chared: Character Encoding Detection with a Known Language

Jan Pomikálek[1] and Vít Suchomel[2]

[1] Lexical Computing Ltd.
73 Freshfield Road, Brighton, UK
jan.pomikalek@sketchengine.co.uk

[2] Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
xsuchom2@fi.muni.cz

Abstract. chared is a system which can detect character encoding of a text document provided the language of the document is known. The system supports a wide range of languages and the most commonly used character encodings. We explain the details of the algorithm, describe the process of creating models for various languages and present results of an evaluation on a collection of Web pages.

Key words: character encoding, character encoding detection, charset, Unicode

1 Introduction

When creating monolingual text corpora from the Web, one of the problems which need to be addressed is character encoding detection. Web pages usually contain the information about the used character encoding in the meta tags. However, this information is not always available and not always correct, and in general cannot be relied upon. Nevertheless, the character encoding can usually be reliably guessed from the textual contents of a Web page.

The problem of character encoding detection can be tackled at two different levels. First, no assumption is made about the language of the input text. In this case both the character encoding and the language of the input may be guessed at the same time. Second, the language of the input is known and only the character encoding is detected. Existing systems which fall into the former category include for instance TextCat[1] and chardet[2]. A notable representative of the latter category is Enca[3].

The disadvantage of the existing systems is that they support only a limited number of languages and encodings. The system we designed, on the other

[1] http://www.let.rug.nl/vannoord/TextCat/

[2] http://pypi.python.org/pypi/chardet

[3] http://gitorious.org/enca

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2011, pp. 125–129, 2011.

hand, covers a wide range of languages and frequently used encodings for each language.

2 chared

Our method assumes that the language of the input is known. For instance, when building monolingual corpora this is always guaranteed.[4] By narrowing the problem down to a single language, higher accuracy can be achieved.

2.1 Algorithm

The system works as follow. First, a model is created for each supported language. This requires two kinds of input:

1. The list of character encodings used for given language
2. A sample text in the language and in a known character encoding

We convert the sample text to all encodings from the list, creating N different files (where N is the number of encodings). Then, for each of the files we build a frequency list of all consequent triples of bytes. The N frequency lists (vectors) are then compared and the items (triples) with the same frequency in all the N vectors are discarded.

When detecting a character encoding of a document, its frequency vector is built and a scalar product is computed with each of the N model vectors. The character encoding associated with the model vector which produces the highest scalar product is returned as the encoding detected for the input document.

The advantage of building model vectors from the same data (only converted to N different encodings) is that the differences among the model vectors are only where the character encodings differ. If for instance all the N encodings are ascii compatible, then any triple of bytes containing only ascii characters will have the same frequency in all model vectors. Thus, this triple would add the same value to the scalar product with a tested document (vector) for all models. As such it cannot affect the order of the final scalar products (and the result of the classification) and thus it can be safely removed from all the model vectors. This both reduces the size of the vectors and speeds up the classification. Alternative approach which builds the models on N different texts in N different encodings prevents such pruning. Apart from that, similarity of the tested document to the texts used for training the models (regardless of their

[4] In fact, in a web corpus processing pipeline, character encoding detection has to be performed before language detection. Thus, the assumption that the input of the character encoding algorithm is in the language in question is not always valid. However, this is not a problem, since all texts in other languages will be refused by the language filter in the next step and it therefore does not matter whether their character encoding has been detected correctly or not.

encoding) may bias the results of the encoding detection. This bias would be the more severe the more ascii characters (or more generally, the more characters with the same representation in all the encodings in question) are used in the training texts (e.g. very severe for English).

The reason for using frequencies of tuples (triples in our case) of characters rather than single characters is that some bytes represent multiple different commonly used characters in different encodings. Such bytes may cause errors in the encoding detection. This is best illustrated on an example. The byte `a9` represents the character © (copyright symbol) in windows-1250 and the character Š (capital s with caron) in iso-8859-2. Both encodings are used for Czech language. The Š character is much more frequent in Czech texts than the copyright symbol. Thus, a windows-1250 text containing a copyright symbol may be easily misclassified as iso-8859-2 since the `a9` byte will have a higher frequency in the iso-8859-2 model. Since the two encodings are very similar, this kind of classification error is fairly likely, especially for short texts.

Building models on tuples of characters rather than on single characters helps to overcome this problem. The copyright symbol will be typically followed by a space whereas the Š character will be typically followed by another letter in a Czech text. Therefore, two bytes `a920` (© followed by a space in windows-1250) will certainly have a higher frequency in a windows-1250 model than in a iso-8859-2 model (for Czech). In our experiments, using triples of characters resulted in a higher accuracy of the encoding detection than using only pairs of characters.

2.2 Building models

In order to create models for our system, we collected a set of about 1000 Web pages for each language. This was done using Corpus Factory tools. [1] We identified the encoding of each page by using the information in the meta tags. Though we are well aware that this information is not always correct we believe the errors are so rare that they cause only an insignificant bias in the built models. We simply discarded all pages for which we have not been able to determine the character encoding from the meta tags.

First, for each language, we computed the relative frequencies of the encodings used in the Web pages and accepted those with a relative frequency above 0.5 % as the encodings commonly used for the language. We then created a model vector for each of these encodings by converting the Web pages to the encoding and building the frequency vector as described above.

3 Evaluation

To evaluate the system, we performed a 5-fold cross-validation on the training data. Table 1 contains results for selected languages.

The average accuracy is weighted by the frequency of occurrence of the encodings. The rationale of the weighting is that it is more important that the

	Czech		English		German		Greek		Italian		Norwegian	
	freq	accuracy	freq	accuracy	freq	accuracy	freq	accuracy	freq	accuracy	freq	accuracy
utf-8	60.2 %	100.0 %	56.9 %	95.8 %	54.6 %	100.0 %	68.5 %	100.0 %	54.2 %	100.0 %	63.0 %	100.0 %
windows-1250	32.2 %	100.0 %	0.3 %	n/a	0.1 %	n/a	0.2 %	n/a	0.0 %	n/a	0.1 %	n/a
windows-1252	0.4 %	n/a	9.4 %	97.5 %	6.5 %	97.3 %	3.1 %	75.8 %	7.1 %	95.7 %	7.0 %	97.4 %
windows-1253	0.0 %	n/a	0.0 %	n/a	0.0 %	n/a	14.3 %	99.3 %	0.0 %	n/a	0.0 %	n/a
iso-8859-1	1.0 %	89.5 %	32.8 %	90.9 %	37.1 %	85.8 %	1.7 %	71.2 %	37.9 %	85.1 %	29.3 %	88.2 %
iso-8859-2	6.0 %	99.6 %	0.0 %	n/a	0.1 %	n/a	0.0 %	n/a	0.1 %	n/a	0.1 %	n/a
iso-8859-7	0.0 %	n/a	0.0 %	n/a	0.0 %	n/a	12.0 %	97.2 %	0.0 %	n/a	0.0 %	n/a
iso-8859-15	0.0 %	n/a	0.0 %	n/a	1.2 %	85.6 %	0.0 %	n/a	0.0 %	n/a	0.4 %	n/a
training docs		801		668		773		879		771		740
w. avg accuracy		99.2 %		93.5 %		93.7 %		97.9 %		93.3 %		95.7 %

Table 1. Character encoding detection accuracy for selected languages.

algorithm correctly identifies the frequently used encodings than the rarely used ones.

Note that the results may be harmed by incorrect "annotation" of the data. We manually inspected some of the misclassified pages and often found out that the algorithm detected the encoding of the page correctly, but the character encoding specified in the meta tags was wrong. We therefore assume that the accuracy of our system is higher than indicated by the numbers in the table.

4 Conclusion

We have presented chared, a system for detecting character encoding for documents of which we know the language. Though our evaluation is only preliminary and lacks comparison with other systems, we are convinced that our approach works better than the general approach to the problem which makes no assumptions about the language of the input. The main advantage of our system compared to other available similar programs is a support for a wide range of languages (currently 51). chared is released under the BSD open source licence and is available for download from Google code[5].

Acknowledgements The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 248307 (PRESEMT project) and from the Czech Science Foundation under the project P401/10/0792.

References

1. Kilgarriff, A., Reddy, S., Pomikálek, J., PVS, A.: A corpus factory for many languages. Proc. LREC, Malta (2010)

[5] `http://code.google.com/p/chared/`