

MayoBMI at ImageCLEF 2016 Handwritten Document Retrieval Task

Sijia Liu, Yanshan Wang, Saeed Mehrabi, Dingcheng Li, and Hongfang Liu

Section of Biomedical Informatics, Mayo Clinic, Rochester MN 55905, USA
{liu.sijia, wang.yanshan, mehrabi.saeed, li.dingcheng,
liu.hongfang}@mayo.edu,
<http://www.mayo.edu/research>

Abstract. In this working note, we introduce our participation at the ImageCLEF 2016 Handwritten Document Retrieval Task. We mainly focused on hyphenation detection using line images and information retrieval using n-best results. The hyphenation detection step utilizes extracted image features from beginning and end of a line and a binary classifier to determine if a line contains hyphenation. Then the spell correction step is used to eliminate spelling errors from the concatenation of a broken word from the end of a line and the beginning of the next line. The final text retrieval step employs a suffix stripping algorithm to normalize the word tense and form and TF-IDF scheme to rank the retrieved relevant segment results of our submission.

Keywords: handwriting recognition, hyphenation detection, text retrieval

1 Introduction

For the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task [1,2], our aim is to develop a document retrieval system to retrieve the relevant segments and word bounding boxes for given string of test queries. An intuitive solution for this task generally includes three components: handwritten text recognition, keyword spotting and text retrieval. To obtain relatively accurate transcripts from document images, image pre-processing methods such as image binarization and text line extraction are generally used [3,4]. Based on whether trying to generate transcripts from the handwritten text images as an intermediate step, there are mainly two categories of solutions: recognition based approaches and keyword spotting based approaches. For the recognition based approaches, there are two kinds of models commonly used in the state-of-art handwritten recognition systems for historical documents: Recurrent Neural Network (RNN) with Connectionist Temporal Classification (CTC) [5,6] and Hidden Markov Model (HMM) [7,8,9]. Both of these models can achieve high recognition accuracy, which is measured in word and character error rates. For keyword based approaches, depending on whether the query is a word image or a string in the dataset, systems can either query the keyword by comparing

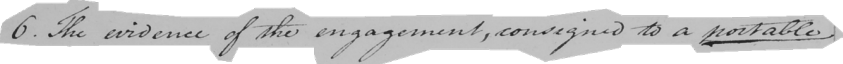
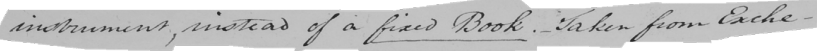
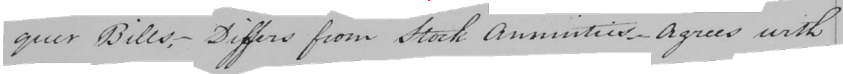
- (a) 
 Ground Truth: 6. The evidence of the engagement, consigned to a portable
 1-best: 6 . The evidence of the engagement consigned to a /*missing*/
- (b) 
 Ground Truth: instrument, instead of a fixed Book. Taken from Exche-
 1-best: instrument , instead of a by Each Silver from The
- (c) 
 Ground Truth: quer Bills,- Differs from Stock Annuities- agrees with
 1-best: our Bills differs from Stock Annuities degrees with=-

Fig. 1: Three line examples of errors in 1-best predictions of development set. The green texts are the ground truth, and the red texts are the recognition errors. In (a), the ending word “portable” is missing in 1-best. In line (b), the prediction of text “fixed Book. Taken” is mistakenly recognized as “by Each Silver” and then ending word by hyphenation “Exche-” is also mistakenly recognized as “The”. In line (c), the recognition system mistakenly added an ending hyphenation “=-” to the line.

the features of query images with the indexed features of existing images in the dataset or transcribe the document before retrieving the query string. Language models like n-gram and HMM [10] can also be used to improve the overall keyword spotting performance.

Although the handwritten document recognition has attracted much attention in the areas of image processing and machine learning, the impact of hyphenated words is not well studied. Hyphenated words, also known as broken words, are the words supposed to appear at the end of a text line while broken and continued at the beginning of the next line because of the manuscript writer’s intention to save space. Hyphens will be used to mark such broken words at the end of line by “-” or “=” and at beginning of a line by “:”. Hyphenations may cause recognition errors if they are not processed properly. Example line images, recognition errors in 1-best transcripts and the corresponding ground truth from the task dataset are shown in Fig. 1. We are interested in this working note to describe a solution to detect words with hyphenation in segmented line images, which can further be used to improve the retrieval result of given strings of queries.

The rest of this working note is organized as follows. First, we introduce our proposed methods in Section 2, including steps of image preprocessing of line segmented image, hyphenation detection, spell correction using training transcript and text retrieval. Second, the evaluation of the effectiveness of

hyphenation detection methods and the official task evaluation of our submitted run are discussed in Section 3. Finally, we conclude our work and proposed some future works and improvements.

2 Proposed Methods

In our proposed solution to ImageCLEF 2016 Handwritten Document Retrieval Task, we mainly focused on how to detect the hyphenated words, how to do spell correction for detected hyphenated words and how to retrieve the relevant segments based on 1-best results. In this section, we will elaborate the details of each step. We first describe the preprocessing step of how the segmented grayscale line images is normalized into fixed-height binary images (Section 2.1). Then using normalized line images, a hyphenation detection methods is proposed (Section 2.2) to detect lines with hyphenation, followed by spell correction (Section 2.3) and text retrieval step (Section 2.4).

2.1 Preprocessing

In this step, our goal for the preprocessing step is to obtain noise and slant free binary line images. In some related works, skew correction may also be necessary before the recognition or word spotting step. However, in the task dataset, lines are well segmented and with only negligible slope, which ease us from skew correction. The slant of written lines are removed by applying a two dimensional affine transformation to the original line images. The transformation matrix is arbitrarily chosen based on the observation from random selected line images in the training set. Afterwards, a global threshold is applied to the slant corrected grayscale line images to generate the binary line images. We also resize the line images to a fixed height of 30, and the width is scaled proportionally.

2.2 Hyphenation Detection

In order to detect lines with beginning and ending hyphenations, image windows at both the beginning and end of each line image are obtained. Several image features are then extracted from the image windows, and various binary classifiers are used to detect hyphenations. As a binary classification problem, according to the writing style of the document writer, lines containing both the end hyphens and beginning hyphens in the next line are considered lines with hyphenation. Lines with such hyphenations are labeled as positive, while the others are labeled as negative. This strict labeling rule is helpful to eliminate false positives in the prediction results.

Several binary classification methods are used on the image features in the training set. To evaluate the performance of these methods, precision, recall and F-score are tested as metrics in the development set. In this task, the ground truth transcripts of both training and development set are provided, thus the

ground truth labels can be obtained and used to compare with the hyphenation detection results.

To represent the binary line image windows as feature vectors, a set of local features are extracted from the beginning windows and ending windows for each line. These features have been used in previous works on keyword spotting approaches [10]. There are 8 features of the image window describing whether they contain hyphens. They are: the horizontal and vertical locations of the first non-background pixel in the window, the horizontal and vertical locations of the last non-background pixel in the window, the average intensity, the second order moment and the coordinates of the window centroid. Further, the window is cropped with the tight rectangular bounding box by removing all the lines with no non-background pixel on the boundary. Then the average and second order moment of intensity and the window centroid are recalculated and combined with previous features. Besides, the number of non-background pixels are summed up by each line and column, which generates pixel histograms horizontally and vertically. In this work, we use a window with width of 20 and height of 30. Therefore, a feature vector with $8 + 4 + 20 + 30 = 62$ dimensions are extracted to represent the window. For each line, both the ending window of the current line and the beginning window of the next line are used to determine whether the line contains hyphenation or not. Thus, the dimension of feature vector of each line is doubled, resulting in a feature vector of $2 \times 62 = 124$ dimensions for each line. The feature vector is then used as the input of various machine learning methods.

The classifiers investigated are implemented in Scikit-Learn [11]. The classifiers which are tested on development set are: 5 Nearest Neighbors, Decision Tree, Random Forest, AdaBoost, Naive Bayes with Gaussian kernel. Among these classifiers, our experiment in development set suggested that AdaBoost can provide the best prediction.

2.3 Spell Correction

Once the hyphenation of a certain line is detected, the last word of the current line and the first word of the next line are concatenated. The text from the two word windows are extracted from the 1-best result, and all the special characters are removed before concatenation.

From the 1-best documents in both the training and development dataset, the accuracy of recognition without hyphen is relatively high. However, for hyphenated words, word spotting algorithms tend to predict the given word image as the most similar complete word, instead of considering it as only the front part or the back part of a word. For example, the word “testimony”, if broken as “testi-mony” into the end and beginning of two lines, the latter part is more likely to be predicted as ”many” which is a complete word, instead of ”mony” which is a suffix in ground truth. Fig. 1 also shows some of such examples.

To correct the recognition errors caused by hyphenation, a spell correction step is applied after the concatenation of two broken words. The ground truth

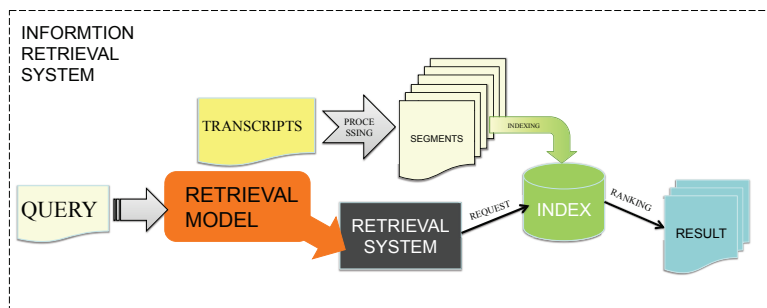


Fig. 2: Information Retrieval System Overview

transcript from the training set is used to generate a dictionary of the dataset. All the words in training and development set are converted to lower case before put into the dictionary. A spell corrector is then used. The spell corrector considers the corrections of edit distance no larger than 2 from the original word. The corrections with maximum likelihood to the current prediction in the dictionary is chosen. If no matched correction under these criteria is found, the original word is remained unchanged. The implementation of the spell corrector can be found in [12].

2.4 Text Retrieval

Figure 2 shows the overview of information retrieval system that we utilize to retrieve the relevant segments. The system basically consists of two components: indexing and retrieval. In the indexing component, we annotate the textual documents and then create the segments for indexing. In the retrieval component, the queries are used as input to the retrieval model, and then the relevant segments are retrieved from the indexing database. In the following we will describe each component in detail.

The textual document are preprocessed before indexing. In the submitted run, we utilized the 1-best transcript document as the input for the retrieval system. According to the definition of segments in this task, 6 consecutive lines in the document are concatenated to create the segments. Specifically, the segments are defined by a sliding window with the size of 6 that moves one line at a time until the bottom of the sliding window is reached at the end of the document. Prior to creating the segments, we add the bounding boxes information into the 1-best transcript documents. More precisely, words are separated in each line, and subsequently attached the line number and bounding boxes to each word as the prefix and suffix, respectively. The format of each annotated word is defined as:

$$L_T_W_H_X_Y, \quad (\text{Format 1})$$

where L is the line number, T represents the word, W and H are the width and height of the bounding box, and X and Y are the left-top coordinate of the box, respectively.

We used the software package Elasticsearch¹ to index the segments. A total of 16939 segments were indexed into three fields: “ID”, “contents” and “annotations”. In the “ID” field, the ID of the first line in each segment was used to distinguish the segments. In the “contents” field, we only indexed the words, i.e., W in format 1, in the segments. We applied the following hyphen-rule to take the hyphenated words into consideration:

If line i ends with any of =, -, and :, and line $i + 1$ starts with either - or :, then the last term in line i and the first term in line $i + 1$ are considered a hyphenated word.

(Rule 1)

The broken words are concatenated together after removing the hyphens. The spell corrector described in previous section is then used. Before indexing, we also applied the Porter stemmer [13] to the “contents” field. By doing so, the words like “abuses”, “abusing” and “abused” were also retrieved given the query “abuse”. In the “annotations” field if rule 1 was found, the corresponding annotation terms represented in Format 1 were connected by “=:”. Therefore, for each segment the number of words in the “contents” field is equivalent to the number of annotation terms in the “annotations” field.

In the retrieval component, we utilize the Term Frequency - Inverse Document Frequency (TF-IDF) scheme [14] to rank the segments in the “contents” field and retrieved the top 30 segments. According to the positions of the matched word in the “contents” field, the corresponding annotation terms are retrieved from the “annotations” field. Using the annotation terms, the bounding boxes as well as the queries are created as the final submission.

3 Experiments and Evaluation

In this section, we elaborate the details of experiments and the performance of hyphenation detection and ImageCLEF 2016 Handwritten Scanned Document Retrieval Task evaluation.

The task dataset is a subset of scanned and manually transcribed manuscripts written by Jeremy Bentham under the Transcribe Bentham project [2,15]. The task dataset consists of three subsets: training, development and test set. The training and development set contains 9645 and 10589 manually segmented line images, respectively. The test set contains the 10589 line images in the development set and 6355 line images exclusive in test set, resulting in a total of 16944 lines.

For the hyphenation detection step, the ground truth are extracted from transcripts of training and development set. As expected, the dataset is

¹ <https://www.elastic.co/products/elasticsearch>

significantly unbalanced. In the training set, there are only 810 positive samples in 9645 lines, the proportion of positive samples is 8.4%. In development set, there are only 853 positive samples in 10589 total samples. The percentage of positive samples is 8.0%. The trained models from training set are used for the classification of the development set. The precision, recall and F-score metrics of the development set is shown in Table 1. From the experiment results we can observe that AdaBoost is the best performed classifier for hyphenation detection in the proposed feature set. The detection algorithm does not perform well in development set, thus we do not include it into the final submission. The spell corrector is still used in text retrieval step to handle the hyphens already in the 1-best results.

Table 1: Hyphenation detection results

	Precision	Recall	F-score
Nearest Neighbor	0.514	0.237	0.325
Decision Tree	0.545	0.225	0.319
Random Forest	0.340	0.412	0.373
AdaBoost	0.301	0.650	0.411
Naive Bayes	0.477	0.263	0.339

We submitted one run as the task submission, and the official evaluation results are shown in Table 2. We noticed there is a significant drop from the results of the development set to those of the test set. The similar performance decreases can be also found in the results of the baseline system, which uses exact string matching and should be robust among different datasets if these datasets are of similar characteristics and quality. The reason for this significant performance difference is because the test set is considerably more difficult than the development set, where the bounding box is much less accurate and the images quality is lower.

Table 2: ImageCLEF Handwritten Scanned Document Retrieval Task evaluation results in percentage (%)

	Development		Test	
	Segment	Box	Segment	Box
gAP	25.76	18.40	2.53	1.02
mAP	23.41	18.37	2.85	1.67
gNDCG	33.05	25.71	6.96	3.42
mNDCG	26.56	22.24	3.62	2.48

4 Conclusion

We discussed our participation of ImageCLEF 2016 Handwritten Scanned Document Retrieval Task. Our submission run is based on a three step process, hyphenation detection, spell correction and information retrieval. The hyphenation detection step utilizes extracted image features from beginning and end of lines and a binary classifiers to determine if a line contains hyphenation or not. Then the spell correction step is used to eliminate spelling errors the concatenation of a broken word from a line ending and the line beginning of the next line. The spell correction step uses only the vocabulary from the transcript of training set. A final information retrieval step employs a suffix stripping algorithm to normalize the word tense and form and TF-IDF scheme to rank the retrieved 30 segments as the output of our system.

There are several future works that can be investigated to improve the performance of our system. First, more line image features can be considered as the input of supervised binary classification methods. Second, larger lexicon can be utilized in both the hyphenation detection step and spell correction step. Due to the task restriction, only the vocabulary in the training set can be used, and the use of external data for learning a language model is prohibited. A larger lexicon of the whole dataset or external data rather than only the training set will improve the effectiveness of the spell corrector.

5 Acknowledgement

The authors gratefully acknowledge the support from the National Library of Medicine (NLM) grant R01LM11934.

References

1. Villegas, M., Müller, H., García Seco de Herrera, A., Schaer, R., Bromuri, S., Gilbert, A., Piras, L., Wang, J., Yan, F., Ramisa, A., Dellandrea, E., Gaizauskas, R., Mikolajczyk, K., Puigcerver, J., Toselli, A.H., Sánchez, J.A., Vidal, E.: General Overview of ImageCLEF at the CLEF 2016 Labs. Lecture Notes in Computer Science. Springer International Publishing (2016)
2. Villegas, M., Puigcerver, J., Toselli, A.H., Sánchez, J.A., Vidal, E.: Overview of the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task. In: CLEF2016 Working Notes. CEUR Workshop Proceedings, Évora, Portugal, CEUR-WS.org (Sep 2016)
3. Saabni, R., Asi, A., El-Sana, J.: Text line extraction for historical document images. Pattern Recognition Letters **35** (2014) 23 – 33 Frontiers in Handwriting Processing.
4. Shi, Z., Setlur, S., Govindaraju, V.: Text extraction from gray scale historical document images using adaptive local connectivity map. In: Proc. Eighth Int. Conf. Document Analysis and Recognition. ICDAR 05, Washington, DC, USA, IEEE Computer Society (August 2005) 794–798 Vol. 2
5. Graves, A., Schmidhuber, J.: Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. Advances in Neural Information Processing Systems 21, NIPS'21 (2008) 545–552

6. Strauß, T., Grüning, T., Leifert, G., Labahn, R.: Citlab ARGUS for historical handwritten documents. *CoRR* **abs/1412.3949** (2014)
7. Fischer, A., Keller, A., Frinken, V., Bunke, H.: Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters* **33**(7) (2012) 934–942
8. Almazán, J., Gordo, A., Fornés, A., Valveny, E.: Efficient Exemplar Word Spotting. *Proceedings of the British Machine Vision Conference 2012* (2012) 67.1—67.11
9. Rodríguez-Serrano, J.A., Perronnin, F.: Handwritten word-spotting using hidden Markov models and universal vocabularies. *Pattern Recognition* **42**(9) (2009) 2106–2116
10. Martí, U.V., Bunke, H.: Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence* **15**(01) (2001) 65–90
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12** (2011) 2825–2830
12. Norvig, P.: How to write a spelling corrector. <http://norvig.com/spell-correct.html>
13. Porter, M.F.: An algorithm for suffix stripping. *Program* **14**(3) (1980) 130–137
14. Salton, G., McGill, M.J.: *Introduction to modern information retrieval*. McGraw-Hill, Inc. (1986)
15. Tim Causer, V.W.: Building a volunteer community: Results and findings from transcribe bentham. *Digital Humanities Quarterly* **6**(2) (2012)