Slovak University of Technology in Bratislava
Faculty of Mechanical Engineering
15th Conference on Applied Mathematics **APLIMAT 2016**
Proceedings

IMPROVEMENT OF CHARACTER SET DETECTOR CHARDET

BOHDAL Róbert (SK)

Abstract. There are many encoding schemes which represent characters in text files. If the program displaying these texts cannot determine the right encoding, the text may become unreadable. Thanks to the widely spread universal charset detector from Netscape, it is possible to display text correctly in any software on any device. Language models for the automatic character set detection have been created only for a small group of languages. Our aim was to create language models for more countries so that the probability of successful determination of the encoding increased. The most problematic was the increase in accuracy of detecting the character set for languages using ISO-8859-1 encoding. The original algorithm was not sufficiently precise, and we have therefore designed a different procedure.

Keywords: character set, character encoding, detector, language identification, n-gram

Mathematics subject classification: Primary 68T50; Secondary 65C50.

1 Introduction

Thanks to the Internet and electronic media, people can access a big amount of documents in the electronic version. Those documents can be read not only on monitors, tablets or smartphones, but also on more and more popular e-readers. However, many old documents have not been converted to the currently used character set UTF-8, and they remain in older encodings such as ISO-8859 and Windows codepage. If the application does not recognise correctly in which character set the document was created, many characters will not be displayed properly, and some of them will not be displayed at all. The user then has to try several different character sets in the settings of the application in order to display all the characters correctly.

One of the first applications which used automatic detection of the character set in documents was a web browser called Netscape. In fact, it was Netscape and later Mozilla, which developed a fast and relatively reliable algorithm for automatic detection of the character set. The algorithm was gradually developed in various programming languages for different applications, and it is used also in the wide-known application called Calibre for e-book management.

While opening many documents written in Slovak and Czech language in the Calibre, we have encountered the inability of the program to correctly detect the character set, which led us to

the decision of improving the existing detector. Our aim was to modify the existing *chardet* detector and create a publicly available tool for generating language models to improve the automatic detection of character sets. Moreover, this detector is a standard part of packages for Python language.

2 Background and Related Work

Only few authors discuss the methods of automatic detection of the character set, in which the document was created. Besides the method by Li and Momoi [8], we can find a method *chared* [11], which uses the comparison of trigram occurrences between the tested set of character and the created language model. The disadvantage of classical methods based on the comparison of the probabilities of occurrences of individual n-grams[1] between the tested sample and the language models created before is the inability to choose the character set correctly in multilingual documents. Suzuki [14] tried to solve this problem using Shift-Codon-Matching process. Kim and Park [7] described a method using a hybrid algorithm, which at first uses the Naive Bayes method for the sequence of characters, which chooses the character set and consequently uses SVM to increase the accuracy of the detection of the correct language. Very good results of the detection of the character set using SVM method were obtained in [13].

We can find quite many articles discussing the methods of choosing the language in which the text was written. Grothe [6] includes a good comparison study of methods identifying the language of the analysed text. According to the authors, the majority of methods uses three approaches for creating the language model. The first of them creates a language model based on short words [5, 12]. The second approach uses the most common words [10], and the third is based on the probability of occurrence of n-grams in the text [5, 2, 12]. The created language model is then used for consecutive classification of the text. Classification methods use ad hoc ranking [2], Markov chains together with the Naive Bayes method [4], SVM [15, 9] and other. We can find a good overview study together with the comparisons of the success of various methods of automatic detection of the language for web documents in *html* format in [3]. The relation between the amount of test data and the length of the test data on the success of language detection was researched in [1].

3 Character set and encoding detection

By character set we understand a table which chooses a mapping between the individual characters and individual bytes (codes of characters in strings). Each character[2] is uniquely assigned a number code consisting one or more bytes. The character sets can be sometimes the same for some languages if we take into account only letters, i.e the letters have the same place in various character set tables. An example of this is Romanian language which uses the character sets ISO 8859-2 and Windows-1250. In such case the detector has to search for the characters which occur in one character set and not in the other. Fortunately, character sets for encoding Windows codepage include characters which are not present in ISO 8859 (e.g. symbol €, see Tab. 1).

A worse situation happens with texts written in languages which use character sets whose characters are at different places in different character set tables (see Tab. 2). If such text does

[1] We mean by n-gram a sequence of n characters of the text.

[2] By character we understand not only letters or numerals, but also any symbol which occurs in the text of the file, including control characters.

not include letters typical only for a particular character set, the algorithm has to find the best language model.

A general way of identifying the language can be split into two steps. At first, a source language model is created[3] from a sufficient amount of test data (corpus) for each language. Next, a target language model is created from the test data, for which we want to find the language. Consequently, the detector compares the target language model with the individual language models and based on some comparison criterion identifies the language of the document. We usually create more language models for each language, for each character set one. If the characters representing the letters are at the same places in different character sets, a new language model is not created, and the detector decides according to the other characters.

Hex code	80	82	84	89	8B	91	92	93	94	95	96	9B	A9	AB	AE	B7	BB
CP-1250	€	‚	„	‰	‹	‘	’	“	”	•	–	›	©	«	®	·	»

Tab. 1. Selected characters from Windows-1250 codepage

Hex code	A1	A3	A5	A6	A9	AA	AB	AC	AE	AF	B1	B3	B5	B6	B9	BA	BB	BC	BE	BF
CP-1250	ˇ	Ł	Ą	¦	©	Ş	«	¬	®	Ż	±	ł	μ	¶	ą	ş	»	Ľ	ľ	ż
ISO 8859-2	Ą	Ł	Ľ	Ś	Š	Ş	Ť	Ź	Ž	Ż	ą	ł	ľ	ś	š	ş	ť	ź	ž	ż

Tab. 2. Comparing the ISO-8859-2 and Windows-1250 character sets

4 The detection of character set by *chardet*

The universal character set detector *chardet* created by Mozilla uses three different approaches, which support each other (see Fig. 1).

At first, a specific sequence of bytes is found, which classifies the document. For example, each document encoded in the encoding of UTF group usually begins with a specific sequence of bytes marked as BOM. Next, if the input document includes character 'ESC' or '~{', the detector knows the input text is probably encoded in one of the escape character sets encoding scheme (HZ-GB-2312, ISO-2022-JP/CN/KR) and all characters between '~{' and '~}' are regarded as national characters. The detector uses the state machine for this input for each relevant character set. Immediately when the state is changed to 'itsMe' for a currently processes character set, the detection finishes. If the input text does not fulfill the previous case and does not contain a character with a code higher than 0x7F, it is encoded in ASCII.

The second and the third approach use statistical distribution of occurrence of characters in the text for various character sets. The *chardet* uses a slightly different method of detection for single byte and multibyte character sets[4]. It uses the statistical distribution of unigrams for multibyte character sets, but it uses bigrams for single byte character sets. The reason is there are relatively few single byte characters for a reliable choosing of the character set based on the statistical distribution. For languages which do not use latin characters, all latin characters are filtered before the processing.

Similarly as in cryptography, the relative frequency table of character occurrencies in the text is used for identifying the language or character set of the document because that is unique for each language in a given encoding. The method of *chardet* creates for the language model of

[3] A language model usually represents a table of n-grams sorted by their occurrences in the text.

[4] Singlebyte (multibyte) character set represents one character by one (more than one) byte.

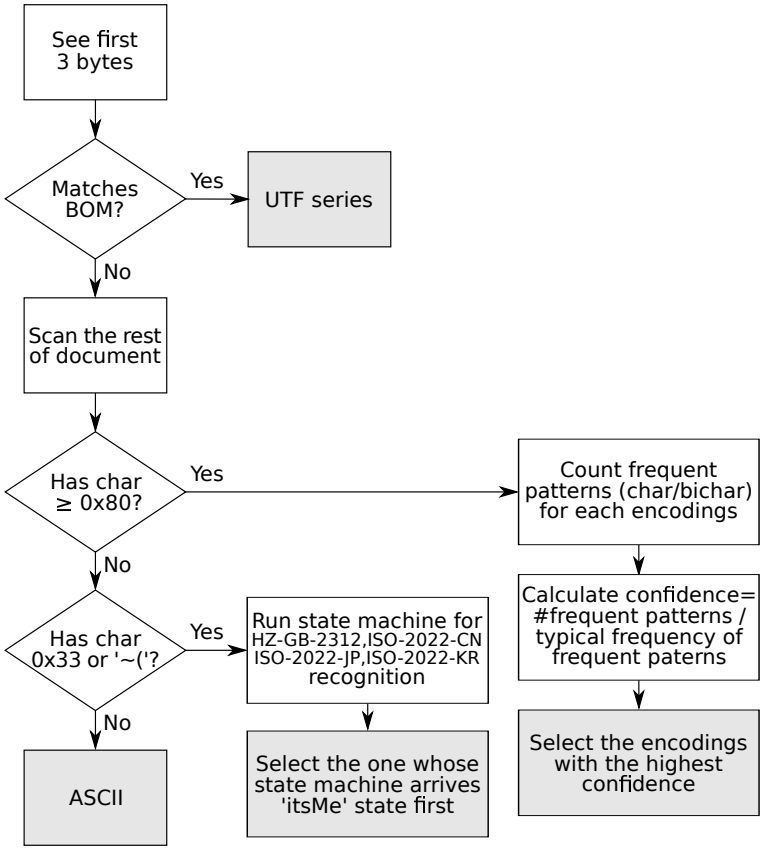

Fig. 1. Flow chart of the chardet detector

single byte character sets a table of 4096 bigrams created from the 64 most common occurring letters sorted by their occurrencies in text. This table is then divided into four categories. The first category represents the first 512 most frequent bigrams and is marked as *positive category*, the second representing the next 512 frequent bigrams is marked as *likely category*. The last *negative category* includes bigrams of letters which occur in the text rarely, and their number in the text, from which the language model is created is lower than three occurrences per bigram. The last but one *unlikely category* includes all bigrams between the second and the fourth category.

While detecting the character set, *chardet* calculates the *confidence level* value from the test file for each language model according to algorithm 1, and it chooses the character set based on the highest value.

5 Our *chardet* modification

The algorithm of *chardet* is sufficiently reliable for detecting the character sets which can be distinguished by searching for specific bytes, but it fails in case of some European languages especially because of the missing language models. Our aim was to create the missing models so that *chardet* could detect the character sets more reliably. The new way of creating the language model is described in algorithm 2. The improvement is that bigrams are now divided into categories based on the thresholds of cumulative probability of their occurrence in the text. The values of thresholds 0.95 and 0.999 were obtained using a heuristic approach so that the resulting detection would give the smallest error on the test files.

Algorithm 1 Confidence Level Calculation
input: *input_text*, *bigrams_frequencies* table, *tpr* (typical positive ratio)
output: *confidence_level*

```
for all char in input_text do
    if char is not a symbol or control character then
        num_chars += 1
    order = bigrams_frequencies[char]
    if order < 64 then
        freq_chars += 1
        if last_order < 64 then
            num_seqs += 1
            category = GetCategoryValue(last_order, order)
            counters[category] += 1
    last_order = order
confidence_level = counters[positive_cat]/(num_seqs/tpr) * (freq_chars/num_chars)
```

We also used a different approach than in the original algorithm for creating the mixed character set ISO 8859-1. At first, we mutually compared the probabilities of the occurrence of individual letters for all studied languages using correlation coefficients (see Tab. 4). Since only a few of the languages sufficiently mutually correlated, we have decided to assign letters into groups so that the mutual correlations between the languages were not smaller than value 0.95 (see Tab. 5). The highest correlation was achieved by assigning the letters into the categories shown in table 3. Similarly as while creating the language models for individual languages, also here the sizes of the categories were chosen so that the detector gave the best results. Analogously, we used optimization to choose the relation for the calculation of *typical positive ratio* value (see algorithm 3). Our improved program also deletes from the text all `html` and `xml` element tags for *html* and *xml* files. Only such filtered text continues to the detection.

Algorithm 2 Language Model Creation
input: *input_text* raw utf-8 data text, *charset* table
output: *chars_map*, *bigrams_frequencies*, *typical_positive_ratio*

```
Create chars_map[char] = 0 table for each char in the given charset
Create letters[] list from chars_map table
Create chars_frequencies[char] table for each char in chars_map table
Sort chars_frequencies[] table from the most frequent to less frequent char
Update chars_map[letter] value for each letter in letters list to a position index value of the letter in
    chars_frequencies table
Create frequent_letters[] list for first 64 most frequent letters from chars_frequencies
Create bigrams_frequencies[bigram] table for each bigram created from frequent_letters list
Sort bigrams_frequencies[] table from the most frequent to less frequent bigram
Update bigrams_frequencies[bigram] value for each bigram to a category value
Calculate typical_positive_ratio =
    (num_positivecat_bigrams + 0.25 * num_likelycat_bigrams)/num_bigrams *
    (num_frequent_letters/num_letters)
```

6 Language models creation

The training data were obtained from publicly available sources on the internet in UTF-8 encoding. The data were a mixture composed of various documents: beletry, professional (technical, scientific, politics, economics) literature and biblical texts. In case of beletry, the

Algorithm 3 Mixed Language Model Creation
input: n, $lang\_text[i]$ for $i = 1$ to n raw utf-8 data text, $letter\_codes$ list, $chars\_to\_lcodes$ table
output: $avg\_bicodes\_frequencies$, $typical\_positive\_ratio$

1: Convert all chars in input $lang\_text[i]$ files to letter codes using $chars\_to\_lcodes$ table
2: Create $bicodes\_frequencies[i][bicode]$ table for each $bicode$ created from $letter\_codes$
3: Calculate average relative frequency for each $bicode$
$avg\_bicodes\_frequencies[bicode] = \sum_{i=1}^{n} bicodes\_frequencies[i][bicode]/n$
4: Update $avg\_bicodes\_frequencies[bicode]$ value for each $bicode$ to a category value
5: Calculate $typical\_positive\_ratio =$
$(num\_positivecat\_bicodes + 0.25 * num\_likelycat\_bicodes - num\_unlikelycat\_bicodes - num\_negativecat\_bicodes)/num\_all\_bicodes/1.1532$

letters category	code	characters
ascii small vowel	0	a e i o u y
small vowel accent	1	à á â ã è é ê ì í î ò ó ô õ ù ú û ý
small vowel other	2	ä å æ ë ï ö ø œ ü ÿ
ascii small consonant	3	b c d f g h j k l m n p q r s t v w x z
small consonant other	4	ç ð f ñ ß š þ ž
ascii capital vowel	5	A E I O U Y
capital vowel accent	6	À Á Â Ã È É Ê Ì Í Î Ò Ó Ô Õ Ù Ú Û Ý
capital vowel other	7	Ä Å Æ Ë Ï Ö Ø Œ Ü Ÿ
ascii capital consonant	8	B C D F G H J K L M N P Q R S T V W X Z
capital consonant other	9	Ç Ð Ñ Š Þ Ž

Tab. 3. Letters categories

	EN	FR	DE	ES	PT	IT	SV	NL	DA	FI
English (EN)		0,931	0,934	0,926	0,886	0,917	0,929	0,940	0,918	0,853
French (FR)	0,931		0,939	0,947	0,899	0,932	0,920	0,937	0,928	0,859
German (DE)	0,934	0,939		0,891	0,817	0,852	0,919	0,972	0,948	0,796
Spanish (ES)	0,926	0,947	0,891		0,971	0,966	0,926	0,919	0,902	0,844
Portuguese (PT)	0,886	0,899	0,817	0,971		0,949	0,880	0,854	0,834	0,828
Italian (IT)	0,917	0,932	0,852	0,966	0,949		0,907	0,893	0,871	0,876
Swedish (SV)	0,929	0,920	0,919	0,926	0,880	0,907		0,930	0,950	0,892
Dutch (NL)	0,940	0,937	0,972	0,919	0,854	0,893	0,930		0,968	0,813
Danish (DA)	0,918	0,928	0,948	0,902	0,834	0,871	0,950	0,968		0,789
Finnish (FI)	0,853	0,859	0,796	0,844	0,828	0,876	0,892	0,813	0,789	

Tab. 4. Crosscorrelation of relative letters frequencies

	EN	FR	DE	ES	PT	IT	NL	DA	FI
English (EN)		0.996	0.997	0.995	0.978	0.985	1.000	0.993	0.990
French (FR)	0.996		0.987	1.000	0.991	0.995	0.996	0.981	0.995
German (DE)	0.997	0.987		0.984	0.959	0.968	0.998	0.999	0.980
Spanish (ES)	0.995	1.000	0.984		0.994	0.996	0.994	0.978	0.995
Portuguese (PT)	0.978	0.991	0.959	0.994		0.999	0.976	0.950	0.991
Italian (IT)	0.985	0.995	0.968	0.996	0.999		0.983	0.960	0.996
Dutch (NL)	1.000	0.996	0.998	0.994	0.976	0.983		0.995	0.989
Danish (DA)	0.993	0.981	0.999	0.978	0.950	0.960	0.995		0.974
Finnish (FI)	0.990	0.995	0.980	0.995	0.991	0.996	0.989	0.974	

Tab. 5. Crosscorrelation of relative letters categories frequencies

	original chardet		improved chardet	
test files	accuracy	calc. time	accuracy	calc. time
260 (european languages)	9.61%	9.681s	99.6%	12.560s
626 (all languages)	61.18%	68.149s	99.84%	40.614s

Tab. 6. Results of tests

documents were mainly current ones, but some were also historical. The minimum length of the training data was 10 MB for each language. Longer parts in other languages were removed from the documents. If a language model did not achieve a sufficient accuracy in the reverse detection of the language, the training data were checked again, and also shorter parts in other languages were removed.

7 Testing and conclusion

The testing data were mainly obtained from news servers from various countries across Europe in *txt* format and UTF-8 encoding. These data were then re-encoded into the chosen character sets. The minimum length of the test files was 1 kB, which represents approximately 150 words. The average length of a test file was 4.7 kB. Besides the newly acquired documents, we used also the original test files of the *chardet*, which were in formats *txt*, *html*, *xml* and *srt* with various length between 136 B to 582 kB.

Overall 17 languages were processed for improving the accuracy of detection of the character set using *chardet*. Thereof 7 new language models were created for character set ISO-8859-2/Windows-1250 (Czech, Croatian, Hungarian, Polish, Romanian, Slovak, Slovenian). A new mixmodel for ISO-8859-1/Windows-1252 (English, French, German, Spanish, Portuguese, Italian, Dutch, Danish, Finnish) was created. Swedish language was not included in this model because of poor correlation with other countries using character set ISO-8859-1. Norwegian language was not included in the model because of missing data for creating frequency table. The language models for Turkish and Greek were repaired because of improving the accuracy of detection.

Overall 18 languages were tested for character sets ISO-8859-1/Windows-1252 and ISO-8859-2/Windows-1250 on 270 files. 8 results were incorrect, from which 1 of 10 was for Finnish language and 7 of 10 was for Swedish language. Since Swedish language model has not been created yet, only 1 result was truly incorrect. Moreover, this incorrect result is not surprising because the detector confused Finnish language with Hungarian language. It is known that both languages belong to the group of Ugro-Finnic languages and are related by their bigrams. After excluding the test files for Swedish language, we got 99.6% accuracy (see Tab. 6). The test of the original unimproved detector on the same files ended very badly. The program received 233 failures from 249 tested files. Moreover, it did not run on two files and did not give any result. The reason is the model for ISO-8859-1 was created without a more detailed analysis of the relation between characters or character sets, and there were missing language models for countries using character set ISO-8859-2. The group ISO-8859-2 was represented only by Hungarian language and even that was detected incorrectly in some cases.

The existing program *chardet* was not only supplemented by new language models, but also some parts of the source code were modified. Despite adding other language models, which means increase in the number of performed operations, the program increased its speed when testing 626 files by more than 40%.

We strongly believe that the proposed changes will be gradually incorporated into the main development branch of widely used *chardet* detector, which will improve the accuracy of automatic detection of character set in many applications.

8 Acknowledgement

This paper arose thanks to the support of the project KEGA 094UK-4/2013 "Ematik+, Continuing education of mathematics teachers", http://elearn.ematik.fmph.uniba.sk.

References

[1] BALDWIN, T., LUI, M. *Language identification: The long and the short of the matter*, in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010, pp. 229–237.

[2] CAVNAR, W. B., TRENKLE, J. M., et al. *N-gram-based text categorization*, Ann Arbor MI, vol. 48113, no. 2, (1994), pp. 161–175.

[3] CHEW, Y. C., MIKAMI, Y., NAGANO, R. L. *Language identification of web pages based on improved n-gram algorithm*, International Journal of Computer Science Issues, vol. 8, no. 3, (2011), pp. 47–58.

[4] DUNNING, T. *Statistical identification of language*, Computing Research Laboratory, New Mexico State University, 1994.

[5] GREFENSTETTE, G. *Comparing two language identification schemes*, 3rd International conference on Statistical Analysis of Textual Data (JADT 1995).

[6] GROTHE, L., De LUCA, E. W., NÜRNBERGER, A. *A comparative study on language identification methods.*, in *LREC*, 2008.

[7] KIM, S., PARK, J. *Automatic detection of character encoding and language*, Tech. rep., CS 229, Machine Learning, Stanford University, 2007.

[8] LI, S., MOMOI, K. *A composite approach to language/encoding detection*, in *19th Unicode Conference*, San Jose, 2001.

[9] LODHI, H., SAUNDERS, C., SHAWE-TAYLOR, J., CRISTIANINI, N., WATKINS, C. *Text classification using string kernels*, The Journal of Machine Learning Research, vol. 2, (2002), pp. 419–444.

[10] MARTINO, M. J., PAULSEN Jr, R. C. *Natural language determination using partial words*, 2001, US Patent 6,216,102.

[11] POMIKÁLEK, J., SUCHOMEL, V. *chared: Character encoding detection with a known language*, in *RASLAN*, 2011, pp. 125–129.

[12] PRAGER, J. M. *Linguini: Language identification for multilingual documents*, in *Proceedings of the 32nd Hawaii International Conference on System Sciences*, IEEE, 1999, pp. 11–pp.

[13] SULLIVAN, S. R. *Character set encoding detection using a support vector machine.*

[14] SUZUKI, I. *A language and character set determination method based on n-gram statistics*, ACM Transactions on Asian Language Information Processing (TALIP), vol. 1, no. 3, (2002), pp. 269–278.

[15] TEYTAUD, O., JALAM, R. *Kernel-based text categorization*, in *In International Joint Conference on Neural Networks*, vol. 3, 2000, pp. 1891–1896.

Current address

Róbert Bohdal, RNDr., PhD.
Department of Algebra, Geometry and Didactics of Mathematics
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava
Mlynska Dolina, 842 48 Bratislava, Slovak Republic
Tel. number: +421-2-602 95 ext. 185, e-mail: robert.bohdal@fmph.uniba.sk