Language Identification of Web Pages Based on Improved N-gram Algorithm

Yew Choong Chew[1], Yoshiki Mikami[2], Robin Lee Nagano[3]

[1] Information Science and Control Engineering, Nagaoka University of Technology
Nagaoka, Niigata 940-2188, Japan

[2] Information Science and Control Engineering, Nagaoka University of Technology
Nagaoka, Niigata 940-2188, Japan

[3] Foreign Language Education Center, University of Miskolc
Miskolc, Egyetemvaros H3515 Hungary

Abstract

Language identification of written text in the domain of Latin-script based languages is a well-studied research field. However, new challenges arise when it is applied to non-Latin-script based languages, especially for Asian languages' web pages. The objective of this paper is to propose and evaluate the effectiveness of adapting Universal Declaration of Human Rights and Biblical texts as a training corpus, together with two new heuristics to improve an n-gram based language identification algorithm for Asian languages. Extension of the training corpus produced improved accuracy. Improvement was also achieved by using byte-sequence based HTML parser and a HTML character entities converter. The performance of the algorithm was evaluated based on a written text corpus of 1,660 web pages, spanning 182 languages from Asia, Africa, the Americas, Europe and Oceania. Experimental result showed that the algorithm achieved a language identification accuracy rate of 94.04%.

Keywords: *Asian Language, Byte-Sequences, HTML Character Entities, N-gram, Non-Latin-Script, Language Identification.*

1. Introduction

With the explosion of multi-lingual data on the Internet, the need and demand for an effective automated language identifier for web pages is further increased. Wikipedia, a rapidly growing multilingual Web-based encyclopedia on the Internet, can serve as a measure of the multilingualism of the Internet. We can see that the number of web pages and languages (both Latin-script and non-Latin-script based) has increased tremendously in recent years, as shown in Figure 1.

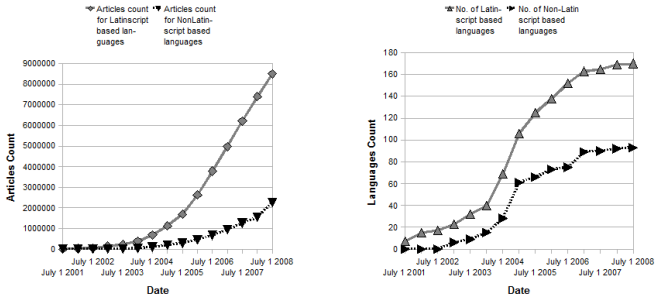

Figure 1 Articles count and number of languages (Latin-script and non-Latin-script based) on Wikipedia's language projects, 2001 to 2008.

1.1 Unreliable HTML and XML's Language Attribute

The Hyper Text Markup Language (HTML) is the standard encoding scheme used to create and format a web page. In the latest HTML 4.01 specification, there is a *lang* attribute that defined to specify the base language of text in a web page. Similarly, the Extensible Markup Language (XML) 1.0 specification includes a special attribute named *xml:lang* that may be inserted into documents to specify the language used in the contents. However, the reality remains that many web pages do not make use of this attribute or, even worse, use it incorrectly and provide misleading information.

Using the validation corpus in this study as a sample, we found that only 698 web pages out of 1,660 contain *lang* attribute, as shown in Table 1. When *lang* attribute is available, it does not always indicate the correct language of a web page. Table 1 shows that 72.49% of web pages with *lang* attribute produced correct language indication. Overall, only 30.48% of web pages in our sample

produced correct language identification result from *lang* attribute. Therefore, we are left with deducing information from the text to determine the language of a given web page. This is the domain of language identification.

Table 1 Number of web pages with *lang* attribute and percentage of correct language identification using *lang* attribute as indicator, based on validation corpus of this study.

	Correct Pages	Total Pages	Percent Correct
Web pages with *lang* attribute	506	698	72.49%
Web pages without *lang* attribute	0	962	0.00%
Total	506	1660	30.48%

1.2 Language Identification

Language identification is the fundamental requirement prior to any language based processing. For example, in a fully automatic machine translation system, language identification is needed to detect the source language correctly before the source text can be translated to another language. Many studies of language identification on written text exists, for example, [Gold 1967] [William B. Cavnar 1994] [Dunning 1994] [Clive Souter 1994] [Michael John Martino 2001] [Izumi Suzuki 2002] [ÖLVECKÝ 2005] [Bruno Martins 2005], just to name a few.

A comparative study on language identification methods for written text was reported in [Lena Grothe 2008]. Their paper compares three different approaches to generate language models and five different methods for language classification.

The first approach generates language model based on "short words". It uses only words up to a specific length to construct the language model. The idea behind this approach is that language specific common words having mostly only marginal length. [Grefenstette 1995] tokenized and extracted all words with a length up to five characters that occurred at least three times from one million characters of text for ten European languages. [Prager 1999] used still shorter words four or fewer characters, for thirteen Western European languages.

The second approach generates language model is based on "frequent words". It uses a specified number of the most frequent words occurring in a text to construct the language model. For instance, the most frequent one hundred words were used in [Clive Souter 1994] and [Michael John Martino 2001], while [Eugene Ludovik 1999] used the most frequent one thousand words.

The third approach generates a language model based on "n-gram". An n-gram is a subsequence of N items from a given sequence. [William B. Cavnar 1994] [Grefenstette 1995] [Prager 1999] used a character-sequence based n-gram method, while [Dunning 1994] used a byte-sequence based n-gram method.

The generated language model is used as the input for language classification method. Many language classification methods had been proposed before, these include Ad-Hoc Ranking [William B. Cavnar 1994], Markov Chains in combination with Bayesian Decision Rules [Dunning 1994], Relative Entropy [Penelope Sibun 1996], Vector Space Model [Prager 1999] and Monte Carlo sampling [Poutsma 2001].

Table 2 shows the information of five selected studies. Previous studies reported excellent results on a few selected Latin-script based languages. Japanese and Russian are the only two exceptional here. The Japanese language, written with the Japanese logographs and syllabaries, and Russian, written in the Cyrillic script, can be easily distinguished from the Latin-script based languages, and also from each other. However, the performance of language identification on non-Latin-script based languages remains unknown.

Most studies in Table 2 are focusing on plain text content. There is only two previous study evaluate its language identification algorithm against web page. Although the proposed heuristics work well on Latin-script based web page, they might not able to effectively handling the non-Latin-script based web page. Usually, non-Latin-script has different bits setting, while many non-Latin-scripts in Asia are encoded in legacy fonts. Besides, none of the studies mentioned about HTML entities, which indeed is commonly used in non-Latin-script based web page.

As previous studies are focusing on Latin-script based languages, most of them adopted a training corpus with limited number of Latin-script based languages only. Thus, our research aims to improve language identification on a broader range of languages, especially for non-Latin-script and added support for web page content. The initial target is set at the 185 languages given in ISO 639-1.

1.3 Hyper Text Markup Language and HTML Parser

[Penelope Sibun 1996] states that language identification is a straightforward task. We argue that their claim is only true for language identification on Latin-script based plain text document. Web pages are different from plain text documents since they contain the HTML tags that are used to publish the document on the Web. In order to correctly identify the language of a web page, a HTML parser is

Table 2 Five selected language identification studies on written text with information of languages coverage, training corpus, validation corpus and accuracy of identification.

Research	Language Coverage	Training Corpus	Validation Corpus	Percent Correct
[William B. Cavnar 1994]	English, Portuguese, French, German, Italian, Spanish, Dutch, Polish	Unspecified	3713 text sample from soc.culture newsgroup	99.8%
[Dunning 1994]	Dutch, Polish	A set of text samples from Consortium for Lexical Research	Another set of text samples from Consortium for Lexical Research	99.9%
[Clive Souter 1994]	Dutch/Friesian, English, French, Gaelic, German, Italian, Portuguese, Serbo-Croat, Spanish	A set of text samples from Oxford Text Archive, each is 100 kilobytes	Another set of text samples from Oxford Text Archive	94.0%
[Poutsma 2001]	Danish, Dutch, English, French, German, Italian, Norwegian, Portuguese, Spanish, Swedish	90% of text samples from European Corpus Initiative Multilingual Corpus	10% of text samples from European Corpus Initiative Multilingual Corpus	Result in chart format
[Bruno Martins 2005]	Danish, Dutch, English, Finnish, French, German, Italian, Japanese, Portuguese, Russian, Spanish, Swedish	Text samples of 23 languages collected from newsgroups and the Web	Web pages of 12 languages collected from newsgroups and the Web	91.25%

needed in order to remove the HTML tags and to extract the text content for language identification.

An HTML parser usually processes text based on character sequences. The HTML parser read the content of a web page into character sequences, and then marked the blocks of HTML tags and the blocks of text content. At this stage, the HTML parser uses a character encoding scheme to encode the text. HTML parser usually depends on a few methods (describes in subsection Character and Byte-sequence based HTML Parser) to determine the correct character encoding scheme to be used. If no valid character encoding is detected, the parser will apply a predefined default encoding.

Today, a common approach is to use UTF-8 (a variable-length character encoding for Unicode) as the default encoding, as the first 128 characters of Unicode map directly to their ASCII correspondents. However, using UTF-8 encoding on non-Latin-script based web pages might cause the application to apply a wrong character encoding scheme and thus return an encoded text that is different from its web origin.

Using the validation corpus of this study as an example, we found that 191 web pages were with doubtful character encoding information. Table 3 shows an example of text rendered by wrongly character encoding. The authors only show one example as the reason for wrong character encoding is identical.

1.4 Unicode and HTML Character Entities

Unicode is a computing industry standard that allowing computers to represent and manipulate text expressed in most of the world's writing systems. The Unicode Consortium has the ambitious goal of eventually replacing existing character encoding schemes with Unicode, as many of the existing schemes are limited in size and scope. Unicode characters can be directly input into a web page if the user's system supports them. If not, HTML character entities provide an alternate way of entering Unicode characters into a web page.

There are two types of HTML character entities. The first type is called character entity references, which take the form *&EntityName;*. An example is *©* for the copyright symbol. The second type is referred as numeric character references, which takes the form *&#N;*, where *N* is either a decimal number (base 10) or a hexadecimal number for the Unicode code point. When *N* represents a hexadecimal number, it must be prefixed by *x*. An

Table 3 Text rendered and language identification results on a selected web page with misleading *charset* information.

Web Page	HTML Parser (Character-sequence based)			Web Origin	
	Detected Charset	Text Rendered	Identified As	Text Rendered	Identified As
chinese-05-newscn.htm	No Match, use default UTF-8	????	English, Latin, Latin1	杭洲旅遊	Chinese, Simplified Chinese,GB2312

examples of these entities is *和* (base 10) or *和* (base 16) for the Chinese and also Japanese character "和".

Using HTML character entities, any system is able to input Unicode characters into a web page. However, this causes a problem for language identification as the language property is now represented by label and numeric references. In order to identify the language of an HTML character-entity-encoded web page, we propose a HTML character entity converter to translate such entities to the byte sequences of its corresponding Unicode code point.

1.5 Organization of this paper

The remaining of this paper is ordered in the following structure. The authors review related works in the next section. In Methodology section, the authors describe the language identification process and the new heuristics. In Data and Experiments section, the authors explain the nature and preparation of training and validation corpus; followed by description on how the experiments are setup and the purposes of them. In the Result and Discussion section, the authors present the results from the experiments. In the last section, the authors draw conclusions and propose a few areas for future work.

2. Related Work

2.1 Martin Algorithm

In [Bruno Martins 2005], the authors discussed the problem of automatically identifying the language of a given web page. They claimed that web page is generally contained more spelling errors, multilingual and short text, therefore, it is harder for language identification on the web pages. They adapted the well-known n-gram based algorithm from [William B. Cavnar 1994], complemented it with a more efficient similarity measure [Lin 1998] and heuristics to better handle the web pages. The heuristics included the following six steps:

i. Extract the text, the markup information, and meta-data.
ii. Use meta-data information, if available and valid.
iii. Filter common or automatically generated strings. For example, "This page uses frames".
iv. Weight n-grams according to HTML markup. For example, n-grams in the title section have more weight than n-grams in meta-data section.
v. Handle situations when there is insufficient data. When a web page has less than 40 characters, the system reports "unknown language".
vi. Handle multilingualism and the "hard to decide" cases. When a document cannot be clearly classified to one language, the system will re-apply the algorithm, and weight the largest text block as three times more important than the rest.

In the experiment, they constructed 23 different language models from textual information extracted from newsgroups and the Web. They tested the algorithm using testing data in 12 different languages, namely Danish, Dutch, English, Finnish, French, German, Italian, Japanese, Portuguese, Russian, Spanish and Swedish, respectively. The total number of documents for testing is 6,000, with 500 documents for each language. The testing data were crawled from on-line newspapers and Web portals. Overall, the best identification result returned accuracy of 91.25%, which was lower than other researches on text document. The authors believe that this is due to the much noisier nature of the text in web page.

2.2 Suzuki Algorithm

In [Izumi Suzuki 2002], the method is different from conventional n gram based methods in the way that its threshold for any categories is uniquely predetermined. For every identification task on target text, the method must be able to respond to either “correct answer” or “unable to detect”. The authors used two predetermined values to decide which answer should respond to a language identification task. The two predetermined values are UB (closer to the value 1) and LB (not close to the value 1), with a standard value of 0.95 and 0.92, respectively. The basic unit used in this algorithm is trigram. However, the authors refer to it as a 3-byte shift-codon.

In order to detect the correct language of a target text, the algorithm will generate a list of shift-codons from the target text. The target’s shift-codons will then compare to the list of shift-codons in training texts. If one of the matching rates is greater than UB, while the rest is less than LB, the algorithm will report that a “correct answer” has been found. The language of the training text with matching rate greater than UB is assumed to be language of the target text. By this method, the algorithm correctly identified all test data of English, German, Portuguese and Romanian languages. However, it failed to correctly identify the Spanish test data.

3. Methodology

The general paradigm of language identification can be divided into two stages. First, a set of language model is generated from a training corpus during the training phase. Second, the system constructs a language model from the target document and compares it to all trained language models, in order to identify the language of the target document during the identification phase. The algorithm used in this study adopted this general paradigm; however, it contains two new heuristics to properly handle web pages. The first heuristic is to remove HTML tags in byte-sequence stream. The second heuristics is to translate HTML character entities to byte sequences of their Unicode code point. The algorithm only takes text and HTML documents as valid input. The overall system flow of language identification process is shown in Figure 2.

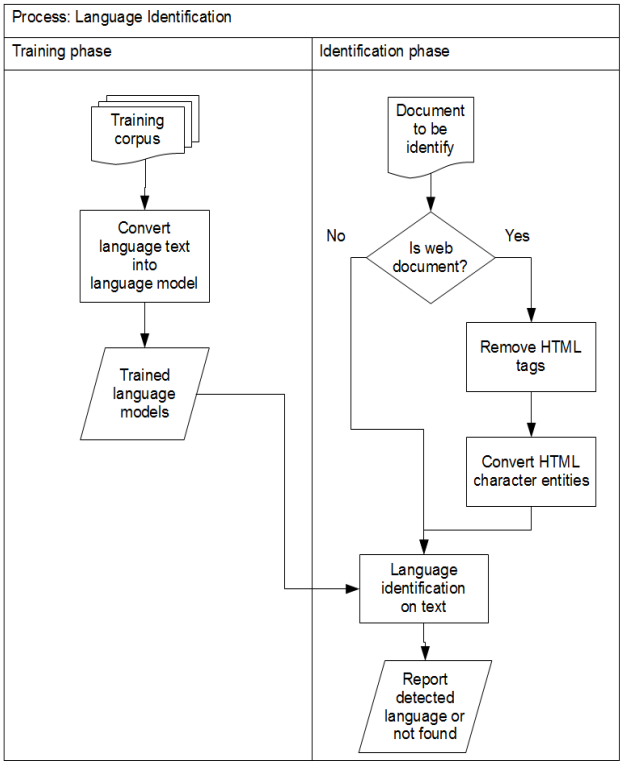

Figure 2 System flowchart for language identification process in this paper.

3.1 Byte-sequence based n-gram algorithm

An n-gram is a sub sequence of *N* items from a longer sequence. An n-gram order 1 (i.e. *N*=1) is referred to as a monogram; n-gram order 2 as a bi-gram and n-gram order 3 as a trigram. Any other is generally referred to as "N-gram". This paper adapted the n-gram based algorithm proposed by [Izumi Suzuki 2002]. The algorithm generates language model from text document into trigrams of byte sequences. For example, the trigrams for the Japanese word "こんにちわ" (or 82 B1 82 F1 82 C9 82 BF 82 CD in the Shift-JIS character encoding scheme) are highlighted as follows:

82 B1 82 F1 82 C9 82 BF 82 CD
82 **B1 82 F1** 82 C9 82 BF 82 CD
82 B1 **82 F1 82** C9 82 BF 82 CD
82 B1 82 **F1 82 C9** 82 BF 82 CD
82 B1 82 F1 **82 C9 82** BF 82 CD
82 B1 82 F1 82 **C9 82 BF** 82 CD
82 B1 82 F1 82 C9 **82 BF 82** CD
82 B1 82 F1 82 C9 82 **BF 82 CD**

The language classification method is based on trigram frequency. The trigram distribution vector of training document has no frequency information. Only the target document has a frequency-weighted vector. In order to detect the correct language of a target document, the algorithm will generates a list of byte-sequence based trigrams from the target document, together with the frequency information of each trigram. The target document's trigrams will then be compared to the list of byte-sequence based trigrams in every training language model. If a target's trigram matches a trigram in the training language model, its frequency value is added to the matching counter. After all trigrams from target document have been compared to trigrams in training language model, the matching rate is calculated by dividing the final matching counter by the total number of target's trigrams.

The matching process for detecting a language can be summarizing as below:

i. Let N be the number of trigrams in target document.
ii. All the trigrams from the target document u_1, u_2, ..., u_N are listed. Let u_j be the j^{th} trigram in the target language model.
iii. Let T_i be the i^{th} language model in the training corpus. R_i (or R-values) is calculated from every i^{th} language model using equation (1), where R_i is the rate at which the set of trigrams in i^{th} language model of the training corpus appears in the target document.

$$R_i = \sum_{j=1}^{n} \frac{f(u_j)}{n}, \quad \text{where } f(u_j) = \begin{cases} 1 & \text{if } u_j \in T_i \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$

3.2 Character and Byte-sequence based HTML Parser

In order to correctly process a web page, a HTML parser must ascertain what character encoding scheme is used to encode the content. This section describes how to detect

Table 4 Possible scenarios of character encoding scheme determination.

Encoding in HTTP content-type	Override HTTP server-side encoding	Encoding in XML declaration	Encoding in HTML meta charset element	Default encoding by User's application	Result of character encoding detection
Correct	No	Any	Any	Any	Correct
Wrong	No	Any	Any	Any	Wrong
Any	Yes	Correct	Any	Any	Correct
Any	Yes	Wrong	Any	Any	Wrong
Any	Yes	Missing	Correct	Any	Correct
Any	Yes	Missing	Wrong	Any	Wrong
Any	Yes	Missing	Missing	Correct	Correct

the character encoding in Hypertext Transfer Protocol (HTTP) header, XML or HTML.

When a web page is transmitted via the HTTP, the Web server will sent the character encoding in the content-type field of the HTTP header, such as *content-type :text/html; charset=UTF-8*. The character encoding can also be declared within the web page itself. For XML, the declaration is at the beginning of the markup, for instance, *<?xml version="1.0" encoding="utf-8"?>* for HTML, the declaration is within the *<meta>* element, such as *<meta http-equiv="content-type" content="text/html; charset=UTF-8">*. If there is no valid character encoding information detected, a predefined character encoding scheme will be invoked. The default character encoding scheme varies depending on the localization of the application. In the case of conflict between multiple encoding declarations, precedence rules apply to determine which declaration shall be used. The precedence is as follows, with HTTP content-type being the highest priority:

i. HTTP content-type
ii. XML declaration
iii. HTML Meta charset element

Since information in the HTTP header overrides information in the web page, it is therefore important to ensure that the character encoding sent by the Web server is correct. However, in order to serve file or files using a different encoding than that specified in the Web server's default encoding, most Web serves allow the user to override the default encoding defined in HTTP content-type. Table 4 illustrates all possible scenarios of character encoding scheme determination.

Table 4 shows that misleading and missing character encoding information would probably lead to the wrong result. Therefore, it is quite possible that a character-sequence based HTML parser might apply an incorrect character encoding scheme to web pages without valid character encoding information, especially on non-Latin-script web pages.

The HTML parser implemented in this paper is unique in that it processes the content of a web page based on byte sequences, thus avoiding the above mentioned problem. By using byte sequences, it eliminates the need to detect and apply character encoding scheme on the content extracted from the web page. The HTML parser parses the web page in a linear fashion. It searches for HTML tags from the beginning to the end of page. It looks for valid HTML start and end tags and marks all blocks of HTML tags. The parser removes all detected HTML blocks and return remaining content in byte sequences for language identification. The parser searches in sequence of bytes instead of characters. For example, in order to determine the locations of *<body>* and *</body>* tags in a web page, the parser searches for 3C 62 6F 64 79 3E and 3C 2F 62 6F 64 79 3E, respectively. The parser keeps a list of byte-sequence based HTML tags and uses them to remove HTML tag's blocks from the target web page.

3.3 HTML Character Entity Converter

The HTML character entity converter is designed to translate HTML entities to corresponding byte sequences of Unicode's code point. The converter is able to handle both character entity references and numeric character references. There are 252 character entity references defined in HTML version 4, which act as mnemonic aliases for certain characters. Our converter maintains a mapping table between the 252 character entity references and their represented byte sequences in hexadecimal number. When a character entity reference is detected by the converter, it replaces the entity with its associated byte sequences.

For numeric character references, the converter performs a real time decoding process on it. The converter will convert the character reference from decimal (base 10)

number to byte sequences if it detects the following pattern: character ampersand (*&*), followed by character number sign (*#*), followed by one or more decimal digits (zero through nine), and lastly followed by character semicolon (*;*). For example, *A* (representing the Latin capital letter A).

Similarly, the converter will convert the character reference from hexadecimal (base 16) number to byte sequences if it detects the following pattern: character ampersand (*&*), followed by character number sign (*#*), followed by character (*x*), followed by one or more hexadecimal digits (which are zero through nine, Latin capital letter A through F, and Latin small letter a through f), and lastly followed by character semicolon (*;*). For example, *A* (again representing the Latin capital letter A).

Table 5 shows the byte sequences output by the HTML character entities converter, using an ampersand sign (*&*), a Greek small letter beta (*β*) and a Chinese character "平" as examples. These examples are carefully selected to show the different ways of conversion based on different number of byte order in UTF-8.

4. Data and Experiments

There are two sets of data used in this study. The first set is the training corpus, which contains training data used to train the language models. The second set is the validation corpus, which is a collection of web pages used as target documents in the experiments.

4.1 Training Corpus

In this paper, the authors prepared two sets of training data. The first set of training data is constructed from 565 Universal Declaration of Human Rights (UDHR) texts collected from the Office of the High Commissioner for Human Rights (OHCHR) web site and Language Observatory Project (LOP). UDHR was selected as it is the most translated document in the world, according to the Guinness Book of Records.

The OHCHR web site contained 394 translations in various languages. However, 80 of them are in Portable Document Format (PDF). As a result, only 314 languages were collected from OHCHR. The LOP contributed 18 new languages. The total size of the first set of training data is 15,241,782 bytes. Individual file size ranged from 4,012 to 55,059 bytes. From here onward this set of training data will be referred to as training corpus A.

The second set of training data, training corpus B, increases the number of languages by 33. It contains 65 (some are same language but in different encoding schemes) Biblical texts collected from the United Bible Societies (UBS). All files have similar content, but written in different languages, scripts and encodings. The total size of the second set of training data is 1,232,322 bytes. Individual file size ranged from 613 to 54,896 bytes.

Most languages have more than one training file in the training corpora. This is because the same language can be written in different scripts and encodings. For example, the Chinese language has five training files in training corpus A. The five training files by language_script_encoding are: Chinese_Simplified_EUC-CN, Chinese_Simplified_HZ, Chinese_Simplified_UTF8, Chinese_Traditional_BIG5 and Chinese_Traditional_UTF8. Likewise, a language might be covered by texts in training corpus A and B.

Table 6 shows the number of languages, scripts, encodings and user-defined fonts of the training corpora, sorted according to geographical regions. The column header (A ∪ B) represents the distinct number of languages, scripts, encodings and fonts in the corpora.

From Table 6, we can observe that the Asian region is more diversity in its written languages. Asia has the highest number of scripts (writing systems), character

Table 5 Example to show output of HTML character entities converter, based on three different types of HTML entities and each using different byte order.

Char-acter	Character Entity References	Numeric Character References	Unicode Code Point	UTF-8 Byte Order			Output in Byte Sequences
				Byte-1	Byte-2	Byte-3	
&	*&*	*&*	U+0026	0xxxxxxx			U+0026 -> 00100110 -> 0x26
β	*β*	*β*	U+03B2	110yyyxx	10xxxxxx		U+03B2 -> 1100111010110010 -> 0xCEB2
平		*平*	U+5E73	1110yyyy	10yyyyxx	10xxxxxx	U+5E73 -> 111001011011100110110011 -> 0xE5B9B3

Table 6 Number of languages, scripts, encodings and user-defined font's information in training corpus A and B, sorted according to geographical region.

	Language			Script			Encoding			Font		
Training Corpus	A	B	A∪B	A	B	A∪B	A	B	A∪B	A	B	A∪B
Africa	90	10	97	4	2	4	2	1	2	1	0	1
Asia	79	27	92	28	17	32	13	4	14	17	6	23
Caribbean	5	1	6	4	1	4	2	1	2	3	0	3
Central America	7	0	7	1	0	1	2	0	2	0	0	0
Europe	64	16	72	4	3	5	6	3	6	1	0	1
Int. Aux. Language(IAL)	3	1	4	1	1	1	3	1	3	0	0	0
Middle East	1	0	1	1	0	1	2	0	2	0	0	0
North America	20	1	21	2	1	2	2	1	2	1	0	1
Pacific Ocean	16	3	18	1	1	1	2	1	2	0	0	0
South America	47	0	47	1	0	1	2	0	2	0	0	3
Unique count			365			40			19			29

encoding schemes and user-defined fonts. Each of these factors makes language identification difficult. In the case of user-defined fonts, many of them do not comply with international standards, hence making language identification an even more challenging task.

4.2 Validation Corpus

The validation corpus is comprised of texts from web pages. The authors predefined three primary sources to search for web pages in different languages. These sources are Wikipedia, the iLoveLanguages gateway and online news/media portals. The source referred here is not necessarily a single web site. For example, a web portal might contain, or link to, many web sites. Table 7 shows more detailed information on each source.

The rule for selection is to collect one web page per web site. The authors believe that in general a web site will apply the same character encoding scheme to the web pages it hosts. Thus, it would be redundant to collect more than one page from the same web site. For each language, we collected a maximum of 20 web pages. Popular languages like Arabic (ar), Chinese (zh), and English (en) are easy to find, while less popular languages, like Fula (ff), Limburgish (li), or Sanskrit (sa) are very difficult to find.

The authors' initial target was to cover all of the 185 languages listed in ISO 369-1. However, three languages, namely Kanuri (kr), Luba-Katanga (lu) and South Ndebele (nr) could not be found from the sources, nor by using search engines on the Web. As a result, the final validation corpus used in the experiments contained 182 languages. There are 1,660 web pages in the validation corpus, occupying 76,149,358 bytes of storage. The authors did not normalize the size of collected web pages as the wide variation reflects the real situation on the Web.

Each web page in the validation corpus has its filename in

Table 7 Information of defined Web's sources for collecting web pages for the validation corpus.

Web Site	Validation corpus			
	No. of Pages	Total Size (bytes)	Min. (bytes)	Max. (bytes)
Wikipedia	171	7,511,972	601	146,131
iLoveLanguages	103	790,934	3,634	18,445
BBC	34	396,292	2,990	61,190
China Radio	13	1,891,896	9,419	222,526
Deutsche Welle	14	1,164,620	5,957	87,907
The Voice of Russia	26	1,832,797	39,198	103,251
Voice of America	22	1,791,145	9,674	87,574
Kidon Media-Link & ABYZ News Links	1,277	60,769,702	135	1,048,314
Total	1,660	76,149,358		

Table 8 Experiments' settings and language identification results.

Experiment	One	Two	Three	Four
Training corpus	A	A and B	A and B	A and B
HTML parser	Character-sequence	Character-sequence	Byte-sequence	Byte-sequence
HTML character entities converter	Disabled	Disabled	Disabled	Enabled
Correct/Total	1,241/1,660	1,444/1,660	1,494/1,660	1,561/1,660
Accuray rate	74.76%	86.99%	90.00%	94.04%

the following format: language-index-source. Language indicates the language of the web page; index represents the accumulated number of texts in each language; and source indicates the original web site of the page.

Unlike many researches listed in Table 2, our validation corpus is totally independent from the training corpus. By selecting validation sample files from different sources, the validation corpus evenly represents the language diversity on the web, while increasing its coverage on language, script, and encoding systems on the web, as wide as possible.

4.3 Experiments

Four experiments were performed. Each experiment was designed to show the baseline performance and the improvement achieved by using training data, the byte-sequence based HTML parser, and the HTML character entity converter. Table 8 provides a summary of the conditions and results of each experiment.

In the first experiment, we trained the language models using training corpus A. The HTML parser adapted in this experiment was based on character sequences, which relies on the mechanism, described in Section Character and Byte-sequence based HTML Parser and integrates the Mozilla Charset Detector algorithm to determine the character encoding scheme of a web page. If no valid character encoding scheme is detected, the parser uses its predefined default encoding, i.e., UTF-8 to encode extracted text. The HTML character entity converter was not used in this experiment.

The second experiment was designed to evaluate the improvement achieved through the extension of the training corpus. Training corpus A and B were used to train the language models. The remaining settings are the same as in the first experiment.

The third experiment applied the same settings as in the second experiment; except that the character-sequence based HTML parser was replaced by a byte-sequence based HTML parser. Besides evaluating the efficacy of the byte-sequence based HTML parser, the authors also analyzed the number of web pages with missing or invalid character encoding scheme information.

The final experiment is designed to evaluate the efficacy of the HTML character entity converter. The converter is enabled while keeping the remaining settings the same as in the previous experiment. In addition, we also examined the number of web pages that are encoded with HTML character entities in the validation corpus.

5. Results and Discussion

The summarized evaluation results of all experiments are presented in Table 8, where different settings are used. The first column showed the result of Experiment one, while the following right columns shows the results of Experiments two, three and four, respectively.

Experiment one was used as a base line for comparison. It adapted [Izumi Suzuki 2002] algorithm for language identification, but the correct language identification rate was only 74.76%. After manually inspecting the results and web pages, the authors found that 217 out of the 419 (i.e., 1660-1241) wrongly identified web pages were due to the unavailability of corresponding language models. This evidence that training corpus A alone was inadequate led to the decision to expand the training corpus. As a result, the authors collected new training data from the United Bible Societies web site in order to increase the number of language models.

5.1 Evaluation on Effectiveness of Training Corpus B

After adding the new language models of training corpus B and repeating the identification process as Experiment two, the algorithm was able to increase its accuracy of language identification from 74.76% to 86.99%, i.e., a 12.23 percent point improvement from the previous test. All of the 217 web pages wrongly identified due to unavailability of corresponding language models in Experiment one were correctly identified. However, there

Table 9 List of files in Validation Corpus that are correctly identified in Experiment one, but wrongly identified in Experiment two.

File in VC	Language Identification					
	Experiment one			Experiment two		
	Language	Script	Encoding	Language	Script	Encoding
bosnian-15-svevijesti.ba	Bosnian	Latin	Latin2	Punjabi	Gurmukhi	UTF8
indonesian-11-watchtower	Indonesian	Latin	UTF8	Aceh	Latin	Latin1
indonesian-19-pontianakpost	Indonesian	Latin	UTF8	Malay	Latin	Latin1
interlingua-01-wikipedia	Interlingua	Latin	UTF8	Spanish	Latin	UTF8
italian-13-rai.it	Italian	Latin	UTF8	Aragonese	Latin	UTF8
ndonga-01-wikipedia	Nepali	Devanagari	UTF8	Kwanyama	Latin	UTF8
persian_dari-08-afghanpaper	Persian-Dari	Arabic	UTF8	Pashto	Arabic	UTF8
portuguese-11-acorianooriental	Portuguese	Latin	Latin1	Galician	Latin	UTF8
portuguese-13-diariodoalentejo	Portuguese	Latin	Latin1	Galician	Latin	UTF8
portuguese-18-falcaodominho.pt	Portuguese	Latin	Latin1	Galician	Latin	UTF8
serbian-10-watchtower_cyrillic	Serbian	Cyrillic	UTF8	Macedonian	Cyrillic	UTF8
tibetan-03-tibettimes.net	Tibetan	Tibetan	UTF8	Dzongkha	Tibetan	UTF8
zulu-01-wikipedia	Zulu	Latin	Latin1	Ndebele	Latin	UTF8
zulu-09-zulutop.africanvoices	Zulu	Latin	Latin1	Ndebele	Latin	UTF8

were 14 web pages that had been correctly identified before, but were wrongly identified in Experiment two due to the problem of over-training. Over-training problem occurs when a language model is over-trained by a larger training data size; and/or a newly trained language model affects the accuracy of other language models. Table 9 shows the list of files that affected by this problem.

5.2 Evaluation on character and byte-sequence based HTML Parser

During the HTML parsing stage of Experiment two, the language identification process detected 1,466 web pages with valid charset information and 191 web pages with doubtful charset information. Of these 191 web pages, fourteen had "user-defined" charset and 177 were missing charset information.

The character-sequence based HTML parser used in Experiment one and two was defined to use UTF-8 encoding to encode web page without valid charset information. When investigated on web pages without valid charset information, it was found that the default UTF-8 character encoding scheme worked well on Latin-script based languages, but did not work well for 11 non-Latin-script based languages: Amharic, Arabic, Armenian, Belarusian, Bulgarian, Chinese, Greek, Hebrew, Macedonian, Russian and Ukrainian, respectively. Fifty wrong classifications occurred after applied UTF-8 to the text extracted from web pages belonging to those languages. Of those 50 pages, Africa(7), Asia(30), Europe(10), International Auxiliary Language(2) and Middle East(1). As a result, the byte-sequence based HTML parser was introduced in Experiment three.

By eliminating the steps of guessing and applying charset to text using charset returned by the charset detector, the byte-sequence based parser was able to improve the accuracy of language identification in Experiment three to 90.00%. All of the previously mentioned 50 web pages were identified correctly in Experiment three.

5.3 Evaluation on HTML Character Entities Converter

Experiment three miss-classified 166 web pages. Among those, 76 web pages are caused by HTML character entities problem. As a result, the HTML character entities converter was introduced in Experiment four.

The accuracy of language identification in Experiment four is 94.04%. The HTML character entities converter improved the algorithm by correctly identified 67 out of the 76 (88.16%) HTML entities encoded web pages. There were 9 HTML entities encoded web pages not correctly identified, where 3 of them were due to untrained legacy font and the remaining 6 were miss identified to another closely related language, like Amharic identified as Tigrinya, Assamese identified as Bengali, Persian identified as Pashto, etc.

Table 10 shows the language identification result based on writing systems. For Non-Latin-Script based languages, the algorithm achieved perfect score on Logographic and Syllabic systems based languages; its accuracy on Abjad (93.33%), and Non-Lain Alphabet (94.17%) based languages is acceptable. The worst performance come under Abugida system based languages due to many of their web pages encoded with legacy fonts. In case of Latin-Script based languages,. The algorithm achieved 95.42% accuracy rate.

Table 10 Language identification result based on writing systems.

Writing System	Language Identification Result	Percent Correct
Abjad (e.g. Arabic)	126/135	93.33%
Abugida (e.g. Indic scripts)	211/242	87.19%
Alphabet (Latin)	938/983	95.42%
Alphabet (Non-Latin)	226/240	94.17%
Logographic (Chinese)	40/40	100.00%
Mixed logographic and syllabic (Japanese)	20/20	100.00%

6. Conclusion and Future Work

The primary aim of this paper was to take into account the practical issues of language identification on non-Latin-script based web pages, especially for Asia and Africa regions; and to propose corresponding methods to overcome the issues. In this paper we have shown that the adaption of UDHR and Biblical texts as training data are simple and yet effective ways of gathering data on a large variety of languages. An initial language identification accuracy rate of 86.99% was obtained based on testing 1,660 web pages in 182 different languages. We proposed and discussed the importance of a byte-sequence based HTML parser and a HTML character entity converter for non-Latin-script web pages. The evaluation results showed that our algorithm with the two new heuristics was able to improve the accuracy of language identification from 86.99% to 94.04%.

The list of future work includes finding the optimal length for training data in order to avoid the over-training problem; improvement of language identification for closely related languages; extending the algorithm to handle multi-lingual web pages; and lastly, finding a method to effectively handle the user-defined font issue.

Acknowledgments

The authors are grateful for the sponsorship of the Japan Science Technology Agency (JST) through the Language Observatory Project (LOP) and Country Domain Governance (CDG) Project, which in turn provided the necessary resources to improve the language identification algorithm.

References

[1] Bruno Martins, Mário J. Silva. "Language identification in web pages." 2005 ACM symposium on Applied computing. Santa Fe: ACM New York, NY, USA, 2005. 764-768.
[2] Clive Souter, Gavin Churcher, Judith Hayes, John Hughes, Stephen Johnson. "Natural Language Identification usion Corpus-Based Models." Hermes - Journal of Language and Communication Studies, 1994: 183-204.
[3] Datong Chen, Hervé Bourlard, Jean-Philippe Thiran. "Text Identification in Complex Background Using SVM." IEEE Conference on Computer Vision and Pattern Recognition. 2001. 621-626.
[4] Dunning, Ted. Statistical Identification of Language. Technical Report, Computing Research Laboratory, New Mexico State University, 1994.
[5] Eugene Ludovik, Ron Zacharski, Jim Cowie. "Language Recognition for Mono- and Multi-lingual Documents." In the Proceeding of the VEXTAL Conference. Venice, 1999.
[6] Gold, E Mark. "Language identification in the limit." Information and Control 10, no. 5 (1967): 447-474.
[7] Gordon, Raymond G. Ethnologue: Languages of the World. 15th. Dallas: SIL International, 2005.
[8] Grefenstette, Greg. "Comparing Two Language Identification Schemes." The proceedings of 3rd International Conference on Statistical Analysis of Textual Data (JADT 95). Rome, 1995.
[9] Izumi Suzuki, Yoshiki Mikami, Ario Ohsato, Yoshihide Chubachi. "A language and character set determination method based on N-gram statistics." ACM Transactions on Asian Language Information Processing (TALIP), 2002: 269-278.
[10] Lena Grothe, Ernesto William De Luca, Andreas Nürnberger. "A Comparative Study on Language Identification Methods." Proceedings of the Sixth International Language Resources and Evaluation (LREC'08). Marrakech, 2008. 980-985.
[11] Lin, Dekang. "An Informaiton-Theoretic Definition of Similarity." Proceedings of the Fifteenth International Conference on Machine Learning. 1998. 296-304.
[12] Matthias Richter, Uwe Quasthoff, Erla Hallsteinsdóttir, Christian Biemann. "Exploiting the Leipzig Corpora Collection." IS-LTC 2006. Ljubljana, 2006.
[13] Michael John Martino, Robert Charles Paulsen Jr. Natural language determination using partial words. United States Patent US 6,216,102 B1. April 10, 2001.
[14] ÖLVECKÝ, Tomáš. "N-Gram Based Statistics Aimed at Language Identification." IIT.SRC. Bratislava, 2005. 1-17.
[15] Penelope Sibun, A. Lawrence Spitz. "Language Identification: Examining the Issues." In Proceedings of the 5th Symposium on Document Analysis and Information Retrieval. Las Vegas, 1996. 125-135.

[16] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, Robert L. Mercer. "Class-Based n-gram Models of Natural Language." Computational Linguistics 18 (1992): 467-479.
[17] Poutsma, Arjen. "Applying Monte Carlo Techniques to Language Identification." Computational Linguistics in the Netherlands. Rodopi, 2001. 179-189.
[18] Prager, John M. "Linguini: Language Identification for Multilingual Documents." Proceedings of the 32nd Hawaii International Conference on System Sciences. IEEE Computer Society, 1999. 2035.
[19] William B. Cavnar, John M. Trenkle. "N-Gram-Based Text Categorization." SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. 1994. 161-175.

First Author Yew Choong Chew received his Master"s Degree of Management and Information Systems Engineering from Nagaoka University of Technology in 2007. He is currently a Ph.D candidate and researching in statistical natural language processing for written text.

Second Author Yoshiki Mikami received his B.Eng (Mathematical Engineering) from Tokyo University in 1975. Joined ministry of International Trade and Industry (MITI) in 1975. Worked in JETRO Singapore Office 1994-1997. Since July 1997, he is a Professor at Nagaoka University of Technology. His research interests include natural language processing and Internet governance.

Third Author Robin L. Nagano is currently an English language teacher at the University of Miskole, Hungary. She holds Master's degrees in Applied Linguistics (Macquarie) and Advanced Japanese Studies (Sheffiled). Her areas of interest include academic writing and language use in engineering.