Regular Paper

Identifying the Coding System and Language of On-line Documents Using Statistical Language Models

Genichiro Kikui[†]

This paper proposes a new algorithm that simultaneously identifies the coding system and language of a code string retrieved from the Internet, especially the World-Wide Web. The algorithm uses statistical language models to select the correctly decoded string as well as to determine the language. The proposed algorithm covers 43 combinations of 15 languages and 11 coding systems used in Eastern Asia and Western Europe. Experimental results show that the level of accuracy of our algorithm is over 95% for 929 on-line documents.

1. Introduction

There is a great demand for intelligent document processing systems (e.g., text search and machine translation) to deal with the enormous number of documents accessible through the Internet.

Identifying the coding system and the language of on-line documents is fundamental for processing documents on the Internet. In fact, these documents are written in a variety of languages and encoded with various character coding systems. Therefore, systems should identify the coding system of individual documents for correct decoding. In addition systems should identify the language so that appropriate language dependent resources such as dictionaries and the grammar can be selected[☆].

Although character coding systems and languages are closely related, identification of these two have been separate research areas.

It is generally considered impossible to identify the coding system of an arbitrary code string among unrestricted candidates of a coding system. This is because some coding systems have almost the same code ranges (i.e., different coding systems map their own characters into the same character codes). Therefore, existing algorithms have several restrictions. Some algorithms limit the set of coding systems they can handle. For example, programs for Japanese coding systems (e.g., by Lunde[5]) choose one coding system among three coding systems widely used in Japan[☆☆]. Other algorithms (e.g., the identifier for Mule[4]) sacrifice selecting a unique coding system but output several candidates. These limitations make it difficult to include identification modules in document processing systems.

Automatic language identification has been discussed in the field of document processing. Previous approaches are classified into two types: statistic-based and heuristics-based. The former uses the n-gram of characters[6], diacritics and special characters[7], and using the word unigram with heuristics[8]. Among these methods, the n-gram model[6] shows the best accuracy over 95%. The latter employs grammatical words peculiar to each language[9] and achieved comparable accuracy.

They, however, presuppose that the input text is correctly decoded, which does not hold true for documents on the Internet as described above.

This paper presents a novel algorithm that simultaneously identifies the coding system and the language. The underlying assumption is that a correctly decoded string must be a document in natural language. Based on this assumption, choosing the correctly decoded string is regarded as selecting the string that belongs to a specific language with the highest likelihood. In our algorithm, statistic-based language models calculate the likelihood that a string is from a particular language.

The next section describes the problem. Section 3 presents our algorithm which handles lan-

[†] NTT Information and Communication Systems Laboratories

[☆] A fundamental solution is to develop international standards for an internationalized coding system and language representation. In fact, there is active discussion on the international coding standards[1)~3)], and the language representation on the WWW[3)]. However, it will require several years before most of the documents are encoded into a unique well-defined coding system.

[☆☆] UJIS, SJIS, JIS.

guages and the coding system used in West-European and East-Asian countries. Sections 4 and 5 describe an example and the experimental results, respectively.

2. Problem

2.1 Identifying Character Coding Systems on the Internet

For historical reasons, documents on the current WWW are encoded with various coding systems. This section briefly explains the current coding systems on the Internet and clarifies the problem.

2.1.1 Terminology

A *character coding system* defines how a text is mapped to a sequence of numbers (called *character codes*). A character coding system presupposes one or multiple *character sets.*

A *character set* is a standardized collection of characters in a particular language (or a set of languages). The most fundamental character set is the ASCII character set[☆]. Some countries, or local communities, define their own character sets that cover characters in their languages. For example, GB 2312 and JIS X 0201 is for Chinese and for Japanese respectively. ISO 88591 covers all West-European languages (e.g., French, German, Dutch, etc.).

Each element in a character set is assigned an identification code (hereafter *character identification code* or *CI code*) unique in the character set. Therefore, if only one character set is presupposed, CI codes have enough information to represent individual characters. In fact, on a closed network, a text is encoded simply by substituting individual characters with their CI codes.

If multiple character sets are presupposed, the same CI code from different character sets should be discriminated. *ISO 2022* is an international standard that defines a flexible framework for handling multiple character sets. It assigns a unique *escape sequence* to every registered character set. The escape sequence marks the corresponding character set in an encoded string. For example, the escape sequence for JIS X 0201 is inserted at the point where a Japanese character begins, as shown in **Fig. 1**.

ISO 2022 defines another way to handle multiple character sets which is particularly useful when (a part of) the text consists of characters from two character sets. Since every CI code

☆ Technically, ISO 646 IRV.

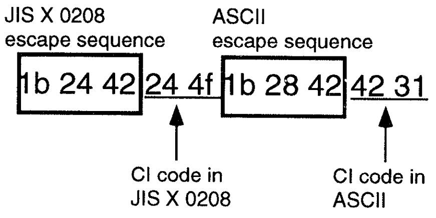


Fig. 1 An example of code string with escape sequences.

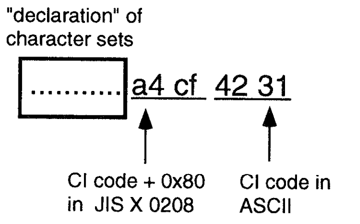


Fig. 2 An example of a code string using the eighth bit.

of a registered character set consists of one or two bytes whose eighth bit is 0, the eighth bit can be used to encode information whether if the code is from one of the two character sets. According to the ISO 2022 standard, the two character sets are 'declared' at the beginning of the code string by using escape sequences, then the eighth bit is employed to discriminate these two character sets. This is exemplified in **Fig. 2**.

2.1.2 Coding Systems on the Internet

The scope of this paper is restricted to coding systems used in North American, West European, and East Asian countries, although this algorithm is applicable to other coding systems.

Most coding systems on the Internet are local coding systems in the sense that they are designed to efficiently handle limited character sets sufficient for local communication. All these local coding systems include the ASCII character set for codes smaller than 0x80 as their default. In other words, if an entire text is written with ASCII characters, all the local coding systems encode it into the same byte string. Therefore the difference lies in how to encode non-ASCII characters, as classified into the following two types.

Type 1: Coding systems that employ escape sequences defined in ISO 2022 (e.g., ISO-2022-kr, ISO-2022-jp).

Type 2: The remaining coding systems (e.g., EUC-GB (Chinese), ISO-8859-1, SJIS, BIG5).

Type 1 coding systems are simplified versions of ISO 2022. As compared with the full ISO 2022 specifications, they handle a limited number of character sets and/or omit control codes such as SI/SO. Since these coding systems explicitly indicate non-ASCII characters by inserting escape sequences, it is easy to decode the encoded string.

Type 2 coding systems mark non-ASCII characters by setting the eighth bits. These are further classified into simplified versions of ISO 2022 and the rest of the coding systems. The former coding systems (e.g., EUC-JIS and EUC-GB) use eighth bits in the same way as ISO 2022, as exemplified in Fig. 2, but they lack escape sequences☆. The rest of the coding systems do not follow ISO 2022 nor do they employ registered character sets (e.g., BIG5 for traditional Chinese). Since the code ranges of these coding systems overlap with each other, it is hard to identify an arbitrary code string as one of these coding systems. It should be noted that ISO 2022 escape sequences do not appear in any Type 2 coding systems, even if they define their own character sets and coding scheme.

2.1.3 Ambiguities in Identifying Coding System

The above observations are summarized into the following statements.

(1) If every byte in a code string is smaller than 0x80, i.e., the eighth bit is zero, and the code string does not contain any escape sequences nor SI/SO, then it is an ASCII text.

(2) If a code string contains ISO 2022 escape sequences, then it is encoded with (a simplified version of) ISO 2022 (Type 1). Therefore, it is uniquely decodable.

(3) If a code string contains bytes greater than 0x7F and does not contain escape sequences, then it is encoded with one of the coding systems belonging to Type 2.

In the last case, we should choose a unique coding system from the Type 2 coding systems. This problem is partly solved by eliminating some of coding system candidates whose code ranges do not cover the given code string. However, the number of candidates cannot always be decreased to one.

The remaining ambiguity is resolved by our algorithm described in Section 3.

☆ In other words, the two character sets are presupposed.

2.2 Identifying the Language on the Internet

Documents on the Internet are written in many languages. The problem is that currently there are no widely accepted protocols for communicating the language of each document.

If we restrict ourselves to HTML documents, then explicit *language tagging*, which represents the language of each portion of a text, will be introduced in a future version of HTML specifications. This, however, is not currently used. Moreover, there are many non-HTML documents on the WWW.

If the character set employed by the text is known, it may provide a good clue for identifying the language because some character sets strongly suggest which language(s) was used. For example, if a document consists of characters in the JIS character set, it must be written in Japanese. However, this is not always the case for the following two reasons. First, the character set of a text is sometimes ambiguous due to the decoding problem described above. Second, some character sets are designed to cover multiple languages (e.g., ISO-8859-1 for several Western-European languages). Sometimes a character set is used in a language that is not the primary candidate suggested by the character set. For example, a document containing only US-ASCII characters, which suggests the document is in English, may be a German document in ASCII-format where umlaut letters are converted into ASCII characters (e.g., "ö" is written as "oe").

The remaining possibility is to identify the language by analyzing its content. As compared with similar problems tackled in previous research (e.g., by Cavnar, et al.[6]), our problem is even more difficult because the text is encoded with an unknown coding system.

Sibun and Spitz[10] proposed a method of determining the language of a text image. The problem tackled by them is similar to ours, in the sense that the input is not a unique character string but a string that potentially corresponds to several different character strings. Their method, however, can not directly applied to our problem.

3. Algorithm

Our algorithm receives a byte (=code) string and outputs the decoded character string as well as its language(s).

The underlying assumption is that the given

Table 1 The scope of coding systems.

Category	Coding Systems
7bit Coding	ISO 646 USA (ASCII), ISO-2022-jp (JIS Code), ISO-2022-kr (ISO-2022-int)
8bit Coding (Type 2)	ISO-8859-1, EUC-KS, EUC-JIS (UJIS), BIG5 (Traditional Chinese), EUC-GB (Simplified Chinese), Shift JIS (MS Kanji code)
Special	Entity Reference for the ISO-8859 characters defined in HTML (or SGML) specifications. (e.g., ö is represented as "&Ouml;")

code sequence is encoded from a document containing natural language texts (or text fragments). This means that the correctly decoded string is more likely to be from some languages than incorrectly decoded strings.

In our algorithm, a statistic-based language identifier calculates how likely a decoded string is from a particular language. The result indicates the level of "correctness" of decoding as well as the language of the string.

The following subsections first present the scope of the algorithm then explain in detail the algorithm.

3.1 The Scope of our Algorithm

The current version of our algorithm can handle 11 coding systems and 15 languages[☆], as shown in **Table 1** and **Table 2** respectively. Table 2 also gives the coding system(s) that can be used to encode each language. For example, Korean text is encoded with EUC-KS or ISO-2022-kr. We allow Chinese document to be encoded with Japanese coding systems because there are several such documents on the Internet[☆☆].

3.2 Outline of the Algorithm

The algorithm is made up of the two major steps.

It first extracts East-Asian language parts (i.e., sub-strings consisting of East-Asian characters) in the input document, if they exist, and identifies the language. The algorithm then identifies the remaining part as one of the West-European languages.

In both steps, a process responsible for identifying the language is employed. This process called the "language identifier" is described in Section 3.4.

☆ We regard traditional and simplified Chinese as two different languages.

☆☆ For example, "fj.soc.chinese".

Table 2 The scope of languages.

Category	Language	Coding Systems
European	Danish, German, English, Spanish, Finnish, French, Italian, Dutch, Norwegian, Portuguese, Swedish	ISO 646, ISO 8859-1, Entity-Reference
East Asian	Chinese (traditional)	Big5
	Chinese (simplified)	EUC-GB ISO-2022-jp EUC-JIS, Shift-JIS
	Korean	EUC-KS ISO-2022-kr
	Japanese	EUC-JIS, ISO-2022-jp, Shift-JIS

3.3 Extracting East-Asian Parts

If the given code string contains escape sequences defined in ISO-2022, East-Asian character strings are easily extracted. This is because East-Asian characters are explicitly marked by escape sequences in the string. In this case, the language identifier for East-Asian languages is applied to the extracted character strings.

If the given code string does not contain such escape sequences (i.e., it must be encoded with Type 2 coding systems), the Eastern-Asian part is identified by the procedure, based on language identification, exemplified in **Fig. 3**.

The procedure first decodes the given code string with every possible decoder for East-Asian character sets (e.g., BIG5, EUC-KS). Then, the East-Asian character parts are sent to the language identifier (for East-Asian languages). The language identifier outputs the language and the likelihood score of the given character string. Finally, the East-Asian part with the highest likelihood score is taken.

Note that if the language identifier fails for every East-Asian part, then the input code string is judged to have no East-Asian language parts (i.e., the code is judged to be ISO-8859-1 by default).

3.4 Language Identifier

Our algorithm employs two language identifiers: one is for East-Asian languages, and the other is for West-European languages. Both identifiers consist of the following three steps.

(1) Selecting possible languages for the given coding system
The coding system (or the character set(s)) of a text is loosely related to the

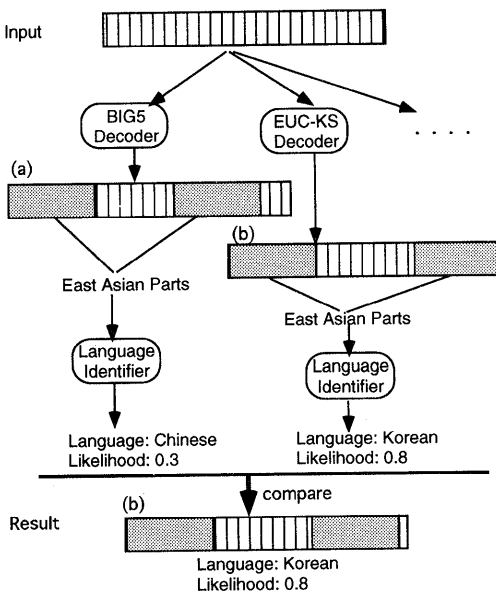


Fig. 3 An example of extracting East-Asian part.

language of the text. For example, a document encoded with US-ASCII is not written in Korean. We made rules to map a coding system to possible languages according to Table 2.

(2) Calculating the 'likelihood' of the decoded string for each language.
For each language, this step calculates how likely the decoded string is to be from that language by comparing the string with the statistic model of the language.

(3) Selecting the language with the highest likelihood
This step compares likelihood scores. If the highest score exceeds predetermined threshold, then the system returns the corresponding language as well as the score. Otherwise, it returns 'ELSE' (i.e., fail).

The second step is the most important. Our system uses a unigram model for both Western-European languages and East-Asian languages, but the models for Western-European languages and the models for the East-Asian languages have different unigram units.

3.4.1 Likelihood Score for Western-European Languages

In order to distinguish Western-European languages, we applied a method proposed by Cavnar, et al.[6]. We assign a class name for each word. The class name of a word longer than n characters is the concatenation of "X-" and the last n characters of the word. If the word is not longer than n characters, the class name is the word itself. For example, if $n = 4$, then class names of "beautiful" and "the" are *X-iful* and *the* respectively.

Let $TEXT$ be the set of words in a text, then the likelihood of $TEXT$ with regard to language l is given as the following $S(TEXT, l)$

$$S(TEXT, l) = \frac{1}{\mid TEXT \mid} \sum_{w \in TEXT} logP(C_w, l)$$

where $P(C_w, l)$ is the unigram probability of C_w in the language l, and C_w is the class name of the word w. $\mid TEXT \mid$ is the total number of words in $TEXT$.

$P(C_w, l)$ is estimated from text corpora in language l.

3.4.2 Likelihood Score for East-Asian Languages

As compared to Western-European languages, East-Asian languages have the following properties:

(1) A large number of characters
East-Asian languages use over 3,000 ideographs or combined characters. A character is normally encoded with two (or more) bytes.

(2) No Explicit Word Boundaries
In East-Asian languages, there are no explicit word delimiters (corresponding to spaces in English) in a sentence. Word-based language models cannot be used.

For East-Asian languages, we use a character unigram, instead of a word unigram, to model a language. Formally,

$$S(TEXT, l) = \frac{1}{\mid TEXT \mid} \sum_{char \in TEXT} P(char, l)$$

where $P(char, l)$ is the unigram probability of *char* in language l. In this case, $\mid TEXT \mid$ corresponds to the number of characters in TEXT.

3.4.3 Threshold Values

We determined threshold values to be slightly lower☆ than the minimum of likelihood scores of all the training documents belonging to the language. These values reject extremely low scores while still accepting all the training data.

☆ e.g., the largest integer.

```
-----------------------------
GB   :咐賂急待の数恕
KS   :잣몄선奸귀鑒匣
JIS  :言語識別の方法
BIG5:蜕賄摹玠及劳哃
-----------------------------
```

Fig. 4 Decoded strings.

4. Example

Suppose the following code sequence is given to the algorithm.

```
b8c0 b8ec bcb1 cacc a4ce cafd cba1 0a49
6465 6e74 6966 7969 6e67 2074 6865 204c
616e 6775 6167 650a
```

The string is first divided into Asian and European parts. Since there is no escape sequence, which begins with "1b", the procedure in Section 3.3 is applied.

The division procedure first tries to extract Eastern-Asian characters from the given string. Since it has no escape sequences, the procedure exemplified in Fig. 3 is applied.

Decoders for four coding systems, namely EUC-GB, EUC-KS, EUC-JIS, BIG5, are successfully applied. The resulting character string of each decoder has East-Asian characters in the first 14 bytes of the input string. This means that these four coding systems are potential candidates and the corresponding decoders extract the same East-Asian character part.

Resulting East-Asian strings are shown in **Fig. 4**.

Next, the statistic-based language identification is applied to each decoded string,

Table 3 shows the score (= log probability) of each character with regard to the language that produced the highest likelihood (i.e., average score). For example, the second column shows the score of each character with regard to Chinese (zho) when the input code string is decoded with EUC-GB. This implies that Chinese (zho) is the most likely language if we presuppose the original string is encoded with EUC-GB.

The bottom row shows that the highest average score is obtained when the input is decoded with EUC-JIS and the language is Japanese (jpn). Since this score exceeds the threshold (−10), the Eastern-Asian part is confirmed to be a Japanese string encoded with EUC-JIS.

The remaining part is decoded into "Identifying the Language" and sent to the European language identifier. **Table 4** gives scores of tokens with regard to three languages.

Table 3 Character likelihood scores of Asian part.

Char.	SCORES (log prob.) [lang]			
(JIS)	GB [zho]	KS [kor]	JIS [jpn]	BIG5 [zho]
1(言)	−11.05	−15.51	−7.25	−12.62
2(語)	−11.15	−15.51	−7.86	−10.86
3(識)	−8.47	−7.69	−8.68	−12.42
4(別)	−11.45	−15.51	−7.71	−15.51
5(の)	−15.51	−15.51	−3.40	−7.50
...	..	..	..	..
Ave.	−11.06	−14.39	−6.9	−12.64

zho=Chinese, kor=Korean, jpn=Japanese

Table 4 Character likelihood scores of European part.

class of	SCORES (log prob.)			
token	eng	deu	ita	..
X-ying	−7.45	−10.80	−10.80	..
the	−3.11	−9.40	−7.21	..
X-uage	−7.20	−10.80	−10.80	..
Ave.	−5.9	−10.3	−9.60	..

eng=English, deu=German, ita=Italian

In this table, English (eng) is the most plausible language for the European part with a sufficient score.

The final result is easily obtained by combining the results of the Asian and the European parts.

5. Evaluation and Discussion

5.1 Data

For evaluation purposes, we collected a set of documents on the WWW. Each document was manually assigned a correct language name. If a document contained more than one Western-European language, the language that covered more than 80% of the document was chosen as the correct language. Documents without a correct language (i.e., documents without a unique main language) were discarded. The same process was applied to East-Asian parts of documents.

The remaining documents were divided into 640 training documents and 876 test documents, where one web document is about 10 Kbytes on average. In addition to the training documents obtained from the WWW, manuals and the Internet news articles in East-Asian languages, about 4 Mbytes in total, were included in the training corpus. For testing purpose, we added 53 special documents written with Latin characters. They were either in languages outside our scope (e.g., Catalan, Turkish) or language independent documents (e.g.,

Table 6 Confusion matrix for Western-European languages.

	dan	deu	eng	esl	fin	fra	ita	nld	nor	por	swe	ELSE
dan	8		1									
deu		47										1
eng		1	96			1					1	4
esl				25								1
fin					29							
fra						48						1
ita							29					1
nld								49				1
nor	1	1	1						29			1
por										8	1	
swe											26	
ELSE			1								1	42

dan=Danish, deu=German, eng=English, esl=Spanish, fin=Finnish, fra=French, ita=Italian, nld=Dutch, nor=Norwegian, por= Portuguese, swe=Swedish

Table 5 Summary of decoding errors.

Correct	System	Count
EUC-JIS	EUC-GB	4
ISO-8859-1	EUC-GB	1
ISO-8859-1	EUC-KS	5
ISO-8859-1	BIG5	1

lists of person names or acronyms). These special documents were labelled with "ELSE" as their language name. Note that the total number of test documents was 929 (876 + 53).

It is important to note that 830 documents, including 430 East-Asian documents, out of 929 test documents are in Type 2 coding systems, which have been considered to be difficult (see Sections 2.1 and 3.1).

5.2 Identifying Coding Systems

The proposed algorithm correctly identified almost all the test documents. The error rate was 0.8%. The errors are summarized in **Table 5**. For example, the first row shows that four documents encoded with EUC-JIS were identified as EUC-GB by the system. These errors were due to incorrect identification of East-Asian languages, as explained in Section 5.4.

5.3 Identifying Western-European Languages

Table 6 shows the confusion matrix for the Western-European language results. The row correspond to the output from the system, and the columns correspond to the correct answers. The value of n[☆] was set to 4, which gave the fewest number of errors for the training set.

The error rate was 4.8%. 90% (= 18 documents) of erroneous documents did not contain sentences (i.e., verbs) but were filled with proper names, acronyms and/or network addresses. Since these words were independent of languages, n-gram scores were not effective. The remaining documents contained natural language sentences but the system confused with the similar language. This remaining error corresponds to the limitation of the algorithm.

☆ The number of suffix characters for determining the word class. See Section 3.4.1.

Table 7 Confusion matrix for East-Asian languages.

	jpn	kor	zho_s	zho_t	ELSE
jpn	145				
kor		171			5
zho_s	4		56		1
zho_t				90	1
ELSE			1		

jpn=Japanese, kor=Korean, zho_s=simplified Chinese, zho_t=traditional Chinese

5.4 Identifying East-Asian Languages

Table 7 shows the confusion matrix for the East-Asian language results. The matrix shows that the system also performs fairly well for East-Asian languages.

The error rate was 4.6%. Most errors occurred when the document included only a few East-Asian characters. For example, four strings in Japanese were confused with those in simplified Chinese. In fact, all the errors were due to proper names consisting of four or five characters embedded in English documents. As far as the above experiment is concerned, the algorithm correctly identifies East-Asian languages if a document contain more than seven East-Asian characters (14 bytes).

Some documents were identified as East-Asian languages although they were written in Western-European languages. These errors came from accented characters (codes greater than 0x7F) arranged to form graphics such as horizontal lines.

Table 8 Confusion matrix for West-European languages using the extended Cavnar's algorithm.

	dan	deu	eng	esl	fin	fra	ita	nld	nor	por	swe	ELSE
dan	7											
deu		45										
eng	1	2	99		1	2					2	1
esl				24								2
fin					28							
fra						47						
ita							28					3
nld		1						48				1
nor	1								28		1	2
por				1						8		
swe								1			25	3
ELSE		1					1		1		1	40

5.5 Comparison with Other Methods

Although previous language identification algorithms were limited to Western-European languages (i.e., languages that have explicit word delimiters) and assumed input string to be correctly decoded, it is worth while comparing them with our algorithm.

5.5.1 Cavnar's N-gram-based Identification

Although both Cavnar's method and ours are based on n-grams, there are two differences.

One difference is in the n-gram unit. Cavnar uses 1 through 5 grams of characters of all the words. Blanks are added to beginning and ending of the word to make complete n-grams. For example, tri-grams of "a boy" are shown as follows:

"__a", "_a_", "a__", "__b", "_bo", "boy", "oy_", "y__".

Since Cavnar's n-gram unit relies on words as he basic unit, it is not simple to apply it to languages without explicit word separators such as Japanese or Chinese. One solution might be assuming a (two-byte) character as a word. In this case, the n-gram unit is as same as ours.

If we restrict ourselves to Western-European languages, our n-gram unit is a subset of Cavnar's because ours uses only the rightmost four characters (i.e., one 4-gram) of each word. Therefore our n-grams are more efficiently computed than Cavnar's.

The other difference lies in classification algorithms. Cavnar's algorithm is summarized as follows (see Cavnar, et al.[6] in detail):

(1) Classifying training data by language.
(2) For each classified data, generating all the n-grams, counting the frequency of every different n-gram and sorting them by reverse order of frequency. The resulting sorted list is called the category profile.
(3) Creates the profile for the input document (i.e., a sorted list of n-grams of the input document).
(4) Calculate the *out-of-place distance* between the input's profile and each category. The out-of-place distance defines a difference measure between two ordered lists.
(5) Picks the language with the smallest distance.

First, the algorithm is computationally inefficient as compared to ours because the former involves sorting all the n-grams of the input document, while the latter consists of only one summing up operation for each word.

Second, the original algorithm does not have the "ELSE" category. Therefore, we extended the algorithm as follows:

(1) Applying the original algorithm.
(2) If the out-of-place value of the selected language exceeds the predefined maximum distance for that language, then returning "ELSE", otherwise returning the original result. The maximum distance is determined to the smallest value that does not reject any training documents.

We conducted language identification experiment with the above extended Cavnar's algorithm by using the same training and test documents as shown above. The resulting confusion matrix is shown in **Table 8**. The error rate is 6.4% which is slightly worse than ours.

In summary, Cavnar's identification method is applicable to Western-European part of our task but it should be extended as proposed above. The accuracy is slightly lower than, but comparable to, ours but Cavnar's method is less efficient.

5.5.2 Rule-based Language Identification

Language identification with manually selected grammatical words[9] achieved over 95% accuracy without training corpus if we can prepare effective (i.e., discriminating) words.

The problem is that the method requires experts with multi-linguistic knowledge who can choose effective grammatical words. Statistic-based methods including Cavnar's and ours need only a set of texts with correct language names, which can be prepared by normal people familiar with one language.

6. Conclusion

This paper proposed an algorithm for simultaneously identifying the coding system and the language of a given code-string. It handles four East-Asian languages as well as eleven Western-European languages with a high level of accuracy. The algorithm uses statistic language models to select the correctly decoded string as well as to determine the language. Since the algorithm uses statistic language models, it is robust and can be easily extended to other languages.

The algorithm is implemented in a cross-lingual search engine for WWW pages which has a language index (i.e., WWW pages are indexed by language).

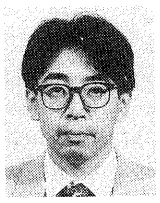
We intend to elaborate on the algorithm so that it can identify languages in multi-lingual text because many documents on the WWW are multi-lingual.

Acknowledgments The author wishes to thank Yoshihiko Hayashi and Seiji Suzaki for their comments on an early version of the paper.

References

1) Yergeau, F., Nicol, G., Adams, G. and Duerst, M.: Internationalization of the Hypertext Markup Language, Internet Draft, draft-ietf-html-i18n-02.txt (1995).
2) Nicol, G.T.: The Mutiliungual World Wide Web, http://fuzine.mt.cs.cmu.edu/mlm/lycos-home.html.
3) Unicode Inc.: The Unicode Standard, http://www.stonehand.com/unicode/standard.html.
4) Nishikimi, M., Takahashi, N., Handa, K. and Tomura, S.: Mule: Multilingual text processing system (in Japanese), SIG-SLUD-9501-7, JSAI (1995).
5) Lunde, K.: Understanding Japanese Information Processing, O'Reilly and Associates (1993). Japanese Translation: Nihongo-Joho-Shori (1995).
6) Cavnar, W.B. and Trenkle, J.M.: N-gram Based Text Categorization, *Proc. Third Annual Symposium on Document Analysis and Information Retrieval*, pp.161–169 (1994).
7) Beesley, K.R.: Language Identifier: A Computer Program for Automatic Natural Language Identification of On-line Text, *Language at Crossroads: Proc. 29th Annual Conference of the American Translators Association*, pp.47–54 (1988).
8) Henrich, P.: Language Identification for the Automatic Grapheme-to-Phoneme Conversion of Foreign Words in a German Text-to-Speech System, *Proc. Eurospeech 1989*, pp.220–223 (1989).
9) Giguet, E.: Multilingual Sentence Categorization according to Language, *Proc. EACL95 SIGDAT Workshop* (1995).
10) Sibun, P. and Spitz, A.L.: Language Determination: Natural Language Processing from Scanned Document Images, *Proc. ANLP 94*, pp.15–21, (1994).

(Received March 7, 1997)
(Accepted September 10, 1997)



Genichiro Kikui was born in 1961. He received his M.E. degree from Kyoto University in 1986. He has been working in NTT Corp. since 1986 and now is a senior research engineer of the Information and Communication Systems Laboratories of NTT. From 1990 to 1994 he had been a visiting researcher of ATR Interpreting Telephony Research Labs. He had also been a visiting researcher of DFKI (German Research Center for Artificial Intelligence) in 1993. He has been engaging in the research areas of natural language processing including machine translation, text revision, and recently, cross-language information retrieval on the WWW. He is a member of IPSJ.