

Crossing the lines: making optimal use of context in line-based Handwritten Text Recognition

J.Tanha, J.D.Does, K.Depuydt

Institute for Dutch Lexicology (INL)

Matthias de Vrieshof 3, 2311 BZ Leiden, The Netherlands

Email: {jafar.tanha, jesse.dedoes, katrien.depuydt}@inl.nl

J.A.Sánchez

Universitat Politècnica de València

Cam de Vera, s/n 46022 València, Spain

Email: jandreu@prhlt.upv.es

Abstract—Hand-written text recognition (HTR) is often carried out line-by-line: the decoding of text lines is carried out independently. This approach is known to deteriorate recognition accuracy of words and characters close to the line boundaries. The present study investigates this issue from the point of view of the language modeling component of the HTR system. Obviously, lack of linguistic context may be one of the reasons for loss of accuracy, but it certainly is not the only factor in play. We seek to clarify to which extent the problem can be influenced by the language modeling component of the system. We first discuss how to develop adapted language models which significantly improve HTR performance in general. We then focus on the deployment of methods to improve accuracy at line boundaries. The final result is an efficient approach which significantly improves HTR accuracy without changing the basic HTR system setup.

Keywords—Domain adaptation; Higher order N-gram model; Hand-written text recognition.

I. INTRODUCTION

The current hidden markov model (HMM-based) approaches to handwritten text recognition (HTR) typically utilize a statistical language model during the decoding process [1], [2]. The statistical language models are basically used to guide the decoding step by ranking and constraining the possible word sequence hypotheses. Language models are usually constructed from large text corpora which – ideally – are *in-domain*, linguistically close to the language of the document collection which is being processed. However, in HTR for historical documents, which is the main goal of this research, obtaining effective models is much less straightforward: models built from the in-domain data are generally unsatisfactory because not enough data can be obtained to avoid overfitting. Therefore, one can use *out-of-domain* data to improve the language model, but in order to exploit the larger pool of out-of-domain data one has to surmount two difficulties: (1) indiscriminate use of out-of-domain data may not benefit, in fact even deteriorate system performance and (2) the use of the complete out-domain data for training may increase the complexity of the system, making the decoding step almost untractable [3], [2].

The above-mentioned issues are typically dealt with by using *domain adaptation* or *language model adaptation* techniques [4][5][3][2]. In this paper we use a language model adaptation approach proposed in [2], which employs a semi-supervised learning method [6] to handel language model adaptation problem for HTR.

Another issue of language models in the HTR systems is the use of a higher N-gram language model. Current HTR systems mainly employ Unigram or Bigram language models [1] [2] in the decoding process of the recognition, which is not unreasonable, because using higher order language models in HTR may lead to complex system and increases substantially the computational cost of the decoding process [7] [1]. However, it is obvious that trigram or higher language models will give better recognition performance. To deal with this problem, we propose a new approach to use a higher N-gram model in HTR without any further computational cost.

In our experiments, we use the *transScriptorium* HTR engine described in [8] on a set of digitised images of manuscripts written by the 18th and early 19th-century British philosopher Jeremy Bentham. We use available *Bentham* transcriptions as in-domain data and the public part of the *Eighteenth Century Collections Online*¹ corpus as out-of-domain data. Analysing the results of the HTR experiments on the *Bentham*² benchmark corpus shows more word error rate near the line boundaries (cf. Fig. 2). From the point of view of language modeling, this issue might be due to: (1) missing context information for words near the line break, (2) language model overfitting of bigram with line starting symbol (<S>) and ending symbol (</S>), or (3) hyphenated forms of words. We propose two paragraph-based approaches to deal with the issues: (1) using a paragraph-based language model, and (2) using a word graph-based approach applying a paragraph-level decoding algorithm, in which scoring recognition hypotheses from different lines is no longer independent. Our experimental results on the Bentham dataset show the performance of the proposed methods.

The rest of the paper is organised as follows. Section II gives an overview about hidden markov model and language modeling. In Section III the used methods for improving language model is addressed. Section IV introduces the paragraph-based approaches and Section V presents the experimental setup. Section VI addresses the results and Section VII concludes the paper.

II. HIDDEN MARKOV MODELS AND LANGUAGE MODELS IN TEXT RECOGNITION

In this section, we first address the problem definition and then state the role of language models in HTR.

¹<http://www.textcreationpartnership.org/tcp-ecco/>

²Images and transcriptions have been produced in the *Transcribe Bentham* project [9], <http://www.ucl.ac.uk/transcribe-bentham>

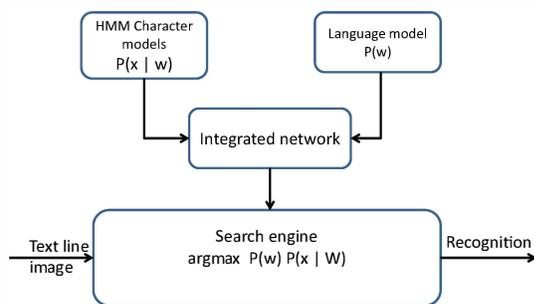


Fig. 1: An overview of the integrated architecture for text line image decoding.

A. Problem definition

The line boundary problems in line-based HTR are non-trivial and may lead to significant loss of accuracy. In principle, it is possible to try circumvent the issue by concatenating line images, feature files, word graphs or N-best lists, but these approaches have their own issues:

- Some toolkits, like HTK, are not able to decode too large “lines” efficiently.
- Concatenating the line images is not a trivial task: it is not obvious how they should be glued together (consider for instance skewed lines), how the interline should be modeled, and how white space should be introduced in the concatenation from the point of view of the optical models.
- In word graph concatenation, combination of optical model weights and LM weights is not trivial. Neither is the treatment of hyphenated words obvious.

We therefore address the following task: how to arrive at an optimal application of language modeling techniques to improve HTR accuracy in general, but more particularly at the line boundary, without changing the underlying HTR system?

B. Hidden Markov Models (HMMs)

In a handwritten text line recognizer the goal is to recognize the most likely word sequence, $W = (w_1, \dots, w_m)$, for a known observation sequence of features extracted from the document image, $X = (X_1, \dots, X_l)$, as follows:

$$\hat{W} = \arg \max_w P(w|X) \quad (1)$$

For simplicity, the resulting hidden markov model (HMM) is reformulated using the Bayes rule as:

$$\hat{W} = \arg \max_w P(X|w) \times P(w) \quad (2)$$

The HMM-based recognizer used in this paper is supported by a statistical language model in the decoding step. Fig. 1 shows the used architecture in the HTR system. The current HMM-based approaches to HTR systems typically utilize a statistical unigram [10] or bigram language model [2], [1] during the decoding process. The main reason is that trigram or higher order models only make sense if they are based on a substantial text corpus, and incorporation of comprehensive language models may substantially increase the computational

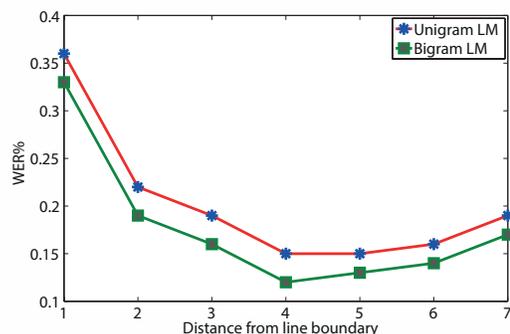


Fig. 2: Word error rate from line boundaries using Unigram and Bigram LM.

cost. However, it is to be expected that trigram or higher language models will give better recognition performance. Especially in our context, where we have extracted useful information from out-of-domain data, much will be lost if we cannot apply more sophisticated models. As a result, we study methods for improving the underlying language model in the HTR system and also to show the impact of the improved language models to reduce the error rate in the line boundaries.

III. IMPROVING THE UNDERLYING LANGUAGE MODEL IN HTR

We start by training two language models LM_i ($i=0,1$) on \mathcal{B}_0 and \mathcal{B}_1 where \mathcal{B}_0 and \mathcal{B}_1 arise from a partition of a corpus \mathcal{B} . In the context of handwritten text recognition, \mathcal{B}_0 could for instance be the HTR training set or some other portion of a transcribed corpus, and \mathcal{B}_1 is the rest of the text. We then use \mathcal{E} corpus as a general large out-of-domain corpus. Our goal here is to find an informative subset \mathcal{E}_1 of resources from the \mathcal{E} corpus, which is relevant to the \mathcal{B}_0 collection, and to exploit this for domain adaptation. The adapted language model can be obtained as follows:

$$P(W) = \lambda_{\mathcal{B}_0} P_{\mathcal{B}_0}(W) + \lambda_{\mathcal{B}_1} P_{\mathcal{B}_1}(W) + \lambda_{\mathcal{E}_1} P_{\mathcal{E}_1}(W) \quad (3)$$

where $\lambda_{\mathcal{B}_0}$, $\lambda_{\mathcal{B}_1}$, and $\lambda_{\mathcal{E}_1}$ are the interpolation weights and W is a word sequence in the test set. We use *SRILM* toolkit [11] to find the optimal values for the coefficients of equation (3).

In (3) the third term is the resulting language model from the \mathcal{E} corpus which is obtained by means of an intelligent sample selection approach [2] for selecting the relevant subset, as addressed in the following sections.

A. Adapted language modeling with the Disagree-Co algorithm

Language models for specific tasks are typically constructed from large in-domain corpora. However, in practice for historical data there is not enough in-domain data available to build a proper language model for HTR. Consequently, we use a new approach to domain adaptation which selects a subset of informative samples using a semi-supervised co-training approach [12]. Co-training is one of the widely used semi-supervised learning methods [6] in practical domains. In co-training, two classifiers based on two views of data are trained and then unlabeled data are classified by the classifiers. Unlabeled data that are labeled with high confidence by one classifier are used as training data for the other.

In order to be able to use the co-training framework for domain adaptation, we need to exploit a set of *in-domain* resources \mathcal{B} and a set of *out-of-domain* resources \mathcal{E} . Without loss of generality, we assume a partitioning of the in-domain data \mathcal{B} in two subsets \mathcal{B}_0 and \mathcal{B}_1 such that $|\mathcal{B}_0| < |\mathcal{B}_1| \ll |\mathcal{E}|$.

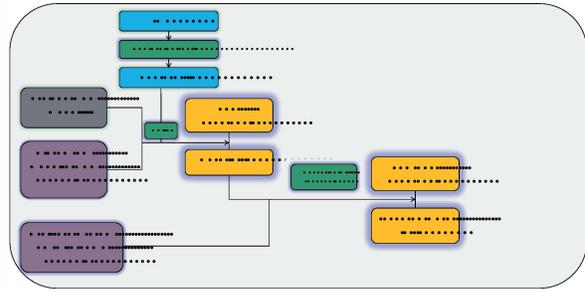
We introduce the Disagree-Co algorithm [2], which gradually exploits a set of informative data from the out-of-domain data, using a disagreement-based co-training approach [13]. We start by training two language models LM_i ($i=0,1$) on \mathcal{B}_0 and \mathcal{B}_1 . We consider these two language models as classifiers in the co-training framework. We then apply the trained models on the \mathcal{E} collection and evaluate and rank the resources by means of a scoring criterion, which is a function of the perplexity and number of Out-Of-Vocabulary (OOVs). The used algorithm then selects an informative subset \mathcal{E}_1 of high-confidence resources from the \mathcal{E} collection for each language model. Next, Disagree-Co adds to the training material of the second model a set of resources which are in the high-confidence set for the first model, but not in the high-confidence set of the second model, and vice versa. After this, the training process is repeated until the stopping condition is reached. At the end, the algorithm gives a subset of high-informative resources to in-domain data which we use for language model adaption in equation 1.

B. Higher-order language models

Another important issue in language modeling for HTR systems is the use of higher order language models, which may substantially increase the computational cost of the decoding process of HTR system [7], [1]. Current HTR systems often rely on small-scale *Unigram* or *Bigram* language models derived from the HTR training set [2], [1]. The use of bigram language models is not unreasonable in this situation. However, in order to exploit the information in the out-of-domain data to its full potential and improve the recognition performance in line boundaries, more advanced language models are vital. To deal with this problem, we use a recognition lattice rescoring approach, which on the one hand enables the HTR system to benefit the use of higher order language models and on the other hand reduces the computational costs of the decoding process in HTR.

We first introduce recognition lattices (word graphs). Recognition lattices are produced during the decoding process by the HTR recognizer. A recognition lattice (word graph) is a data structure that represents different hypotheses of a hand-written text recogniser in a compact way as a finite state network. The lattice typically represents the most promising subspace of recognition results produced by the decoding process. The edges of word graph are labeled with a word hypothesis, a score provided by the HMM recognizer, and a score provided by the language model (namely Bigram LM). Our goal here is to rescore the language model scores provided by Bigram models using *higher N-gram* models.

Using this idea can substantially decrease the computational cost of the decoding process in HTR; the re-scoring operation is much faster than the full decoding process. We use this idea and propose an algorithm to handle the problem. Fig. 3 gives an overview of the proposed method to deploy the power of higher-order models in an efficient way. We first train



the Hand-written Text Recognizer. It then generates the n-best hypotheses as word lattices using a bigram LM. In the meantime the N-gram ($N > 2$) language model is generated. Next, this language model is used to re-score the word lattices. The re-scored recognition lattices are then applied to evaluate the performance of HTR. We use *lattice-tool* of the *SRILM* toolkit to implement the rescoring algorithm and tune the related parameters.

IV. DEALING WITH LINE BOUNDARIES ISSUES

We first address a paragraph-based language modeling approach. We then propose a paragraph level decoding to handle the line boundary problem.

A. Paragraph-Based language model

As shown in Fig. 2, the word error rate at the line boundaries is significantly higher than in other positions. To deal with this issue, one could concatenate lines and use a completely paragraph-based segmentation in conjunction with a paragraph-based language modeling. However, as discussed in section II-A, concatenating lines is not without problems, while a paragraph-based language model is easily obtained. We consequently apply paragraph-based higher order language models in conjunction with the proposed rescoring approach.

B. Paragraph-level decoding approach

In this section, we propose an algorithm to decode a sequence of line-based word lattices using a language model. The goal here is to find a way to exploit context from the previous and next line to improve the recognition performance at line boundaries.

As is well known, decoding of a lattice \mathcal{L} requires a forward and a backward pass to obtain the best hypothesis. The forward pass produces graph $G(\mathcal{L})$ which has, for each relevant position p in a text line, a list $L_p = \{v_i\}$ of scored vertices labeled with word hypotheses w_i . Each vertex v_i has a backward edge to an earlier vertex v_j belonging to a list L_q , where q precedes p , in such a way that some optimal path to v_i passes through v_j . From this graph, an optimal path is found in the backward pass, by tracing back a path from the best-scoring node in the list of nodes L_{p_f} corresponding to the final line position.

We can decode a sequence of lattices $\{\mathcal{L}_i\}$ by interconnecting the corresponding $\{G(\mathcal{L}_i)\}$, cf. algorithm 1.

Algorithm 1 Paragraph-level decoding

```

- let  $\text{forward}(\mathcal{L}, LM) :=$  produce the intermediate graph  $G(\mathcal{L})$  by forward
  evaluation using  $LM$ 
- let  $\text{forward}(\mathcal{L}, LM, G(\mathcal{L}_{i-1})) :=$  produce the intermediate graph  $G(\mathcal{L})$ 
  by forward evaluation using  $LM$ , but initialize from the final vertex
  list of the predecoding of the previous line, connecting the initial
  vertices of  $G(\mathcal{L}_i)$  to the final vertices of  $G(\mathcal{L}_{i-1})$ 
Main loop:
for each line  $l_i$  in page
  obtain a word lattice  $\mathcal{L}_i$  for this line by HTR
  for each paragraph  $P_j$  in page
    let  $L_j :=$  the list of line lattices  $\mathcal{L}_i : l_i$  is a line in  $P_j$ 
    for each  $\mathcal{L}_i, i < N$ :
      if  $(i = 0) : \text{forward}(\mathcal{L}, LM)$ 
      else:  $\text{forward}(\mathcal{L}, LM, G(\mathcal{L}_{i-1}))$ 
    finally: trace back from the final vertices of  $G(\mathcal{L}_N)$  through all previous
      predecoded lattices to obtain the best paragraph decoding

```

V. EXPERIMENTS

In this section we perform several experiments on linguistic resources to show the effect of the proposed methods for improving the language model on the performance of the HTR system.

A. Dataset

We make use of the TRANSCRIPTORIUM ([8]) Bentham HTR competition dataset for the evaluation of HTR performance.

This collection consists of a set of images and with ground truth transcriptions of “Transcribe Bentham” manuscripts [9]. The dataset for the competition³ is composed of 433 pages; most of the pages consist of a single block with many difficulties for line detection and extraction (see page samples below). The dataset is divided for the competition in three different parts: training, validation, and test. The training part consists of about 9,200 lines, whereas the validation partition is about 1,400 lines. The training part of the set, and the remaining transcriptions from “Transcribe Bentham” are used as in-domain LM training material. We use the public part of the ECCO (Eighteenth Century Collections Online), about 70m words, as out-of-domain LM training data.

B. Experimental Setup

We first perform the baseline experiments. We then apply the addressed methods for improving language models. We set up three sets of experiments: line-based and paragraph-based N-gram language models, and paragraph-level decoding approach.

VI. RESULTS

We compare the baseline methods to the proposed approaches on the Bentham collection. We have considered four main evaluation criteria for each experiment, the general word error rate (WER), the word error rate without taking the first/last word of each line into account (WER-WOF and WER-WOL), and the word error rate without both first and last words (WER-WOFL). In the experiments, we improve the underlying language model of the transcriptorium HTR engine [8] in two ways: (1) we first apply the Disagree-Co algorithm for domain adaptation [2] on ECCO dataset as out-of-domain data, and

(2) we then build a higher N-gram language model from the resulting resources of Disagree-Co and evaluate them. Next, we apply the proposed paragraph-based approaches. Tables I-III report the related results. In each table the best result has been boldfaced.

A. Line-based language modeling

In this experiment, we build the language model based on the lines in the original resources. Different order language models are constructed in this experiment to show the effect of higher order language models on the performance of the HTR system. Table I shows the results. As shown, using adapted and higher order language model improves significantly the performance. The best result has been achieved by the adapted Trigram model, which is 16.03%.

TABLE I: THE RESULTS OF THE LINE-BASED LM.

Method	WER	WER-WOF	WER-WOL	WER-WOFL
Unigram LM	24.63	23.97	24.02	23.22
Bigram LM	22.43	21.74	21.84	20.94
Adapted Bigram	18.92	17.77	18.10	16.70
Adapted Trigram	16.03	15.09	15.16	14.05
Adapted 4-gram	16.57	15.62	15.58	14.42
Adapted 5-gram	16.42	15.48	15.53	14.38

B. Paragraph-based language modeling

In this experiment, the underlying language model in HTR has been constructed based on the paragraphs of the original resources. We expect to achieve better performance in the line boundaries in this experiment. Table II gives the related results. As shown, the used adapted Trigram model achieves the best performance. Consistent with our hypothesis, the paragraph-based language model performs better than the line-based approach.

TABLE II: THE RESULTS OF THE PARAGRAPH-BASED LM.

Method	WER	WER-WOF	WER-WOL	WER-WOFL
Bigram LM	22.57	21.86	21.94	21.02
Adapted Bigram	18.73	17.59	17.83	16.49
Adapted Trigram	15.97	15.08	14.98	13.90
Adapted 4-gram	16.12	15.06	15.21	13.97
Adapted 5-gram	15.89	14.94	14.89	13.74

C. Paragraph-level decoding model

With this approach, 4-grams give the best result. The adapted bigram model profits most from the added context.

TABLE III: THE RESULTS OF THE PARAGRAPH-LEVEL MODEL.

Method	WER	WER-WOF	WER-WOL	WER-WOFL
Bigram LM	21.96	21.40	21.31	20.60
Adapted Bigram	17.33	16.48	16.53	15.52
Adapted Trigram	15.68	14.9	14.65	13.76
Adapted 4-gram	15.65	14.91	14.71	13.88
Adapted 5-gram	16.05	15.24	15.1	14.18

³<http://www.transcriptorium.eu/~htrcontest/>

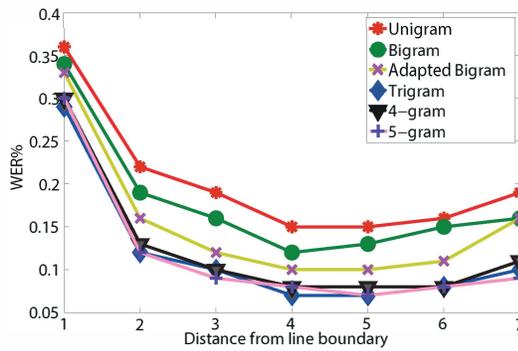


Fig. 4: Word error rate using different LMs.

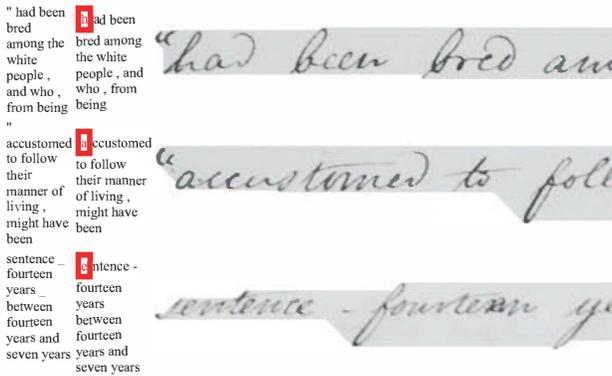


Fig. 5: Main issues in the recognition results.

D. Analysis of results

We here analyse the results in more detail. As shown in Tables I, II, and Fig. 5, the word error rate in the line boundaries is higher than the other positions in the lines. Fig. 6 depicts the error rate analysis in more details. As seen, although the proposed language models improve the performance of HTR in the line boundaries, however, the error rate is still high at line boundaries. We now compare the recognition results to images to identify the main issues, see Fig. 5. Based on our analysis the main issues are:

- In many cases where errors occur, the line segmentation follows the character contour so tightly that (presumably) feature extraction will give a different result for line-initial instances of the same character than for line-internal instances, see Fig. 5.
- A relatively large portion of the errors is related to punctuation symbols, for which the current HTR system is not optimized.
- Hyphenated words which are not supported by the language model and dictionary.

VII. CONCLUSION

We have studied and tested several ways in which adapted and higher order N-gram language models, when trained and applied in a suitable way, can contribute to the solution of the typical loss of HTR accuracy at the line boundary. The proposed methods for the combination of in-domain and out-of domain data and the application higher order N-gram models, used in conjunction with a specially developed paragraph-level

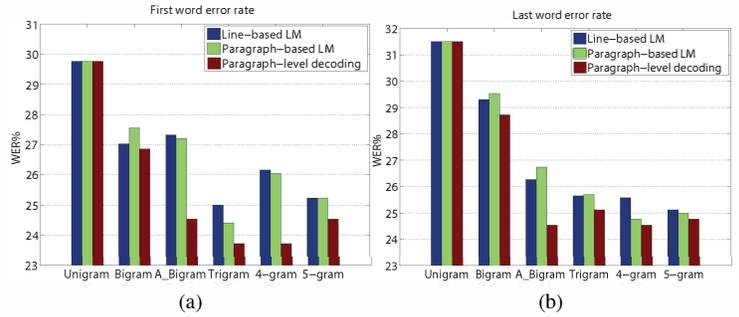


Fig. 6: Comparison of error rate in line boundaries.

lattice rescoring approach, have been shown to yield significant improvement in HTR results.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union’s Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 600707 - transcriptorium.

REFERENCES

- [1] M. Zimmermann and H. Bunke, “Optimizing the integration of a statistical language model in hmm based offline handwritten text recognition,” in *ICPR*, vol. 2, 2004, pp. 541–544.
- [2] J. Tanha, J. de Does, and K. Depuydt, “An intelligent sample selection approach to language model adaptation for hand-written text recognition,” *the ICFHR conference, 2014*.
- [3] A. Axelrod, X. He, and J. Gao, “Domain adaptation via pseudo in-domain data selection,” in *Proceeding of Conference on EMNLP*, 2011, pp. 355–362.
- [4] G. Foster, C. Goutte, and R. Kuhn, “Discriminative instance weighting for domain adaptation in statistical machine translation,” in *Conference on EMNLP, 2010*.
- [5] J. Jiang and C. Zhai, “Instance weighting for domain adaptation in nlp,” in *ACL*, vol. 2007, 2007, p. 22.
- [6] J. Tanha, “Ensemble approaches to semi-supervised learning,” *Ph.D thesis, Informatics Institute, University of Amsterdam*, 2013.
- [7] U.-V. Marti and H. Bunke, “Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system,” *J.Pattern Recognition and AI*, vol. 15, no. 01, pp. 65–90, 2001.
- [8] J. A. Sánchez, G. Mühlberger, B. Gatos, P. Schofield, K. Depuydt, R. M. Davis, E. Vidal, and J. de Does, “transcriptorium: a european project on handwritten text recognition,” in *ACM symposium on Document engineering*. ACM, 2013, pp. 227–228.
- [9] M. Moyle, J. Tonra, and V. Wallace, “Manuscript transcription by crowdsourcing: Transcribe bentham,” *LIBER Quarterly*, vol. 20, no. 3, 2011.
- [10] J. F. Pitrelli and A. Roy, “Creating word-level language models for large-vocabulary handwriting recognition,” *International Journal on Document Analysis and Recognition*, vol. 5, no. 2-3, pp. 126–137, 2003.
- [11] A. Stolcke, J. Zheng, W. Wang, and V. Abrash, “Srlm at sixteen: Update and outlook,” in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, 2011, p. 5.
- [12] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in *ICML*. ACM, 1998, pp. 92–100.
- [13] J. Tanha, M. van Someren, and H. Afsarmanesh, “Disagreement-based co-training,” in *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*. IEEE, 2011, pp. 803–810.