

# **Web Page Classification Using Features from Titles and Snippets**

By

**Zhengyang Lu**

Thesis submitted to the  
Faculty of Graduate and Postdoctoral Studies  
in partial fulfillment of the requirements for the degree of  
Master of Science  
in Electronic Business Technologies



University of Ottawa  
Ottawa, Ontario, Canada  
May 2015

© Zhengyang Lu, Ottawa, Canada, 2015

## **Abstract**

Nowadays, when a keyword is provided, a search engine can return a large number of web pages, which makes it difficult for people to find the right information. Web page classification is a technology that can help us to make a relevant and quick selection of information that we are looking for. Moreover, web page classification is important for companies that provide marketing and analytics platforms, because it can help them to build a healthy mix of listings on search engines and large directories. This will provide more insight into the distribution of the types of web pages their local business listings are found on, and finally will help marketers to make better-informed decisions about marketing campaigns and strategies.

In this thesis we perform a literature review that introduces web page classification, feature selection and feature extraction. The literature review also includes a comparison of three commonly used classification algorithms and a description of metrics for performance evaluation. The findings in the literature enable us to extend existing classification techniques, methods and algorithms to address a new web page classification problem faced by our industrial partner SweetIQ (a company that provides location-based marketing services and an analytics platform).

We develop a classification method based on SweetIQ's data and business needs. Our method includes typical feature selection and feature extraction methods, but the features we use in this thesis are largely different from traditional ones used in the literature. We test selected features and find that the text extracted from the title and snippet of a web page can help a classifier to achieve good performance. Our classification method does not require the full content of a web page. Thus, it is fast and saves a lot of space.

## **Acknowledgements**

I would like to take this opportunity to express my appreciation to my supervisor Dr. Morad Benyoucef for his incredible support. He is the person who guided and encouraged me all the way through the research process, and who spent much time reading my thesis drafts and giving me valuable suggestions.

I am thankful to all my friends who have offered my help during the difficult times of my graduate study.

Last but not least, I would like to give special thanks to my parents who have always encouraged me throughout my studies and more importantly my life. Without their love and unconditional support, I could never have come this far.

## Table of Contents

Abstract .....	II
Acknowledgements .....	III
Table of Contents .....	IV
List of Tables.....	VII
List of Figures .....	VIII
List of Abbreviations.....	IX
Chapter 1: Introduction .....	1
1.1 Research Motivation.....	2
1.2 Research Objective .....	3
1.3 Research Methodology .....	4
1.4 Thesis Organization.....	8
1.5 Thesis Contribution .....	9
Chapter 2: Literature Review .....	12
2.1 Text classification.....	12
2.1.1 Definition of text classification.....	12
2.2 Web page classification .....	13
2.2.1 Definition of web page classification .....	14
2.2.2 Categories of web page classification.....	15
2.2.3 Features used in web page classification .....	16
2.3 Feature Selection .....	20
2.4 Feature Extraction .....	21
2.5 Classifiers .....	23
Chapter 3: Classification Method.....	29
3.1 Background.....	29
3.2 Data Collection .....	32
3.3 Definition of Classes .....	37
3.4 Feature Selection .....	45
3.4.1 Classical Text Features .....	46
3.4.2 Term Frequency-Inverse Document Frequency .....	46

3.4.3 Custom Features .....	47
3.4.4 Feature Set .....	48
3.5 Feature Extraction .....	50
3.5.1 Text Tokenization .....	51
3.5.2 Text Normalization .....	51
3.5.3 URL Tokenization .....	51
3.6 Class Imbalance Problem .....	52
3.6.1 Oversampling .....	53
3.6.2 Undersampling .....	53
3.6.3 Ensemble Theory Techniques .....	54
3.7 Conclusion .....	55
Chapter 4 Experiments and Results .....	55
4.1 Experimental Settings .....	55
4.2 Feature Selection and Extraction Process .....	56
4.2.1 Text Feature Selection and Extraction .....	56
4.2.2 URL Feature Selection and Extraction .....	59
4.3 Performance Evaluation .....	61
4.4 Classification with Word Tokens in Title and Snippet .....	64
4.4.1 Experiment 1: Comparing SVM with Random Forest .....	65
4.4.2 Experiment 2: Classification with Selected Word Tokens Using SVM .....	66
4.5 Classification with All Text Features from Title and Snippet .....	67
4.5.1 Experiment 3: Classification with Word Tokens and Custom Text Features .....	67
4.5.2 Experiment 4: Classification with Selected Word Tokens and Custom Text Features .....	68
4.6 Classification with URL .....	69
4.6.1 Experiment 5: Classification with URL Features Using SVM and Random Forest .....	69
4.7 Classification with Balanced Data Set .....	70
4.8 Discussion .....	70
Chapter 5: Conclusion .....	73
5.1 Summary of the Research .....	73

5.2 Limitations and Future Works.....	74
Appendix A: Python Code of Extracting Title and Snippet from JSON File .....	75
Appendix B: Python Code of Extracting URL from JSON File .....	76
Appendix C: Python Code of Searching for City in Text .....	77
Appendix D: Python Code of Searching for Phone Number in Text.....	78
Appendix E: Python Code of Searching for Region in Text.....	79
Appendix F: Python Code of Searching Street Type in Text.....	80
Appendix G: Python Code of Search for Target Business Name in Text and URL Domain .	85
Appendix H: Python Code of Searching for Zip Code and Postal Code in Text.....	86
Appendix I: Python Code of Splitting URL into Different Components.....	87
Appendix J: Python Code of Searching for Directory Provider in URL Domain .....	88
Appendix K: Python Code of Searching for Review Provider in URL Domain .....	92
References .....	94

## List of Tables

Table 1: Web Page Classification Decision Matrix .....	15
Table 2: Example of Bag of Words Representation.....	22
Table 3: Information of SweetIQ's Ten Clients .....	33
Table 4: Search Queries .....	34
Table 5: Data Distribution.....	45
Table 6: Feature Set .....	49
Table 7: Definition of TP, TN, FP, P and N (Han & Kamber, 2012) .....	62
Table 8: Confusion Matrix (Han & Kamber, 2012).....	62
Table 9: Evaluation Measures (Han & Kamber, 2012).....	64
Table 10: Addition Evaluation Metrics (Han & Kamber, 2012) .....	64
Table 11: Result of Experiment 1 Using SVM .....	65
Table 12: Result of Experiment 1 Using Random Forest .....	66
Table 13: Result of Experiment 3 Using SVM .....	68
Table 14: Result of Experiment 3 Using Random Forest .....	68
Table 15: Result of Experiment 4 .....	69
Table 16: Result of Experiment 5 Using SVM .....	70
Table 17: Result of Experiment 5 Using Random Forest .....	70
Table 18: Result of Experiment 6 Using SVM .....	70

## List of Figures

Figure 1: A Typical Process Model of Design Science Research (Vaishnavi & Kuechler, 2004) .....	5
Figure 2: Knowledge Contribution Framework (Vaishnavi & Kuechler, 2004) .....	7
Figure 3: All Phases in Our Classification Method .....	29
Figure 4: A Web Page Returned by Bing with a Specific Search Query .....	35
Figure 5: Example of a Profile Page .....	38
Figure 6: Example of a Business Search Result Page .....	39
Figure 7: Example of a Profile Page .....	39
Figure 8: Example of A Review Page .....	42
Figure 9: Example of a Target Profile Page .....	42
Figure 10: Example of a Target Review Page .....	44
Figure 11: Result of Experiment 2 Using SVM (X-axis represents the number of features) ..	67



## **List of Abbreviations**

BPN: Back Propagation Network

DSR: Design Science Research

DSRM: Design Science Research Methodology

FOIL: First Outer Inner Last

FP: False Positive

GA: Genetic Algorithm

HTML: HyperText Markup Language

IDF: Inverse Document Frequency

k-NN: k-Nearest Neighbours

KPI: Key Performance Indicator

LSA-SVM: Latent Semantic Analysis Support Vector Machine

JSON: JavaScript Object Notation

N: Negative

NLTK: Natural Language Toolkit

P: Positive

PILFS: Predicate Invention for Large Feature Spaces

SVM: Support Vector Machine

TF: Term Frequency

TN: True Negative

TP: True Positive

URL: Uniform Resource Locator

WVSVM: Weighted Voting Support Vector Machine

## **Chapter 1: Introduction**

Over the past two decades, we have witnessed an exponential growth of the Internet. People all over the world can easily get access to billions of web pages. Internet is a powerful tool for people to get information across the world but it is not a tool for locating or organizing massive information. As a result, search engines such as Google, Yahoo and Bing have been developed to assist people in locating information on the Internet. These search engines are good at locating information but provide limited ability in organizing web pages (Qi & Davison, 2009). Search engines only return a ranked list of web pages based on a keyword and web pages of different topics and functions are mixed together in the list. People have to carefully sift through the long list to find pages of interest (H. Chen & Dumais, 2000). The large amount of noisy information makes it more difficult to find the right information. As a result, we need tools that can help us to make a relevant and quick selection of information that we are seeking (Choi & Yao, 2005); and then companies can take advantage of such tools to develop online marketing strategies as they can pinpoint the information they are looking for in the web. One of the most promising approaches to achieve this goal is web page classification.

Web page classification is the process of assigning a web page to one or more predefined categories (Qi & Davison, 2009). Classification is often described as a supervised learning problem in which a set of labeled data is used as training set to train a classifier which later is applied to new data to determine their categories (Qi & Davison, 2009). Web page classification techniques can help search engines to effectively deal with web pages and put them into categories (Slawski, 2010a). There is evidence that web page classification is

expected to play a significant role in search services in the future. Google filed a patent that focuses on classification of documents based on a wider range of information, including user behavior data and query profiles. The patent describes a method of creating profiles for web pages that include classification information. This information can later be used to unclassified web pages through query profiles. This kind of user-based information can be used with ranking algorithms to improve the quality of search results and provide personalized results to people (Oztekin & Chiu, 2014)(Slawski, 2010b).

SweetIQ is a company in local search marketing and provides location-based marketing and analytics platforms. For companies like SweetIQ, it is important to build a healthy mix of listings on search engines, large directories, niche directories, blogs, and wikis, because this can provide local analytics and insights for large brands and marketing agencies. In order to build such listings, are tasked with proposing a web page classification method based on the business needs and requirements of SweetIQ.

## **1.1 Research Motivation**

Classification is a supervised learning problem where a classifier is trained using labeled training examples with predefined classes and then applied to predict the classes of new examples. Web page classification means much more than merely assigning labels to web pages. It is crucial to focused crawling and it also supports web search and advertising, making it a significant topic of broad interest (Qi, 2012).

Currently at SweetIQ, the crawling system crawls all pages from the web when given a search query and classification of web pages is done manually or partly supported by software, which costs time and effort. It is becoming increasingly important to have an

automatic classification system that can identify web pages of different interests and disregard other web pages which are not relevant.

A lot of research focuses on how to improve the existing classification algorithms used to classify web pages, and most of the research is based on web page content. However, there is little focus on exploring new features that can be used to classify web pages without the content of web pages. Classification algorithms proposed in the literature are usually tested on a few public data sets, such as the 4 Universities data set<sup>1</sup> and the 20 Newsgroups data set<sup>2</sup>. However, our data is from the industry which makes it quite different from the public data. We have a unique way to crawl the data and the data content and structure are special. Therefore, features selected and then extracted from our data will also be different. As a result, we need to do research about features and classifiers that are suitable in our case.

## **1.2 Research Objective**

In order to develop a web page classification method that can meet the organization's (i.e., SweetiQ) requirements, existing methods, approaches and models from the literature will be reviewed and analyzed in detail. First, we will review the definition and categories of web page classification. Second, we will analyze what features are useful for classifying web page. Third, we will discuss feature selection and feature extraction methods. Fourth, we will evaluate some classifiers that have been widely used in research and industry. At last, we will review a set of metrics used to measure classification performance. Based on the review

---

<sup>1</sup> <http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/>

<sup>2</sup> <http://qwone.com/~jason/20Newsgroups/>

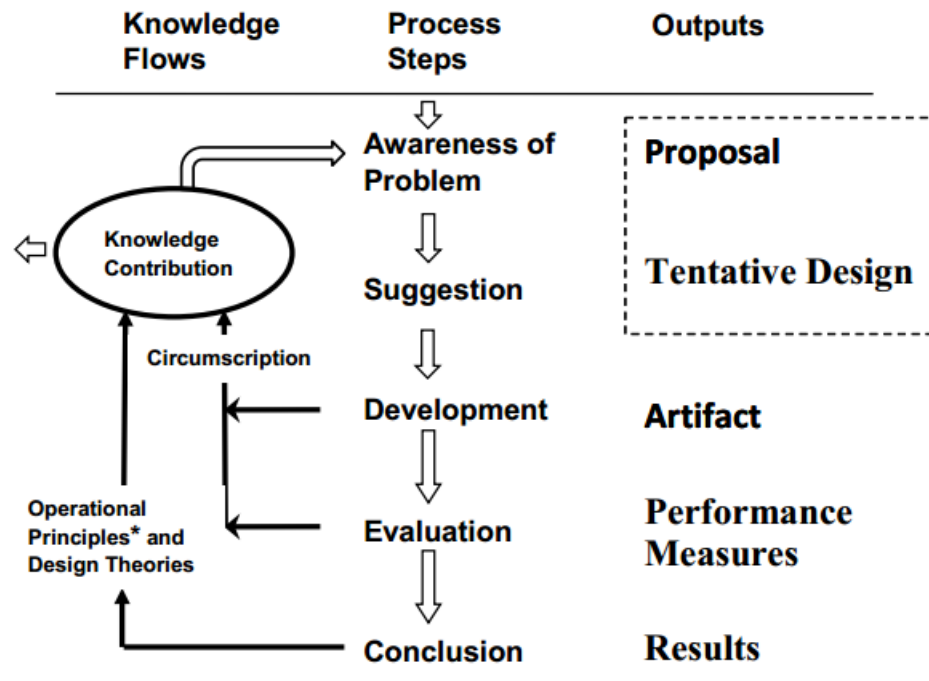
and analysis, features will be extracted from our data and classifiers will be implemented and tested. Therefore, there are two main objectives in this research. One is that we need to find the useful features other than features from web page content that can lead to a good classification performance. It is mainly because that to get the content of a web page, we need to access the web page, download it, and store it. The process takes a lot of time and therefore is not efficient. Downloading web pages also requires a lot of storage space. The other one is that we need to test and tune classifiers and find a suitable one for web pages classification based on our data and features.

### **1.3 Research Methodology**

One research methodology that is appropriate for this research is design science research methodology (DSRM). Design science research (DSR) is a set of synthetic and analytical techniques and perspectives for performing research in information systems (IS). DSR involves creating new knowledge through the design of novel or innovative artifacts and analyzing the use/performance of such artifacts with reflection and abstraction to improve and understand the behavior of aspects of information systems (Vaishnavi & Kuechler, 2004). In this research, such artifacts include algorithms and classifiers. These artifacts need to be built and evaluated to satisfy the requirements of DSRM (Peppers, Tuunanen, Rothenberger, & Chatterjee, 2007).

The typical process model of DSR, which includes five steps, is described in Figure 1 (Vaishnavi & Kuechler, 2004). It starts with the awareness of the problem that may come from multiple sources and will result in a proposal. Based on the problem in the first step, a possible suggestion is raised in the second step with a tentative design as output. The third

step concentrates on developing a solution and building an artifact, the performance of which will be tested and evaluated in the fourth step using predefined criteria. The outcome of the last step is typically the result.

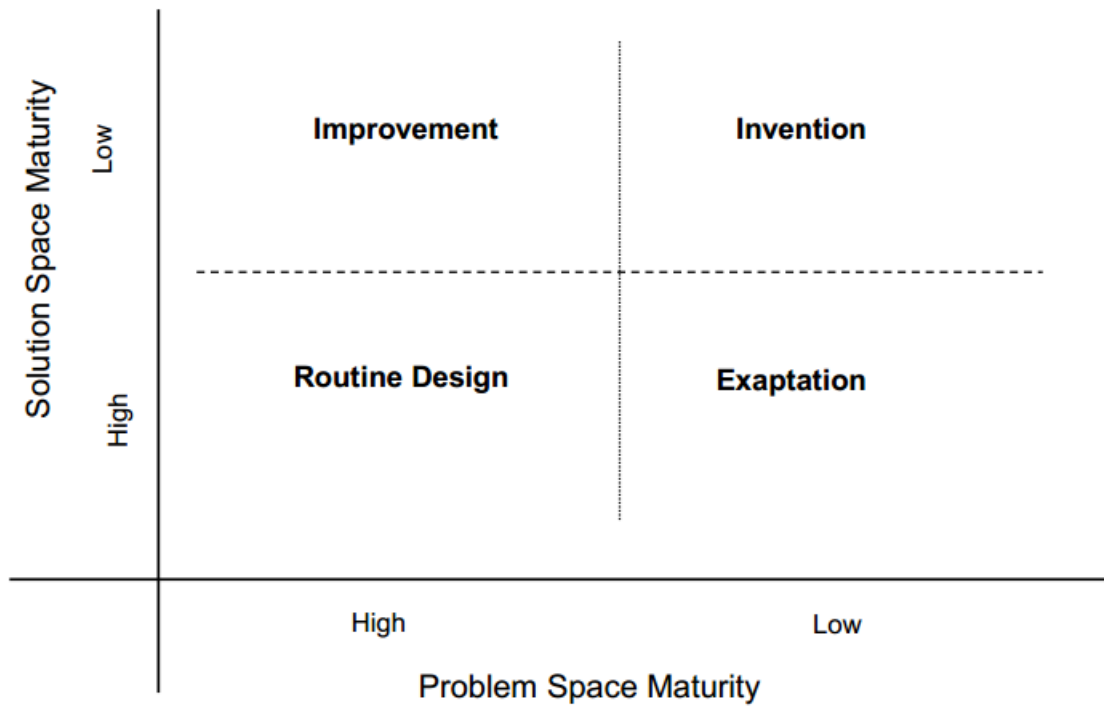


**Figure 1: A Typical Process Model of Design Science Research (Vaishnavi & Kuechler, 2004)**

The typical levels of knowledge derived from DSR can be explicated through four outputs that are constructs, model, method, and instantiation (Vaishnavi & Kuechler, 2004). Constructs are the conceptual vocabulary of a domain. Models represent the set of propositions or statements expressing relationships among constructs. Methods are the set of steps used to perform a particular task. Instantiations enable the realization of artifacts in their own environment to demonstrate the feasibility and effectiveness of the models and methods they contain.

This research starts by studying existing techniques, methods and algorithms in the context of web page classification. Both advantages and disadvantages of these techniques, methods and algorithms are analyzed and used to explore new problems that may not be fully addressed in terms of web page classification. For example, new features may be found to be useful for classification and features are added or deleted depending on the results of experiments. What's more, selected classifiers built on different algorithms are implemented for performance evaluation. In the development phase, new classification methods are proposed. Then in the evaluation step, these methods are implemented and tested with real data collected and made available to us by the organization (SweetIQ). It is worth noticing that the data in question are crawled from the web by a crawler developed by the organization. We introduce metrics that are used to evaluate classification performance and new methods will be tested based on the metric. Finally, in the conclusion step, we present our findings in the experiment step.

A knowledge contribution framework for DSR has been proposed by Gregor and Hevner (2013). In this framework, "Improvement (new solutions for known problems), Invention (new solution for new problems), Exaptation (non-trivial extension of know solutions for new problems) can be research contributions while Routine Design by itself would be considered as a research contribution. (Vijay Vaishnavi and Bill Kuechler, 2013)"



**Figure 2: Knowledge Contribution Framework (Vaishnavi & Kuechler, 2004)**

Web page classification has been studied for over a decade. Different problems of web page classification have been raised and then solved; classification algorithms have been improved and applied to web page classification; and traditional classifiers for text classification have been tweaked for web page classification. However, in this research, we are facing a new and unique problem from a company. The problem requires new methods that can tackle it from a new angle and can meet the company's expectations. Therefore, our proposed method is a research contribution as an exaptation.

In this research, we also follow the Cross Industry Standard Process for Data Mining (CRISP-DM) methodology. CRISP-DM is a data mining process model that describes commonly used approaches that data mining experts use to tackle problems (Shearer, 2000).



CRISP-DM has six major phases – business understanding, data understanding, data preparation, modeling, evaluation, and deployment. The first phase focuses on understanding the research or the project’s objective and requirement. In our research, the objective has been clarified in section 1.2 and the business requirement is to have an automatic web page classification process that can classify web pages into profile, search result and review. The second phase starts with an initial data set, which in our research contains web pages, crawled using specific queries. We then identify potential problems in the data set and discover the insights into data. In our research, we have data pre-process to make sure the data is clean and of high quality. The third phase covers all activities to construct the final data set. These activities can include table, record, and attribute selection. In this phase, we select features and extract them from our data. In the fourth phase, various modeling methods can be applied and we applied several machine learning algorithms and natural language process techniques in this process. In the fifth one, the methodology tells us that the model needs to be evaluated before the deployment. In our research, we have introduced several evaluation metrics for classification problems and applied them to our model. In the last phase, the company carries out the deployment.

## **1.4 Thesis Organization**

This thesis is organized in five chapters. In chapter 2, we review the literature of related concepts on web page classification such the definition of text classification and web page classification, and features that can be used in web page classification. Then we discuss the importance of feature selection. We present commonly used feature selection methods and techniques. We also discuss feature extraction techniques. We analyze and compare some popular classification algorithms that can be used in web page classification. At last, we

introduce some metrics used to evaluate a classifiers' performance such as precision, recall and F-measure.

In chapter 3, we introduce the organization where this research took place, namely SweetIQ as well as the techniques currently used by SweetIQ in areas related to web page classification. Then we describe the organization's problem that we need to solve in our research. We present our data collection method and describe the structure of our data. We analyze the feature selection methods and list the features we have selected. We present the feature extraction techniques and apply them to our data. Finally, we discuss the class imbalance problem and provide several solutions to this problem.

In chapter 4, we conduct several experiments with different feature sets and test Support Vector Machine and Random Forest classification algorithms on our data set, because they outperform other algorithms in most cases (Fernández-Delgado, Cernadas, Barro, & Amorim, 2014). Then we list and discuss the results of these experiments.

In chapter 5, we conclude the thesis by providing a summary of our research contributions and discussing the research limitations and future works.

## **1.5 Thesis Contribution**

In this thesis, we provide a generic method to classify web pages with features extracted from titles and snippets. Most web page classification methods are developed based on the content of web pages. This means, in order extract features, we need to access web pages and download them to a database or disk. The accessing and downloading process require extra time. It also costs much storage space to store web pages. In our method, we don't need to access any web page, and we only need to get the basic information of web pages— titles and

snippets that are returned by search engine. The features extracted from titles and snippet also provide good classification performance. As a result, it saves a lot of time and space, making the classification process more efficient.

Even though there are many classification algorithms and methods proposed in the literature and used in the industry, we did not know initially whether or not these algorithms or methods can be extended to address our problem. Therefore, we reviewed and compared classification algorithms to find the best suitable one. We also extracted and selected different features to find the most effective ones.

With respect to the analysis and findings of our research, the following objectives we defined at first have been achieved:

1. Find useful features other than features extracted from web page content that can lead to a good classification performance

After cleaning and analyzing our data, we first defined 16 candidate features that we thought might be helpful. These features include text related features and URL related features. Then we performed several experiments to test these features and we came to a conclusion that we can achieve over 80% precision and recall using text features extracted from title and snippet. Compared to the whole content of a web page, the title and snippet take much less space to store and much less time to crawl. This helps us to use less data to train a classifier as well as to predict web pages in the future. It also saves time to access and download every web page.

2. Find a suitable classification algorithm for this project

In order to find a suitable classification algorithm, we reviewed and compared commonly used algorithms and found in the literature that Support Vector Machine and Random Forest are the two best classification algorithms in most cases. Then we tested the two algorithms using our data set and features. We found that both algorithms yielded good performance, but SVM gave us better precision while Random Forest beat SVM in terms of recall. Both algorithms may be adopted depending on different situations.

In addition, we showed that applying the oversampling technique to our imbalanced data set can improve the classification precision.

## **Chapter 2: Literature Review**

In this chapter, we first review the concept of web page classification first by addressing text classification as web pages mostly consist of text. Then we introduce different categories of web page classification because different methods are applied when classifying web pages based on types and topics. As features extracted from web pages are used in every classification method, we also analyze different features that can be extracted from web pages and used in web page classification. Most research papers focus on features extracted from web content and there is less research on other features that can be used for web page classification. With a complete feature set, we need to apply feature selection and extraction techniques to convert features to the data that can be read by classifiers. Therefore, we review various feature selection and extraction techniques in terms of web page classification. At last, we analyze and compare different classifiers that are commonly used in web page classification.

### **2.1 Text classification**

Classification is a supervised machine learning task where a model or classifier is constructed to predict class labels. Classification is a two-step process. First is a learning step where a classifier is built from training data. Second is a classification step where the classifier built in the first step is used to predict class labels for given data.

Automatic text classification (or categorization) was developed over several years and now is an important application in industry and a research topic that has been widely studied.

#### **2.1.1 Definition of text classification**

According to (Sebastiani, 2002), text classification is defined as the task of “assigning a Boolean value to each pair  $\langle d_j, c_i \rangle \in D \times C$ , where  $D$  is a domain of documents and  $C = \{c_1, \dots, c_{|C|}\}$  is a set of predefined categories”. A TRUE value is assigned to  $\langle d_j, c_i \rangle$  if file  $d_j$  belongs to class  $c_i$ , while a FALSE value is assigned to  $\langle d_j, c_i \rangle$  if file  $d_j$  does not belong to class  $c_i$ . More precisely, the task is to “approximate a target function that describes how documents ought to be classified” (Sebastiani, 2002).

## 2.2 Web page classification

Even though web page classification shares a lot of similarities with text classification, it is much more complicated than text classification. Web pages are highly erratic in nature because of the variable size, a large number of HTML tags, different formats and a lot of noise content such as advertisement banners and flash, all of which make it clear that learning over web pages is difficult and has new characteristics. Therefore, web pages should be preprocessed before the application of any classification algorithm. Moreover, since web pages are presented in multiple ways, different web page preprocessing methods have been proposed and used in early research. After preprocessing, web pages are usually represented as multi-dimensional vectors that may contain different types of values such as numeric values and strings. Each dimension in a vector represents a single feature (or attribute) of the web page. If all possible features that can be used to represent a web page are extracted and converted into a vector, the dimensionality of the vector will be very high. This will result in a high consumption of time and space, or in other words, a high time and space complexity. In order to reduce the dimensionality, various dimensionality reduction methods have been proposed and evaluated in different cases. Feature selection is one of the most useful dimensionality reduction methods. Dimensionality reduction can also help to reduce the

problem of overfitting. Overfitting is a phenomenon where a classifier is tuned into the training set, rather than just the essential characteristic of the training set to classify new instances (Sebastiani, 1999).

### 2.2.1 Definition of web page classification

Web page classification, also known as web page categorization, is a type of supervised machine learning problem that aims to classify a web page into one or more predefined classes based on labeled training data (Qi & Davison, 2009). Machine learning techniques on text documents have been well studied during the last two decades and these techniques are referred to as text learning. Machine learning techniques are similar to text learning, because web pages can be treated as text documents. Therefore, one definition of web page classification can be derived from text classification.

Web page classification is the task of assigning a value  $a_{ij} \in \{0, 1\}$  to each pair  $(d_i, c_j) \in D \times C$ , where  $1 \leq i \leq N, 1 \leq j \leq K, D = \{d_1, \dots, d_N\}$  is a set of web pages to be classified and  $C = \{c_1, \dots, c_K\}$  is a set of predefined classes.  $A = D \times C$  can be described as the following decision matrix:

Web Pages		Classes				
		$c_1$	$\dots$	$c_j$	$\dots$	$c_K$
$d_1$		$a_{11}$	$\dots$	$a_{1j}$	$\dots$	$a_{1K}$
$\dots$		$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$d_i$		$a_{i1}$	$\dots$	$a_{ij}$	$\dots$	$a_{iK}$
$\dots$		$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

$d_N$	$a_{N1}$	$\dots$	$a_{Nj}$	$\dots$	$a_{NK}$
-------	----------	---------	----------	---------	----------

**Table 1: Web Page Classification Decision Matrix**

$a_{ij} = 1$  indicates that web page  $d_i$  belongs to the class  $c_j$ , while  $a_{ij} = 0$  indicates that web page  $d_i$  does not belong to class  $c_j$ . A web page can belong to more than one class. More formally, the task is to approximate the unknown function  $f: D \times C \rightarrow \{0, 1\}$  by means of a learned function  $f': D \times C \rightarrow \{0, 1\}$  called a classifier, a hypothesis, a model or a rule, such that  $f'$  coincides to  $f$  as much as possible (Sebastiani, 2002)(Choi & Yao, 2005).

### 2.2.2 Categories of web page classification

The problem of web page classification can be divided into several categories, such as subject-based classification, functional classification (also called genre-based classification), and sentiment classification (Qi & Davison, 2009). The most cited classification problems in the literature are subject-based classification. Subject-based classification is concerned with the subject (or topic) of a web page. A subject can be “business”, “technology” or “sports”, etc. Functional-based classification cares about the role that the web page plays. Examples of web page functions (genres) are “home page”, “product catalogue”, and “frequently asked questions”.

Web page classification is further categorized into binary classification and multi-class classification, based on the number of classes (Qi & Davison, 2009). Binary classification categorizes a web page into exactly one of two classes, while multi-class classification considers more than two classes. For example, in a binary classification problem, a web page can only be classified as “commercial” or “non-commercial”, while in a multi-class classification problem, a web page can be classified into one class among “business”, “arts”,



and “sports”. In multi-class classification, a web page can be either single-label or multi-label (Qi & Davison, 2009). In single-label classification, one and only one label can be assigned to a web page. In multi-label classification, more than one label can be assigned to a web page. For example, if there is a web page involving three classes that are “business”, “technology” and “sports”, it can either be single-label, where only “business” is assigned to the web page, or multi-label, where both “business” and “technology” are assigned to the web page.

### **2.2.3 Features used in web page classification**

Even though web page classification is similar to text classification, it is clear that machine learning on web pages has new characteristics that cannot be overlooked. Web pages are semi-structured text documents written in HTML; and an HTML file is much more complex than a simple text file. Therefore, web pages have their own structure to present content with HTML tags. These tags are important in web page classification since they can be used as features of web pages. Furthermore, unlike text documents, web pages are usually connected to each other via hyperlinks that are useful for determining which class a web page belongs to. Finally, the sources of web pages are numerous, heterogeneous, distributed and dynamically changing (Choi & Yao, 2005). Regardless of the great effort that has been made in the area of text classification, text mining techniques or text classification algorithms cannot be directly used for web page classification.

The first step in web page classification is to transform a web page. A web page usually consists of strings of characters, HTML tags, images, hyperlinks and anchor text (the text to be clicked on to activate and follow a hyperlink to another web page, placed between HTML

<A> and </A> tags), all of which can be considered as features that may be useful for web page classification (Choi & Yao, 2005)(Qi & Davison, 2009). Since a web page is represented by a feature vector in web page classification, and feature selection and extraction are equally important to classification algorithms, web-specific features need to be first reviewed and analyzed.

Web-specific features can be divided into two categories - on-page features and features of neighbors. On-page features are directly located on the web page to be classified. These features can be further divided into textual features and visual features (Qi & Davison, 2009). Textual content is the most straightforward feature that can be considered to use. Researchers have tried various methods to take advantage of textual features in order to classify web pages, such as feature selection, bag-of-words representation and N-gram representation. These methods are mostly inspired by text mining research. Since each of them has its own merits and drawbacks, one method is usually performed in combination with another one. Besides the textual content of web pages, another on-page feature that must be considered is HTML tags. It has been demonstrated that using information contained in HTML tags and the HTML tags themselves can largely boost a classifier's performance. Since HTML tags are used to assign certain properties to the text on the web page, if fractions of the text appear between two HTML tags, the portion of the included text assumes such tags. These HTML tags can be selected to calculate the relevance of words on a web page (Ribeiro, Fresno, Garcia-Alegre, & Guinea, 2003). Kwon and Lee (2000) proposed a web page approach based on the k-Nearest Neighbor (k-NN) algorithm, in which terms are assigned different weights using HTML tags annotated with terms (Kwon & Lee, 2000). This approach has improved the efficiency of the traditional k-NN approach as well as the performance of the classifier.

Özel (2011) developed a web page classification system based on a genetic algorithm (GA) using tagged-terms as features. The system takes both terms and HTML tags at the same time on a web page as features, while other systems consider only terms or only HTML tags, not both as features. In the research in question, Özel examined HTML tags including <title>, <h1>, <h2>, <h3>, <a>, <em>, <strong>, <b>, <i>, <p>, and <li> and found out that most of the domain specific terms are under these tags and these terms are quite important for classifier learning. The research also compared the system with other GA-based systems that take only terms as features and the result shows that it performs better than other GA-based systems (Özel, 2011). In addition, each web page has at least two representations – a textual representation and a visual representation. Although most classification approaches focus on textual representations, visual representations can be useful as well when textual features are not available. Shen et al. found that summarization of web content can improve accuracy. They proposed a new summarization algorithm that extracted the main topic of a web page and enhanced the performance by about 12.9%. However, this algorithm focuses on topic-based web page classification and may not be useful in genre-based web page classification (Shen et al., 2004).

Besides on-page features, feature of neighbors, or in other words, other pages linked to the page to be classified, sometimes can be more reliable and can largely improve the performance of a classifier. For instance, some home pages only contain images and flash objects instead of text. In this case, it is difficult for classifiers to utilize on-page features; hence features must be extracted from neighboring pages. Among many connections between pages, the most obvious and commonly used connection is the hyperlink.

Hyperlinks among web pages demonstrate certain patterns that can be helpful for web page

classification and are usually hard to capture with traditional statistics models. Oh, Myaeng and Lee proposed a categorization method for hypertext documents. When compared to the conventional method using Naïve Bayesian and the HyperClass method, their method enhanced the quality and speed for hypertext categorization using hyperlinks (Oh, Myaeng, & Lee, 2000). Lu and Getoor (2003) proposed a framework for modeling link distributions, a link-based model that supports discriminative models describing both the link distribution and the attributes of linked objects. By using the link structure, the model has been shown to improve classification accuracy (Lu & Getoor, 2003). This link-based model can also be extended to link-based web page classification problems as a guideline. Furthermore, Kan and Thi (2005) demonstrated the usefulness of the uniform resource locator (URL) alone in performing web page classification. The authors considered a classifier that is restricted to using the URL as the sole source input. Such a classifier magnitudes faster than traditional classifiers as it does not require fetching pages or parsing the text (Kan & Thi, 2005). Kan and Thi (2005) proposed a web page classification approach that is restricted to using the URL as the only source of input (Kan & Thi, 2005). In their approach, a URL is first tokenized using information-theoretic measures and then tokens are fed into a module where useful features are extracted for classification. They developed a set of feature classes including baseline; segments of entropy reduction; URL components; length; orthographic; Sequential Bi-, Tri-, 4-gram; and precedence bigram. Their experiment results have showed that URL features perform well on web page classification tasks and in certain cases their approach has paralleled or exceeded the performance of full-text approaches.

Another feature that is used to classify web pages is the hierarchical structure. For example, Dumais and Chen used hierarchical structure to classify hierarchical-related web pages

(pages in “Computer” category and pages in “Computer/Software” category are hierarchical-related). And they found a small advantage in classification performance when using hierarchical models, compared with a non-hierarchical model. However, their method is only useful when web pages are hierarchically related (Dumais & Chen, 2000).

## **2.3 Feature Selection**

Feature selection (or dimensionality reduction) is used to reduce the dimensionality of feature space in order to improve classification accuracy or boost classifiers’ performance on very high dimensional data sets. Since emphasizing features that are more discriminative usually boosts performance of classifiers, feature selection plays an important role in classification (Qi & Davison, 2009).

There are a variety of feature selection methods and many of them have proved effective. Wibowo and Williams (2002) used a simple feature selection approach that extracts the first fragment of each training document (Wibowo & Williams, 2002). The rationale behind this approach is that a summary of each document is presented at its beginning and their experiment has proved it. There are also some feature selection approaches developed from text classification using well-known metrics such as document frequency, information gain, and mutual information. Joachims (1998) explored the use of support vector machine in text categorization, in which features are ranked and selected using information gain (Joachims, 1998). Calado et al. (2003) also used information gain to select features when they were studying the combination of the link-based and content-based measures in web document classification (Calado et al., 2003). Kwon and Lee (2000) used expected mutual information and mutual information measurements to reduce the noise of a training document when they proposed a classification approach based on the k-Nearest Neighbour (Kwon & Lee, 2000).

Riboni proposed a new structure-oriented weighting technique that combined the number of occurrences of terms in web pages and the HTML element the terms are presented in. This technique guaranteed better performance than techniques only using term frequency (Riboni, 2002).

## **2.4 Feature Extraction**

Feature extraction is very different from feature selection. Feature extraction consists in transforming arbitrary data, such as text and image, into numerical features that can be taken by classifiers, while feature selection is a machine learning technique applied on these features.

Feature extraction techniques are used to extract features in a certain format that is supported by machine learning algorithms from data sets consisting of formats such as text and image. One popular text feature extraction technique is the Bag of Words representation. Since the text as raw data is a sequence of symbols that cannot be directly fed to machine learning algorithms as most algorithms take numerical feature vectors with a fixed size rather than the text documents with variable length. In order to address this, we usually apply several common ways to extract numerical values from text, such as tokenizing, counting and normalizing. These are explained below.

1. Tokenizing means that strings are first tokenized and then an integer id is given to each possible token. For instance, text is usually tokenized by using white space and punctuations as separators.
2. Counting means that the occurrences of tokens or words are counted in each document.

3. Normalizing means the text is transformed into a single canonical form before it is analyzed. For example, all the words in a text document are transformed to lower case.

In the Bag of Words representation, “each individual token occurrence frequency is considered as a feature, and the vector of all the token frequencies for a given document is considered as a multivariate sample” (Qazi, Raj, Tahir, Cambria, & Syed, 2014). Therefore, a corpus of text documents can be represented by a matrix with one row per document and one column per token occurring in the corpus. Table 2 shows an example of this matrix.

Each cell in the table represents the occurrence of a word in a document.

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
Document 1	0	4	0	0	3	1	0
Document 2	3	0	0	3	0	0	1
Document 3	0	0	1	6	3	2	0
Document 4	0	0	0	0	1	3	1

**Table 2: Example of Bag of Words Representation**

This specific method (tokenization, counting and normalizing) is called the Bag of Words representation. Text documents are described by word occurrences when the relative positions of words in the document are entirely ignored.

## 2.5 Classifiers

**Decision Tree** is a non-parametric supervised learning method used for classification. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A decision tree is a tree-like structure where each internal node denotes a test on a feature, each branch represents an outcome of the test, and each leaf node holds a class label (Pant, Pant, & Pardasani, 2009).

Some advantages of decision trees are:

1. Decision trees are simple to understand and to interpret. The construction of a decision tree does not require any domain knowledge or parameter setting. Moreover, decision trees can be visualized.
2. Decision trees require little data preparation. Other techniques often require data normalization; and dummy variables need to be created and blank values to be removed.
3. The cost of using a decision tree is logarithmic in the number of data points used to train the tree.
4. Decision trees are able to handle both numerical and categorical data. Other techniques are usually specialized in analyzing datasets that have only one type of variable.
5. Decision trees are able to handle multi-dimensional data.
6. Decision trees use a white box model, which means the explanation for a condition can be easily obtained by Boolean logic because the given situation is observable in the model. On the contrary, in a black box model that may be used by other classifiers, results may be more difficult to interpret (Bouyer & Mousavi, 2009).



Representations of acquired knowledge in tree form are intuitive and generally easy to assimilate by humans (Pant et al., 2009).

7. Decision trees make it possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.

Among the disadvantages of decision trees we cite the following:

1. Overfitting can easily emerge if decision learners create too complex trees that generalize the data well (Patel Brijain & Rana, 2014).

Decision trees have been used for classification in various fields, such as medicine, financial analysis, manufacturing and production, and molecular biology. Decision trees are the basis of several commercial rule induction systems (Kamber, Winstone, Gong, Cheng, & Han, 1997). However, very little research has applied decision trees to web page classification.

**Support Vector Machine (SVM)** is one of the supervised learning models with associated learning algorithms used for classification and regression analysis. SVM can perform linear classification by building a model that assigns new instances into one class or the other. In addition, SVM can efficiently perform a nonlinear classification using kernel methods. This makes SVM easily applicable to both linear and nonlinear data.

SVM has been used in various classification problems in the real world. SVM has been widely applied to classify text as it can significantly reduce the size of the training set in both the standard inductive and transductive settings (Du & Swamy, 2013). SVM is also used for image classification. A support vector machine active learning (SVMActive) algorithm proposed by Edward Chang and Simon Tong “has achieved significantly higher search accuracy than traditional query refinement schemes” (Chang & Tong, n.d.).

Some advantages of SVM are:

1. SVM is more effective than most classifiers in many applications involving very high dimensional data.
2. SVM can still be effective in cases where the number of dimensions is greater than the number of samples.
3. SVM is also memory efficient because it uses a subset of training points in the decision function (called support vectors).
4. SVM is versatile as different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

Some disadvantages of SVM are:

1. SVM is likely to perform poorly if the number of features is much greater than the number of samples.
2. SVMs do not directly provide probability estimates, and these are calculated using an expensive cross-validation, especially for large data sets.

SVM has been proven to yield a good performance in many cases when it comes to text classification. Since most web pages consist of text, SVM also achieves good performance in classifying web pages, which makes it one of the most popular classifiers in web page classification. Sun, Lim and Ng (2002) used SVM to classify web pages using text and context features and tested their method on the WebKB<sup>3</sup> data set. Their method has been

---

<sup>3</sup> <http://www.cs.cmu.edu/~WebKB/>

proven to get a better performance than FOIL-PILFS (for FOIL with Predicate Invention for Large Feature Spaces) method on the same data set (Sun, Lim, & Ng, 2002). Chen and Hsieh (2006) proposed a web page classification method called Weighted Voting Support Vector Machine (WVSVM) based on the SVM model. Sports news pages were used to test the WVSVM system in comparison with Latent Semantic Analysis Support Vector Machine (LSA-SVM) and Back Propagation Network (BPN). WVSVM achieved higher accuracy than LSA-SVM and BPN even with a small data set (R.-C. Chen & Hsieh, 2006). Glover et al. (2002) trained an SVM classifier when using web structures as features to classify web pages (Glover, Tsioutsoulis, Lawrence, Pennock, & Flake, 2002); and Kan and Thi (2005) also used an SVM classifier to demonstrate their web page classification method using URL features (Kan & Thi, 2005).

**Naïve Bayes** is a supervised learning algorithm based on applying Bayes' theorem with the "naïve" assumption of independence between every pair of features.

Some advantages of naïve bayes are:

1. Naïve bayes is probably the simplest of classification models, because it assumes that all features of the examples are independent of each other given the context of the class (McCallum, Nigam, & others, 1998).
2. Naïve bayes classifiers can be extremely fast compared to more sophisticated methods.

One important disadvantage of naïve bayes is:

1. Naïve bayes has strong feature independence assumptions. In some situation, naïve bayes classifier may fail because they see features as independent contributors to a classification (de Kok & Brouwer, 2011).

There are several different naïve bayes models adopted in the industry and studied in the literature. Some of them are Gaussian naïve bayes, multinomial naïve bayes, and Bernoulli naïve bayes<sup>4</sup>. McCallum et al. (1998) conducted a comparison between the multi-variate Bernoulli model and the multinomial model in terms of text classification. Both models make the naïve bayes assumption. The authors found that multi-variate Bernoulli performs well with a small size of vocabulary, while multinomial model is better for a large vocabulary size (McCallum et al., 1998). Riboni used a naïve bayes classifier to test a Structure-oriented Weighting Technique that takes into account both the number of occurrences of terms in documents and the HTML elements the terms are present in (Riboni, 2002). Haleem et al. (2014) presented a naïve bayesian probabilistic model for web page classification. This model is optimized and works effectively to classify web pages. It has provided an accuracy of around 94% for a given test data (Haleem, Niyas, Verma, Kumar, & Ahmad, 2014).

The comparison of different text and web page classification methods is very difficult because there is no cohesive methodology for the matter-of-fact evaluation, which means current methodologies are bias to some extent. There are a limited number of methodologies that have been mentioned in the literature. However, these comparisons are one-sided due to the particular data and methods used. Small-scale comparisons lead to highly

---

<sup>4</sup> [http://scikit-learn.org/stable/modules/naive\\_bayes.html](http://scikit-learn.org/stable/modules/naive_bayes.html)

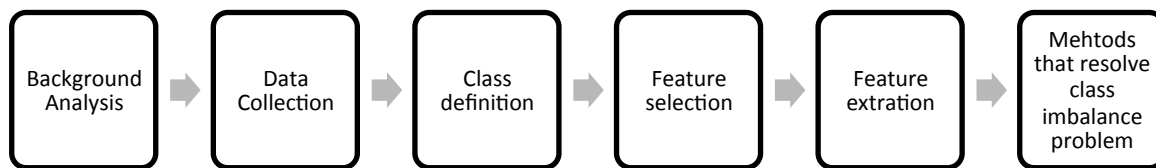
comprehensive statements based on inadequate observations; and provide limited insight into an all-round comparison among a wide range of methods (Marath, Shepherd, Milios, & Duffy, 2014).

In a recent research, Fernández-Delgado, Cernadas, Barro, & Amorim have evaluated 179 classifiers from a wide collection of 17 families over the whole UCI machine learning classification database and they find out that the best results are achieved by the parallel random forest built in R with caret, tuning the parameter mtry. This classifier “achieves in average 94.1% of the maximum accuracy over all the data sets, and overcomes the 90% of the maximum accuracy in 102 out of 121 data sets” (Fernández-Delgado et al., 2014).

Another good classifier they found in their research is the “LibSVM implementation of SVM in C with Gaussian kernel, tuning the regularization and kernel spread” and it achieves 92.3% of the maximum accuracy (Fernández-Delgado et al., 2014). Overall, in their research, six RFs and five SVMs are included among the 20 best classifiers (Fernández-Delgado et al., 2014). It is obvious that random forest and support vector machine outperform other classifier in most cases.

## Chapter 3: Classification Method

In this chapter, we discuss the various phases of our classification method. These phases can be explained with the following diagram.



**Figure 3: All Phases in Our Classification Method**

### 3.1 Background

This research is in cooperation with SweetIQ ([www.sweetiq.com](http://www.sweetiq.com)) a company specializing in local web search marketing. SweetIQ provides location-based marketing services and an analytics platform. It has developed a scalable, enterprise-grade local marketing platform allowing efficient management of one to thousands of locations with comprehensible dashboards and flexible reports. Its local marketing analytics and automation platform delivers local execution for national brands and their marketing agencies. Since 2010 SweetIQ has been helping its clients grow their sales by increasing their online visibility and findability and giving them the tools to manage their local search engine optimization at

scale. The platform's focus is to optimize the online to offline funnel (O2O funnel) ultimately converting online searches to in-store shoppers (SweetIQ, 2014).

SweetIQ's data is gathered via web crawlers that automatically identify, analyze, and extract content from web pages that are relevant to the local businesses it is analyzing. As part of the company's mandate to provide the most structured local search marketing data possible, its crawlers need to acquire the ability of classifying web pages so it can provide more insight into the distribution of the types of web pages their local business listings are found on. This allows marketers to make better-informed decisions about marketing campaigns and strategies. This is recognized as the business understanding phase in the CRISP\_DM methodology.

SweetIQ owns and operates technology that allows it to crawl and analyze, in real-time, local marketing data from hundreds of online sources. Its crawlers are able to target specific channels, as well as automatically discover new channels where local businesses may be listed, have reviews, social interactions, and other relevant local data points and (key performance indicators) KPIs. The technology was all built in house, and is SweetIQ's own custom stack, made up of processes of many cutting-edge technologies, including codebases in Python, NodeJS, GoLang, on top of Mongo, Redis, Riak, and Memcache amongst others (SweetIQ, 2014).

The infrastructure used by SweetIQ is made up on five main types of processors:

1. Data Flow Control System - Finite state machines that control the flow of requests, data, validation, and aggregation of information.

2. Data Gathering Systems - Processes that receive instructions to retrieve specific data points from specific sources.
3. Data Extractions Systems - Processes that know how to extract data from defined sources, as well as undefined sources.
4. Data Validation and Transformation Systems - Processes that validate the correctness and accuracy of the information retrieved, and transform them into application-specific data.
5. Data Aggregation Systems - Processes that compute and aggregate analytics from the data retrieved.

These systems gather and analyze many KPIs and local marketing data points, for example:

1. Listings - A listing is a business profile web page created for a local business on a local business directory, social website, or search engine. A typical listings page will include the business' name, address, phone number, website, hours of operation, business description, category, as well as some additional "enhanced data", such as products and services, years in business, menu items, photos, videos, etc.
2. Reviews - A review is a user-submitted piece of content that includes the user's experience with the local business, typically accompanied with some sort of a grade, such as a star rating, or a numeric value.
3. Keyword rank - A keyword rank is a numerical value that denotes a local business' presence in a list of the first 100 results for a search term. Each keyword provides 2 ranks, one for the business' website (organic), and one for the business' listing on the search engine the rank is being gathered for (local).



4. Competitors - As the crawlers index local marketing data for a local business, they are also able to identify local competitors.
5. Social metrics - The crawlers are also able to automatically identify a local business' social profiles on Twitter, Foursquare, and Facebook, and gather metrics relevant to each social profile.

As part of SweetIQ's continued efforts to provide highly accurate and pertinent data for its clients, it is currently in need of researching better ways of performing some data analysis and extraction tasks that will better validate the data its crawlers retrieve. Unfortunately, as it deals with highly unstructured, disorganized data (web pages from all across the web), this is not a trivial task. One of these tasks is web page classification.

An important aspect of the data gathering process is to identify what type of web page we extracted this data from. This is an important marker for marketing strategies, as having a healthy mix of listings on search engines, large directories, niche directories, blogs, and wikis has an immediate impact on keyword rank. We need to develop a process that, given the contents of a page or at a higher level the URL pattern, automatically infers and classifies the type of page we extracted the data from.

### **3.2 Data Collection**

In the second phase of CRISP\_DM, we are asked to start with an initial data collection. We have identified ten most valuable clients associated with their addresses and phone numbers, which are listed in table 3. Our data is collected from search engines such as Yahoo and Bing by search for these ten clients.

<b>Business name</b>	<b>Address</b>	<b>Phone number</b>
FedEx	1 Place Ville Marie, Montreal, QC H3B 3Y1	1-800-463-3339
Dominos	490 Mapleview Dr. W, Barrie, Ontario L4N 6C3	1-705-728-0330
Bell	2154 W 4th Ave, Vancouver, BC V6K 1N6	604-678-5873
Union Oyster House	41 Union Street, Boston, Massachusetts 02108	1-617-227-2750
1-800-Got-Junk?	9 Dibble St., Toronto, Ontario M4M 2E7	1-800-468-5865
McDonald's	675 Yonge St, Toronto, ON M4Y 2B2	416-413-1442
Wishbone Restaurant	1001 W Washington Blvd, Chicago, IL 60607	1-312-850-2663
Wells Fargo Bank	501 SOUTH COLLEGE ST, CHARLOTTE, NC 28202	1-800-869-3557
Bass Pro Shop	300 Cincinnati Mills Dr, Cincinnati, OH 45240	1-513-826-5200
Fontainebleau Miami Beach	4441 Collins Avenue, Miami Beach, FL 33140	1-305-538-2000

**Table 3: Information of SweetIQ's Ten Clients**

In order to crawl web pages that are related to target business, we provide a search query that consists of two parts – a business name and a geomodifier that is a city in our case. A list of search queries we are using right now is described in table 4:

Business Name	Geomodifier	Query
FedEx	Montreal	<b>Google:</b> q=BBB&oq=BBB&aqs=chrome.0.57l2j60l3j 65.2645j0&nord=1&sourceid=chrome&ie= UTF-8&as_qdr=all&num=100&near=CCC
Dominos	Barrie	
Bell	Vancouver	
Union Oyster House	Boston	<b>Bing:</b> setmkt=en-US&q=BBB near CCC&count=100"
1-800-Got-Junk?	Toronto	
McDonald's	Toronto	
Wishbone Restaurant	Chicago	<b>Yahoo:</b> p="BBB" near CCC&n=100"
Wells Fargo Bank	Charlotte	
Bass Pro Shop	Cincinnati	
Fontainebleau Miami Beach	Miami	where <b>BBB</b> = Business Name <b>CCC</b> = Geomodifier which is, in this case, city

**Table 4: Search Queries**

After we have built a query, we can feed a URL to our crawler and the crawler will send requests to the search engine and get a list of results from the search engine. For example, if we want to get web pages related to 1-800-Got-Junk? in Toronto from Bing, then we first build a query: “setmkt=en-CA&q=1-800-Got-Junk? near Toronto&count=100”. The URL constructed from this query is “<http://www.bing.com/search?setmkt=en-CA&q=1-800-Got-Junk? near Toronto&count=100>”. It is equivalent to typing “1-800-Got-Junk? near Toronto” in the search box on [www.bing.com](http://www.bing.com). The returned web page is a list of search results provided by Bing. Once the URL is provided to our crawler, the crawler will see the web page shown in figure 4.

The screenshot shows a Bing search results page for the query "1-800-Got-Junk? near Toronto". The search bar at the top contains the query, and the results are categorized under "Web". The left sidebar lists several search results, including "1-800-GOT-JUNK?@ - Fully Insured & Up Front Rates!", "1-800-GOT-JUNK? | Junk Removal Toronto | Book Online & ...", "1-800-GOT-JUNK? Toronto/GTA has 188 reviews and ...", "1-800-GOT-JUNK? Toronto and G.T.A. - Riverdale - Toronto ...", "Junk Removal | Hauling 1-800-GOT-JUNK? Trash Waste ...", and "1-800-Got-Junk? Toronto, 800-468-5865, Toronto, Ontario ...". The right sidebar features a map of the area around 9 Dibble St, Toronto, with a blue dot indicating the location. Below the map, the business information for "1-800-GOT-JUNK?" is displayed, including the address "9 Dibble St, Toronto ON M4M 2E7", phone number "800-468-5865", hours "Mon - Sun 8:00 AM to 8:00 PM", and a Yelp review section with a 4-star rating and a review from May 19, 2014.

Figure 4: A Web Page Returned by Bing with a Specific Search Query

As we can see in figure 3, this web page has 8660 results listed, but we only consider the top 100 pages as our data and that is why we have “count=100” in our query so that our crawling system only crawls the top 100 pages. For each result, we can see a title with a hyperlink, and a small paragraph of words that describe the result web page. We call this paragraph a snippet of this result. By analyzing the result, we can find three elements that we can extract, which are the title of the result, the URL of the result, and the snippet of the result. Therefore, along with the business name and city of the business, we can build a dictionary with these elements. The dictionary is written in JSON format. For example, the dictionary built from the first result shown in figure 4 is:

```
{  
  
  "businessName": "1-800-Got-Junk?",  
  
  "city": "Toronto",  
  
  "item": {  
  
    "snippet": "1-800-GOT-JUNK? is a leading junk removal provider, servicing  
TorontoToronto and the GTA. Book online and SAVE! ... Junk Removal Toronto. You are  
here. Ontario;",  
  
    "title": "1-800-GOT-JUNK? | Junk Removal Toronto | Book Online & ...",  
  
    "url": "http://www.1800gotjunk.com/ca_en/locations/junk-removal-toronto"  
  
  },  
  
  "searchEngine": "bing"
```

}

In this JSON code, we can see a “key:value” format. Five elements in this dictionary are used in our classification task, which are businessName, city, snippet, title and url.

### **3.3 Definition of Classes**

This section describes the various classes of web pages we are interested in. We also give an example for each class.

#### **1. Profile:**

A business profile page is a page that has at least one business name and one address. A profile page can also include one or more phone numbers, even though a phone number is not a determinant in classifying a page as a profile page. If the page has multiple business names and addresses then it is also considered a profile page.

Example:

# FEDEX CORP

1080 MONTREAL AVE  
SAINT PAUL, MN 55116-2386 | [view map](#)  
[www.fedex.com](http://www.fedex.com)

Looking for more information? [Sign up for FREE!](#)



## Company Details

Location Type: Branch  
Industry: Air Courier Services  
Ownership: Public  
Year Founded: 1971  
Sales Range: Over \$1,000,000,000  
Employees: 100,000 to 150,000  
Have fresher information? [Update](#)

## Recent Alerts

On this company:

Credit Risk Increase	No
Payment Decline	No
Purchase Behavior Decline	No
Public Records	No
Financial News	<a href="#">YES ▶</a>
Growth Clues	<a href="#">YES ▶</a>

**Figure 5: Example of a Profile Page**

On this web page, we can find a business name “FEDEX CORP” and a corresponding address “1080 MONTREAL AVE, SAINT PAUL, MN 55116-2386”. Therefore, this page is classified as a business profile page.

## 2. Search result:

A business search result page is a page that contains multiple business names and corresponding links to business profile pages. Sometimes, there is a list of business names

and links on a search result page, but those links lead us to pages that do not have business names or addresses. In such cases, the search result page will be classified as “other”.

Example:

BBB > Home > Accredited Business Directory > Waste Containers > Toronto, ON

## Toronto Waste Containers

Localize Results:

City/province or postal code

---

All BBB Accredited businesses in the category of  
Toronto Waste Containers

---

	<b>1-800-Got-Junk?</b> 9 Dibble St, Toronto, ON <a href="#">Request a Quote</a> <a href="#">Map</a> <a href="#">Web</a>
	<b>Dufferin Disposal Ltd.</b> 65 Milvan Drive, North York, ON (10.0 miles) <a href="#">Request a Quote</a> <a href="#">Map</a> <a href="#">Web</a>

---

**Figure 6: Example of a Business Search Result Page**

On this web page, we can find a list of two results that both have business names and links. When we click the link of the second business, we can see a profile page (see figure 6) of this business.

**BBB ACCREDITED BUSINESS SINCE 15/02/2013**

**Dufferin Disposal Ltd.**

Phone: (416) 746-2290

Fax: (416) 746-4204  
65 Milvan Drive, North York, ON M9L 1Y8  
[www.dufferindisposal.com](http://www.dufferindisposal.com)

**Figure 7: Example of a Profile Page**



As a result, this page is classified as a business search result page.

### **3. Review:**

A review page is a page that has at least one user-submitted piece of content that includes the user's experience with the local business, typically accompanied by some sort of a grade, such as a star rating or a numeric value.

Example:

## FedEx Authorized Shipcentre

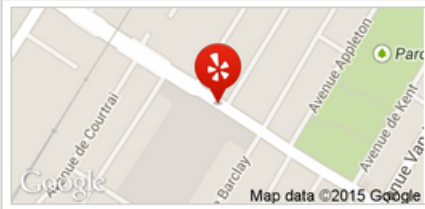
★ ★ ★ ★ ★ 1 review

[Details](#)

★ [Write a Review](#)

Couriers & Delivery Services

[Edit](#)



6700 Ch Cote-des-neiges  
Montreal, QC H3S 2B2  
Canada

[Get Directions](#)

+1-800-463-3339

[fedex.com/ca\\_english](http://fedex.com/ca_english)

[Edit](#)

[Add Photo](#)



**Ad StorageMart**

[Info](#)

3.8 miles away from FedEx Authorized Shipcentre

**Jon L. said** "This Storage unit is very organized and helpful. They let you know when your packages are in, they are kind and are always..." [read more](#)



**Ad Entrepot Beaumont Mini-Storage** ★ ★ ★ ★ ★ 3 reviews

[Info](#)

1.9 miles away from FedEx Authorized Shipcentre

**Tammy F. said** "Great location for a storage place, very close to parc ave and provides parking for your short trips to your locker. The..." [read more](#)

## Recommended Reviews

Search reviews



[Yelp Sort](#) [Date](#) [Rating](#) [Elites](#)

English 1



**Your trust is our top concern**, so businesses can't pay to alter or remove their reviews. [Learn more.](#)



**Jean-Marc B.**  
Outremont, Canada

0 friends

★ 5 reviews

★ ★ ★ ★ ★ 4/4/2014

[First to Review](#)

CHANGE IN THEIR DELIVERY POLICY MAKE THEM Suck.!!!!#@#%@%

FedEx found a great way to save money: they deliver the courier to the address where they have to only once! Forget about a second and third trial, only once and if not, too bad for you, you have to go to a service point (in my case, Montreal, located just in the middle of downtown) with the result that I will have to drive their, pay the parking, lose time to pick it up and then all back home for a 3.5\$ item!!!! Tell me that when they are delivering you a courier they advise you in advance (nope!!!), telling you the time they would be there (Nope)..... FEDEX is crap!! Better is UPS, at least they try several time with a short window time to indicate when they will be back, and also they allow you to get the item without having to sign!!

### **Figure 8: Example of A Review Page**

On this web page, we can find a user's review about the business "FedEx Authorized Shipcentre". Thus, this page is classified as a review page.

#### **4. Target profile:**

If a profile page contains a business name similar to the name provided in the search and also a city similar to the one provided in search, then this profile page will also be classified as a target profile page.

Example:

#### **1-800-GOT-JUNK? Toronto And G.T.A.**

Address:

[9 Dibble St.](#)  
[Toronto,](#)  
[ON](#)  
[M4M2E7](#)

Phone:

1-800-468-5865

### **Figure 9: Example of a Target Profile Page**

On this web page, we see that the business name "1-800-GOT-JUNK?" is the same as the one we searched for and the city "Toronto" is the same as the one we searched for. Hence, this page is classified as target profile page.

#### **5. Target review:**

If a review page mentions a business name similar to the business name provided in the search and also contains a target city similar to the one provided in the search, then this review page is also classified as a target review page.

Example:

Union Oyster House, Boston

21 Reviews

41 Union St., Boston, MA 617-227-2750

Boston Overview

Things to Do

Hotels

Restaurants

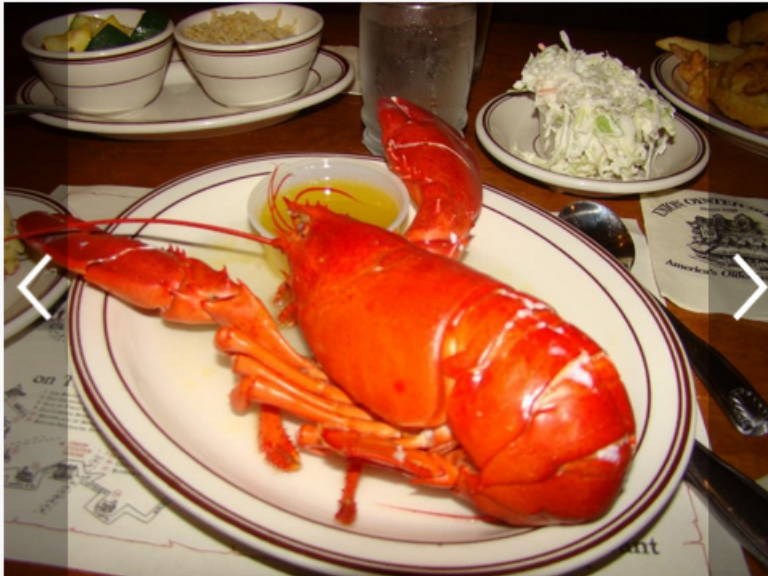
Travel Insurance




Nightlife

Transportation

Been here? Rate It!

hide




See 8,504 Photos

Sort by:

Most recent

Most helpful

Write a Review



Union Oyster House: ye oldest Restaurant in America (2)

by machomikemd Updated Sep 3, 2014

Part Two of My Union Oysterhouse tips with more pictures.

This Iconic Joint claim to be the oldest restaurant in America and the decor and ambiance inside definitely proves it plus is has been into numerous reviews in the boob tube courtesy of such heavyweights like the Discovery Travel Channel and The Food Network.Hence is definitely on my list of where to eat the famed new england seafoods here in boston! Dating back to 1826, the Union Oyster House maintains a suitably old-fashioned tavern decor, with lots of weathered wood and a casual atmosphere. As the owners would say "located on the Freedom Trail near Faneuil Hall, enjoys the unique distinction of being America's oldest restaurant. This Boston fixture is housed in a building dating back to pre-revolutionary days and started serving food in 1826. The stalls and oyster bar where Daniel Webster was a regular customer are in their original positions. Today's politicians, celebrities and tourists still come in to enjoy the food".

**Figure 10: Example of a Target Review Page**

44

On this webpage, we can see that the business name “Union Oyster Bay” is the same as the business name provided in the search, and a city name “Boston” the same as the one we searched for and a review about this business. As a result, this page is classified as a target review page.

After the crawled system has crawled different pages from Bing and Yahoo, we manually classified pages into five classes based on the definitions we mentioned above. We prepared a data set of 599 web pages for training and testing. The distribution of the pages is described in the following table.

	Search result	Profile	Review	Target profile	Target review
Number of pages	76	349	52	119	27

**Table 5: Data Distribution**

### 3.4 Feature Selection

In the third phase of CRISP\_DM, data need to be prepared for modeling tools. Therefore, features should be selected and extracted. For the purpose of classification, documents are represented by their features. A feature is simply a decimal value, a measure of a given aspect of a document. For instance, the frequency of a word in the document could be a feature. Note that features don’t necessarily have a decimal part. They could be Boolean, integer, or some sort of label. But all of these can be represented as a decimal number.

When considered as a whole, these features form a vector of decimal numbers. The length of the vector is equal to the number of features chosen to model the documents. Using this

representation, we can think of a document as a point in  $R^n$ , with its position determined by its feature vector. The space  $R^n$  is called the feature space.

### **3.4.1 Classical Text Features**

For text classification, the classical approach to feature selection is to use Bag of Words. Since the data we have collected contains a large part of text, we consider words in all examples and count how many times each word occurs in each example. This approach is known as the Bag of Words approach because it does not matter where each word appears in the document. The document is simply a bag with all of its words randomly mixed inside.

### **3.4.2 Term Frequency-Inverse Document Frequency**

Term Frequency-Inverse Document Frequency (tf-idf) weight is a weight widely used in information retrieval and text mining. It is a statistical measure to evaluate how important a word is to a document in a collection or corpus (Wen, Fang, & Guan, 2008). “The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus” (Wen et al., 2008).

Tf-idf weight consists of two terms: the first one, Term Frequency (TF) “calculates the number of times a word appears in a document divided by the total number of words in that document”; and the second one, Inverse Document Frequency (IDF) is “computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears” (Jayalekshmi, 2014).

“Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided

by the document length (aka. the total number of terms in the document) as a way of normalization” (Agrawal & Gupta, 2014).

$$TF(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}}$$

While computing TF, all terms are considered equally important. However, certain terms that may not have much meaning, such as "the", "of", and "this", may appear a lot of times but are not important. Thus we need to weigh down the frequent terms while scaling up the rare ones (Agrawal & Gupta, 2014).

$$IDF(t) = \log \frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}}$$

Below is a simple example to explain tf-idf.

Consider a document containing 100 words where the word “apple” appears 3 times. The term frequency (tf) for “apple” is  $3/100 = 0.03$ . Now, assume we have 10 million documents and the word “apple” appears in one thousand of these. Then, the inverse document frequency (idf) is computed as  $\log(10,000,000/1,000) = 4$ . Thus, the tf-idf weight is the product of these quantities:  $0.03 \times 4 = 0.12$ .

### 3.4.3 Custom Features

However, our problem is not to classify documents according to their content, but according to their type. If a document is a jug, we want to classify the jug, not the liquid inside it. For instance, one of the contexts we wish to identify is the profile of a business. A profile page is a type of web page that matches our definition, but its content could be anything. It could be a profile page about a restaurant that may have words like “food”, “menu”, and “dinner”; or



it could be a profile page about a bank that may have words like “loan”, “credit”, and “banking”. As a result, we cannot determine the type of these two web pages only by looking at their content. We need to look for some specific information that can help us determine whether the web page is a profile page or not. According to our definition of a profile page, business name and address are two fundamental elements that make a profile page distinctive from other pages. Therefore, we want to know if the web page we are analyzing has business name and address, and these two elements are two important features when classifying web pages. In addition to business name and address, we think phone number is another useful feature because most profile pages will have phone numbers associated with business names.

We also have collected a list of directory providers and a list of review providers. These two lists are used to help us to know if one of the providers appears in the URL of the web page, because if there is a directory provider on a web page, this page is likely to contain profile information; and if there is a review provider, it is probably a review page. We have also considered other URL features, such as business name in URL, and frequent tokens appearing in URLs of a certain class.

### 3.4.4 Feature Set

We have identified useful features for this web page classification problem based on title, snippet and URL. The following table summarizes these features to classify web pages.

There are two kinds of features: text-based features and URL-based features.

Feature Number	Feature
Feature 1	Street type in title and snippet
Feature 2	City in title and snippet

Feature 3	Target city in title and snippet
Feature 4	Region in title and snippet
Feature 5	Postal code (zip code) in title and snippet
Feature 6	Phone number in title and snippet
Feature 7	Target business name in title and snippet
Feature 8	Bag of Words in title and snippet from profile instances
Feature 9	Bag of Words in title and snippet from search result instances
Feature 10	Bag of Words in title and snippet from review instances
Feature 11	Director provider in URL domain
Feature 12	Review provider in URL domain
Feature 13	Target business name in URL domain
Feature 14	Target business name in rest after URL domain
Feature 15	Bag of Words in rest after URL domain from profile instances
Feature 16	Bag of Words in rest after URL domain from search instances

**Table 6: Feature Set**

Features 1 to 10 are text-based features. Feature 1 is assessing whether there is a street type (examples are st, blvd, and ave) in title or snippet, and we provide a list of street types for

matching. Feature 2 is assessing whether there is a city in title or snippet. We are using a python library “geotext” to look for cities in title or snippet. “geotext” uses data from <http://www.geonames.org>. Besides city, we have feature 4 to identify target profile pages particularly, which is target city - a city of a specific business. We also have a list of Canadian provinces and US states to look for any region that appears in title or snippet, which is feature 3. We also build regular expressions to match feature 5 and 6 which are postal code (or zip code) and phone number in title or snippet. We apply the same logic to extracting feature 2 in the remaining part after URL domain. We also need to know if feature 7 – business name – has appeared in the text. Features 8, 9, and 10 are Bag of Words features. When we classify profile pages, pages from other classes – review, and search results – are considered as non-profiles. Profile pages and non-profile pages are mixed together as our data set in which we select the most valuable words that distinguish profile pages from other pages. We put these words into a “bag” and this “bag” is feature 8. Then we apply the same logic to search result and review pages and we get feature 9 and 10.

Features 11 to 16 are URL-based features. URLs are first segmented into several components according to different requirements. In order to extract feature 11, 12, and 13 from a URL, the domain of the URL will be extracted. For example, given a URL “[http://www.tripadvisor.co.uk/Restaurant\\_Review-g154980-d5093278-Reviews-Domino\\_s\\_Pizza-Barrie\\_Ontario.html](http://www.tripadvisor.co.uk/Restaurant_Review-g154980-d5093278-Reviews-Domino_s_Pizza-Barrie_Ontario.html)”, “[www.tripadvisor.co.uk](http://www.tripadvisor.co.uk)” will be extracted. Everything after the domain will also be considered as a string that will be broken into tokens so as to extract feature 14, 15 and 16.

### **3.5 Feature Extraction**

After we have selected features to use in our classification task, we need to extract them from data. Feature extraction is very different from feature selection. Feature extraction consists in transforming arbitrary data like text into numerical features usable for classification. We have used several feature extraction techniques including text tokenization, text normalization, and URL tokenization.

### **3.5.1 Text Tokenization**

In order to be able to match business name or find out most frequent words in title or snippet, we consider all the text in a document as a string and divide the string into substrings by splitting on the specified characters such as “,” “+”, “-”, “\_”, and space. For example, “Domino's Pizza, Barrie: See unbiased reviews of Domino's Pizza” will be tokenized to [“Domino’s”, “Pizza”, “Barrie”, “See”, “unbiased”, “reviews”, “of”, “Domino’s”, “Pizza”].

### **3.5.2 Text Normalization**

The next step after text tokenization is text normalization. We normalize the text to lowercase so that the distinction between “The” and “the” is ignored. We have gone further than this, and stripped off any affixes, a task known as stemming. We have used the Porter stemmer<sup>5</sup> in this task.

### **3.5.3 URL Tokenization**

Similar to text tokenization, a URL is also considered as a string and it is divided into strings. A URL is first segmented into its components as given by the URL protocol (e.g., scheme://host/path/document.extension?query#fragment). For each component, we split it on

---

<sup>5</sup> <http://tartarus.org/~martin/PorterStemmer/>

specific characters such as “/”, “+”, and “-“. We also break the component at URL encodings like “%20”.

### **3.6 Class Imbalance Problem**

As we can see in our data set, the numbers of instances of each class vary a lot. We have 349 profile pages out of 599 pages, 79 search result pages and only 27 review pages. When considering search result class only, we are building a training set based on 79 positive instances (profile pages) and 520 negative instances (non-profile pages). The number of positive instances and that of negative instances are highly imbalanced. The problem becomes even worse when it comes to review class, because we only have 27 positive instances compared to 572 negative instances. We did an experiment trying to classify three classes (profile, search result, and review) at the same time with the data set. We got a 100% recall (true positive rate) on profile pages but extremely low recalls on the other classes, which means that almost all the pages were classified as profile pages. This result is caused by the class imbalance problem.

Traditional classification algorithms usually assume balanced distribution of classes and equal error costs (the costs of false positives and false negatives are equal), and therefore these algorithms are not suitable for class-imbalanced data sets. At least, they don't yield good results. Several approaches are used to improve the classification performance of class-imbalanced data sets. These approaches include oversampling, undersampling, threshold moving and ensemble techniques. Oversampling and undersampling are resampling techniques that only change the distribution of instances in the data set without changing the

classification model. In our case, we have tested oversampling and undersampling and also used the random forest algorithm<sup>6</sup> which is an ensemble theory technique.

### **3.6.1 Oversampling**

Oversampling is the simplest way to make the sizes of classes equal by increasing the size of the minority class (Mollineda & Sotoca, 2007). Suppose that the original data set has 100 positive and 500 negative instances. In oversampling, we duplicate instances of the minority class and therefore we build a new data set that contains 500 positive and 500 negative examples. Instead of merely duplicating instances of the minority class, Chawla et al. developed an oversampling method called Synthetic Minority Over-Sampling (SMOTE) that generates new synthetic minority instances by interpolating between some positive instances that lie close to each other (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

Batista et al. have proved oversampling to be convenient to apply to highly imbalanced data sets (Batista, Prati, & Monard, 2004). However, it may increase the computational burden of classification algorithms as it increases the size of the whole data set.

### **3.6.2 Undersampling**

Undersampling is the method that randomly removes the negative examples in order to generate balanced data sets. Suppose that the original data set is the same as the one in 3.6.1. In undersampling, we randomly remove 400 negative examples so that we have a data set that contains 100 positive and 100 negative examples. It is as simple as oversampling, moreover it has empirically been proven to be efficient among resampling techniques (Mollineda & Sotoca, 2007). Many undersampling methods in the literature use intelligent

---

<sup>6</sup> [https://www.stat.berkeley.edu/~breiman/RandomForests/cc\\_home.htm#intro](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#intro)

selection of examples of the majority class to eliminate. For instance, Kubat and Matwin's undersampling method only removes redundant negative cases that they consider as noise (Kubat, Matwin, & others, 1997).

The primary problem of undersampling is that it may discard data potentially important in the classification process (Mollineda & Sotoca, 2007). It may be even worse if we use this method when classifying review pages, because we only have a few review pages in our data set and we may have to remove a lot of instances of other classes to match the size of the review class. We would end up having very few instances to train the classifier. Therefore, undersampling is quite convenient when the level of imbalance is very low.

### **3.6.3 Ensemble Theory Techniques**

Ensemble theory techniques are solutions to the class imbalance problem at the algorithmic level. Three popular ensemble theory techniques are bagging, boosting, and random forest. We mainly focus on random forest in our research. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control overfitting. The random forest utilizes bagging in tandem with random attribute selection (Khoshgoftaar, Golawala, & Van Hulse, 2007)(Han & Kamber, 2012). The random forest gives accuracy similar to or better than Adaboost (a machine learning meta-algorithm) (Schapire, 2013) and it is relatively robust to errors and outliers (Breiman, 2001)(Khoshgoftaar et al., 2007). The random forest is also faster than either bagging or boosting (Breiman, 2001). The tendency to overfit decreases when using the random forest classifier because the generalization error for a forest converges as long as a large number of trees are added to the forest (Breiman, 2001)(Han & Kamber, 2012).

### **3.7 Conclusion**

In this chapter, we introduced various phases in our classification methods including background analysis, data collection, data pre-processing, feature selection and extraction. We also explained the class imbalance problem that we found in our research and then proposed several techniques to resolve this problem, including oversampling, undersampling and ensemble techniques.

## **Chapter 4 Experiments and Results**

### **4.1 Experimental Settings**

We have performed experiments on the classification of the 599 web pages we crawled from two search engines – Bing and Yahoo. We manually classified them into profile pages and non-profile pages before the classification experiments. We had 349 profile pages and 250 non profile pages. We used both SVM and random forest as our classification algorithms in each experiment. We fed different features to the classification algorithms to evaluate the results. We also used different techniques to balance out the data set and compared the results with those obtained from the original data set. We used a 10-fold cross validation in each experiment to avoid overfitting.

The software tools used in our experiments were as follows. We used Python programming language to process data and select and extract features from data. We used several Python built-in packages to perform different tasks. For example, we used the urlparse package<sup>7</sup> to

---

<sup>7</sup> <https://docs.python.org/2/library/urlparse.html>



parse URLs, the json package<sup>8</sup> to parse JSON files and the re package to match regular expression. We also used a few third party packages such as the Natural Language Toolkit<sup>9</sup> (NLTK) which is a Python package for processing human language data. We used nltk to tokenize and normalize text. We also used goetext<sup>10</sup> to extract city names from the text. The package has a database that contains cities in north America. At last, we feed our data to Weka<sup>11</sup>, a tool for machine learning and data mining, to train classifiers.

## **4.2 Feature Selection and Extraction Process**

Web pages cannot be directly fed to classification algorithms and they must be converted to features. The same feature extraction process should be applied to both training data and test data. In section 3.4.3, we have identified 16 features as candidates and they can be put into two categories – text features and URL features. In this section we describe the process of extracting these features.

### **4.2.1 Text Feature Selection and Extraction**

Text feature extraction mostly concerns natural language processing which is a set of techniques for approaching text problems. In our process, we load the text from title and snippet and then apply Bag of Words model to get the prediction of whether a web page is a profile page or not.

## **Data Cleaning and Text Processing**

---

<sup>8</sup> <https://docs.python.org/2/library/json.html>

<sup>9</sup> <http://www.nltk.org/>

<sup>10</sup> <https://geotext.readthedocs.org/en/latest/>

<sup>11</sup> <http://www.cs.waikato.ac.nz/ml/weka/>

When considering how to clean the text, we need to think about the data problem we are trying to solve. For many problems, we only need to remove punctuation. In other cases, we also need to tackle a sentiment analysis problem. For example, when we are dealing review pages, words like “great”, “fantastic”, and “disappointed” can be very important, but when analyzing profile pages, these adjectives don’t give us much information.

To remove punctuation, we use a Python built-in package called **re** that deals with regular expressions. We use “[^A-Za-z0-9]” to find anything that is not a lowercase letter (a-z), an upper case letter (A-Z) or a number (0-9). Then we replace them by space. We use regular expressions to extract some other features as well. In order to get Canadian and US phone numbers, we use a regular expression “D\*([2-9]\d{2})(\D\*)([2-9]\d{2})(\D\*)(\d{4})\D\*”. We use another regular expression “(^(\d{5})(-\d{4})?\$)(^[ABCEGHJKLMNPRSTVXY]{1}\d{1}[A-Z]{1} \*\d{1}[A-Z]{1}\d{1}\$)” to find all Canadian postal codes and US zip codes. If there is a phone number or a postal code/zip code, we assign value “1”; otherwise, we assign value “0”.

In order to know if there is region or a business name in the text, we need to compare the words in the text to the regions and names we provide at first. In order to make the comparison easier, we also convert the text to lower case and split them into individual words (call tokenization).

Finally, we need to decide how to deal with frequently occurring words without much meaning. These words are called “stop words”, and in English they include words such as “and”, “the”, “a”, and “is”. With the help of NLTK, we can easily get rid of stop words because NLTK has a list of stop words and provide a nice function to remove them. After all

these steps, the text “Setting the standard for romance, elegance and lasting memories, seal your ceremony with an emblem of history at the Biltmore of Mia... Read More” will be converted to a list of words like:

[‘Setting’, ‘standard’, ‘romance’, ‘elegance’, ‘lasting’, ‘memories’, ‘seal’, ‘your’, ‘ceremony’, ‘emblem’, ‘history’, ‘Biltmore’, ‘Mia’]

A list of words makes it easier to find a match of business name and region. If there is a business name or a region, we assign value “1”; otherwise, we assign value “0”.

### **Bag of Words**

After we have cleaned all the words, we put them together to make it easier to use in the Bag of Words model.

Now that we have our training text tidied up, the next step is to convert them to some kind of numeric representation for machine learning algorithms. The Bag of Words model learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. For example, consider the following two sentences:

Sentence 1: “The cat sat on the hat.”

Sentence 2: “The dog ate the cat and the hat.”

From these two sentences, our vocabulary is as follows:

{the, cat, sat, on, hat, dog, ate, and}

To get our bags of words, we count the number of times each word occurs in each sentence.

In Sentence 1: “the” appears twice, and “cat”, “sat”, “on”, and “hat” each appear once, so the feature vector for Sentence 1 is:

{2, 1, 1, 1, 1, 0, 0, 0}

Similarly, the features for Sentence 2 are: {3, 1, 0, 0, 1, 1, 1, 1}

### **Term Frequency-Inverse Document Frequency**

Occurrence count in Bag of Words model is a good start but there is an issue: large size documents will have higher average count values than shorter documents, even though they might talk about the same topics. What’s more, we have a lot of web pages in our case, which will give us a large vocabulary. However, some words are not good enough to distinguish a kind of web pages from others and we only need the most important words. To avoid these potential issues, we use a numerical statistic called term frequency-inverse document frequency (tf-idf) that we mentioned in section 3.4.2 to select the most important words.

#### **4.2.2 URL Feature Selection and Extraction**

Like the text, we need to split the URL and remove unnecessary parts in order to get features from it. We also use the regular expression to split the URL. In our case, when encountering the following symbols, we split the URL and remove the symbol.

Protocol and domain: “`^w+:\V/?[\w+.]*(?=\V|:\d+)`”

Slash: “`\`”

Dash: “`\-`”

Underscore: “\\_”

Dot: “\.”

Question mark: “\?”

Equal sign: “\=”

Plus sign: “\+”

Space: “\s”

Ampersand: “\&”

Quotation: “\”|”

Comma: “\,”

Colon: “\:”

Pipe: “\|”

URL encoding: “%\d{2}|\%\d{1}[ABCDEFGH]|\%[ABCDEFGH]\d{1}|\%[ABCDEFGH]{2}”

For example, given the URL

“http://portalsso.vansd.org/portal/page/portal/Building\_Pages/COLUMBIA\_RIVER\_HIGH\_SCHOOL”, “http://portalsso.vansd.org/” will be removed first and the following part will be split into a list of words like:

['portal', 'page', 'portal', 'Building', 'Pages', 'COLUMBIA', 'RIVER', 'HIGH', 'SCHOOL']

Finally, we apply the Bag of Words model to URLs and create a bag of words for training purpose.

### 4.3 Performance Evaluation

After a classifier has been built, we want to evaluate the performance of the classifier from different aspects. For instance, we have a classifier trained from web pages that are manually labeled by looking at their content. We would like an estimate of how accurately the classifier can predict incoming web pages from the web, which are web pages on which the classifier has not been trained. We also have built multiple classifiers using different classification algorithms and we want to compare their accuracy. As a result, we need some measures to compare and evaluate classifiers.

TP, TN, FP, P and N refer to the number of true positive, true negative, false positive, positive, and negative samples. These terms are described in Table 7. In this research, we mainly use precision, recall, and F-measure to evaluate our results.

Positives (P)	These are positive tuples.
Negatives (N)	These are negative tuples.
True positives (TP)	These are the positive tuples that were correctly labeled by the classifier.
True negatives (TN)	These are the negative tuples that were correctly labeled by the classifier.
False positives (FP)	These are the negative tuples that were incorrectly labeled as positive.
False negatives (FN)	These are the positive tuples that were

	incorrectly labeled as negative.
--	----------------------------------

**Table 7: Definition of TP, TN, FP, P and N (Han & Kamber, 2012)**

These terms are usually summarized in a confusion matrix (see Table 8). A confusion matrix is a tool for analyzing how well a classifier can recognize tuples of different classes.

	Classified as yes	Classified as no
Yes	TP	FN
No	FP	TN

**Table 8: Confusion Matrix (Han & Kamber, 2012)**

Table 9 describes various evaluation metrics for assessing how good or how accurate a classifier is at predicting the class label of tuples.

Measure	Definition	Formula
Accuracy	This is referred to as the overall recognition rate of the classifier. It reflects how well the classifier recognizes tuples of the various classes.	$\frac{TP + TN}{P + N}$
Error rate	Error rate is simply 1 – accuracy.	$\frac{FP + FN}{P + N}$
Recall (or sensitivity)	This is referred to as the true positive rate.	$\frac{TP}{P}$

	It is a measure of completeness (i.e., what percentage of positive tuples are labeled as such).	
Specificity	This is referred to as the true negative rate.	$\frac{TN}{N}$
Precision	This is considered as a measure of exactness (i.e., what percentage of tuples labeled as positive are actually such).	$\frac{TP}{TP + FP}$
F-measure, harmonic mean of precision and recall	This is an alternative way to use precision and recall by combining them together. It is the harmonic mean of precision and recall.	$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$
$F_{\beta}$ , where $\beta$ is a non-negative real number	This is a weighted measure of precision and recall.	$\frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$



**Table 9: Evaluation Measures (Han & Kamber, 2012)**

In addition to the accuracy-based metrics mentioned above, it can also be important to assess classifiers with respect to some additional aspects that are described in the following table.

Speed	This refers to the computational costs involved in generating and using the given classifier.
Robustness	This refers to the ability of the classifier to make correct predictions, give noisy data or data with missing values.
Scalability	This refers to the ability to construct the classifier efficiently given a large amount of data.
Interpretability	This refers to the level of understanding and insight that is provided by the classifier or predictor.

**Table 10: Addition Evaluation Metrics (Han & Kamber, 2012)**

In this research, we are not going to compare classifiers based on these four additional metrics, mainly because the implementation of a classification algorithm in a system can largely affect the classifier's speed, robustness, scalability, and interpretability.

#### **4.4 Classification with Word Tokens in Title and Snippet**

In this section, we show that word tokens extracted from title and snippet can be used to train a web page classifier that achieves good performance. First, we extract text from title and

snippet from every web page in our data set. Second, we tokenize the text into words. Then we make some normalization. For example, we replace “ ‘ ” and “ “ ” by “ \’ ” and “ \“ ”. We also convert every word to lower case. At last, we apply tf- idf to all the tokens in order to select the most important tokens. From the titles and snippets of 599 web pages, we have got 3092 word tokens and each of them is considered as an element in the feature vector in the experiment.

#### 4.4.1 Experiment 1: Comparing SVM with Random Forest

In Experiment 1, we use all 3092 word tokens to train two classifiers using SVM and the random forest and compare their performance. We first train a classifier with SVM. We use Nu-Support Vector Classification ( $\nu=0.5$ ) with radial basis function kernel. The classification result is shown in table 11.

Class	Precision	Recall	F-Measure
Profile	0.741	0.851	0.793
Non-profile	0.736	0.582	0.650

**Table 11: Result of Experiment 1 Using SVM**

Then we use a random forest of 100 trees in order to compare with SVM. Each tree is constructed with 12 random features to train a classifier. We have tested different numbers of randomly selected features and found that the random forest with 12 random features in each tree yields the best result. The result is shown in table 12.

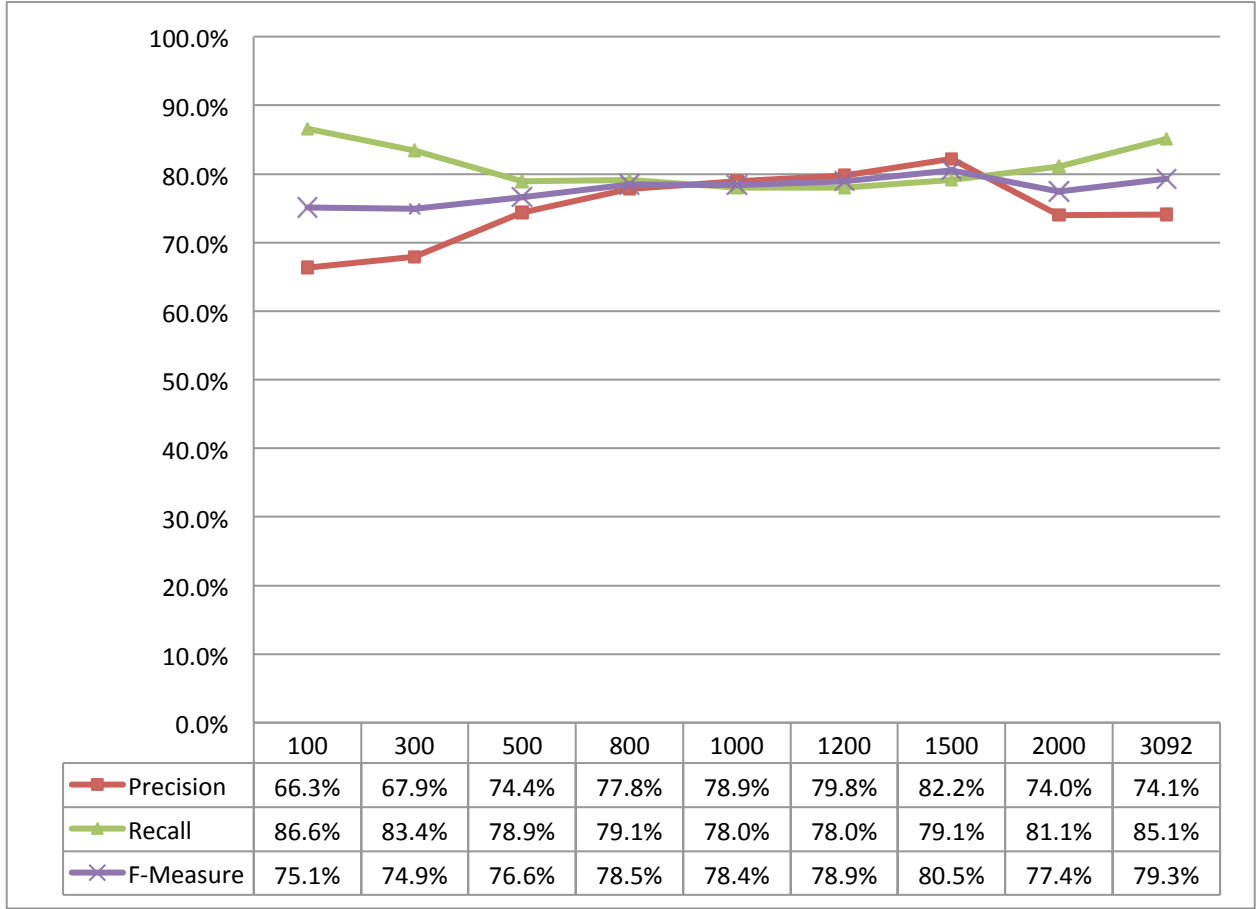
Class	Precision	Recall	F-Measure
Profile	0.707	0.891	0.789
Non-profile	0.759	0.482	0.590

**Table 12: Result of Experiment 1 Using Random Forest**

We can see from the results that even though SVM and the random forest outperform other classification algorithms in most cases (Fernández-Delgado et al., 2014), they are different with regards to different metrics. SVM is more accurate than the random forest when predicting profile pages, while the random forest tends to identify as many profile pages as possible, which can be reflected by a higher recall (the recall for SVM is 0.851, while the recall for random forest is 0.891).

#### **4.4.2 Experiment 2: Classification with Selected Word Tokens Using SVM**

In Experiment 1, we use all the tokens extracted from the text and we have a feature set of 3092 features. We think that the feature space may be too large and the features that are less important may not be useful and therefore yield a low precision. In Experiment 2 we use the information gain technique to reduce the dimensionality by only selecting features with higher rankings with the hope to get a better performance. In the experiment we try different numbers of features: 100, 300, 500, 800, 1000, 1200, 1500, 2000 and 3092. The result is shown in figure 11. In this experiment, we can see that the classifier trained with all features doesn't yield better result. However, the classifier trained with 1500 features selected with the information gain technique has the best result (precision is 0.82, recall is 0.791 and F-measure is 0.805).



**Figure 11: Result of Experiment 2 Using SVM (X-axis represents the number of features)**

## 4.5 Classification with All Text Features from Title and Snippet

In addition to the text extracted from title and snippet, we also have built several custom text features which are derived from the definition of classes. These custom features include street type in title and snippet, city in title and snippet, US states and Canadian provinces in title and snippet, postal code and zip code in title and snippet, phone number in title and snippet and target business name in title and snippet. We need to know if these features can improve the classification performance.

### 4.5.1 Experiment 3: Classification with Word Tokens and Custom Text Features

We feed custom text features along with text extracted from title and snippet to SVM and random forest. We apply the same methods used in section 4.2.1 when processing text from title and snippet and take all the tokens as features. Therefore, we have 3098 features, 3092 of which are word tokens and 6 of which are custom text features. We also use the same classification algorithms we used in section 4.2.1.

First we get a result from SVM which is presented in table 13.

Class	Precision	Recall	F-Measure
Profile	0.747	0.843	0.792
Non-profile	0.730	0.598	0.658

**Table 13: Result of Experiment 3 Using SVM**

Then we have a result from random forest which is shown in table 14.

Class	Precision	Recall	F-Measure
Profile	0.724	0.877	0.793
Non-profile	0.754	0.530	0.623

**Table 14: Result of Experiment 3 Using Random Forest**

Compared with the results of Experiment 1, the results obtained here from all the text features from title and snippet are not much better. Hence, word tokens extracted from title and snippet are good enough to predict profile pages.

#### **4.5.2 Experiment 4: Classification with Selected Word Tokens and Custom Text Features**

However, we don't know whether or not custom text features are more important than some tokens used in Experiment 2. Therefore, we apply the information gain technique to the data and select the top 1500 features in this experiment. We test the selected 1500 features with SVM. The result is shown in table 15.

Class	Precision	Recall	F-Measure
Profile	0.795	0.797	0.796
Non-profile	0.714	0.711	0.712

**Table 15: Result of Experiment 4**

Compared with the 82.2% precision and the 79.1 recall when classifying profile pages with the top 1500 features in Experiment 2, the result in this experiment is slightly worse. Again, it shows that word tokens extracted from text are good enough to train a classifier.

## **4.6 Classification with URL**

In addition to features extracted from title and snippet, features can also be extracted from URL. We have selected a few URL-related features that are tokens extracted from the rest after URL domain, target business name in domain, directory provider in domain and review provider in domain. These features also contain a large amount of information on a web page.

### **4.6.1 Experiment 5: Classification with URL Features Using SVM and Random Forest**

In order to get tokens from the rest after URL, we consider it as a string and apply the same text processing methods as in section 4.2.1. We feed the features to SVM and random forest classification algorithms and get the following results.

Class	Precision	Recall	F-Measure
-------	-----------	--------	-----------

Profile	0.796	0.703	0.747
Non-profile	0.734	0.820	0.775

**Table 16: Result of Experiment 5 Using SVM**

Class	Precision	Recall	F-Measure
Profile	0.816	0.657	0.728
Non-profile	0.713	0.851	0.776

**Table 17: Result of Experiment 5 Using Random Forest**

#### **4.7 Classification with Balanced Data Set**

As we can see that we have an imbalanced data set with 350 profile pages and 259 non profile pages, and sometimes class imbalance problem will have negative impact on accuracy.

We would like to see how the accuracy will be when trained with a balanced data set. We still use the 1500 features selected in Experiment 2 and train a classifier with SVM as it usually yields a higher precision than the random forest. The result is shown in table 18. We can find in the result that the precision has increased from 0.822 in Experiment to 0.852 in this experiment.

Class	Precision	Recall	F-Measure
Profile	0.852	0.754	0.800
Non-profile	0.779	0.869	0.822

**Table 18: Result of Experiment 6 Using SVM**

#### **4.8 Discussion**

In the above experiments, we have compared the classifiers trained using SVM and the random forest algorithms. From the results, we can see that both algorithms have their own

merits when dealing with our data. SVM tends to give a better precision than the random forest when predicting profile pages, while the random forest beats SVM in recall. However, when we consider the precision and the recall together, in other words the F-measure, SVM and the random forest have a very close performance. In Experiment 1, the F-measure of SVM is 0.793 and the F-measure of the random forest is 0.789. In Experiment 3, the F-measure of SVM is 0.792 and the F-measure of the random forest is 0.793. The advantage of having both precision and recall is that one is more important than the other in many circumstances. Nevertheless, the two metrics clearly trade off against one another because you can always get recall 1 (but very low precision) by classifying all web pages as profile pages. So it's very difficult to get a good precision while keeping the recall constant. We can see this clearly from figure 11: the highest recall we have got is 0.866 at a precision of 0.663. When the precision is getting higher, the recall declines. Thus, it really depends on the domain and what we intend to do with the classifier.

We also compared different features in order to find the best features. When we proposed some custom features such as street type, postal code and business name in chapter 3, we thought that these features could be helpful because they were quite related to a business's profile. However, what we find from the results is that custom features haven't improved the classification performance. When trained using 3092 word tokens with SVM and the random forest, the classifiers' F-measures are 0.793 and 0.789 accordingly. However, after adding custom features to the feature vector, the classifiers' F-measures are 0.792 and 0.793 accordingly. In addition, our experiment on URL features didn't give us a good enough performance either. It tells us that the title and snippet contain enough information to predict a profile page. Although word tokens from title and snippet can predict profile pages, we can



still improve both precision and recall by reducing the number of word tokens (i.e. the dimensionality of the feature vector). In our experiment, the top 1500 word tokens selected using information gain yield the highest precision and recall.

At last, considering that the original data set is imbalanced, we adopted the oversampling technique to create a new balanced data set hoping that the result will be more accurate. Our experiment on balanced data set proves that the precision has increased from 0.822 to 0.852. It proves that in our case a balanced data set is better than an imbalanced one in terms of accuracy. Therefore, in circumstances where accuracy matters much more than recall, methods like resampling should be adopted to deal with the class imbalance problem.

## **Chapter 5: Conclusion**

### **5.1 Summary of the Research**

Web page classification is important to online marketing companies like SweetIQ as it provides more insight into the distribution of the types of web pages the company is crawling and saves a lot of recourse to pinpoint the information the company is searching for.

In this research, we first reviewed major concepts of web page classification such as feature selection and feature extraction. Then we introduced several web page classification algorithms and metrics used to evaluate classification performance. Later we analyzed the background of SweetIQ and described the structure of data and the data collection method. Based on the data, we proposed a web page classification method that achieved a good performance.

The current literature on text classification and web page classification indicates that the type of classification method used in the classification task is not the most important thing and in most cases SVM and Random Forest achieve the best performance. The features used in the classification task play a more crucial role than the classification algorithm itself. In this research, in addition to good classifiers, we look for features that can boost classification performance.

After understanding the business needs of SweetIQ, we defined the classes and collected the data. Then we analyzed the data and built an original feature set of 16 features. The features come from two sources - the text in titles and snippets; and the URLs. We conducted several experiments with different selected features in the feature set to train classifiers and compare their performance.

We also examined the class imbalance problem as we have much more profile pages than other types of pages. We reviewed a few techniques that help solve this problem and selected oversampling as our solution because of its simplicity and fitness to our data. We did an experiment to compare the classification performance using a new data set after oversampling to that using the original data set.

## **5.2 Limitations and Future Works**

Although our research on web page classification has reached its aims, there are some limitations. The generalization of our findings is unknown due to the fact that our classification method and experiments' results are constrained by the data and SweetIQ's business needs. We spent much time manually classifying 599 pages and half of them were considered negative examples. We could not test the classification performance on search result pages and review pages because the numbers of these pages are too small. In addition, the topics of the web pages are not equally distributed. Most pages are about restaurants such as McDonald's, Domino's Pizza and Union Oyster House. This means that we have a lot of food-related words in our Bag of Words, which to some extent might affect the classification results.

Our method – using text from title and snippet to classify web pages – can provide a fast yet accurate result. However, we need to evaluate this method with other data sets in different scenarios in the future. Also, we can further analyze the data to explore new features that can improve the classification performance. Finally, research on cutting-edge machine learning technologies such as deep learning will also be considered in order to achieve a better result.

## Appendix A: Python Code of Extracting Title and Snippet from JSON File

```
import sys
import json
import csv

filelist = ["NAME_OF_JSON_FILE"]

def write_to_individual_file(filelist):
    j = 1
    for i in filelist:
        text = []
        f = json.load(file(i))
        for key in f.keys():
            title = f[key]["item"]["title"].encode("utf8")
            snippet = f[key]["item"]["snippet"].encode("utf8")
            text.append(title)
            text.append("\n")
            text.append(snippet)
            text.append("\n")
            with open('profile_text_'+str(j)+'.txt', 'w+') as p:
                for line in text:
                    p.writelines(line)
            j+=1

def write_to_excel(filelist):
    for i in filelist:
        f = json.load(file(i))
        for key in f.keys():
            title = f[key]["item"]["title"].encode("utf8")
            snippet = f[key]["item"]["snippet"].encode("utf8")
            text = title + ' ' + snippet
            with open('profile_text_excel.csv', 'ab') as csvfile:
                writer = csv.writer(csvfile, dialect='excel')
                writer.writerow([text, 'Profile'])

def main():
    write_to_excel(filelist)

if __name__ == '__main__':
    sys.exit(main())
```

## Appendix B: Python Code of Extracting URL from JSON File

```
import csv
from urlparse import urlparse

filelist = ["NAME_OF_FILE"]

def write_to_file(filelist):
    urls = []
    for i in filelist:
        f = json.load(file(i))
        for key in f.keys():
            urls.append(f[key]["item"]["url"].encode('utf8'))
    with open('profile_url.txt', 'w+') as p:
        for url in urls:
            o = urlparse(url)
            if o.path is not None:
                tokens = split_url(o.path)
                for token in tokens:
                    p.writelines(token+' ')
                p.writelines('\n')

def write_to_excel(filelist):
    for i in filelist:
        f = json.load(file(i))
        for key in f.keys():
            url = f[key]["item"]["url"].encode('utf8')
            splited_url = split_url(url)
            text = ' '.join(splited_url)
            with open('profile_url_excel.csv', 'ab') as csvfile:
                writer = csv.writer(csvfile, dialect='excel')
                writer.writerow([text, 'Profile'])

def main():
    write_to_excel(filelist)

if __name__ == '__main__':
    sys.exit(main())
```

## Appendix C: Python Code of Searching for City in Text

```
from geotext import GeoText

def has_city(text):
    City = {'hasCity':0}

    places = GeoText(text)
    if places.cities != []:
        City['hasCity'] = 1
    else:
        pass

    return City
```

## Appendix D: Python Code of Searching for Phone Number in Text

```
import re

def has_phone_number(text):
    regex = r"D*([2-9]\d{2})(\D*)([2-9]\d{2})(\D*)(\d{4})\D*"

    PhoneNumber = {'hasPhoneNumber':0}

    if re.findall(regex, text):
        PhoneNumber['hasPhoneNumber'] = 1
    else:
        pass

    return PhoneNumber
```

## Appendix E: Python Code of Searching for Region in Text

```
import nltk
from nltk.corpus import stopwords
import string

def get_tokens(text):
    no_punctuation = text.translate({ord(k): None for k in string.punctuation})
    tokens = nltk.word_tokenize(no_punctuation)
    return tokens

def has_region(text):
    region_list =
["ON", "QC", "NS", "NB", "MB", "BC", "PE", "SK", "AB", "NL", "AL", "AK", "AZ", "AR", "CA", "CO", "CT",
"DE", "FL", "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD", "MA", "MI", "MN", "MS",
"MO", "MT", "NE", "NV", "NH", "NJ", "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC", "SD",
"TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY"]
    Region = {'hasRegion':0}
    tokens = get_tokens(text)
    filtered = [w for w in tokens if not w in stopwords.words('english') and not w.isdigit()]
    for region in region_list:
        for word in filtered:
            if region == word:
                Region['hasRegion'] = 1
                break
            else:
                pass
    return Region
```



## Appendix F: Python Code of Searching Street Type in Text

```
import nltk
from nltk.corpus import stopwords
import string

def get_tokens(text):
    lowers = text.lower()
    # no_punctuation = lowers.translate(string.punctuation)
    no_punctuation = lowers.translate({ord(k): None for k in string.punctuation})
    tokens = nltk.word_tokenize(no_punctuation)
    return tokens

def has_street_type(text):
    street_type = ["ABBEY",
                   "ACCESS",
                   "ACRES",
                   "AIRE",
                   "ALLEY",
                   "ALLÉE",
                   "AUT",
                   "AV",
                   "AVE",
                   "BAY",
                   "BEACH",
                   "BEND",
                   "BLOC",
                   "BLOCK",
                   "BLVD",
                   "BOUL",
                   "BOURG",
                   "BRGE",
                   "BROOK",
                   "BYPASS",
                   "BYWAY",
                   "C",
                   "CAMPUS",
                   "CAPE",
                   "CAR",
                   "CARREF",
                   "CDS",
                   "CERCLE",
                   "CH",
                   "CHASE",
```

"CIR",  
"CIRCT",  
"CLOSE",  
"COMMON",  
"CONC",  
"CÔTE",  
"COUR",  
"COURS",  
"COVE",  
"CRES",  
"CREST",  
"CRNRS",  
"CROFT",  
"CROIS",  
"CROSS",  
"CRSSRD",  
"CRT",  
"CTR",  
"DALE",  
"DELL",  
"DESSTE",  
"DIVERS",  
"DOWNS",  
"DR",  
"DRPASS",  
"ÉCH",  
"END",  
"ESPL",  
"ESTATE",  
"EXPY",  
"EXTEN",  
"FARM",  
"FIELD",  
"FOREST",  
"FRONT",  
"FWY",  
"GATE",  
"GDNS",  
"GLADE",  
"GLEN",  
"GREEN",  
"GRNDS",  
"GROVE",  
"HARBR",

"HAVEN",  
"HEATH",  
"HGH LDS",  
"HILL",  
"HOLLOW",  
"HTS",  
"HWY",  
"ÎLE",  
"IMP",  
"INLET",  
"ISLAND",  
"KEY",  
"KNOLL",  
"LANDNG",  
"LANE",  
"LANEWY",  
"LINE",  
"LINK",  
"LKOUT",  
"LMTS",  
"LOOP",  
"MALL",  
"MANOR",  
"MAZE",  
"MEADOW",  
"MEWS",  
"MONTÉE",  
"MOOR",  
"MOUNT",  
"MTN",  
"ORCH",  
"PARADE",  
"PARC",  
"PASS",  
"PATH",  
"PEAK",  
"PINES",  
"PK",  
"PKY",  
"PL",  
"PLACE",  
"PLAT",  
"PLAZA",  
"POINTE",

"PORT",  
"PROM",  
"PT",  
"PTWAY",  
"PVT",  
"QUAI",  
"QUAY",  
"RAMP",  
"RANG",  
"RD",  
"RDPT",  
"REACH",  
"RG",  
"RIDGE",  
"RISE",  
"RLE",  
"ROUTE",  
"ROW",  
"RTE",  
"RTOFWY",  
"RUE",  
"RUIS",  
"RUN",  
"SECTN",  
"SENT",  
"SIDERD",  
"SQ",  
"ST",  
"STROLL",  
"SUBDIV",  
"TERR",  
"THICK",  
"TLINE",  
"TOWERS",  
"TRACE",  
"TRAIL",  
"TRNABT",  
"TRUNK",  
"TSSE",  
"VALE",  
"VIA",  
"VIEW",  
"VILLAS",  
"VILLGE",

```
"VISTA",  
"VOIE",  
"WALK",  
"WAY",  
"WHARF",  
"WOOD",  
"WYND"]
```

```
Streets = {'hasStreetType':0}  
tokens = get_tokens(text)  
filtered = [w for w in tokens if not w in stopwords.words('english') and not w.isdigit()]  
for streettype in street_type:  
    for word in filtered:  
        if streettype.upper() == word.upper():  
            Streets['hasStreetType'] = 1  
            break  
        else:  
            pass  
  
return Streets
```

## Appendix G: Python Code of Search for Target Business Name in Text and

### URL Domain

```
import nltk
from nltk.corpus import stopwords
import string

def get_tokens(text):
    lowers = text.lower()
    no_punctuation = lowers.translate({ord(k): None for k in string.punctuation})
    tokens = nltk.word_tokenize(no_punctuation)
    return tokens

def has_target_business_name_in_text(target_business_name, text):
    TargetBusinessName = {'hasTargetBusinessNameText': 0}
    tokenized_business_names = get_tokens(target_business_name)
    tokens = get_tokens(text)
    filtered = [w for w in tokens if not w in stopwords.words('english') and not w.isdigit()]
    for business_name in tokenized_business_names:
        for word in filtered:
            if business_name.upper() == word.upper():
                TargetBusinessName['hasTargetBusinessNameText'] = 1
                break
        else:
            pass

    return TargetBusinessName

def has_target_business_name_in_domain(target_business_name, domain):
    TargetBusinessName = {'hasTargetBusinessNameDomain': 0}
    tokenized_business_names = get_tokens(target_business_name)
    for business_name in tokenized_business_names:
        if business_name.upper() in domain.upper():
            TargetBusinessName['hasTargetBusinessNameDomain'] = 1
            break
        else:
            pass

    return TargetBusinessName
```

## Appendix H: Python Code of Searching for Zip Code and Postal Code in

### Text

```
import re

def has_zip_code(text):
    regex = r"^(^d{5}(-d{4})?$)|(^[ABCEGHJKLMNPRSTVXY]{1}\d{1}[A-Z]{1} *d{1}[A-Z]{1}\d{1}$)"

    ZipCode = {'hasZipCode':0}

    if re.findall(regex, text):
        ZipCode['hasZipCode'] = 1
    else:
        pass

    return ZipCode
```

## Appendix I: Python Code of Splitting URL into Different Components

```
import re
import sys

def split_url(url):
    # request_protocol = r'^\w+:\V\V'
    # domain = r'^\w+(?=:\/\V)'
    protocol_domain = r'^\w+:\V\V?[\w+.]+(?:=\/|:\d+)'
    slash = r'\V'
    dash = r'\-'
    underscore = r'\_'
    dot = r'\.'
    question_mark = r'\?'
    equals_sign = r'\='
    plus = r'\+'
    space = r'\s'
    ampersand = r'\&'
    quotation = r'\"|\\"'
    comma = r'\,'
    colon = r'\:'
    pipe = r'\|'
    url_encoding = r'%\d{2}|\%\d{1}[ABCDEF]|\%[ABCDEF]\d{1}|\%[ABCDEF]{2}'

    regex = protocol_domain
    rest = filter(None, re.split(regex, url))
    # print rest
    regex2 =
    slash+'|'+dash+'|'+underscore+'|'+dot+'|'+question_mark+'|'+equals_sign+'|'+plus+'|'+
    space+'|'+ampersand+'|'+quotation+'|'+comma+'|'+colon+'|'+pipe+'|'+url_encoding
    return filter(None, re.split(regex2, rest[0]))
```



## Appendix J: Python Code of Searching for Directory Provider in URL

### Domain

```
def has_directory_provider(domain):  
  
    directory_providers = ['123poi',  
                           '2findlocal',  
                           '411',  
                           'activediner',  
                           'addyourpoint',  
                           'adsonnow',  
                           'akama',  
                           'allmenus',  
                           'americantowns',  
                           'angieslist',  
                           'apartmentratings',  
                           'autotrader',  
                           'bbb',  
                           'bing',  
                           'bing_maps',  
                           'bizvotes',  
                           'blogthishere',  
                           'bluedoorway',  
                           'brownbook.net',  
                           'business.intuit',  
                           'business.intuit_boorah',  
                           'businessdirectory.bizjournals',  
                           'businessmention',  
                           'canadaplus',  
                           'canadian-lawyers',  
                           'canadianbusinessdirectory',  
                           'canadianhotelguide',  
                           'canadianplanet',  
                           'canpages',  
                           'canplus',  
                           'cardealercheck',  
                           'cars',  
                           'cdnpages',  
                           'citysearch',  
                           'cityseekr',  
                           'citysquares',  
                           'companylist',
```

'cybo',  
'cylex',  
'dakitaki',  
'demandforce',  
'dinehere',  
'directory',  
'discoverourtown',  
'duckduckgo',  
'easyfind',  
'ebusinesspages',  
'ecarnegietech',  
'edmunds',  
'elocal',  
'enrollbusiness',  
'expedia',  
'expressupdate',  
'ezlocal',  
'facebook-pages',  
'facebook',  
'findhere',  
'foodmafia',  
'foodpages',  
'foursquare',  
'fyood',  
'fyp',  
'geckogo',  
'generic\_source',  
'goldbook',  
'google',  
'grubhub',  
'homestars',  
'homestead',  
'ibegin',  
'incpages',  
'infobuz',  
'insiderpages',  
'judysbook',  
'kudzu',  
'lifescrpt',  
'linktown.wcnc',  
'local.botw',  
'local',  
'local.mapquest',  
'local.yahoo',

'localeze',  
'localguides',  
'localstore',  
'lonelyplanet',  
'makbiz.net',  
'makemeheal',  
'manta',  
'maps.google',  
'maps.yahoo',  
'menuism',  
'merchantcircle',  
'mojopages',  
'n49',  
'neustarlocaleze.biz',  
'number',  
'openlist',  
'openorclosed',  
'ourbis',  
'pagesjaunes',  
'patch',  
'phonepages',  
'placestars',  
'plus.google',  
'profilecanada',  
'restaurantica',  
'restomontreal',  
'sample',  
'search.yahoo',  
'showmelocal',  
'superpages',  
'thecanadiandirectory',  
'townrenowned',  
'travel.yahoo',  
'tripadvisor',  
'tupalo',  
'ucomparehealthcare',  
'usa121',  
'voices.yahoo',  
'weddingbee',  
'wellness',  
'wikidomo',  
'worldweb',  
'www.dexknows',  
'yahoo.yellowpages',

```
'yalwa',  
'yellowbook',  
'yellowbot',  
'yellowee',  
'yellowise',  
'yellowpages',  
'yelp',  
'yelp_map',  
'zvents',  
]
```

```
DirectoryProvider = {'hasDirectoryProvider':0}  
for directory_provider in directory_providers:  
    if directory_provider.upper() in domain.upper():  
        DirectoryProvider['hasDirectoryProvider'] = 1  
        break  
    else:  
        pass  
  
return DirectoryProvider
```

## Appendix K: Python Code of Searching for Review Provider in URL

### Domain

```
def has_review_provider(domain):  
    review_providers = ['addyourpoint',  
                        'apartmentratings',  
                        'bing',  
                        'bizvotes',  
                        'business.intuit',  
                        'canpages',  
                        'cardealercheck',  
                        'cars',  
                        'citysearch',  
                        'citysquares',  
                        'cylex',  
                        'demandforce',  
                        'dinehere',  
                        'ebusinesspages',  
                        'edmunds',  
                        'facebook',  
                        'findhere',  
                        'findthebest',  
                        'foodmafia',  
                        'foodpages',  
                        'foursquare',  
                        'fyood',  
                        'fypie',  
                        'generic_source',  
                        'goldbook',  
                        'grubhub.',  
                        'homestars',  
                        'insiderpages',  
                        'judysbook',  
                        'just-eat',  
                        'kudzu',  
                        'lacartes',  
                        'linktown.wcnc',  
                        'listings.findthecompany',  
                        'local.botw',  
                        'local',  
                        'localguides',  
                        'localstore',
```

```
'local.yahoo',
'makemeheal',
'manta',
'menuism',
'mojopages',
'mysheriff',
'n49',
'ourbis',
'pagesjaunes',
'patch',
'plus.google',
'profilecanada',
'restaurantica',
'restomontreal',
'superpages',
'townrenowned',
'tripadvisor',
'tupalo',
'ucomparehealthcare',
'weddingbee',
'wellness',
'dexknows',
'yellowbook',
'yellowbot',
'yellowee',
'yellowise',
'yellowpages',
'yelp',
'zvents',
]
```

```
ReviewProvider = {'hasReviewProvider':0}
for review_provider in review_providers:
    if review_provider.upper() in domain.upper():
        ReviewProvider['hasReviewProvider'] = 1
    else:
        pass

return ReviewProvider
```

## References

- Agrawal, A., & Gupta, U. (2014). Extraction based approach for text summarization using k-means clustering. *International Journal of Scientific and Research Publications*, 4(11).
- Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 6(1), 20–29.
- Bouyer, A., & Mousavi, S. M. (2009). A Predictive Approach for Selecting Suitable Computing Nodes in Grid Environment by Using Data Mining Technique. In T. Kim, L. T. Yang, J. H. Park, A. C.-C. Chang, T. Vasilakos, Y. Zhang, ... Y.-S. Jeong (Eds.), *Advances in Computational Science and Engineering* (pp. 190–205). Springer Berlin Heidelberg. Retrieved from [http://link.springer.com/chapter/10.1007/978-3-642-10238-7\\_16](http://link.springer.com/chapter/10.1007/978-3-642-10238-7_16)
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32. <http://doi.org/10.1023/A:1010933404324>
- Calado, P., Cristo, M., Moura, E., Ziviani, N., Ribeiro-Neto, B., & Gonçalves, M. A. (2003). Combining Link-based and Content-based Methods for Web Document Classification. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (pp. 394–401). New York, NY, USA: ACM. <http://doi.org/10.1145/956863.956938>
- Chang, E., & Tong, S. (n.d.). SVM Active – Support Vector Machine Active Learning for Image Retrieval. In *Proceedings of the 9 th ACM international conference on Multimedia* (pp. 107–118).
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1), 321–357.
- Chen, H., & Dumais, S. (2000). Bringing Order to the Web: Automatically Categorizing Search Results. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 145–152). New York, NY, USA: ACM. <http://doi.org/10.1145/332040.332418>
- Chen, R.-C., & Hsieh, C.-H. (2006). Web page classification based on a support vector machine using a weighted vote schema. *Expert Systems with Applications*, 31(2), 427–435. <http://doi.org/10.1016/j.eswa.2005.09.079>
- Choi, B., & Yao, Z. (2005). Web Page Classification\*. In P. W. Chu & P. T. Y. Lin (Eds.), *Foundations and Advances in Data Mining* (pp. 221–274). Springer Berlin Heidelberg. Retrieved from [http://link.springer.com/chapter/10.1007/11362197\\_9](http://link.springer.com/chapter/10.1007/11362197_9)
- De Kok, D., & Brouwer, H. (2011). *Natural language processing for the working programmer*. Del.
- Du, K.-L., & Swamy, M. N. S. (2013). *Neural Networks and Statistical Learning*. Springer Science & Business Media.
- Dumais, S., & Chen, H. (2000). Hierarchical classification of Web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 256–263). ACM.

- Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, 15, 3133–3181.
- Glover, E. J., Tsioutsoulis, K., Lawrence, S., Pennock, D. M., & Flake, G. W. (2002). Using Web Structure for Classifying and Describing Web Pages. In *Proceedings of the 11th International Conference on World Wide Web* (pp. 562–569). New York, NY, USA: ACM. <http://doi.org/10.1145/511446.511520>
- Haleem, H., Niyas, C., Verma, S., Kumar, A., & Ahmad, F. (2014). Effective Probabilistic Model for Webpage Classification. In S. Sengupta, K. Das, & G. Khan (Eds.), *Emerging Trends in Computing and Communication* (pp. 277–290). Springer India. Retrieved from [http://link.springer.com/chapter/10.1007/978-81-322-1817-3\\_29](http://link.springer.com/chapter/10.1007/978-81-322-1817-3_29)
- Han, J., & Kamber, M. (2012). *Data Mining: Concepts and Techniques*. Elsevier. Retrieved from <http://books.google.ca/books?id=dfs2kgEACAAJ>
- Jayalekshmi, K. (2014). Information Retrieval for Enhancing Retention Management and Decision Making In Telecom Sector. *Indian Journal of Research in Management, Business and Social Sciences*, 2(1(A)).
- Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. In C. Nédellec & C. Rouveilol (Eds.), *Machine Learning: ECML-98* (pp. 137–142). Springer Berlin Heidelberg. Retrieved from <http://link.springer.com/chapter/10.1007/BFb0026683>
- Kamber, M., Winstone, L., Gong, W., Cheng, S., & Han, J. (1997). Generalization and decision tree induction: efficient classification in data mining. In *Seventh International Workshop on Research Issues in Data Engineering, 1997. Proceedings* (pp. 111–120). <http://doi.org/10.1109/RIDE.1997.583715>
- Kan, M.-Y., & Thi, H. O. N. (2005). Fast Webpage Classification Using URL Features. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management* (pp. 325–326). New York, NY, USA: ACM. <http://doi.org/10.1145/1099554.1099649>
- Khoshgoftaar, T. M., Golawala, M., & Van Hulse, J. (2007). An Empirical Study of Learning from Imbalanced Data Using Random Forest. In *19th IEEE International Conference on Tools with Artificial Intelligence, 2007. ICTAI 2007* (Vol. 2, pp. 310–317). <http://doi.org/10.1109/ICTAI.2007.46>
- Kubat, M., Matwin, S., & others. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML* (Vol. 97, pp. 179–186). Nashville, USA.
- Kwon, O.-W., & Lee, J.-H. (2000). Web Page Classification Based on K-nearest Neighbor Approach. In *Proceedings of the Fifth International Workshop on on Information Retrieval with Asian Languages* (pp. 9–15). New York, NY, USA: ACM. <http://doi.org/10.1145/355214.355216>
- Lu, Q., & Getoor, L. (2003). Link-based Classification. In *International Conference on Machine Learning*.



- Marath, S. T., Shepherd, M., Milios, E., & Duffy, J. (2014). Large-Scale Web Page Classification. In *2014 47th Hawaii International Conference on System Sciences (HICSS)* (pp. 1813–1822). <http://doi.org/10.1109/HICSS.2014.229>
- McCallum, A., Nigam, K., & others. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization* (Vol. 752, pp. 41–48). Citeseer.
- Mollineda, V. G. J. S. R., & Sotoca, R. A. J. (2007). The class imbalance problem in pattern classification and learning. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.329.4200&rep=rep1&type=pdf>
- Oh, H.-J., Myaeng, S. H., & Lee, M.-H. (2000). A practical hypertext categorization method using links and incrementally available class information. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 264–271). ACM.
- Özel, S. A. (2011). A Web page classification system based on a genetic algorithm using tagged-terms as features. *Expert Systems with Applications*, 38(4), 3407–3415. <http://doi.org/10.1016/j.eswa.2010.08.126>
- Oztekin, B. U., & Chiu, P.-W. A. (2014, May 6). Generating improved document classification data using historical search results.
- Pant, B., Pant, K., & Pardasani, K. R. (2009). Decision Tree Classifier for Classification of Plant and Animal Micro RNA's. In Z. Cai, Z. Li, Z. Kang, & Y. Liu (Eds.), *Computational Intelligence and Intelligent Systems* (pp. 443–451). Springer Berlin Heidelberg. Retrieved from [http://link.springer.com/chapter/10.1007/978-3-642-04962-0\\_51](http://link.springer.com/chapter/10.1007/978-3-642-04962-0_51)
- Patel Brijain, R., & Rana, K. K. (2014). A Survey on Decision Tree Algorithm for Classification. In *International Journal of Engineering Development and Research* (Vol. 2). IJEDR.
- Peffer, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3), 45–77. <http://doi.org/10.2753/MIS0742-1222240302>
- Qazi, A., Raj, R. G., Tahir, M., Cambria, E., & Syed, K. B. S. (2014). Enhancing Business Intelligence by Means of Suggestive Reviews. *The Scientific World Journal*, 2014, e879323. <http://doi.org/10.1155/2014/879323>
- Qi, X. (2012). Web page classification and hierarchy adaptation.
- Qi, X., & Davison, B. (2009). Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41(2), 1–31. <http://doi.org/10.1145/1459352.1459357>
- Ribeiro, A., Fresno, V., Garcia-Alegre, M. C., & Guinea, D. (2003). Web Page Classification: A Soft Computing Approach. In E. Menasalvas, J. Segovia, & P. S. Szczepaniak (Eds.), *Advances in Web Intelligence* (pp. 103–112). Springer Berlin Heidelberg. Retrieved from [http://link.springer.com/chapter/10.1007/3-540-44831-4\\_12](http://link.springer.com/chapter/10.1007/3-540-44831-4_12)

- Riboni, D. (2002). *Feature Selection for Web Page Classification*.
- Schapire, R. E. (2013). Explaining AdaBoost. In B. Schölkopf, Z. Luo, & V. Vovk (Eds.), *Empirical Inference* (pp. 37–52). Springer Berlin Heidelberg. Retrieved from [http://link.springer.com/chapter/10.1007/978-3-642-41136-6\\_5](http://link.springer.com/chapter/10.1007/978-3-642-41136-6_5)
- Sebastiani, F. (1999). A tutorial on automated text categorisation. In *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence* (pp. 7–35). Buenos Aires, AR.
- Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Comput. Surv.*, 34(1), 1–47. <http://doi.org/10.1145/505282.505283>
- Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *Journal of Data Warehousing*, 5(4), 13–22.
- Shen, D., Chen, Z., Yang, Q., Zeng, H.-J., Zhang, B., Lu, Y., & Ma, W.-Y. (2004). Web-page classification through summarization. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 242–249). ACM.
- Slawski, B. (2010a, October 12). How Google May Use Categories as a Search Ranking Factor. Retrieved January 22, 2014, from <http://www.seobythesea.com/2010/10/how-google-may-use-categories-as-a-search-ranking-factor/>
- Slawski, B. (2010b, October 19). Improved Web Page Classification from Google for Rankings and Personalized Search. Retrieved from <http://www.seobythesea.com/2010/10/improved-web-page-classification-from-google-for-rankings-and-personalized-search/>
- Sun, A., Lim, E.-P., & Ng, W.-K. (2002). Web Classification Using Support Vector Machine. In *Proceedings of the 4th International Workshop on Web Information and Data Management* (pp. 96–99). New York, NY, USA: ACM. <http://doi.org/10.1145/584931.584952>
- SweetIQ. (2014). *SweetIQ Research Projects*.
- Vaishnavi, V., & Kuechler, W. (2004). Design Research in Information Systems. Retrieved from <http://www.desrist.org/design-research-in-information-systems/>
- Wen, H., Fang, L., & Guan, L. (2008). Automatic Web Page Classification Using Various Features. In Y.-M. R. Huang, C. Xu, K.-S. Cheng, J.-F. K. Yang, M. N. S. Swamy, S. Li, & J.-W. Ding (Eds.), *Advances in Multimedia Information Processing - PCM 2008* (pp. 368–376). Springer Berlin Heidelberg. Retrieved from [http://link.springer.com/chapter/10.1007/978-3-540-89796-5\\_38](http://link.springer.com/chapter/10.1007/978-3-540-89796-5_38)
- Wibowo, W., & Williams, H. E. (2002). Simple and Accurate Feature Selection for Hierarchical Categorisation. In *Proceedings of the 2002 ACM Symposium on Document Engineering* (pp. 111–118). New York, NY, USA: ACM. <http://doi.org/10.1145/585058.585079>