日本ソフトウェア科学会第 29 回大会 (2012 年度) 講演論文集

A Repository System for Cross-lingual Documents

Wenting Gu Koichi Sasada Shigeru Chiba

This paper describes a repository system for showing differences in line in corresponding cross-lingual documents to help users keep documents in multiple languages synchronized more easily in a collaborative working environment. This system is able to track changes to documents in different languages, locate the corresponding areas in sentences of cross-lingual documents by using existing sentence alignment techniques, and perform version control. Our system is developed for the reason that we found it is difficult for users to keep the contents of cross-lingual documents synchronized with existing tools. Firstly, there are few tools that support tracking changes and showing them in cross-lingual documents. Secondly, existing tools for cross-lingual documents only inform users of the changes but do not show their location, so it takes users much time and effort to locate the corresponding area of a document in different languages. In this paper, we present the prototype of our system and how our system solves the problems mentioned above.

1 Introduction

In this paper, we present a system for supporting collaborative work on keeping corresponding cross-lingual documents synchronized. It calculates the differences of cross-lingual documents to help users synchronize them and when such a document in one language is modified, it can indicate where modifications need to be made within the documents for the other languages.

Documents written in multiple languages, such as software manuals, Wikipedia pages and so on, have become increasingly prevalent. At the same time the maintenance of these cross-lingual documents which can be modified by collaboratively working has recently become a problem. For example, after a user has modified some parts of a document in a certain language, other users first need to find the paragraphs or sentences that have been modified, and then find the same respective paragraphs or sentences within the associated documents in other languages to perform the same modification.

Some tools were developed for solving this problem. For example, CLWE[1], a system proposed by Huberdeau et al., provides support for informing users of changes in cross-lingual Wiki pages and listing those changes. However, we found that users still need to spend time and effort on finding the exact place that corresponds to each change. This is especially difficult in those cross-lingual documents for which paragraph or sentence order does not match.

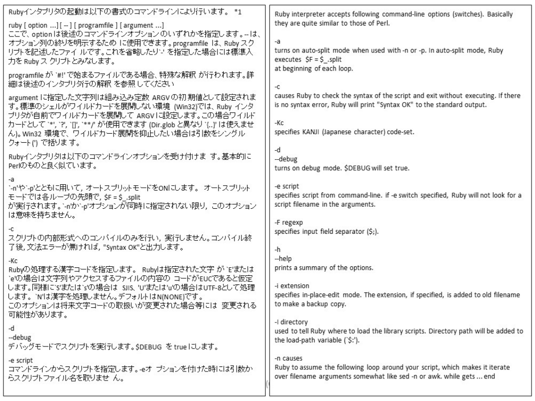
We believe it is useful to help users keep cross-lingual documents in a consistent state with collaboration by offering a tool that locates and displays differences of these documents. In order to display these localized differences, we import the existing sentence alignment techniques employed by [3] to calculate the similarity of sentences and para-

多言語文書支援リポジトリシステム

コ モンテン　千葉 滋, 東京大学大学院情報理工学系研究科, Graduate School of Information Science and Technology, The University of Tokyo.

笹田 耕一, Heroku, Inc., Heroku, Inc..

graphs and establish a correspondence relationship between them. Owing to the limitations of current sentence and paragraph alignment techniques, the relationship between these sentences and paragraphs may not be 100% correct. If necessary, users can manually re-adjust this correspondence relationship themselves. Also, our system provides version control functionality such as tracking changes, locking files, merging versions, and rolling back, to facilitate collaborative work.



Rubyインタプリタの起動は以下の書式のコマンドラインにより行います。 *1

ruby [option ...] [--] [programfile] [argument ...]
ここで、option は後述のコマンドラインオプションのいずれかを指定します。-- は、オプション列の終りを明示するため に使用できます。programfile は、Ruby スクリプトを記述したファイ ルです。これを省略したり`-' を指定した場合には標準入力を Ruby ス クリプトとみなします。

programfile が `#!' で始まるファイルである場合、特殊な解釈 が行われます。詳細は後述のインタプリタ行の解釈 を参照 してください

argument に指定した文字列は組み込み定数 ARGV の初 期値として設定されます。標準のシェルがワイルドカードを展開しない環境 (Win32)では、Ruby インタプリタが自前でワイルドカードを展開して ARGV に設定します。この場合ワイルドカードとして `*', `?', `[]', `**/' が使用できます (Dir.glob と異なり `{..}' は使えません)。Win32 環境で、ワイルドカード展開を抑止したい場合は引数をシングルクォート (') で括りま す。

Rubyインタプリタは以下のコマンドラインオプションを受け付けま す。基本的にPerlのものと良く似ています。

-a
`-n'や`-p'とともに用いて, オートスプリットモードをONにします。 オートスプリットモードでは各ループの先頭で, $F = $_.split
が実行されます。`-n'か`-p'オプションが同時に指定されない限り, このオプションは意味を持ちません。

-c
スクリプトの内部形式へのコンパイルのみを行い, 実行しません。コンパイル終了後, 文法エラーが無ければ, "Syntax OK"と出力します。

-Kc
Rubyの処理する漢字コードを指定します。 Rubyは指定された文字 が `E'または `e'の場合は文字列やアクセスするファイルの内容の コードがEUCであると仮定します。同様に`S'または`s'の場合は SJIS、'U'または'u'の場合はUTF-8として処理します。 `N'は漢字を処理しません。デフォルトはN(NONE)です。
このオプションは将来文字コードの取扱いが変更された場合等には 変更される可能性があります。

-d
--debug
デバッグモードでスクリプトを実行します。$DEBUG を true にします。

-e script
コマンドラインからスクリプトを指定します。-eオ プションを付けた時には引数からスクリプトファイル名を取りませ ん。

Ruby interpreter accepts following command-line options (switches). Basically they are quite similar to those of Perl.

-a
turns on auto-split mode when used with -n or -p. In auto-split mode, Ruby executes $F = $_.split
at beginning of each loop.

-c
causes Ruby to check the syntax of the script and exit without executing. If there is no syntax error, Ruby will print "Syntax OK" to the standard output.

-Kc
specifies KANJI (Japanese character) code-set.

-d
--debug
turns on debug mode. $DEBUG will set true.

-e script
specifies script from command-line. if -e switch specified, Ruby will not look for a script filename in the arguments.

-F regexp
specifies input field separator ($;).

-h
--help
prints a summary of the options.

-i extension
specifies in-place-edit mode. The extension, if specified, is added to old filename to make a backup copy.

-I directory
used to tell Ruby where to load the library scripts. Directory path will be added to the load-path variable (`$:').

-n causes
Ruby to assume the following loop around your script, which makes it iterate over filename arguments somewhat like sed -n or awk. while gets ... end

Figure1 Part of the Ruby documentation in English and Japanese

2 Editing Cross-lingual Documents

In this section, we present a detailed statement to explain why it is currently difficult for users to keep the contents of cross-lingual documents the same with existing tools which only list the changes to the original language text. We analyze the problems with an example of the documentation for the Ruby programming language in English and Japanese.

The programming manual of Ruby is now being authored in both English and Japanese. As Figure1 shows, the contents and order of paragraphs of these two documents are different. For example, the first paragraph in the English document corresponds to the fifth paragraph in the Japanese one. As both documents are part of the Ruby documentation, developers of Ruby want to synchronize the two documents and keep them the same, that is, modifications to a text region R in one document needs to be translated and reflected in that region in the other document that corresponds with region R. However, there is no effective tool to help them accomplish this task.

In order to synchronize the two documents, the first step is to identify those parts within the documents that differ from each other. For the reason that existing tools for supporting collaborative work on cross-lingual documents are not able to figure out the differences between the two documents, synchronizing documents is hard work for users to do by themselves because the Ruby documentation documents in Japanese and English are quite long.

Also, when the two documents are synchronized by user efforts, they might be modified with different content at the same time. For instance, when the user named Sato, who is a Japanese, added documents of the two commands, "-C directory" and "–copyrigh", one before "-c" command and the other after "-c" command, and gave some explanation of them. At the same time, a user called Mary, who is an American, deleted the content of "-Kc" command in the English document because this command is no longer used. After doing this, the order of the command explanations as well as the contents within the two documents are different again. In this case, CLWE can inform users the modifications made in the two documents, but it can not point out the specific region that corresponds to the modification. Thus, users still need to find the exact region to be updated in the corresponding document by themselves.

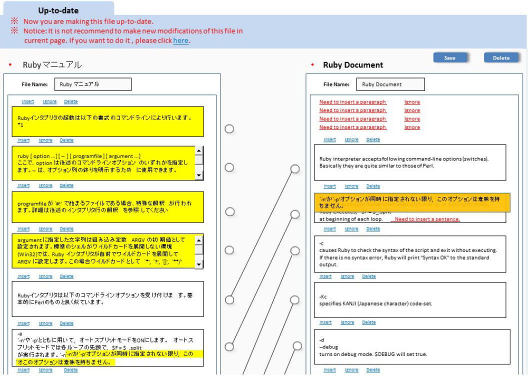
To sum up, the problems are as follows. First, it is difficult to synchronize two documents with different contents in different languages because the locations of differing content parts are not known. Second, when a document in a certain language is modified, it is hard to make the same modification to documents in other languages for the reason that the exact location requiring updating in them is not known. The example we gave is only about two languages, English and Japanese. We believe it will get even more complicated when documents written in more than two languages become involved.

3 A Repository System for Cross-lingual Documents

We consider that if there is a system that can identify differences between cross-lingual documents automatically and establish a correct correspondence relationship between sentences, the problem in Section 2 can be solved easily. Also, we consider it is helpful to import version control functions such as file locking, version merging, and rolling back functions into our system to help users do collaborative work.

3.1 Keeping Cross-lingual Documents Synchronized

We import existing techniques for finding sentence similarity into our system to indicate differences between cross-lingual documents and establish a correspondence relationship between sentences. Owing to the limitations of current sentence and paragraph alignment techniques, the relationship between these sentences may not be 100% correct. In order to improve the accuracy of the correspondence relationship, we not only calculate the similarity of sentences, but also calculate the similarity of paragraphs. Also, we provide a direct visualization displaying the relationship between paragraphs (Figure2) for users so that they can manually adjust the relationship. Once the correspondence relationship is changed, differences will be re-calculated. Then we track changes of each document and locate regions correspond to these changes in its corresponding documents through the established relationship.



※ Now you are making this file up-to-date.
※ Notice: It is not recommend to make new modifications of this file in current page. If you want to do it , please click here.

Ruby マニュアル

File Name: Ruby マニュアル

Rubyインタプリタの起動は以下の書式のコマンドラインにより行います。*1

ruby [option ...] [--] [programfile] [argument ...]
ここで、option は後述のコマンドラインオプション のいずれかを指定します。-- は、オプション列の終りを明示するため に使用できます。

programfile が `#!' で始まるファイルである場合、特殊な解釈 が行われます。詳細は後述のインタプリタ行の解釈 を参照してください

argument に指定した文字列は組み込み定数 ARGV の初期値として設定されます。標準のシェルがワイルドカードを展開しない環境(Win32)では、Ruby インタプリタが自前でワイルドカードを展開して ARGV に設定します。この場合ワイルドカードとして `*', `?', `[]', `**/'

Rubyインタプリタは以下のコマンドラインオプションを受け付けます。基本的にPerlのものと良く似ています。

-a
`-n'や`-p'とともに用いて，オートスプリットモードをONにします。 オートスプリットモードでは各ループの先頭で，$F = $_.split
が実行されます。`-n'か`-p'オプションが同時に指定されない限り，このオプションは意味を持ちません。

Ruby Document

File Name: Ruby Document

Need to insert a paragraph Ignore
Need to insert a paragraph Ignore
Need to insert a paragraph Ignore
Need to insert a paragraph Ignore

Ruby interpreter accepts following command-line options (switches). Basically they are quite similar to those of Perl.

`-n'か`-p'オプションが同時に指定されない限り，このオプションは意味を持ちません。
at beginning of each loop. Need to insert a sentence.

-c
causes Ruby to check the syntax of the script and exit without executing. If there is no syntax error, Ruby will print "Syntax OK" to the standard output.

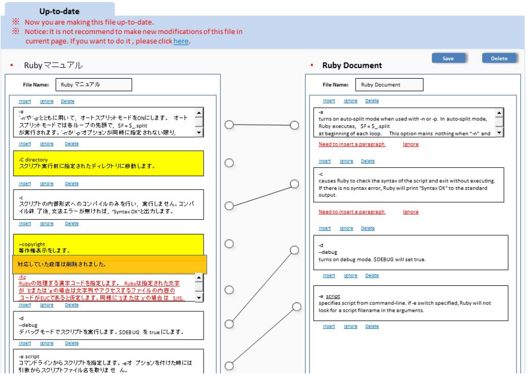
-Kc
specifies KANJI (Japanese character) code-set.

-d
--debug
turns on debug mode. $DEBUG will set true.

Figure2 Showing the difference between two documents

Also, we provide a friendly user interface (Figure2) to display differences to help users easily keep these documents the same. In our system, documents that should be updated are showed with a mark to inform users. Moreover, sentences presented underlined and in red color signify that they should be updated. Content shaded in yellow (background color) signifies it is a difference. When users move their mouse cursor and hover over a sentence in red, a box will appear that shows the difference to the corresponding sentence in the original document. Figure2 is an example of synchronizing two documents in different languages and Figure3 is an instance of keeping two synchronized documents the same.

Our system is helpful to users to keep cross-lingual documents the same for the reason that they do not need to manually compare the documents to find within them the parts in which they differ and locate the corresponding regions to be modified. In addition, we consider that the more languages documents are written in, the more valuable our system becomes, because the difficulty of synchronizing these documents and keeping them up-to-date is expected to increase exponentially. Moreover, we believe our system will become even more effi-



※ Now you are making this file up-to-date.
※ Notice: It is not recommend to make new modifications of this file in current page. If you want to do it , please click here.

- Rubyマニュアル

File Name: Rubyマニュアル

-a
`-n'や`-p'とともに用いて，オートスプリットモードをONにします。 オートスプリットモードでは各ループの先頭で，$F = $_.split
が実行されます。`-n'か`-p'オプションが同時に指定されない限り，

-C directory
スクリプト実行前に指定されたディレクトリに移動します。

-c
スクリプトの内部形式へのコンパイルのみを行い，実行しません。コンパイル終 了後，文法エラーが無ければ，"Syntax OK"と出力します。

--copyright
著作権表示をします。

対応していた段落は削除されました。

-Kc
Rubyの処理する漢字コードを指定します。 Rubyは指定された文字が `E'または`e'の場合は文字列やアクセスするファイルの内容のコードがEUCであると仮定します。同様に`S'または`s'の場合は SJIS,

-d
--debug
デバッグモードでスクリプトを実行します。$DEBUG を trueにします。

-e script
コマンドラインからスクリプトを指定します。-eオ プションを付けた時には引数からスクリプトファイル名を取りま せ ん。

- Ruby Document

File Name: Ruby Document

-a
turns on auto-split mode when used with -n or -p. In auto-split mode, Ruby executes, $F = $_.split
at beginning of each loop. This option mains nothing when "-n" and

-c
causes Ruby to check the syntax of the script and exit without executing. If there is no syntax error, Ruby will print "Syntax OK" to the standard output.

-d
--debug
turns on debug mode. $DEBUG will set true.

-e script
specifies script from command-line. if -e switch specified, Ruby will not look for a script filename in the arguments.

Figure3 Showing the difference in the correspondent area

cient with potential future improvement of sentence alignment techniques in cross-lingual text.

3.2 Sentence and Paragraph Alignment

The most important function of our system is to display differences in corresponding places within documents in other languages.

The algorithm of aligning Japanese and English news articles and sentences (SIM(J,E) [3] [5] [6]) proposed by Muchi et al. [3] has high appreciation so it is being used in our system to calculate the similarity of the paragraphs and sentences.

$$SIM(J,E) = \frac{co(J \cap E)+1}{|J|+|E|-2co(J \cap E)+2}$$

J is the collection of words of a sentence of a language. E is the collection of words of a sentence of another language. $f(x)$ is defined to the frequency word x in collection X and $|X| = \sum_{x \in X} f(x)$. $co(J \cap E) = \sum_{(j,e) \in J \cap E} min(f(j), f(e))$. $J \cap E$ means that if there is a union $(j,e) j \cap J \wedge e \cap E$, the translation of j belongs to E, and the translation of e belongs to J.

We take advantage of Kuromoji[†1] to get the words of a Japanese document. Then we use Microsoft Translator and WordNet[†2] (both English and Japanese version) to retrieve the phrases, words, and their translations. Finally, we calculate the similarity of two paragraphs from the result of the similarity of sentences in the two paragraphs.

According to the evaluation by Muchi et al. [3], if the measured similarity of two paragraphs is below 30%, we consider the two paragraphs not to be corresponding. If the similarity is above 30% and less than 50%, the two paragraphs can be regard as ordinarily corresponding. If it is above 50%, the two paragraphs can be regarded as well corresponding. In this case, the accuracy of the correspondence relationship of news from 1989 to 2001 (about 944,404 pieces) in English and Japanese is 95%.

4 Related Work

There exists some prior work related to our system that supports collaborative working on cross-lingual documents.

Huberdeau et al. proposed a system called CLWE [1], which can assist users in synchronizing the contents of Wiki pages to a certain degree. The CLWE system allows communities to break out of the constrained mold imposed by conventional translation processes, and allows contributors to follow an open-ended work-flow that is more consistent with modern collaborative environments. However, it only lists the changes so that users still need to spend time on finding the corresponding areas especially in the case of content with different internal order. Moreover, it does not support documents which already have differing content like the Ruby manual because CLWE is not able to judge whether the contents in different languages are equal. Our system solves these problems by using existing techniques for calculating sentence similarity to help users do collaborative working on cross-lingual documents easily.

Another related work is an application presented by Kulkarni et al. [7]. This application

[†1] An open source Japanese morphological analyzer written in Java

[†2] http://wordnet.princeton.edu/

just calculate the similarity of the attributes of Wikipedia Info Boxes, not the general content, by using existing text similarity measures [8]. However, Wikipedia engines do not always provide the same attributes of Info Boxes in different languages [2] [4], thus the correspondence relationship provided by this application is not so accurate. We think it is hard to use that system to support collaborative working on general cross-lingual documents for the reason that general documents lack such additional semantic tags. In contrast, our system judges the similarity of content in different languages in paragraphs and sentences, and shows the differences in line to make it easy to keep cross-lingual documents the same.

5 Conclusion

We have presented a system designed to support collaborative working on cross-lingual documents. Our system allows users to collaboratively author and translate cross-lingual documents easily. Users can easily know the synchronization status of cross-lingual documents, and with little effort discover those parts in which they differ within the documents in different languages.

Because our system is currently still under development, the first step for our future work is to finish the development of our system. Subsequently, we will attempt to make our system support more languages such as Chinese, French, and so on. Additionally, we plan to add support for translation such as Microsoft Translation and translation memory to further help users accomplish their task in their corresponding work.

参考文献

[1] Louis-Philippe Huberdeau, Sebastien Paquet, Alain Desilets, The Cross-Lingual Wiki Engine: enabling collaboration across language barriers. In Int. Sym. Wikis (2008), ACM, Article No. 13.

[2] Alain Desilets, Lucas Gonzalez, Sebastien Paquet, Marta Stojanovic, Translation the Wiki way. In Int. Sym. Wikis (2006), ACM, pp. 19–32.

[3] 内山将夫, 井佐原均, 日英新聞の記事および文を対応付けるための高信頼性尺度. 自然言語処理, Vol. 10 (2003), pp. 201–220.

[4] Joachim Kimmerle, Johannes Moskaliuk, Ulrike Cress, Understanding learning: the Wiki way. In Int. Sym. Wikis and Open Collaboration (2009), ACM, Article No. 3.

[5] Takehito Utsuro, IIiroshi Ikeda, Masaya Yamane, Yuji Matsumoto, Makoto Nagao, Bilingual TextMatching using Bilingual Dictionary and Statistics. In COLING (1994), pp. 1076–1082.

[6] William A. Gale, Kenneth W. Church, A Program for Aligning Sentences in Bilingual Corpora. Computational Linguistics, Vol. 19 (1993), pp. 75–102.

[7] Ranjitha Gurunath Kulkarni, Gaurav Trived, Tushar Suresh, Miaomiao Wen, Zeyu Zheng, Carolyn Rose, Supporting collaboration in Wikipedia between language communities. In ICIC.(2012), pp. 47–56.

[8] Michael Mohler, Rada Mihalcea, Text-to-text semantic similarity for automatic short answer grading. In Proc. EACL (2009), pp. 567–575.

[9] Evgeniy Gabrilovich, Shaul Markovitch, Wikipedia-based semantic interpretation for natural language processing. Journal of Artificial Intelligence Research, Vol. 34, No. 1 (2009), pp. 443–498.

[10] Donald Metzler, Susan Dumais, Christopher Meek, Similarity measures for short segments of text. In Proc. ECIR (2007), pp. 16–27.