page.

We convert the HTML layout so that the width of the re-rendered HTML page is smaller than the screen. To maximize the similarity between the original page and the re-rendered page, we need to retain the HTML hierarchy structure of the original page. Our algorithm can be described as from Figure 3.2 and 3.3:



Original HTML page **Layout tree of the**

Figure 3.2: The original HTML web page and its corresponding layout tree structure of the selected area.



Web gamblers wager on Saddam's future

Think Saddam Hussein's days are numbered? Care to put a little money on it? If so, tune up your Web browser to one of the many Internet gambling sites taking bets on how long America's most wanted despot will hold onto the presidency of Iraq.

FULL STORY

MORE NEWS

Al-Jazeera Web site hacked

Anti-war protesters go digital

Massive online sky survey completed

Sunscreen for your computer?

Scam casts doubt on eBay software

Soldier 'bloggers' report from war

FREE FILING

2 million log on to free IRS tax site

Figure 3.3: The web content with the layout optimization for small screen device

In Figure 3.2, the algorithm indicates a larger area that will be returned to a mobile device, the system will put the elements in the selected rectangle in a “Layout Tree” structure. The layout tree has the following features.

1. Each element maintains hierarchical relationship in the original HTML tree.
2. Each node has a rectangle data that records the area that it occupies in original HTML page.
3. Children of the same parent node are at the same hierarchical level in the original HTML tree.
4. The parent node’s rectangle is the minimal rectangle that covers all the children’s rectangles.

The layout tree is built bottom-up, As seen in layout tree in Figure 3.2, suppose node *B*, C, *D* is at the same hierarchical in the original HTML tree, so we put it under the node *M* and set the rectangle of *M* as the minimal rectangle that covers *B C D*, then *M* and *A* shares the same parent nodes, and so on.

The algorithm to re-render the HTML can be described as the following recursive algorithm

```
Render(node)
{
   if( node.child=NULL )   return node.HTML;
   if(node.width>Screen_Width)
   { //if larger than the screen, recursively call each child.
      for(each child for node)
           result  + = render (node.nextChild);
    }
   else
   {   //The screen is wide enough to put all child nodes
          for( each child for node)
           result  + = node.nextChild.HTML;
   }
}
```

If we call *Render()* with root node as parameter, it will return the optimized HTML page. Suppose the root's width is wider then the small screen, the algorithm recursively call render with to nodes *N* and *I* to process each sub tree. If the node *N*'s width is still wider than the screen, we recursive call *Render(A)* and *Render(M).* At the node *M*, suppose it is not wider than the screen width, so we return *B C D's* HTML source. In the case when the width of basic element is larger than the window, we will zoom in the content to fit the screen. Figure 3.3 shows the result.

3.4 Experiment and analysis

We have implemented the system to test our ideas. We have two goals in the system. The first is to satisfy the user's information need. We try to deliver all the information that a user wants in a web page. The second is to save the bandwidth and minimize scrolling in the mobile device. We will use the following measure to evaluate the effectiveness of the system.

1. The recall value R:

R=(retrieved elements that are relevant)/(all the relevant elements)

(We calculate by the area it occupies in the web page)

2. The percentage of returned elements of the extraction:

Return=(number of retrieved elements)/(number of elements on the web page)

It is preferred that the system deliver as little content as possible while achieving the high average recall.

Because of the novelty of the problem, the author's survey failed to find public available and well-accepted test data set that is suitable for evaluation. We created the test data in the following manner.

Firstly, we randomly selected 158 websites from Google directory, under the category of news. From each web site, we chose an average of 5 web pages and recorded the anchor text of the links that lead to the pages.

Secondly, for each web page, we allowed a user to read the anchor text. Here we require that the anchor text is made up of meaningful sentence, rather than link like “read” or “click”. We ask the user to use the mouse to specify the area that she wanted to read on mobile device. We recorded the web site name, the anchor text, and the area specified by the user. We use a total of three different users for this task.

In this manner, we collected altogether 788 web pages. We further divided the set for design and evaluation purposes. We set aside 580 sample pages to use in designing the system and adjusting the parameters. The remaining 208 samples are never seen and used only for evaluation. The design set and evaluation set do not share any page from the same web site.

We set a target average recall rate as the goal, and try to obtain a system that satisfies the recall rate using the design set. We then evaluate the return rate in the delivery on both the design and evaluation set. The return rate on the evaluation set should be a fair indication on future performance as we have never seen the pages during the design process. For the experiment we set the target average recall to be 85%. We did not use 100% because firstly, many of the elements selected by the user are actually ambiguous and hence the additional benefits of achieving total recall are small. Secondly, we believe on the mobile device people not would prefer to use the limited resource to read the most important content. In most cases we do not attach the similar importance for completeness as we do for the desktop computer. We compare our system with three different algorithms.

1. Simple Match: We calculate cosine similarity between the anchor texts with every basic element. We select all the elements where the similarity is above a threshold and return elements in the smallest rectangle surrounding the selected elements.

2. Extended Match: Based on the result of the simple match, we do a second round calculation to calculate the cosine similarity between selected elements with the other elements. If the similarity is above certain threshold, we add the new element to selected list and return elements in the smallest rectangle surrounding the selected elements.

3. Initial Ranking: We give each element an initial rank and weight as described in section 3.2.2 and return the elements in the smallest rectangle surrounding the elements with initial ranking above a threshold.

4. Full implementation of our algorithm, the initial condition is set the same as 3. We improve it with our random walk and extraction algorithm.

Method	Recall_1	Return_1	Recall_2	Return_2
1	0.55	20.8%	0.62	32.4%

2	0.71	85.9%	0.72	78.1%
3	0.72	65.9%	0.75	63.5%
4	0.86	39.3%	0.85	38.2%

Table 3.1: The recall of different random and traction algorithm

In Table 3.1, “Recall_1” and “Return_1” are the experiment result on design set which is used to select thresholds and to set the values of other parameters. “Recall_2” and “Return_2” are the result on the evaluation set. In our experiment, we set β=0.25, α=0.85, W1=1, W2=1, W3=3, W4=1 and W5=2. The “word match” and “extended match” algorithms can only reach to maximum recall of 0.55 and 0.71 respectively no matter how we set the parameters. For the “initial ranking” method a recall of 0.72 can be achieved, 65% of the elements on the web page need to be delivered. Initial ranking really provide valuable additional information, but it is not sufficient. We need information from the graph in order to achieve better performance. With our algorithm we just need to deliver 39.3% to achieve a better recall. In the evaluation set, we observed a similar pattern.

On average the algorithm need only return about 38% of all the elements in a web page to reach a recall above 0.85. We believe this result is encouraging for mobile devices. First of all, 38% of the elements do not mean only 62% of traffic savings. The savings in bandwidth is actually much higher because most of the elements that are removed are usually multimedia elements or advertisements.

Secondly, 0.85 of recall does not mean that user normally does not get a full article. We checked the samples where the algorithm fails to work well. It is usually because the anchor text of the link is irrelevant to the topic of the web page. In addition to that, under our current data collecting methods, it is likely that no algorithm can get a 100% recall, as errors caused by the ambiguity of the selection are probably unavoidable. For example, consider the following web page shown in Figure 3.4.

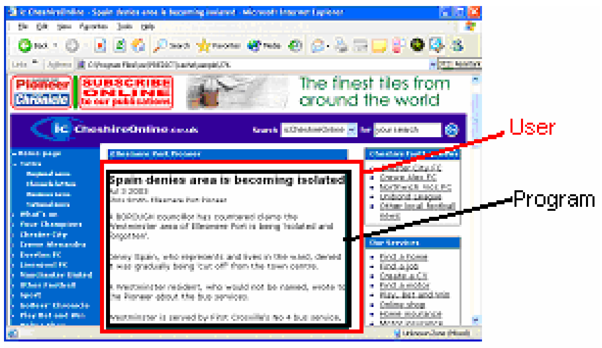

The finest tiles from around the world
CheshireOnline.co.uk
Spain denies area is becoming isolated
User
Program

Figure 3.4: Potential error that introduced in the data collection process

Figure 3.4 explains the limitation of our evaluation. The red rectangle is what user defined as relevant by using the mouse to drag and draw an area on the screen. We can tell that it is no different from what the program returns as positive, which means the algorithm is 100% correct. However, if we calculate the recall by the overlapping physical area, the recall by our definition will only be 0.90.

The algorithm performs consistently on both the design and evaluation case. This shows that the algorithm is stable over different websites. Based on these results, we are confident that our system and implementation achieves the design goal. However, all the sample websites are chosen from Google directory. It is likely that most of them are well organized and designed. The tester selected links with meaningful anchor text to click on which is also helpful for the algorithm. More research needs to be done in the future for the real world Internet where a lot of irregular web pages and misleading anchor text might exist. We also do not have public available data set to evaluate on and hence we are not able to compare the effectiveness for this system with other researchers' systems.

Ideally, the algorithm will be loaded in a personal gateway, which can be our own desktop computer. It will retrieve and render the page in its memory on the behalf of the mobile device, and use the algorithm to optimize the webpage before sending the optimized page to the mobile device wirelessly. Normally optimization of a web page can be done within 1 second on a normal Pentium IV computer. Since the desktop is connected to Internet with cable and only small part of the page is delivered wirelessly, adding the optimization part will not greatly decrease the performance. A personal gateway will also facilitate personalization with less privacy issues.

3.5 Conclusion

Our goal is to design a system that can deliver device independent content to mobile devices from any web page, and to fulfill the user's information need on devices that have minimal computing power, screen and bandwidth available. We achieve this by ranking the importance of each element in a given web page and generating a customized "web" for mobile devices. We proposed three interesting ideas. Firstly, it is possible to represent the HTML web page with a graph structure. Secondly, based on our ranking algorithm that is similar to Google's PageRank, the system can understand what the most important topic of a web page is. Thirdly, we develop an algorithm to reformat and optimize the subset of the original web page for different mobile device. Our experiments show that in the vast majority of cases, the proposed system provides the expected results, making it a useful system.

With the current system, it is possible to navigate by following links that are located within the main article. However, on many sites, special navigation links are provided for navigating within the site. Most of these links are located on the top or side of the web page and will be removed by the current algorithm. In the next chapter we are going to handle these navigation requirements and make the system truly friendly for surfing on mobile devices.

With the development of wireless technology and emergence of various mobile devices, people will not be limited to the desktop computer. We will access the Internet through all possible devices. Instead of building different webs for different devices, we strongly believe that the right direction is to convert and deliver the same content in different ways to different devices.

Chapter 4

Content Classification with Random Walk

4.1 Introduction

In this chapter, we focus on the second fundamental challenges for accessing Internet from mobile devices: How to classify and extract certain type of content in a web page, and deliver only that part to the mobile device, which will exactly satisfy the user's information needs, no more and no less.

As we have shown in last chapter, a web page is made up of hundreds of basic elements. The functional role of each element can be different depending on the context in which it is used. For example, an image can be the banner of a website, an advertisement, or a picture for a news article. If we can accurately understand the functional role of each element, we will be able to develop a new method to optimize the web page for mobile devices. For example, if the user wants to read only the main content of a web page, the system will extract and deliver the main content in the web page. This is desirable for mobile devices which have many constraints, including very small screens, small memory capacities and wireless networks that are slower and more expensive.

In this chapter, we define five content categories: Main Content (*C*), Related Links (*R*), Navigation and Support (*N*), Advertisement (*A*) and Form (*F*). For each web page, we will build five graphs for each content category, and we call it CategoryGraph. The CategoryGraph is specifically designed such that most of the probability mass of the stationary distribution is concentrated in nodes that belong to its corresponding category. We use random walks on specially designed graphs to automatically categorize the web elements. In analogy to Google's PageRank [1], we call the score of each element in the graph after random walk its CategoryRank. The random walk value represents the probability that the object belongs to one specific category; hence each object will come with five rank values. We compare an element's CategoryRank in the five graphs and classify the element into the category where it has maximum value. We present a simple user interface for effective display of the elements in these categories and describe a method for automatically partitioning the web elements into the categories.

We organize this chapter in the following way. In Section 4.2, we will discuss our criteria for functional category selection. We will present an innovative user interface that allows a satisfying surfing experience on mobile devices through category classification. In Section 4.3, we will discuss how to divide a webpage into elements, and build the five category graphs from a single web page. In Section 4.4, we will discuss how we use random walk to obtain the CategoryRank for classification. In Section 4.5 we will discuss the dataset and the evaluation of the system. Section 4.6 gives out the conclusion.

4.2 Functional categories

One web page can contain many different types of content. Each of them will relate to the reader's interest and surfing pattern in a different way. For example, the reader usually uses a navigation link to enter a web page, reads the main content, ignores the advertisement and clicks on related links to further explore a topic. Our objective is to classify web content into meaningful functional categories to facilitate the surfing experience on mobile devices. As a mobile device has very limited memory, screen size, and narrow bandwidth, the system is expected to classify and deliver only one category of the content, for example the main content to the mobile device, while at the same, time, user can switch among content of different category.

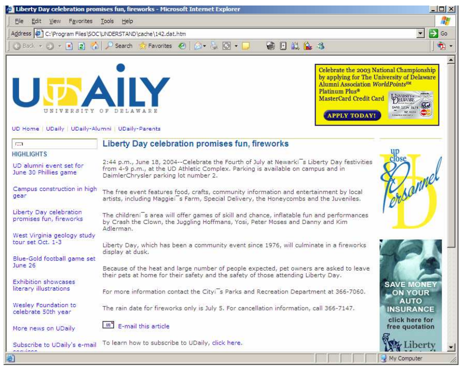
The criteria we use to select the actual categories are very important. Firstly, the functional categories we select need to have a meaningful and useful theme. It needs to be related to the user's daily surfing experience. Secondly, each functional category should have a distinct and coherent meaning. It needs to have a predictable implication on the user behavior while minimizing the confusion for the reader. For example, it would be appropriate to divide links into Navigation (*N*) and Related (*R*) links, because Navigation (*N*) links leads to another section of the site which may or may not have the similar topic with current page, while the Related Links (*R*) is leading to another page that tells a related story. At the same time, further dividing an article into categories like author, article, date, title may not be so necessary for our objective.

We will classify the content into five categories:

- Content (*C*): Main content that is the centre topic of a web page, which includes the main article, title, the author, date, supporting picture or any other material used to describe the topic. The main content is very important, as it is the default content that will be delivered to the mobile device, and we are able to further process the main content, for example text summarization in the next chapter for further facilitate the surfing experience on mobile device.
- Related Link (*R*): A web page normally contains links to other web pages that tell the related story. We want to identify these links to help user to further explore a topic.
- Form (*F*): Forms can be used to enable the reader to interact with the webpage. For example, normally the form is embedded in the web page, by classifying the form that the user can instantly type search with keyword, or submit certain request in web pages from mobile.
- Advertisement (*A*): Web page is filled with internal or external advertisements. They may or may not be useful to the readers especially from the mobile devices. It is ideal if we can separate them and give reader a choice on how to deal with them, especially on mobile devices.
- Navigation and Support (*N*): A website is normally well-structured and divided into different sections. On each web page there are fixed links to those sections, which helps a reader to navigate in the site. Some of the content, for example, the banner, copyright information and the logo, are not informative, are normally invisible and do not have much influence on user's behavior unless user seek for such information, so we put them into this category.

We propose an innovative browser user interface based on functional category classification. Instead of sending the whole web page to the mobile device, which generally wastes bandwidth and clutters the target device, the system first classifies the content in the web page into five categories, and generates five web pages out of the content in the original web page. Mobile users can specify the content to be delivered and rendered in the mobile device. The system allows a user to read exactly what he wants without being distracted by the irrelevant content.

For example, if a user wants to read a web page shown in Figure 4.1. On the mobile device, he will not get the same nice experience as on personal computer.


Liberty Day celebration promises fun, fireworks - Microsoft Internet Explorer

Address C:\Program Files\SOC\UNDERSTAND\cache\142.dat.htm

UDAILY UNIVERSITY OF DELAWARE

Celebrate the 2003 National Championship by applying for The University of Delaware Alumni Association *WorldPoints*℠ Platinum Plus® MasterCard Credit Card APPLY TODAY!

UD Home | UDaily | UDaily-Alumni | UDaily-Parents

HIGHLIGHTS

UD alumni event set for June 30 Phillies game

Campus construction in high gear

Liberty Day celebration promises fun, fireworks

West Virginia geology study tour set Oct. 1-3

Blue-Gold football game set June 26

Exhibition showcases literary illustrations

Wesley Foundation to celebrate 50th year

More news on UDaily

Subscribe to UDaily's e-mail

Liberty Day celebration promises fun, fireworks

2:44 p.m., June 18, 2004--Celebrate the Fourth of July at Newarki¯s Liberty Day festivities from 4-9 p.m., at the UD Athletic Complex. Parking is available on campus and in DaimlerChrysler parking lot number 2.

The free event features food, crafts, community information and entertainment by local artists, including Maggiei¯s Farm, Special Delivery, the Honeycombs and the Juveniles.

The childreni¯s area will offer games of skill and chance, inflatable fun and performances by Crash the Clown, the Juggling Hoffmans, Yosi, Peter Moses and Danny and Kim Adlerman.

Liberty Day, which has been a community event since 1976, will culminate in a fireworks display at dusk.

Because of the heat and large number of people expected, pet owners are asked to leave their pets at home for their safety and the safety of those attending Liberty Day.

For more information contact the Cityi¯s Parks and Recreation Department at 366-7060.

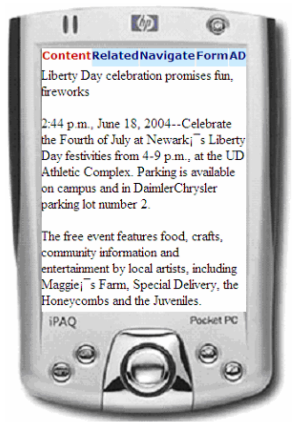
The rain date for fireworks only is July 5. For cancellation information, call 366-7147.

E-mail this article

To learn how to subscribe to UDaily, click here.

Figure 4.1: The original web page on the normal computer browser

In our system, instead of transferring the original web page to the mobile device, we will decompose it into five web pages: Navigation and Support (*N*), Content (*C*), Advertisement (*A*), Related Links (*R*) and Forms (*F*). By default, user will read only the Content (C) page, as shown in Figure 4.2



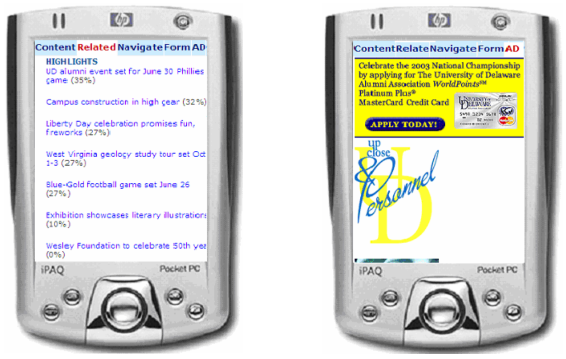
Liberty Day celebration promises fun, fireworks

2:44 p.m., June 18, 2004--Celebrate the Fourth of July at Newark¡¯s Liberty Day festivities from 4-9 p.m., at the UD Athletic Complex. Parking is available on campus and in DaimlerChrysler parking lot number 2.

The free event features food, crafts, community information and entertainment by local artists, including Maggie¡¯s Farm, Special Delivery, the Honeycombs and the Juveniles.

Figure 4.2: The Content view on a mobile device.

As we can see in the Content page, only the main content will be presented, the page is optimized and users can easily read it on a mobile device. At the top of the generated page there are links to other categories. The user can click the links “Related”, “Navigate”, “Form” and “AD” to switch to the other view.


Content Related Navigate Form AD
iPAQ
Pocket PC
Content Relate Navigate Form AD
APPLY TODAY!
up close Personnel

HIGHLIGHTS

UD alumni event set for June 30 Phillies game (35%)

Campus construction in high gear (32%)

Liberty Day celebration promises fun, fireworks (27%)

West Virginia geology study tour set Oct 1-3 (27%)

Blue-Gold football game set June 26 (27%)

Exhibition showcases literary illustrations (10%)

Wesley Foundation to celebrate 50th yea (0%)

Celebrate the 2003 National Championship by applying for The University of Delaware Alumni Association *WorldPoints*℠ Platinum Plus® MasterCard Credit Card

Figure 4.3: The Related view and the Advertisement view of the original web page

As shown in Figure 4.3, if the user clicks the “Related” link in the generated web page the server will return only the related links in the original web. The “AD” link contains all the advertisement in the original webpage. The system allows the user to freely surf the exact the type of the material from mobile device by simply clicking.

4.3 Building category graphs

Using the same method described in the section 3.2.1, we first convert a web page into basic elements. In the second step, we will construct graphs for each functional category. Every pair of elements in the web page is connected with directed edges in both directions. We normalize the sum of the weights coming out of each node to 1. The weight of edge *(i, j)* is the probability of a random walker at node *i* moving to node *j* at the next time instance.

In Chapter 3, we proposed to build single graph for each web page, where the probability mass concentrated on elements that are more important for the mobile user. We use the features like size, similarity with anchor text, and so on. In this chapter, in order to construct different graph for each functional category, we will adopt the similar concept: in each of the graph, the stationary distribution will have most of its probability mass concentrated only on nodes that belong to the corresponding category. As a result, for each graph we will use the features that are more important to that particular category. For example, the feature "external link to advertisement site" is very important for advertisement graph. The probability mass for an element in a graph becomes its score for the category. We categorize each node by selecting the graph that gives the element the highest score.

We note that we effectively solve two problems at the same time: segmenting complex objects in the web page as well as deciding the class that the objects belong to. Motivated by this, we construct each graph from the sum of two distinct graphs: a category independent graph and a category dependent graph. The category independent graph tries to prevent complex objects from being split up into separate classes. On the other hand, the category dependent graph is used for each category. Each of these graphs will attempt to move the probability mass to nodes that belong to its corresponding category. The stationary distribution on the combined graph has probability mass concentrated only on nodes that belong to its corresponding category.

4.3.1 Category independent graph

Based on the features of two basic elements, we will connect them with a weight that increases with the likelihood that the two nodes belong to the same object. We take the following features into consideration:

1. Similarity *S(i, j)*: We calculate the cosine similarity between the visible texts of the two elements. The rationale is that two elements are more likely to belong to same category if they are similar.

2. Distance $Pd(i,j)$: The edge between elements closer to each other shall have a higher weight to encourage walking between them. Let the screen size be (Xc, Yc), the distance between element i and j is:

$$Pd(i,j) = 1 - \frac{\sqrt{(Xi - Xj)^2 + (Yi - Yj)^2}}{\sqrt{4Xc^2 + 4Yc^2}}$$

3. Neighborhood $Nb(i, j)$: If two elements are positioned next to each other, we will add weight to the edge

$$Nb(i,j)=\begin{cases}1 & \mathit{if\ Neighbour\ \ to\ each\ other}\\ \\ 0 & \mathit{otherwise}\end{cases}$$

4. Same Edge $Eg(i, j)$: If two elements have either same left, right top or bottom edge, we will add weight to the edge

$$Eg(i,j)=\begin{cases}1 & \mathit{if\ Same\ Edge}\\ \\ 0 & \mathit{otherwise}\end{cases}$$

5. Same Tag $Tg(i, j)$: If two elements have either same tag, we will add weight to the edge

$$Tg(i,j)=\begin{cases}1 & \mathit{if\ Same\ Tag}\\ \\ 0 & \mathit{otherwise}\end{cases}$$

6. Same Parent $\Pr(i, j)$: If two elements are the children of the same parent node in the HTML hierarchy, we will add weight to the edge

$$\Pr(i,j)=\begin{cases}1 & \text{if Same Parent}\\ 0 & \text{otherwise}\end{cases}$$

7. The weight from the element i to element j is calculated by the function $W(i, j)$ where

$$W(i,j)=I(i,j)/\sum_{n=1}^{N}I(i,n)$$

$$\mathrm{I(i,j)=S(i,j)+Pd(i,j)+Nb(i,j)+Pr(i,j)+Eg(i,j)+Tg(i,j)}$$

The weight represents the probability that the random walker move from one element to another. It is based on features that are independent on the category. For each of the category independent graph, we will add category dependent factor to each edge and thus construct separate graph for each category.

4.3.2 Content (C) graph

For each of the edges in the basic graph, we will add weight to edges whose target is more likely to be a content element; in this way, we can construct the content graph. The following content related features are added:

1. Element Size $Sz(i,j)$: Content object normally have a larger size than other type of element. We use the following formula to calculate the weight for size factor:

$$Size(i)=Width\ (i)*Height\ (i)$$

$$Sz(i,j)=\begin{cases}1 & if\ Size(j)>2*Size(i)\\ \\ 0 & otherwise\end{cases}$$

2. Tag Category $Tag(i,j)$: The web designer tends to use plain text for the main content. We will add weight to the edge which points to a text block. According to the HTML specification, we define an element is s text element if its HTML string starts with tag including "<p>, <Br> <H>, <Strong>, <title>", or it is a <img> tag with capital and alt text

$$Tag(i,j)=\begin{cases}1 & if \quad node\ j\ is\ text\ element\\ \\ 0 & otherwise\end{cases}$$

3. Direction *Dr(i, j):* if we connect the center point of element *i* and element *j* with an arrow, and the arrow is pointing towards the center of the web page, it is more likely to walk to the content we will set *Direction (i, j) =1* otherwise 0.

4. Text length $Tx(i,j)$: The length of visible plain text in an element is often a strong indication of the content element. We use the length as one feature.

$$Tx(i,j)=\begin{cases}1 & if\ TextLength(j)>2*TextLength(i)\\ \\ 0 & otherwise\end{cases}$$

5. The weight for Content graph from the element i to element j is calculated by the function $Wc(i, j)$. We consider both the feature which are category dependent *Ic(i, j)* as well as features which are category independent *W(i, j):*

$$\mathrm{Ic(i, j) = Sz(i, j)} + 4Tag\mathrm{(i, j)} + Dr\mathrm{(i, j)} + 2Tx\mathrm{(i, j)}$$

$$Wc(i, j) = Ic(i, j) / \sum_{j=1}^{N} Ic(i, j) + W(i, j)$$

The constants in our formulas, for example the constant 4 and 2 in the above formula, are set empirically and adjusted through our experiment. It is the same case with the constant in the other formulas in this chapter.

4.3.3 Advertisement (A) graph

For Advertisement (*A*), we use the following features to construct a category dependent graph.

1. $Tag(i, j)$: Certain element, for example Flash object, is mainly used for advertisements. In our graph, we consider the following tags
   - Image: Image with URL or alt text contains “ad”
   - Animated GIF: whether the image is animated GIF
   - Flash object
   - <iFrame> or <Script> object

If element j has one of these features we will set $Tag(i,j)$ to one otherwise it is zero

2. $Text(i,j)$: We collect a collection of words that are often be used in advertisement, for example “advertisement”, “sponsor”, “sale”, “classified”, etc, we count the appearance of such word in element j *CountAD(j)*

$$Text(i,j) = CountAD(j)/5.0$$

3. *Link(i, j)*: the advertisement link presents many clues:
   - *AdSite(i,j):* We collect list of advertisement sites, for example doubleclick.com. The presence of such domain names is a strong evidence for advertisement. *AdSite(i, j)* will be set as one if it is a link to those sites, otherwise zero.
   - *AdURL(j)* The URL structure is under a folder “/ads/” or “ad=” etc. *AdURL(i,j)* will be set as one if its URL contains that structure component, otherwise zero.
   - *AdDomain(j)* If the URL is made up of two domain names that are from different sites, it is normally an advertisement referring program, we will set *AdDomain(j)=1*, otherwise zero.

$$\mathrm{Link(i,j)} = AdSite(\mathrm{i,j}) + AdURL(\mathrm{i,j}) + AdDomain(\mathrm{i,j})$$

The weight for the Advertisement graph for an edge pointing to element *j,* is calculated by the function $Wa(i,j)$:

$$\mathrm{Ia(i,j)} = \mathrm{Tag(i,j)} + 4\mathit{Text}(\mathrm{i,j}) + \mathit{Link}(\mathrm{i,j})$$

$$Wa(i,j) = Ia(i,j) / \sum_{j=1}^{N} Ia(i,j) + W(i,j)$$

4.3.4 Relate (R) graph

For the Related (*R*) graph, we use the following features to construct a category dependent graph.

1. $Tag(i,j)$: Tag <a> or with <Li> is likely to be a related link, but it is also possible that plain text with a small link "read more" can also be related links. We will set $Tag(i,j) = 1$ if it contains such tag, otherwise zero.

2. $Length(i,j)$: Related link (*R*) tends to have a longer anchor text to tell the topic of a story. To calculate this, will count the number of useful word, after removing stop word like "click" or "more",

$$Length(i,j) = CountWord(AnchorText) - 4$$

3. *Link(i, j):* Related link have different URL structure from that of advertisement and a navigation link.

- *FileLink(i, j)* If it is normally a link to a specific file instead of folder or an index file, we will set it as one, otherwise zero.
- *LinkURL(i, j)* If URL normally contains folder name like "/story/", "/news/", we will set it as one other wise zero.
- *InternalURL(i, j)*: if the link is point to a URL within the same domain name, it is more likely to be a related link and will be set as one, otherwise zero.

$$\mathrm{Link(i,j)} = \mathit{FileLink}(\mathrm{i,j}) + \mathit{LinkURL}(\mathrm{i,j}) + \mathit{InternalURL}(\mathrm{i,j})$$

The weight for Advertisement graph for edge pointing to element *j*, is calculated by the function $Wr(i, j)$,

$$\mathrm{Ir(i,\ j)} = \mathrm{Tag(i,\ j)} + 2\mathit{Length}\ \mathrm{(i,\ j)} + \mathit{Link}\ \mathrm{(i,\ j)}$$

$$Wr(i, j) = Ir(i, j) / \sum_{j=1}^{N} Ir(i, j) + W(i, j)$$

4.3.5 Navigation and support (N) graph

For Navigation and Support (*N*) we use the following features to construct category dependent graph

1. $Tag(i, j)$: If the edge is pointing to element with Tag <a>, it can be a navigation link. Image <img> can also be navigation links Space image: if the image file

name contains word like “blank”, “corner”, “space” etc, which indicate the image is not related to the content.

If element *j* has one of these features we will set *Tag(i, j)* to one otherwise it is zero

2. $Length(i, j)$: Navigation link (*R*) tends to have a shorter anchor text, normally a just one or two words, to calculate this we will use the following formula:

$$Length(i, j) = 4 - CountWord(AnchorText)$$

3. *Link(i, j)*: The Navigation and Support element’s URL is normally link to a index page rather than a file.
   - *FolderLink(i, j)* If it is normally a link to a folder or an index file, we will set it as one, otherwise zero.
   - *InternalURL(i, j)*: if the link is point to a URL external domain name, it is more likely to be a related link and will be set as one, otherwise zero.

$$\mathrm{Link(i, j)} = FolderLink\mathrm{(i, j)} + ExternalURL\mathrm{(i, j)}$$

4. $Text(i, j)$: We collect a collection of words that are often be in the anchor text of used in link, for example “global”, “weather”, “business”, “copy right”, “logo”, “banner” etc, we count the appearance of such word as *CountNavigate(j)*

$$Text(i,j) = CountNavigate(j)/3.0$$

5. *Pos(j)* Element in the edge, which tends to be Navigation (*N*) element. So we use four features LeftEdge, RightEdge, TopEdge and BottomEdge, and use a binary value to represent whether the element's border touches one of the four edges of the web page,

$$LE(j) = \begin{cases} 1 & if\ leftBoard(j) < 0.1 * ScreenWidth \\ 0 & else \end{cases}$$

$$RE(j) = \begin{cases} 1 & if\ rightBoard(j) > 0.9 * ScreenWidth \\ 0 & else \end{cases}$$

$$TE(j) = \begin{cases} 1 & if\ topBoard(j) < 100 \\ 0 & else \end{cases}$$

$$BE(j) = \begin{cases} 1 & if\ bottomBoard(j) < ScreenHeight - 100 \\ 0 & else \end{cases}$$

$$\mathrm{Pos(j)} = LE(\mathrm{j}) + RE(\mathrm{j}) + TE(\mathrm{j}) + BE(\mathrm{j})$$

The weight for Navigation graph for edge pointing to element j is calculated by the function $Wn(i,j)$:

$$\mathrm{In(i,j)} = \mathrm{Tag(i,j)} + Length(\mathrm{i,j}) + Link(\mathrm{i,j}) + 2\mathrm{Text(i,j)} + \mathrm{Pos(j)}$$

$$Wn(i,j) = In(i,j) / \sum_{j=1}^{N} In(i,j) + W(i,j)$$

4.3.6 Form (F) graph

Rules for Form (*F*) is simpler than the rest, we use the following features to construct category dependent graph

1. $Tag(i,j)$: Tag <input> or with <textarea> <form> is a part of the form. In some websites, form is connected with a database, so appearance of database inquiry language will be treated as proof of the Form, we will set it as one if such feature is presented, otherwise zero.

2. $Text(i,j)$: The collection of indicative words for Form (*F*) include "search", "submit", "form", "login" etc, we count the appearance of such word in element j *CountForm(j)*

   $$Text(i,j) = CountForm(j) / 3.0$$

   The weight for Advertisement graph for edge pointing to element j, is calculated by the function $Wf(i,j)$,

   $$\mathrm{If(i,j)} = 5\mathrm{Tag(i,j)} + Link\mathrm{(i,j)}$$

   $$Wf(i,j) = If(i,j) / \sum_{j=1}^{N} If(i,j) + W(i,j)$$

4.4 Random walk on the graphs

We propose the idea of "CategoryRank" to calculate the likelihood that an element in a web page belongs to certain category. "CategoryRank" can be thought of as a model of user behavior similar to "PageRank", where random walking separately takes place in five graphs. Each element will get five "CategoryRank" from the five graphs. We then compare the element's "CategoryRank" in five graphs and classify an element to a category with the maximum "CategoryRank".

In our implementation we use the following formula to calculate the CategoryRank iteratively; it is very similar to PageRank formula

$$CR^{t}(i) = (1-d)CR^{t-1}(i) + d\sum_{(j,i)\in E} W(j,i) * CR^{t-1}(j)$$

At each iteration, the CategoryRank of element *i* will be updated with the contribution from itself and all linking vertices in the graph. Following the Google's PageRank [1] formula setting, we set *d* as 0.85. We set a threshold to check whether the probability distribution has converged in each round of iteration. This is done by summing up all the changes in the CategoryRank of every element and stopping the iterations if the sum is smaller than 0.0001. It normally takes about 10 rounds before the scores stabilize.

Once we get the CategoryRank for each of the elements in the web page, we will use that rank to classify the web content into different categories, and generate a new web page that contains only one type of content and deliver to mobile.

4.5 Experiment result and analysis

We implemented the system to validate our ideas and evaluate the performance of this approach. We use the area of the correctly classified elements as a measure of performance, rather than counting the number of correctly classified elements, because the elements in the web page are not equally important. For example, a text block is more important than a navigation icon. Secondly, for the mobile devices where the screen size is so small, we will focus on delivering as much important information as possible, rather than present as many relevant elements in as possible.

To compare with simple machine learning approaches, we used the off-the-shelf system WEKA [6] for this experiment. We follow the general practice of using precision (P), recall (R) and F-measure as the evaluation.

1. The precision (P) for a certain category C:

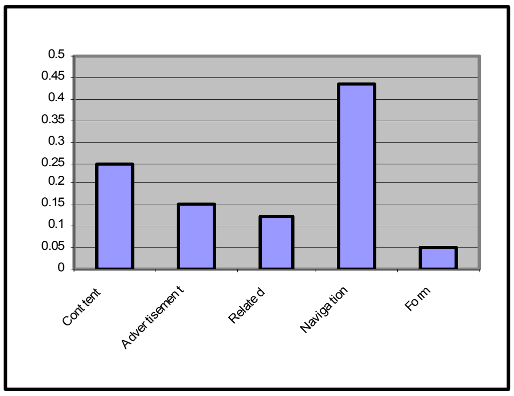
P = (Total area of elements correctly classified as belonging to C)/(Total area of elements classified as belonging to C)

2. The recall (R) for certain category:

R= (Total area of elements correctly classified as belonging to C)/(Total area of elements that actually belong to C)

3. F-measure: 2PR/(P+R)

Because of the uniqueness of the research problem, there is no public available and well-accepted test data set that is suitable for evaluation. The author created the data in the following manner. We first randomly selected 150 websites from the Google directory as the training set, under the category of news. We developed a specialized annotation software tool. For each web site, we chose 2 pages and manually label each of the 18,864 elements into one of the five categories. As we can see from the distribution of all the elements in the training set, only 24% of the elements are content.



The distribution pattern

Content	24%
Advertisement	14%
Related Link	12%
Navigation	43%
Form	7%

Figure 4.4: The distribution of category element in our dataset

When we looked into the large number of elements, we find some of the elements are not really useful to display. For example, an element that is either extremely small. Removing them from the page will improve the page loading speed without hurting the surfing experience on the mobile device. We come up with two rules to remove such elements. Firstly, the element with either width or height less than 5 pixels will be removed. This helps to remove very small icons and narrow bars. Secondly, a picture (decided by source image file name) will be removed if it is used more than three times in a page. These are elements like “space.gif”, which is normally used to fill empty space on the web page or to separate useful information.

After preprocessing, 12,134 elements remain. We set aside the first 10,009 as the training set and the rest as the test set. The training set is used in the process of designing the feature sets and weights while the unseen test set is used to test the effectiveness of the algorithm. The test set does not have any common web sites with the training set, and the elements are labeled with the same tool.

Besides changing the edge weights, we can also improve performance by initializing the nodes to different values for different graphs. We found that initializing the node values to 0.15 for the Form graph and to 0.20 for all other graphs works well. This is consistent with Form containing the smallest number of elements in the web pages. The corresponding results are showed in Table 4.1.

Class	Precision	Recall	F-Measure
Content	0.90	0.94	0.92
Advertisement	0.83	0.90	0.87
Relate	0.65	0.64	0.64
Navigation	0.66	0.56	0.61
Form	0.92	0.56	0.70

Table 4.1: Experiment result with training set for all five category contents

The results on the 2,125 test data elements are shown in Table 4.2. The F value for Content (C) and Advertisement (A) and Form (F) are as high as 0.93, 0.82 and 0.82 respectively. The result on the training and test set shows that the multi-graph random walk method is an effective and practical for method for classifying and extracting the main content and advertisement for mobile devices.

Class	Precision	Recall	F-Measure
Content	0.92	0.93	0.93
Advertisement	0.75	0.90	0.82
Relate	0.58	0.66	0.62
Navigation	0.76	0.56	0.64
Form	0.93	0.74	0.82

Table 4.2: Experiment result with test set for all five category contents

To compare the experiment result, we use the WEKA package J48 classifier. We tried many other classifiers provided by WEKA, including NB and SMO, and J48 performed the best on the dataset. We use the same feature vector used in constructing the graph to build the J48 classifier using the training set. We use it to evaluate the 2,125 elements in the test set. The results are:

Class	Precision	Recall	F-Measure
Content	0.85	0.77	0.80
Advertisement	0.40	0.60	0.48
Related Link	0.12	0.58	0.20
Navigation	0.85	0.68	0.76
Form	0.47	0.75	0.58

Table 4.3: The WEKA result with test set for all five category contents

As we can see from the Table 4.1 and Table 4.2, the performance of our approach is better than the machine learning approach for all categories except for the navigation. However, the current result is based on the experiment data set that the author managed to collect with the help of independent focus group users. Whether the system outperforms the machine leaning methods throughout all web pages on the Internet still need future research to explore.

4.6 Conclusion

Our goal in this chapter is to solve the second fundamental challenges for accessing Internet from mobile devices: classify and deliver only part of web content to satisfy the user's information needs on the mobile device. Based on the analysis of surfing patterns of normal users on the common websites, we narrow our content classification into five functional categories, which includes Content (C), Related Link (R), Navigation and Support (N), Advertisement (A), Form (F).

We design a system that can divide a webpage into basic elements, and build five category graphs. We propose a novel method based on random walk models for this task. Experimental results are promising, showing that the method outperforms common machine learning methods on this problem. However, due to the lack of commonly available dataset for this problem, the author has to manually create data to verify the idea. The universal effectiveness the proposed method still needs to be examined when larger data set or focus group users are available.

With the system introduced in this chapter, it is possible to extract only certain categories of content within a web page to be delivered to a mobile device. Furthermore, result of this system proposed in this chapter can also facilitate other web information processing application. It serves as the foundation of next chapter, as the main content may still be too long for some small screen devices. We will develop text summarization system to make web content more readable on mobile devices.

Chapter 5

Text Summarization with Random Walk

5.1 Introduction

In this chapter, we focus on using graph and random walk to solve the third fundamental challenge in the mobile Internet: text summarization for mobile devices.

The system in chapter 4 extracts the main article from a web page. However, full article may not be ideal for mobile device. First, the screen of mobile devices is too small to present the full article, so we will need to scroll and scroll in order to read the long text on mobile. Second, the wireless bandwidth is expensive and slow, and it is not cost effective to transfer the full text when users typically read only a small portion of the whole. In this chapter, we will propose a text summarization system for mobile device, so that mobile users can choose to read the summary rather than the full text.

Compared with traditional text summarization tasks, the challenge of automatic summarization for mobile device is unique. Most of the existing summarization systems are designed to work for specific document sets. They make use of the features that are calculated based on all articles in that document set. For example, the average sentence length. However, the summarization system for mobile device needs to summarize a single web article, and there is no other document to refer to. Second, most existing summarization systems need to be trained before generating useful results, and this process makes the summarization system less adaptable to different web sites on the Internet. As a conclusion, traditional text summarization method is not suitable for the mobile Internet.

Researchers have proposed to use graphical model for text summarization. The basic approach contains two steps: First, convert the text article into a graph, normally the node in the graph is the sentence. Second, use graph technique to select sentence as summary. For example, Salton et al. [75] used degree centrality to extract the important paragraphs of an article as summary. Erkan et al. [51] proposed LexRank, the system converts text into a fully connected graph and then random walks to select summary sentences. Mihalcea et al. Mihalcea et al. [27] proposed the TextRank model, which is based on a unidirectional backward pointing graphical model and random walks to rank the sentences. All these work have delivered promising results for text summarization. However, existing graph based text summarization systems also have limitations. As we will explain later, the graphical models they choose may not be the most suitable for text summarization. We will develop a new graphical model for single document summarization, which further improves the performance.

The system we propose will act as a gateway between the mobile device and the Internet. When the user clicks on a link from the mobile device, the system will retrieve the web page on behalf of the mobile device. Rather than delivering the full article, it will summarize the article and return only the summary to the mobile devices. This framework has clear advantages: first, it saves the bandwidth and speeds up the surfing experience by transmitting only the compressed summary. Secondly, the summary captures the main idea of the article, and is more suitable to display on small screens.

We have two contributions in this research. First, we analyze two existing graphical models for text summarization. We believe the right graphical model must be compatible with how human beings read and write articles. Second, we propose a new "citation model" for text summarization. It is a simple and effective for text summarization. Our experiments show that our model outperforms other graphical model in the single document text summarization.

In section 5.2 we will discuss our assumptions about how users read and write articles. Then we will analyze the two existing graphical models, and explain their advantages and disadvantages. In section 5.3, we will propose our citation graphical model. In section 5.4, we discuss the experiment and the results which confirm that our model outperforms its peers.

5.2 The graphical models

We want to build graphs from text articles, and then use the graph technique to pick the most important sentences as the summary. Graphical models have also been proposed in the area of text summarization.

To understand how human beings read or write articles is helpful. We believe normal people will read an article from the beginning towards the ending, and the attention is more focused in the beginning. Similarly, in order to match the reading pattern of the normal people, the authors normally write from the beginning part in the direction towards the end, even though they may not write sentence by sentence. Secondly, the authors normally put important ideas or concepts in the beginning, and leave the details or less important content near the end. Most authors first make a claim, then use examples or explanation to reinforce the claim before jumping to the next claim. The assumptions for reading and writing are true for most human being. A good graphical model for text summarization needs to take this into consideration.

5.2.1 The fully connected graph

A fully connected graph model was proposed by Erkan et al. [51], where every two sentences that share common words will be linked with an edge. The weight of the connectivity is based on intra-sentence cosine similarity, where between each pair of the sentences, we will connect them by their cosine similarity. An article with ten sentences generates a graph as shown in Figure 5.1.

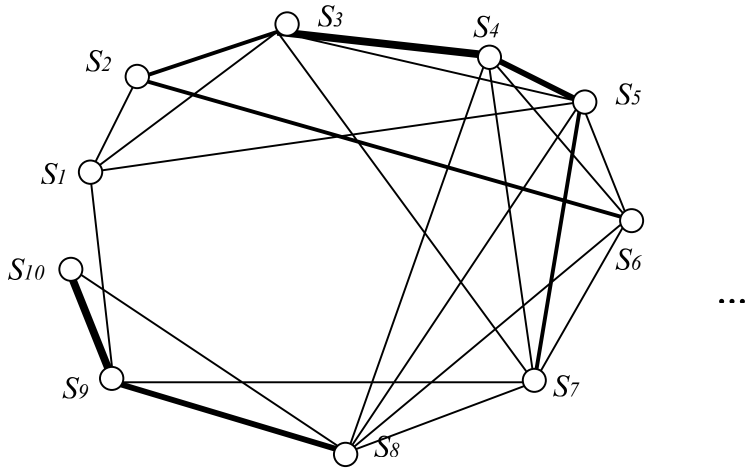


Figure 5.1 The fully connected graph model

In Figure 5.1, even with fully-connect every sentence in the article, the result graph may not be fully-connected as the similarity between some pair of sentences would be zero. The fully-connected graph focuses on the "centrality" of a sentence, which is calculated by the similarity property. A fully-connected graph model is incompatible with how users read or write articles as it assumes all the sentences are equal regardless of their position in the article. We understand that this may not be true, as the readers normally pay more attention to earlier content, and authors start an article with important materials. The graphical model should be built in a way that somehow is in favor of earlier sentences. Although we believe that "centrality" is important, it does not model the real world well enough. It is not the most suitable graphical model for text summarization.

5.2.2 The backward directed graphical model

According to the backward graphical model proposed by Mihalcea et al. [27], the authors write article from the first sentence to the last. All the later sentences “backward” link to earlier sentences, as shown in Figure 5.2.

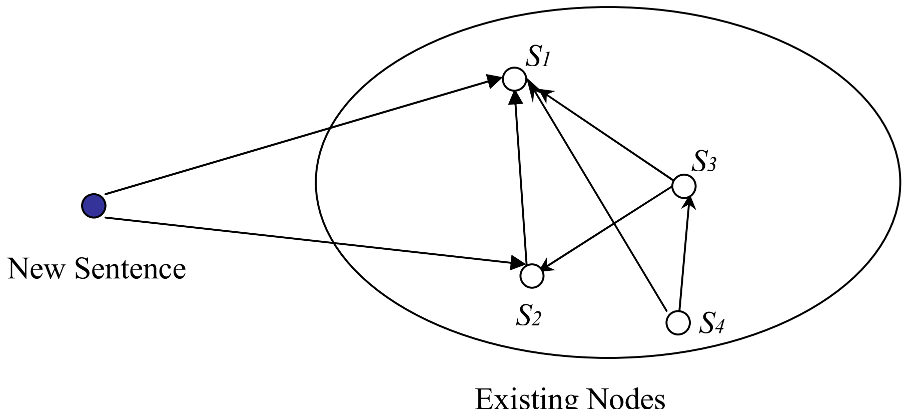


Figure 5.2: Scheme of the growth of the backward graph model

As we can see in the Figure 5.2, when a new sentence is added to the graph, the backward link to earlier sentences will be added. The “TextRank” system [27] is one kind of backward graph. It is built by having all the links pointing backward to earlier sentences, and the weight of directed edge from sentence *i* and sentence *j* is:

$$W(i,j) = S(i,j) \times P(i,j)$$

$$P(i,j) = \begin{cases} 1 & if \;\; j < i \\ 0 & otherwise \end{cases}$$

$S(i,j)$ is the similarity between the sentence *i* and sentence *j*. The $P(i,j)$ ensures that only the backward link will be used in the graph.

The backward similarity graph is compatible with our assumption about how the human beings read and write articles. The content of the article is developed by first making a statement or claim, and then support or explain it by evidences or details. The sentences with the most support from later sentences are important. In the generated graph, there will be directed links from the supporting sentences to the statement and claims. The backward similarity graph rightly gives earlier sentence a higher priority.

We believe the backward graph model is a better model for text summarization. However, it is still not the perfect graph model. It gives every earlier sentence equal prior importance. In particular, when creating a link from a sentence to earlier sentences, it does not assign the sentences at the start of the article more importance.

5.3 The citation graph

We propose a new graphical model for the text summarization. It is inspired by the citation network formed by academic papers, so we call it citation graph. In a citation network, the probability of a new paper citing an old paper is proportional to the in-degree of the old paper [85]. This naturally gives strong preference to linking to nodes which appears very early in the evolution of the network. We build the citation graph from an article in five steps:

1. Add the first sentence to the graph as first node (N_1) with initial significance value W_1=1;
2. Add the next sentence to the graph as node N_i, and calculate the similarity of this sentence to all the earlier nodes.

   $$\begin{aligned} &S_{i1} = Similarity(i,1); \\ &S_{i2} = Similarity(i,2); \\ &\ldots \\ &S_{ii-1} = Similarity(i,i-1); \end{aligned}$$

3. Add a directed link from the new node to all earlier nodes. The weight is calculated by the similarity and the original significance of the cited nodes;

   $$\begin{aligned} &W_{i1} = W_1 \times S_{i1}; \\ &W_{i2} = W_2 \times S_{i2}; \\ &\ldots \\ &W_{ii-1} = W_i \times S_{ii-1}; \end{aligned}$$

4. The introduction of the new citing sentence will lead to increase of the significance value of cited sentence.

$W_1' = W_1 + S_{i1};$
$W_2' = W_2 + S_{i2};$
$...$
$W_{i-1}' = W_i + S_{ii-1};$

5. Go to step 2 and repeat with each subsequent sentence, until the last sentence of the article is reached.

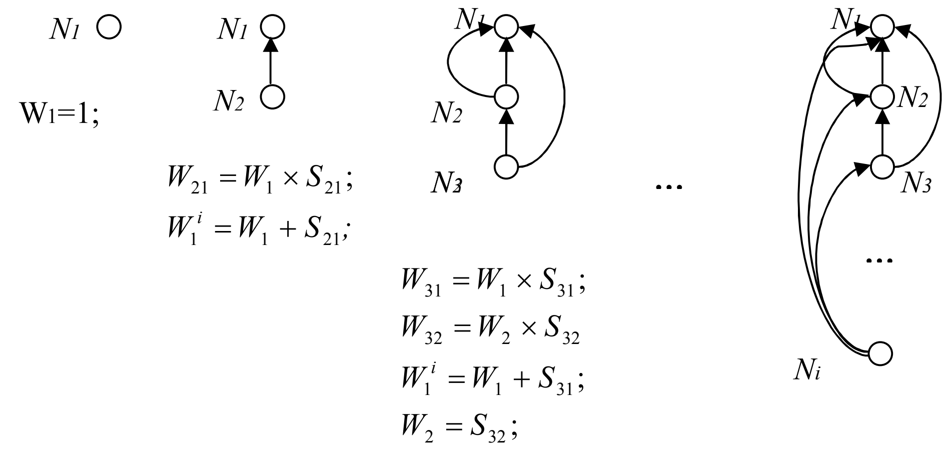


Figure 5.3: The process of constructing the citation graph

As shown in the graph 5.3, in the first step the first sentence is the only node N_1 in the graph, with an initial weight W_1=1. In the second step we process the second sentence N_2. We connect the two nodes with a directed edge from N_2 to N_1 with the weight $W_{21} = W_1 \times S_{21}$.

At each step, we add a new sentence to the graph, and modify the weight of the existing edges and nodes. In this way we gradually incorporate the citation relationship introduced by the new sentences into the graph. We will repeat this procedures until all sentences are added, and text document is fully converted into a citation model graph.

We believe the citation model is a better model for the text summarization problem, because it precisely matches our assumption about how user read and write articles. First, the citation graph is different from the fully connected graph model, where the newly introduced sentence is treated equally. Under our assumption, we believe earlier sentences are more important. Secondly, the citation model is a variation of the backward graph. In the backward graph, new nodes are gradually added into the graph, but the link and node value remain static. In the citation graph model, we dynamically update the node value as we introduce each new node into the graph. Most articles have important ideas or concepts in the beginning, and leave the details or less important content near the end. The citation model reflects that; as details or supporting materials are added, the importance of corresponding earlier sentence changes accordingly.

After the graphical model is built upon the text article, we will perform random walk on it and rank all the sentences in the article. Top sentences are chosen as the summary. We will use the same random walk iteration formula and parameter setting as the Google system [1].

5.4 Experiment result and analysis

5.4.1 The dataset and evaluation package

In order to provide a fair comparison of our system and the other summarization systems in this field, we will use the standard dataset and evaluation package.

The Document Understanding Conference (DUC) [34] was started by the National Institute of Standards and Technology as an evaluation conference for research in the area of text summarization. NIST produced 60 reference sets, 30 for training and 30 for testing. Each set contains documents, single document summaries, and multi-document summaries on clusters of related articles. The datasets define standard single summarizations with fixed length of 50, 100, 200 and 400 words, which we utilize for our experiments. We will use the DUC 2001 training set for selecting the best graph model to use while the DUC 2002 dataset was used for comparison with other systems[1]. Considering the screen size of the average mobile device, we will focus on 100 words summary.

This dataset is suitable for evaluating our proposed method for several reasons. First, the articles are from many different sources, for example, The Financial Times, FBI, LA, San Jose Mercury News etc., It helps to test whether the proposed method is independent of the source. Second, the articles are written by different authors about different topics; so that we can use it to test whether our graphical model performs equally well for articles in

[1] Training set is not available for DUC 2002.

different domains. Third, although the data set is still small compared to all the articles in the Internet, it is being used by many other researchers, so it is a fair comparison to test whether the citation graph outperforms other graphical models and summarization systems.

For evaluation, we will use the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) toolkit. ROUGE package is introduced by Lin et al. [35], which proposed that the summarization system can be evaluated by the unigram co-occurrence with human judges. ROUGE was adopted by the DUC conference only from 2004 onwards. However, researchers have used it for single document DUC 2001 and 2002 datasets to compare results.

Before conducting our experiment, we verified the DUC dataset and ROUGE package. We discovered some bugs in both the DUC dataset and the earlier ROUGE package. The bug has resulted in the inaccurate reports by other researchers on the performance of their algorithms. We explain the bugs in the document sets as well as in the ROUGE package in Appendix I.

The top 100 word is a well accepted common baseline for text summarization, so we will extract the top 100 words of the article as the baseline. We will use the ROUGE 1.5.5 package, which has been verified correct for bugs found in the earlier versions of ROUGE. The parameter setting is: "-n 1 -l 100 -m –a". The "-n" parameter sets the max-ngram length that will be computed. In our experiment we use "- n 1" to request the

ROUGE package to evaluate using unigram. The “-l” parameter sets the length of the words in the system and peer summary for the evaluation. In our experiment we will generate a summary longer than 100 words and use“-l 100” to have ROUGE package cut the summary length limit to 100 words for us. We will also use “-m” to stem both model and system summaries using Porter stemmer before computing various statistics. We will use “-a” to set ROUGE package to evaluate all systems specified in the ROUGE-eval-config-file. The baseline result or all different dataset is shown in Table 5.1.

Dataset	Baseline
DUC 2001 Training	0.44464
DUC 2001 Test	0.44252
DUC 2002	0.48057

Table 5.1: The baseline performance for DUC 2001 and 2002 using ROUGE 1.5.5.

5.4.2 Fully connected graphical model

In the first step, we will build fully connected graph with a single feature, and then use random walk to rank the sentences. In our experiment, for any two sentences S_i and S_j, let N_i be the number of words inside S_i. A sentence can be represented by a vector of non-trivial words $S_i = W_1^i, W_2^i, W_3^i, W_4^i, ..., W_n^i$. We remove stop words from the list. The complete list of stop words is listed in Appendix 2. The similarity between sentence *i* and *j* is then defined as:

$$Similarity(i,j)=\frac{|\{Wk \mid Wk \in Si \,\&\, Wk \in Sj\}|}{\log(|Si|)+\log(|Si|)}$$

In this formula, the similarity is calculated by the number of overlapping of the words in the two sentences, and it is further normalized by the sum of the sentence length. The formula is proposed by the TextRank system [27]. We have also implemented other similarity schemes such as cosine similarity. However, this formula outperforms the rest. We will use this formula through out our experiment.

The result is listed in Table 5.2.

Graph	Performance
Base line (top100)	0.44464
Similarity graph	0.43468

Table 5.2: Fully connected graph performance for DUC 2001 training set using ROUGE 1.5.5.

The result shows that the fully connected graph does not outperform the baseline.

5.4.3 The backward graphical model

In the second part of the experiment, we implement the backward graph on the DUC 2001 training data, the result is listed in the following Table 5.3.

Graph	Performance
Base line (top100)	0.44548
Backward Similarity	0.43024

Table 5.3: The backward graph performance for DUC 2001 training dataset using ROUGE 1.5.5.

As we can see the performance of the backward directed graphical model does not outperform the baseline. It reconfirms our assumption about normally how the articles are written: most of the articles begin with the most important sentences. As we will see in next section, the top 100 word is very difficult to beat; none of the paper in the DUC 2001 has achieved that goal.

5.4.4 The citation model

We use the citation model on the DUC 2001 training set, resulting in a ROUGE score of 0.45218. It is promising as it is higher than the baseline 0.44651. We then deploy the trained model on the DUC 2001 testing set and DUC 2002, in standard open testing.

Graph	Baseline (top100)	Citation Model
DUC 2001 Training	0.44651	**0.45218**
DUC 2001 Testing	0.44252	**0.45136**
DUC 2002 Testing	0.48196	**0.49024**

Table 5.4: The citation model performance on all datasets using ROUGE 1.5.5.

As we see from the Table 5.4, the citation model outperforms all the baselines in every dataset. If we compare the result with other systems, as shown in Table 5.5:

System No	Rouge 1.5.5
Group 31	0.46544
Group 29	0.4641
Group 28	0.48079
Group 27	0.45733
Group 25	0.41409
Group 23	0.43179
Group 21	0.4779
Group 19	0.46204
Group 18	0.44298
Group 16	0.43717
Group 15	0.45374
Group 1	0.47103
TextRank[2]	0.48293
baseline	0.48196
Citation	**0.49024**

Table 5.5: The performance comparison of all system using ROUGE 1.5.5 on DUC 2002

In Table 5.5, the "Group 1" to "Group 31" represent different teams participating in the DUC2002, Out of all the systems, the citation graph model achieves the highest result throughout DUC2001 training and testing set as well as DUC2002 dataset.

[2] The creator of TextRank has kindly provided the output of their algorithm for our comparison.

5.5 Conclusion

In this chapter we focus on text summarization for mobile device. Different graphical models have been proposed for text summarization, for example, the fully connected model and the backward model. However, we find they are not the most suitable model for the text summarization. We propose a new citation model with random walk to create an automatic single document summarization system

We use the Document Understanding Conference (DUC 2001, 2002) as the dataset and ROUGE as evaluation package. Our experiment shows most of the earlier system cannot outperform the baseline. The fully-connected graph model and the backward graph model fail to generate better result than the baseline. However, the citation model can generate summaries that are better than the baseline. Especially, the result of the citation graph is consistently better than baseline and that of the earlier systems. It is simple, elegant, with no parameter to adjust. It requires no domain knowledge and it can work without a training process.

Chapter 6

Conclusion

Accessing the web from mobile devices is faced with three fundamental challenges. Firstly, most web pages are designed for screen that is much bigger than that of the mobile device. Secondly, most web pages contain more information than people are willing to process on a mobile device. Thirdly, the main article of a web page may be too lengthy for the mobile device to display.

Summary

In this thesis, we propose a graph and random walk based method to solve these three fundamental challenges. After the discussion background knowledge and related work, in Chapter 3, we focus on using random walk to solve the first fundamental challenge in the mobile Internet: by using random walk on the elements that makes up a web page, we can rank the page elements in order to optimize the layout for small screen device. In Chapter 4, we focus on the second fundamental challenges for accessing Internet from mobile devices: how to classify and extract certain type of content in a web page, and deliver only that part the mobile device, which will exactly satisfy the user's information needs, no more and no less. In Chapter 5, we focus on using graph and random walk to solve the third fundamental challenge in the mobile Internet: text summarization for web articles.

The methodology of this thesis is simple. We believe web page is too complex to analyze as a whole, we will first divide the entire web page into basic elements such as text blocks, pictures, etc. Next, based on the relationship between the elements, we will connect the elements with edges to make them into a graph. Finally, we will use random walk to conquer the three challenges.

The result of the method proposed in this thesis is very effective. Our experiments of web page optimization show that from randomly selected websites, the system need only deliver 39% of the objects in a web page in order to fulfill 85% of a viewer's desired content. In the experiments of web content classification, the system generates good performance with the F value for main content and advertisement (A) as high as 0.93 and 0.82 respectively. In the experiments of text summarization, we propose a new citation based graph model. We use the well-accepted DUC dataset for single document summarization, and ROUGE as evaluation package. Our performance of the DUC 2001 and DUC 2002 single document top 100 words is higher than the baseline and it has outperformed all the other participants in the conference and other graphical models.

Limitations

This research proposed an interesting solution for three challenges in the mobile Internet; however, the method has its limitation. First, its effectiveness is only verified in the lab environment. The data set is small if compared to the whole Internet. For example, in the web content optimization and classification task, there is no well-accepted data set and the author has to manually create a data set for the purpose of verification. Even though the data collection is designed to be unbiased, due to limited resource, the data may still be biased towards the a few hundred website we have used. We also need to take in consideration of small number of people involved in the collection process. In the text summarization task, even though we use a well-accepted data set, it is still a limited size of data.

Secondly, the method works well only for static web pages. As in all the three challenges, we assume the user click a link and open a web page, our task is to optimize the layout and content in the page for a mobile device. A lot of web sites are using technology like Ajex or Java Script to enhance user interactivity, while our method completely ignores the dynamic content in the web page. The system does not work well for those dynamic and interactive webpage. Finally, the system assumes each web page is a separate and independent browsing session. However, many web servers keep user session to support "user flow", for example, the Yahoo Web Mail.

Future Work

Our research is based on the thin-client computing concept. Instead of directly accessing the Internet, the mobile devices will access the proxy server. Upon receiving the request from mobile devices, the server will retrieve the HTML page and render the page in the memory. Based on the methods proposed in this thesis, the proxy server may optimize, classify, filter or summarize the content in the webpage, and generate a new web page that is optimized for the mobile device. The optimized webpage is then sent back to the mobile device to be viewed properly on the small screen. Instead of building different webs for different devices, we strongly believe that the right direction is to convert and deliver the same content in different ways to different devices.

Looking into the future, we believe people's activity on Internet is not longer purely surfing, and the difference between an application and a web page will diminish with the concept of "Web OS" being well accepted. Optimizing and simplifying the page layout for mobile device is just the first step towards a useful mobile Internet. The new challenge is how to simply the session, dynamic interaction and user flow of traditional web application and fit it to the limited screen and interaction capability of mobile devices. This is the major challenge that we will further explore.

Appendix 1: DUC and ROUGE bug analysis

Bugs in DUC Data

The bugs in the DUC data are related to file name typo or mismatch. We did a manual check for each of the document. The typical the bug that we identify in the DUC 2001 includes.

- The file mentioned as “APP890515-0232" doesn’t exist, but actual file it is named AP890515-0232, where the typos is “APP” vs. “AP”
- The manual summary for "AP870611-0085" exists, however the original file disappear, after we manually check, we find it is the summary for the file"WSJ870611-0085", where the “AP” is mistaken for “WSJ”
- "AP891125-0090" is available in training set, but its summary does not exist. which make it valueless for the evaluation, so we have to remove it.
- "WSJ870611-0085" in the dataset but summary not exist, after a manual search we find the summary for"AP870611-0085" is actually for "WSJ870611-0085"
- "FBIS3-57782" in dataset, but its summary is mentioned as "FB153-57782", where ‘S’ is typed as ‘5’.

To our best effort we identified in total 24 bugs in the DUC dataset, which is more than 5% of the total documents. As most of the reported system result differs only in 10% level, so how they handle the bugs may have some impact.

Bugs in ROUGE Package

Earlier version of the ROUGE package has bugs related to the sentence length. The bug is inside every earlier version of ROUGE package, and it is only fully solved in the version 1.5.5. To demonstrate the bug the evaluation package， we conduct a small experiment.

1. We create a simple function to truncate and return the first *n* words in an article, where *n* is a parameter.
2. We set n=100, and return 100 words from the original text in the DUC dataset as summary to evaluated by the ROUGE package.
3. We gradually increase the *n* , that we will output longer text as summary.

We gradually output extra text as a summary to the ROUGE 1.2.2 and ROUGE 1.5.4.1. We use a parameter "-l 100" to require ROUGE package to consider only the top 100 words. We expect that the ROUGE value will be stabilized at certain fix value, despite the more text in the summary. We output the ROUGE value vs. the number of extra words in the Figure A.1.

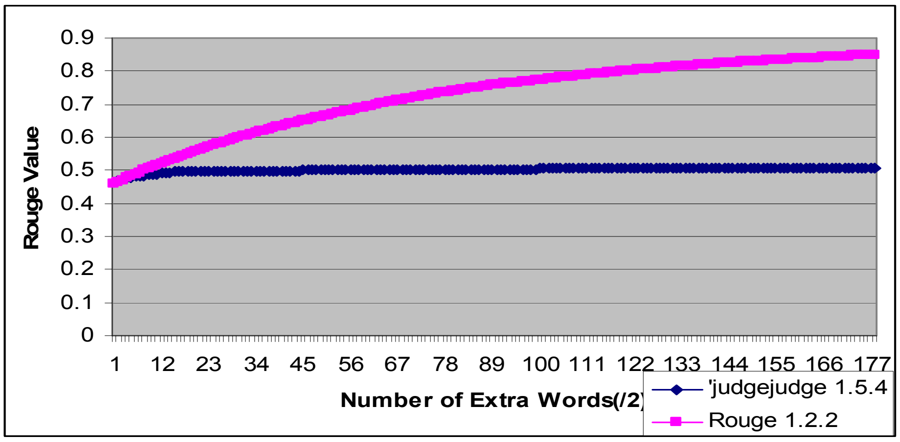


Figure A.1: The extra words and the ROUGE value comparison

In version 1.2.2 the ROUGE score increase with the length of the summary. In version 1.5.4.1 the ROUGE score is very sensitive to the actual summary length. Previous researchers might have been misled by the result and have been very encouraged to have outperformed the base line. In order make a solid foundation for our comparison, we identify the bugs and redo the experiment with the latest package, and we get the following result.

Team	Rouge 1.2.2	Rouge 1.5.5
Group 31	0.49142	0.46544
Group 29	0.46812	0.4641
Group 28	0.48906	0.48079
Group 27	**0.50117**	0.45733
Group 25	0.41112	0.41409
Group 23	0.42784	0.43179
Group 21	0.48693	0.4779
Group 19	0.45855	0.46204
Group 18	0.45102	0.44298
Group 16	0.4327	0.43717
Group 15	0.45859	0.45374
Group 1	0.47672	0.47103
Base line	0.47791	**0.48196**

Table A.1: The performance differentiation of all system using Rouge 1.2.2 and 1.5.5.

Table A.1 shows the results obtained on this dataset of 567 news articles from all the available groups. In the column 1 lists the systems from different group. The column 2 is the reported result that is evaluated by the ROUGE 1.2.2 package. While column 3 is the result generated by the latest ROUGE 1.5.5, which we have verified to be free of the sentence length bug.

As we can see, in total 4 systems outperforms the base line 0.47791. However, most of the system output more than 100 words as summary, and the ROUGE 1.2.2 package failed to cut to exact 100 words before evaluation. This bug is removed in ROUGE 1.5.5, and we use that package to evaluate. The actual performance is different; we see that not a single system has outperformed the baseline. According to ROUGE package evaluation, simply returning the 100 words of an article is still the best summarization we can achieve for single document.

Appendix 2: Stop word list

a	better	exactly	in	nevertheless	right	then	what
able	between	example	inasmuch	new	said	thence	whatever
about	beyond	except	inc	next	same	there	when
above	both	far	indeed	nine	saw	thereafter	whence
according	brief	few	indicate	no	say	thereby	whenever
accordingly	but	fifth	indicated	nobody	saying	therefore	where
across	by	first	indicates	non	says	therein	whereafter
actually		five	inner	none		theres	whereas
after	came	followed	insofar	noone	second	thereupon	whereby
afterwards	can	following	instead	nor	secondly	these	wherein
again	cannot	follows	into	normally	see	they	whereupon
against	cant	for	inward	not	seeing	think	wherever
all	cause	former	is	nothing	seem	third	whether
allow	causes	formerly	it	novel	seemed	this	which
allows	certain	forth	its	now	seeming	thorough	while
almost	certainly	four	itself	nowhere	seems	thoroughly	whither
alone	changes	from	just	obviously	seen	those	who
along	clearly	further	keep	of	self	though	whoever
already	co	furthermore	keeps	off	selves	three	whole
also	com	get	kept	often	sensible	through	whom
although	come	gets	know	oh	sent	throughout	whose
always	comes	getting	knows	ok	serious	thru	why
am	concerning	given	known	okay	seriously	thus	will
among	consequently	gives	last	old	seven	to	willing
amongst	consider	go	lately	on	several	together	wish
an	considering	goes	later	once	shall	too	with
and	contain	going	latter	one	she	took	within
another	containing	gone	latterly	ones	should	toward	without
any	contains	got	least	only	since	towards	wonder
anybody	corresponding	gotten	less	onto	six	tried	would
anyhow	could	greetings	lest	or	so	tries	would
anyone	course		let	other	some	truly	yes
anything	currently	had	like	others	somebody	try	yet
anyway		happens	liked	otherwise	somehow	trying	you
anyways	definitely	hardly	likely	ought	someone	twice	your
anywhere	described	has	little	our	something	two	yours
apart	despite	have	look	ours	sometime	un	yourself
appear	did	having	looking	ourselves	sometimes	under	yourselves
appreciate	different	he	looks	out	somewhat	unfortunately	zero
appropriate	do	hello	ltd	outside	somewhere	unless	
are	does	help	mainly	over	soon	unlikely	
around	doing	hence	many	overall	sorry	until	
as	done	her	may	own	specified	unto	
aside	down	here	maybe	particular	specify	up	
ask	downwards	hereafter	me	particularly	specifying	upon	
asking	during	hereby	mean	per	still	us	
associated	each	herein	meanwhile	perhaps	sub	use	

at	edu	hereupon	merely	placed	such	used
available	eg	hers	might	please	sup	useful
away	eight	herself	more	plus	sure	uses
awfully	either	hi	moreover	possible	take	using
be	else	him	most	presumably	taken	usually
became	elsewhere	himself	mostly	probably	tell	uucp
because	enough	his	much	provides	tends	value
become	entirely	hither	must	que	th	various
becomes	especially	hopefully	my	quite	than	very
becoming	et	how	myself	qv	thank	via
been	etc	howbeit	name	rather	thanks	viz
before	even	however	namely	rd	thanx	vs
beforehand	ever	ie	nd	re	that	want
behind	every	if	near	really	thats	wants
being	everybody	ignored	nearly	reasonably	the	was
believe	everyone	immediate	necessary	regarding	their	way
below	everything		need	regardless	theirs	we
beside	everywhere		needs	regards	them	welcome
besides	ex		neither	relatively	themselves	well
best			never	respectively		went
						were

References

[1] Sergey Brin, Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proceedings of the 7th International World Wide Web Conference, 1998*.

[2] Yudong Yang, HongJiang Zhang. HTML Page Analysis Based on Visual Cues. *In Proceedings of the International Conference on Document Analysis and Recognition 2001, Seattle, 2001.*

[3] Shipeng Yu, Deng Cai, Ji-Rong Wen, Wei-Ying Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. *In Proceedings of the 11th World Wide Web Conference (WWW 12), 2003*.

[4] Yu Chen, Wei-Ying Ma, Hong-Jiang Zhang. Detecting web page structure for adaptive viewing on small form factor devices. *In Proceedings of the 11th World Wide Web Conference (WWW 12), 2003*.

[5] Xiao-Dong Gu, Jinlin Chen, Wei Ying Ma, Guo-Liang Chen Visual Based Content Understanding towards Web Adaptation. *In Proceedings of 2nd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Spain, 2002.*

[6] Trevor, J. Hilbert, D.M., Schilit, B.N., Koh, T.K: From desktop to phone top, a UI for web interaction on very small devices. *In Proceedings of the 14th annual ACM symposium on user interface software and technology (UIST2001) , 2001*

[7] Buyukkokten, O., Garcia-Molina, H., Paepcke, A., T. Winograd. Power Browser: Efficient Web Browsing for PDAs. *In Proceedings of the ACM Conference on Computers and Human Interaction (CHI'00) , 2000.*

[8] Buyukkokten, O., Garcia-Molina, H., Paepcke, A. Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices. *In the Proceedings of the 10th World Wide Web Conference (WWW 10), 2001.*

[9] Bickmore, T., Schilit, B. Digester. Device Independent Access to the World Wide Web. *In the Proceedings of the Sixth International World Wide Web Conference (WWW 6), 1997.*

[10] H. Bharadvaj, A. Joshi, S. Auephanwiriyakul. An active transcoding proxy to support mobile web access. In Proceedings of 17th IEEE Symposium on Reliable Distributed Systems, West Lafayette, USA, 1998.

[11] Natasa Milic-Frayling, Ralph Sommerer. SmartView: Flexible Viewing of Web Page Contents. *In Proceedings of the 11th World Wide Web Conference (WWW 11), 2002.*

[12] Corin R. Anderson and Eric Horvitz. Web Montage: A Dynamic Personalized Start Page. *In Proceedings of the 11th World Wide Web Conference (WWW 11), 2002.*

[13] Corin R. Anderson, Pedro Domingos, and Daniel S. Weld. Adaptive Web Navigation for Wireless Devices. *In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), 2001.*

[14] Lan Yi, Bing Liu. Eliminating Noisy Information in Web Pages for Data Mining. *In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2003), 2003.*

[15] Lan Yi, Bing Liu. "Web Page Cleaning for Web Mining through Feature Weighting". *In the Proceedings of Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), 2003.*

[16] Ziv Bar-Yossef, Sridhar Rajagopalan. Template Detection via Data Mining and its Applications. *In Proceedings of the 11th World Wide Web Conference (WWW11), 2002.*

[17] Soumen Chakrabarti. Integrating the Document Object Model with Hyperlinks for Enhanced Topic Distillation and Information Extraction. *In the Proceedings of the 10th World Wide Web Conference (WWW 10), 2001.*

[18] Albert M., Jason N, Bhagyashree B,Vijayarka N, Abhishek P. Surana, and Suchita V. Improving Web Browsing on Wireless PDAs Using Thin-Client Computing. *In Proceedings of the 13th International World Wide Web Conference, 2004.*

[19] D. Billsus and M. J. Pazzani. A hybrid user model for news story classification. *In Proceedings of the Seventh Intl. Conference on User Modeling, pages 99–108. Springer-Verlag New York, Inc., 1999.*

[20] N. Kushmerick. Learning to remove internet advertisement. In O. Etzioni, J. P. M¨uller, and J. M. Bradshaw, editors, *In Proceedings of the Third International Conference on Autonomous Agents (Agents'99), USA, 1999.*

[21] Deng Cai, Xiaofei He, Ji-Rong Wen, and Wei-Ying Ma. Block-level link analysis. *In ACM SIGIR Conference (SIGIR), 2004.*

[22] R. Mihalcea Graph-based ranking algorithms for sentence extraction, applied to text summarization. *In Proceedings of the 42nd Annual Meeting of the Association for Computational Lingusitics (ACL),2004.*

[23] Ian H. Witten and Eibe Frank, Data Mining: Practical machine learning tools with Java implementations," Morgan Kaufmann, San Francisco, 2000.

[24] Jinlin Chen, Baoyao Zhou, Jin Shi, Hongjiang Zhang , Qiu Fengwu. Function-based Object Model towards Website Adaptation. *In Proceedings of 10th Thirteenth International World Wide Web Conference, 2001.*

[25] Zaiqing Nie, Yuanzhi Zhang ,JiRong Wen, WeiYing Ma. *Object Level Ranking: Bringing Order to Web Objects. In Proceedings of the 14th International World Wide* Web *Conference, 2005*

[26] Lawrence Kai Shih and David R. Karger. Using URLs and Table Layout for Web Classification Tasks. *In Proceedings of the 13th International World Wide Web Conference, 2004.*

[27] Mihalcea and P. Tarau. TextRank: bringing order into texts. *In Proceedings of Empirical Methods in Natural Language Processing (EMNLP) , pages 404-411, 2004.*

[28] Ruihua Song, Haifeng Liu, Jirong Wen, Wei-Ying Ma. Learning Block Importance Models for Web Pages. *In Proceedings of 13th International World Wide Web Conference, 2004.*

[29] Xinyi Yin, Wee Sun Lee, Zhenqiang Tan. Personalization of Web Content for Wireless Mobile Device. *IEEE Wireless Communications and Networking Conference 2004.*

[30] Larry Page, Sergey Brin, R. Motwani, T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *In Proceedings of the 7th World Wide Web Conference, 1998.*

[31] Suhit Gupta, Gail Kaiser, David Neistadt, Peter Grimm. DOM-based Content Extraction of HTML Documents. In *Proceedings of the 12th International Conference on World Wide Web*, 2003.

[32] Xinyi Yin, Wee Sun Lee. Using Link Analysis to Improve Layout on Mobile Devices. In *Proceedings of 13th International World Wide Web Conference*, 2004.

[33] Xinyi Yin, Wee Sun Lee. Towards Understanding the Functions of Web Element. *In Proceedings of Asia Information Retrieval Symposium, 2004.*

[34] DUC: Document understanding conference. http://www-nlpir.nist.gov/projects/duc/.

[35] *Chin-Yew Lin, Eduard Hovy*. Automatic evaluation of summaries using n-gram co-occurrence statistics. *In Proceedings of Human Language Technology Conference (HLT-NAACL) 2003.*

[36] P.J. Herings, G. van der Laan, and D. Talman. 2001. Measuring the power of nodes in digraphs. Technical report, Tinbergen Institute.

[37] R. Mihalcea, P. Tarau, and E. Figa. 2004. PageRank on semantic networks, with application to word sense disambiguation. *In Proceedings of the 20st International Conference on Computational Linguistics (COLING), 2004*.

[38] Orkut Buyukkokten, Hector Garcia-Molina, Andreas Paepcke. Seeing the whole in parts: text summarization for web browsing on handheld devices. *In Proceedings of 10th Thirteenth International World Wide Web Conference, 2001*

[39] H. P. Luhn, "The automatic creation of literature abstracts," IBM Journal of Research and Development, pp. 155--164, April 1958.

[40] Mark T. Maybury, Inderjeet Mani, Advances in Automatic Text Summarization, MIT Press, Cambridge, MA, 1999.

[41] P. Edmundson. "New methods in automatic extracting," Journal of the ACM, vol. 16, no. 2, pp. 264--285, 1969.

[42] J. J. Pollock and A. Zamor. "Automatic Abstracting Research at Chemical Abstracts Service," Journal of Chemical Information and Computer Sciences, 1975.

[43] Müürisep, Kaili and Pilleriin Mutso. ESTSUM - Estonian newspaper text summarizer. *In Proceedings of the Second Baltic Conference on Human Language Technologies. April 4-5, 2005.*

[44] Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Jamie Callan. Creating and evaluating multidocument sentence extract summaries. *In Proceedings of CIKM 2000.*

[45] T.W. Bickmore and B.N. Schilit, "Digester: Deviceindependent Access to the World Wide Web", *In Proceedings of 6th Thirteenth International World Wide Web Conference*, *1997.*

[46] Adam L. Berger and Vibhu O. Mittal. OCELOT: a system for summarizing web pages. In Research and Development in Information Retrieval, 2000.

[47] Sentence Extract Summaries. in CIKM'00: *In Proceedings of 9th International Conference on Information Knowledge Management. 2000.*

[48] Kaasinen E, Aantonen M, Kolari J, Melakoski S, Laakko T, two approaches to bringing Internet services to WAP devices. *In Proceedings of the ninth International World Wide Web conference, 2000.*

[49] Oren Kurland, Lillian Lee: PageRank without hyperlinks: structural re-ranking using links induced by language models. *In Proceedings of the 28th Annual Inernational ACM SIGIR, 2005.*

[50] Inderjit Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. *In Proceedings of the Seventh ACM SIGKDD Conference, 2001.*

[51] Gunes Erkan and Dragomir R. Radev. LexRank: Graph-based lexical centrality as salience in text summarization. Journal of Artificial Intelligence Research, 22:457-479, 2004.

[52] Eugene Garfield. Citation analysis as a tool in journal evaluation. Science, 178:471-479, 1972.

[53] Winfried K. Grassmann, Michael I. Taksar, and Daniel P. Heyman. Regenerative analysis and steady state distributions for Markov chains. Operations Research, 1985.

[54] Thorsten Joachims. Transductive learning via spectral graph partitioning. *In Proceedings of the International Conference on Machine Learning (ICML), 2003.*

[55] Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. Learning random walk models for inducing word dependency distributions. *In Proceedings of the International Conference on Machine Learning, 2004.*

[56] Jon Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM, 46:604 – 632, 1999.

[57] Wessel Kraaij and Thijs Westerveld. TNO-UT at TREC9: How different are web documents? *In Proceedings of the Ninth Text Retrieval Conference (TREC-9), 2001.*

[58] Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. Learning random walk models for inducing word dependency distributions. *In Proceedings of the International Conference on Machine Learning, 2004.*

[59] David R. H. Miller, Tim Leek, and Richard M.Schwartz. A hidden Markov model information retrieval system. *In Proceedings of SIGIR, pages 214 –221, 1999.*

[60] Barzilay, R., & Elhadad, M. Using Lexical Chains for Text Summarization. In Mani, I., & Maybury, M. T. (Eds.), Advances in Automatic Text Summarization. The MIT Press, 1999.

[61] Brandow, R., Mitze, K., & Rau, L. F. Automatic condensation of electronic publications by sentence selection. Information Processing and Management, 1995.

[62] Erkan, G., & Radev, D. R. Lexpagerank: Prestige in multi-document text summarization.In Lin, D., & Wu, D. (Eds.), *In Proceedings of Association for Computational Linguistics EMNLP, 2004.*

[63] Erkan, G., & Radev, D. R. The University of Michigan at DUC 2004. *In Proceedings of the Document Understanding Conferences Boston, MA, 2004.*

[64] Hatzivassiloglou, V., Klavans, J., Holcombe, M., Barzilay, R., Kan, M., & McKeown, K. Simfinder: A flexible clustering tool for summarization, 2001.

[65] Jing, H. (). Using hidden Markov modeling to decompose Human-Written summaries, 2002.

[66] Knight, K., & Marcu, D. Statistics-based summarization | step one: Sentence compression. *In Proceeding of the 17th National Conference of the American Association for Artificial Intelligence, 2000.*

[67] Kupiec, J., Pedersen, J. O., & Chen, F. A trainable document summarizer. *In Research and Development in Information Retrieval, 1995.*

[68] Chin-Yew Lin. Training a Selection Function for Extraction. *In Proceedings of the Eighteenth Annual International ACM Conference on Information and Knowledge Management (CIKM), 1999.*

[69] Mani, I., & Bloedorn, E. Multi-document summarization by graph search and matching. *In Proceedings of the Fourteenth National Conference on Artifcial Intelligence (AAAI-97) , 1997.*

[70] Kathleen R. McKeown, Vasileios Hatzivassiloglou, Regina Barzilay, Barry Schiffman, David Evans, Simone Teufel Columbia Multi-Document Summarization: Approach and Evaluation. *In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001.*

[71] Mihalcea, R., Tarau, P., & Figa, E. Pagerank on semantic networks, with application to word sense disambiguation. *In Proceedings of the 20th International Conference on Computational Linguistics, 2004.*

[72] Miles Osborne. Using Maximum Entropy for Sentence Extraction. *In Proceedings of the ACL-02 Workshop on Automatic Summarization, 2002.*

[73] Radev, D., Blair-Goldensohn, S., & Zhang, Z. Experiments in single and multidocument summarization using MEAD. *In First Document Understanding Conference, 2001.*

[74] Radev, D. R., Jing, H., & Budzikowska, M. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. *In ANLP/NAACL Workshop on Summarization, 2000.*

[75] G. Salton, A. Singhal, M. Mitra, and C. Buckley. Automatic text structuring and summarization. Information Processing and Management 1997.

[76] Toutanova, K., Manning, C., & Ng, A. Learning random walk models for inducing word dependency distributions. *In Proceedings of the International Conference on Machine Learning (ICML) , 2004*.

[77] Dagan, I., Lee, L., & Pereira, F. Similarity-based models of cooccurrence probabilities. Machine Learning, 1999.

[78] I-Mode NTT DOC http://www.nttdocomo.com/services/imode/index.html

[79] Baxendale, P. Man-made index for technical literature - an experiment. IBM J. Res. Dev., 2(4), 354–361, 1958.

[80] Sparck-Jones, K. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28(1), 11–20,1972.

[81] Hal Daumé III and Daniel Marcu. A phrase-based HMM approach to document/abstract alignment. *In Proceedings of ACL, EMNLP, 2004*.

[82] Moens, M.-F., Uyttendaele, C., & Dumortier, J. Abstracting of legal cases: the potential of clustering based on the selection of representative objects. J. Am. Soc, 1999.

[83] H. Zha, Generic Summarization and Keyphrase Extraction Using Mutual Reinforcement Principle and Sentence Clustering, SIGIR '02, August 11-15, 2002.

[84] Salton, V., Allan, J., Buckley, C., and Singhal, A. Automatic analysis, theme greeneration, and summarization of machine-readable texts, Readings in information retreival 1997, pp 478-483

[85] S.N. Dorogovtsev and J.F.F. Mendes. Evolution of networks. Submitted to Advances in Physics on 6th March 2001

[86] Paul E. Black and Paul J. Tanenbaum, "graph", in *Dictionary of Algorithms and Data Structures*, Paul E. Black, ed., U.S. National Institute of Standards and Technology. 1 September 2006.

[87] Florian Wolf and Edward Gibson. Representing Discourse Coherence: A Corpus-Based Study Computational Linguistics 31(2) , 2005, pp 249-287

[88] P-E E. Bergner. Dynamics of Markovian Particles; A kinetics of macroscopic particles in open heterogeneous systems , (2005)

[89] Sam Chapman. String Similarity Metrics for Information Integration http://www.dcs.shef.ac.uk/~sam/stringmetrics.html