Automatic Evaluation Framework for Word Spotting

Kengo Terasawa[1,2], Hajime Imura[3] and Yuzuru Tanaka[3]

[1] *Department of Media Architecture, Future University-Hakodate, Japan*
[2] *PRESTO, Japan Science and Technology Agency, Japan*
[3] *Meme Media Laboratory, Hokkaido University, Japan*
kterasaw@fun.ac.jp, {hajime,tanaka}@meme.hokudai.ac.jp

Abstract

Word spotting is the task of retrieving a text region that has a similar appearance to a query image specified by the user. This paper proposes an automatic evaluation framework for word spotting methods. In order to make our framework available to researchers around the world, we discuss some standard definitions and representations that are suitable for most word spotting methods, regardless of the assumptions and settings on which the individual methods depend. We also design a protocol for interprocess communication between a parent process and a word spotting engine. This protocol can modularize individual spotting methods to become interchangeable parts in a larger application. Our framework will promote the development of such a word spotting method and improve its usability.

1. Introduction

This paper proposes an automatic evaluation framework for word spotting methods. We discuss what kind of interface should be developed in order to share common evaluation tools among researchers around the world. A prototype software we have implemented will also be introduced.

Word spotting is no doubt one of the main interests in the ICDAR community. The aim of word spotting is to retrieve a text region that has a similar appearance to the query image specified by the user. Word spotting provides many benefits for users reading manuscripts to which optical character recognition (OCR) does not provide enough accuracy, as is often the case with handwritten, historical or polluted document images. Because of its advantages, word spotting is expected to be useful in such fields as history, humanities and archival records management.

Many studies about word spotting have been published in recent years [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. These studies vary in assumptions and settings on which they depend. For example, some studies assume that every word in the image is segmented in the preprocessing, while others require only line-segmentation or sometimes do not require any segmentation. While some studies developed methods that target a particular language, other studies have been oriented to treat arbitrary languages. Some studies required substantial prior learning to make word recognition possible, while others carried out a spotting task to avoid prior learning, which is sometimes problematic with a small corpus of historical documents.

It seems that having such a wide array of word spotting methods, in terms of the variety of procedures and targets, has prevented a common evaluation criterion from being established. Among the substantial contributions presented above, it appears that no widely-used evaluation criteria or datasets exist. Instead, each author uses his own material manuscript and provides his own evaluation. For example, Kołcz *et al.* [2] used an "Archive of the Indies" as their material manuscript and depicted Receiver Operating Characteristics (ROC) curve for the evaluation. Marti and Bunke [3] evaluated their word recognition method for English manuscripts by using recognition rate under the first ranked choice and the top ten ranked choices. Lavrenko *et al.* [4] developed a holistic (character-segmentation-free) word recognition method and evaluated it by using word error rate (WER). Marinai *et al.* [5] evaluated their word spotting method by counting relevant retrieval results that were ranked higher than the first false positive result. In [6] and [7] Rath and Manmatha developed a profile feature-based word spotting method for historical handwritten English manuscripts under the assumption that complete word-segmentation is available, and evaluated their method using mean average precision scores produced by the popular trec_eval program. Gatos *et al.* [8] also developed word spotting method under the assumption that complete word-segmentation is available. They targeted historical typewritten Greek documents and depicted a recall-precision curve to evaluate their method. Terasawa *et al.* [9, 10] de-

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.119

veloped language independent word spotting method under the assumption that only line-segmentation is available, and they also used mean average precision scores and a recall-precision curve for evaluation. Leydier *et al.* [11] developed a word spotting method that does not require any segmentation, and evaluated their method using a recall-precision curve, an R-precision score and an ultimate recall rate that the system could output. Bilane *et al.* [12] proposed a promising method for Syriac manuscripts but they did not provide quantitative evaluation because of the lack of ground-truth datasets. Rodríguez and Perronnin [13] used mean average precision scores and DET (detection error tradeoff) curves where false acceptance rate vs. false rejection rate was plotted.

As enumerated above, there exist many evaluation criteria. It is presumed that each author reproduced his own code to make an evaluation. The studies above lead us to believe that having a readily available evaluation framework would be beneficial. The trec_eval tool used in TREC (Text REtrieval Conference) community is a good example. Unfortunately, the trec_eval tool cannot be applied to an arbitrary word spotting method because it requires a well-defined identifier for query/resultant words and well-defined criteria for relevancy judgment, however such definitions have not yet been introduced to word spotting studies.

In the section following, we will start with discussing some standard definitions and representations that are suitable for most word spotting methods regardless of the assumptions and settings. After we propose a word spotting protocol for interprocess communication in section 3, the prototype software will be introduced in section 4. The advantage of our framework in the view of modularization will be described in section 5. Finally, in section 6 we will conclude the paper.

2. What to define?

We aim to develop an automatic evaluation tool (AET) that is applicable to most word spotting engines. This section provides some definitions and representations.

The performance evaluation with AET is as follows. The most basic process in the evaluation is to execute a retrieval for a certain word and to determine whether the retrieval result is relevant or not. This basic process should be iterated several times and the result should be averaged to obtain a reliable evaluation result. To make these processes automatic, the ground-truth data must be prepared in advance. A retrieval trial should be executed with one occurrence of the ground-truth dataset used as a query and the rest of the ground-truth dataset used as occurrences to be retrieved. After sufficient numbers of occurrences in the ground-truth dataset have been used as a query, a performance index such as average precision scores for each query should be summarized and output as the result.

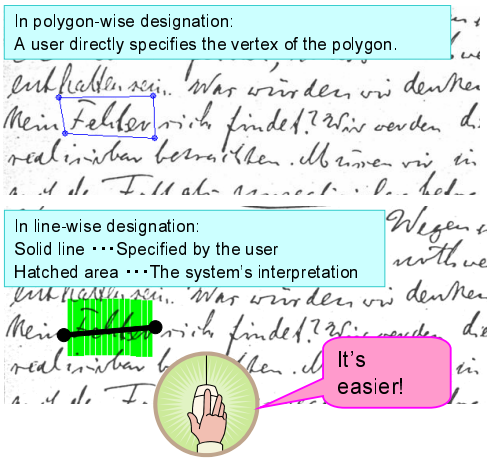

Figure 1. Line-wise designation is easier for users than polygon-wise designation

In the process above, the word spotting engine to be evaluated has to understand the query input that is sent from the AET, and has to output the retrieval result to the AET. For this usage, we defined what information should be communicated between AET and word spotting engines (Secs. 2.1–2.2). Another definition we have to determine is the standard format for the ground-truth dataset (Sec. 2.3) and the criteria for relevancy judgment (Sec. 2.4).

2.1. Query representation

How should we standardize query representation so that it can be commonly used for word spotting engines regardless of their segmentation assumptions? The simplest way of providing integers as identifiers is only valid for a word-segmentation-based method. Considering applicability to arbitrary search engines, the query should be represented using coordinates in the image.

One possible way to specify the query word is to specify the polygon as in the top of Fig. 1. However, instead of the polygon-wise designation, we propose a line-wise designation as depicted in the bottom of Fig. 1. The advantage of line-wise designation is that it provides an intuitive interface for users. Assume there is a graphic user interface (GUI) for word spotting applications. In line-wise designation, a user can designate his intended word just by dragging the mouse cursor over the word. Accepting the line-wise designation input, the search engine should translate it into a polygon representation, if needed.[1]

[1]The behavior for pathological input such as putting a line over the white region is just a detail and not the scope of this paper.

Figure 2. The example where a single line is not enough to designate an intended word

To be more exact, the query should be represented as a list of lines rather than a single line. This happens when the intended word spans two lines like the word "supermarket" in Fig. 2. Therefore, our final definition for query representation is: a list of lines.

2.2. Retrieval Result Representation

After accepting the query, the search engine should output its retrieval result. The result should be a ranked list of records, with each record specifying some region in the images along with its dissimilarity. The region specifier should be a list of polygons. The reason why a list of polygons rather than a single polygon should be used is exactly the same as the query representation: to represent a hyphenated word. In summary, our definition for retrieval result representation is: a ranked list of records, where a record is a pair of dissimilarity scores and a region specifier, where a specifier is a list of polygons.

2.3. Ground-Truth Representation

A ground-truth dataset is used for both making a query and judging the relevancy of the result. It should be a set of specifier each of which specifies the region in the image representing the same word. We determined that each entry of ground-truth should be represented in exactly the same way as a query representation, i.e., the ground-truth dataset should be a set of lists of lines.

This definition is convenient in ground-truth making for just the same reason as it is in query representation. If we intended to make a ground-truth dataset for our own material, all the preparation we need is to drag over the ground-truth occurrence in the image using some software with a GUI.

2.4. Relevancy Judgment Criteria

As mentioned above, the entry of the retrieval result is represented as the list of polygons and the entry of the ground-truth is represented as the list of lines. Here, we discuss how we should judge the relevancy of them.

Our judgment criteria are summarized in Fig. 3. If the retrieved result and ground-truth do not intersect as depicted

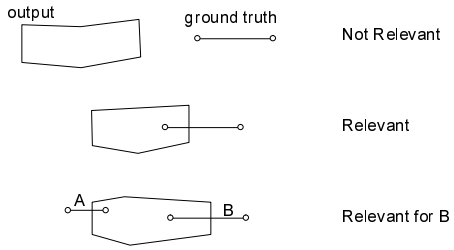

Figure 3. The relevancy judgment criteria

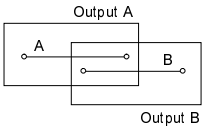

Figure 4. The case where two entries of the ground-truth overlap

in the top of the figure, the retrieval result is regarded as irrelevant. If the retrieved result and ground-truth do intersect as in the middle of the figure, the retrieval result is regarded as relevant and the entry of the ground-truth is removed from the list. This removal is required to avoid a duplicate count. A optional parameter could be set, if needed, which determines the minimum length of the intersection required to be judged relevant. The bottom of the figure represents a bit complicated case. If the retrieved result intersects with two entries of the ground-truth, it is regarded as relevant to the entry with a longer intersection. In this case, since line B has a longer intersection than line A, the retrieved result is regarded to be relevant to B. In this case B is removed from the list but A remains in the list. This is the safe definition in the complicated case involving searching for $abab$ in $xxababababxx$ as in Fig. 4. Through our method, we avoid making a false removal no matter how complicated the case.

3. Word Spotting Protocol

In the previous section we determined *what* information should be communicated. Next we have to determine a protocol for *how* the information should be communicated. The Word Spotting Protocol (WSP) is our proposal, which is publicized in our website [14]. This section provides the outline of the WSP.

3.1 Interprocess Communication

To communicate information between AET and the word spotting engine, we decided to use the standard streams.

The parameter *QueryLineList* should specify the page and the coordinates of both ends of the line, like the following example: p5x278y13x305y14.
The example above specifies the line between (278,13) and (305,14) on page 5. To represent multiple lines, simply concatenate the single line expression like the following example: p5x278y13x305y14p5x305y14x339y11

The output is a list of retrieval records (see Sec. 2.2), and each record should be represented as the following example: r1d44.6985p4x12y13x144y15x156y44x13y44
The example above means the record is ranked first, has the dissimilarity 44.6985, and specifies the quadrangular region with vertex (12,13)-(144,15)-(156,44)-(13,44) on page 4. If multiple polygons need to be described, simply concatenate the single polygon expression. The final output should be the concatenation of the record's expression.

Figure 5. Input and output format for command search

The search engine has to accept the commands listed in the next subsection from *stdin*, and has to output the result into *stdout*.

3.2 Commands

In general, the search request for information retrieval is represented as the following expression.

$$\text{Retrieve } p \text{ from the set } X. \tag{1}$$

From this expression, we have observed that the minimal commands the search engine should accept are only these two: *assign* command to specify X and *search* command to specify p. It is true that a few additional trivial commands are needed in practice. However, we will not explain them here because they are just small details. This paper only explains two main commands; for the details of other commands, refer to our website.

3.2.1 assign

A command *assign* is used to specify the dataset among which the query should be searched. It should be used with one argument in the following format:

$$\text{assign } \mathit{ListFileName} \tag{2}$$

where the parameter *ListFileName* is a string that specifies the filename where the list of the image files to be searched is saved in. Every listed image is assigned an integer as an identifier called *page* to be used in the search command.

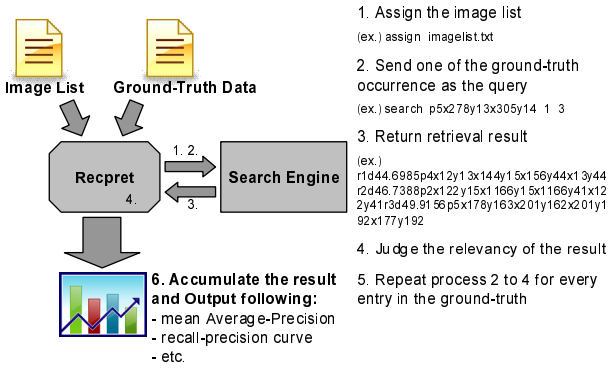

Figure 6. Communication between Recpret and the search engine

3.2.2 search

A command *search* is used to specify the query string represented by a list-of-line format (see Sec. 2.1). This command should be used with three arguments in the following format.

$$\text{search } \mathit{QueryLineList}\ \mathit{iFirst}\ \mathit{nCount} \tag{3}$$

When this command is accepted, the search engine shall start the search process and output the retrieval result with the rank from *iFirst* to $\mathit{iFirst}+\mathit{nCount}-1$.

The parameter *QueryLineList* should specify the page and the coordinates of both ends of the line like the example in Fig. 5. The format for output is also listed in Fig. 5.

4. Recpret: a prototype software

According to the definitions and protocol described in Sec. 2 and Sec. 3, we have already developed a prototype software. We named our software 'Recpret', which is short for 'Recall-Precision Evaluation Tool,'

To use Recpret, you need only prepare three files. Your search engine that follows the WSP is the first one. The second one is the list of the image files among which the query should be searched, and the last one is the ground-truth data file. Specify these three file names, and click the RUN button. That's it! The Recpret will output the evaluation of your search engine in the meantime.

The data communication between Recpret and your search engine is summarized in Fig. 6.

5. One more virtue: modularization

The WSP produces one more virtue for the word spotting application. The word spotting engine that follows the WSP is a module in the sense that it can be interchanged as a unit.

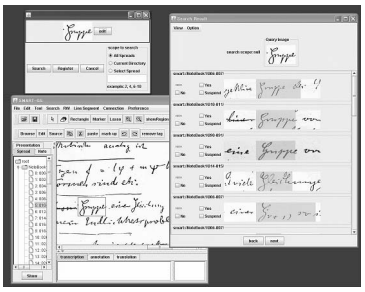

Figure 7. Screenshot of Smart-GS application

Regarding the practical use of word spotting, probably there is not an ideal method that shows the highest performance regardless of written language, the date, the degree of pollution, writing variation, and so on. Moreover, accuracy and computational cost are generally in a trade-off relationship. Such considerations bring us to this conclusion: It would be convenient for users if they could choose a word spotting engine that meets their demands, instead of having all users choose one 'ideal' engine.

The word spotting engine that follows the WSP will have opportunities to acquire more users. Suppose a developer who is developing an application that manipulates document images, and the application has various functions including but not limited to word spotting. Even if a certain word spotting engine is suitable for his application and the source code of the engine is publicized, he may give up using the code because of the language difference such as with Java, C, BASIC, etc. However, if both the application and engine follow the WSP, the engine can be easily integrated into application software irrespective of the language that is developed.

In fact we have already conformed our word spotting engine to follow the WSP, and our engine is integrated into a SMART-GS application, a tool for humanistic studies developed by Prof. Hayashi at Kyoto University (Fig. 7).

6. Conclusion

We have discussed the automatic evaluation framework for a word spotting engine. We defined the standard representation for a query, a retrieval result, and ground-truth. A criterion for relevancy judgment is also defined. In addition, we defined a protocol named WSP that specifies the interprocess communication between the parent application and the search engine. Using this framework, any word spotting engine independently developed could be evaluated by the same standard. We anticipate that our framework will facilitate more in depth word spotting research.

In addition, our framework can modularize any word spotting engine. Our framework will facilitate the utilization of word spotting due to its convenience of interchangeability, and it will facilitate the distribution of word spotting engines; thus, making the framework a common resource around the world.

References

[1] R. Manmatha, C. Han, and E. M. Riseman, "Word spotting: a new approach to indexing handwriting," Proc. CVPR'96, pp. 631–637, 1996.

[2] A. Kołcz, J. Alspector, M. Augusteijn, R. Carlson, and G. Viorel Popescu, "A line-oriented approach to word spotting in handwritten documents," Pattern Analysis and Applications, vol. 3, no. 2, pp. 153–168, 2000.

[3] U.-V. Marti and H. Bunke, "Handwritten sentence recognition," Proc. ICPR'00, pp. 3467–3470, 2000.

[4] V. Lavrenko, T. M. Rath, and R. Manmatha, "Holistic word recognition for handwritten historical documents," Proc. DIAL'04, pp. 278–287, 2004.

[5] S. Marinai, E. Marino, and G. Soda, "Indexing and retrieval of words in old documents," Proc. ICDAR2003, vol. 1, pp. 223–227, 2003.

[6] T. M. Rath and R. Manmatha, "Word image matching using dynamic time warping," Proc. CVPR'03, vol. 2, pp. 521–527, 2003.

[7] T. M. Rath and R. Manmatha, "Word spotting for historical documents," Int. J. on Document Analysis and Recognition, vol. 9, no. 2, pp. 139–152, 2007.

[8] B. Gatos, T. Konidaris, K. Ntzios, I. Pratikakis, and S. Perantonis, "A segmentation-free approach for keyword search in historical typewritten documents," Proc. ICDAR2005, vol. 1, pp. 54–58, 2005.

[9] K. Terasawa, T. Nagasaki, and T. Kawashima, "Eigenspace method for text retrieval in historical document images," Proc. ICDAR2005, vol. 1, pp. 437–441, 2005.

[10] K. Terasawa and Y. Tanaka, "Locality sensitive pseudo-code for document images," Proc. ICDAR2007, vol. 1, pp. 73–77, 2007.

[11] Y. Leydier, F. Lebourgeois, and H. Emptoz, "Text search for medieval manuscript images," Pattern Recognition, vol. 40, no. 12, pp. 3552–3567, 2007.

[12] P. Bilane, S. Bres, and H. Emptoz, "Local orientation extraction for wordspotting in Syriac manuscripts," Proc. ICISP 2008, LNCS 5099, pp. 481–489, 2008.

[13] J. A. Rodríguez and F. Perronnin, "Local gradient histogram features for word spotting in unconstrained handwritten documents," Proc. ICFHR2008, 2008.

[14] Kengo Terasawa's Word Spotting Home Page, http://km.meme.hokudai.ac.jp/people/terasawa/WS/