Construction of language models for an handwritten mail reading system

Olivier Morillot, Laurence Likforman-Sulem, Emmanuèle Grosicki

Construction of language models for an handwritten mail reading system

Olivier Morillot[a], Laurence Likforman-Sulem[a] and Emmanuèle Grosicki[b]

[a]Télécom ParisTech and CNRS LTCI; [b]DGA;

ABSTRACT

This paper presents a system for the recognition of unconstrained handwritten mails. The main part of this system is an HMM recognizer which uses trigraphs to model contextual information. This recognition system does not require any segmentation into words or characters and directly works at line level. To take into account linguistic information and enhance performance, a language model is introduced. This language model is based on bigrams and built from training document transcriptions only. Different experiments with various vocabulary sizes and language models have been conducted. Word Error Rate and Perplexity values are compared to show the interest of specific language models, fit to handwritten mail recognition task.

Keywords: Offline Handwriting recognition, handwritten mail, language modeling, Hidden Markov Models, text-line recognition, n-grams

1. INTRODUCTION

Handwritten text recognition is currently a very active area of research and is being thoroughly studied.[1–3] Among its different applications, postal mail automatic processing is one of the most considered question. Companies and administrations are now interested in sorting, searching through and even answering the large amount of letters they receive.

Analyzing handwritten mail has been mainly studied as a document classification.[4] Specific words are searched within text blocks using key-word spotting methods. In the present work, we aim at recognizing the text blocks, i.e. recognizing all words and their sequence. While low error rates are now reached in isolated word recognition,[5] there is still considerable scope for progress on word sequences.

This recognition task presents many specific characteristics. First, the vocabulary used on companies and administrations' mail has often an intermediate size (∼7.000 words) since it is specific to its activity field. Due to the formal aspects of letters, some of its sentences are nearly idiomatic. Although there are some codified structures, unknown words such as family names and zip codes appear in almost every mail. We use Rimes, a French Handwritten Database*, which reflects industrial expectations because it only contains unconstrained free handwriting.

We choose to build a recognition system based on Hidden Markov Models (HMMs) with an analytical strategy, since they have been successfully used to model handwriting.[2, 6] Basically, characters are modeled as a left-right sequence of states. Words are then modeled as concatenations of those character HMMs. At word level, we have the choice between two strategies: with or without explicit segmentation. Character models can either be learnt on pre-segmented characters or segmentation can be implicitly performed using a sliding window approach when decoding. In this paper we consider the approach without segmentation because segmentation into is not fitted to cursive characters. In addition to that, we build contextual characters models[7] to reckon with character neighborhood.

Two main approaches can also be considered at line level: with or without explicit segmentation. The segmentation-based approach starts with splitting the line into words and then the recognition system is run at

Further author information: E-mail: morillot@telecom-paristech.fr, Telephone: +33 (0)1 45 81 71 48

*Reconnaissance et Indexation de données Manuscrites et de fac similés - Recognition and Indexing of handwritten documents and faxes (http://www.rimes-database.fr/)

Document Recognition and Retrieval XIX, edited by Christian Viard-Gaudin, Richard Zanibbi,
Proc. of SPIE Vol. 8297, 82970S · © 2012 SPIE · CCC code: 0277-786X/12/$18 · doi: 10.1117/12.911965

the word level. The other approach consist of processing directly the entire line. Using an explicit segmentation has the advantage of simplifying the combination of methods, since there is no need to align the different outputs when re-scoring. Seeing that RIMES database includes very irregular spacing between words and many script writings, we choose an approach without initial segmentation. Segmentation into words will be provided when decoding lines.

Our motivation is also to take into account language properties by building a statistical language model adapted to the given task. Only a few works have been made at line level: Vinciarelli notably proposed a large vocabulary handwritten text recognizer based on HMMs and statistical language models.[8, 9] Then, HMM-RNN hybrid approaches have been experimented at line level.[10] BLSTM architectures have also been recently tested at the line level.[11] Contrary to previous works, which use large vocabularies (from 10.000 to 50.000 words) and large corpora to compute language models, we develop our system only on the training database. Thus, even if we work on an open vocabulary task, our recognizer has an intermediate dictionary size ($\sim$7.000 words). We try to show that small dictionary and language model can be efficient for specific tasks, such as automatic mail processing.

This paper is organized as follows: Section 2 describes the preprocessing and the feature extraction steps. Hence, Section 3 gives a description of the HMM-based proposed recognition system. Language modeling questions are detailed in Section 4. Experiments results on the Rimes database are given in Section 5. Conclusion and some perspectives are given in Section 6.

2. HMM CONTEXTUAL MODELING

2.1 Preprocessing and feature extraction

The HMM approach we choose is based on a sliding window strategy: Images are transformed into a sequence of features frames, on which models will be learnt. Given the variability of writing, preprocessing images is essential in an effort to reduce variation between samples.

2.1.1 Noise removal

Our starting point is the coordinate boxes of lines within the sheet. Those coordinates were provided by the database creators. So we have cropped lines from the sheet grayscale images.

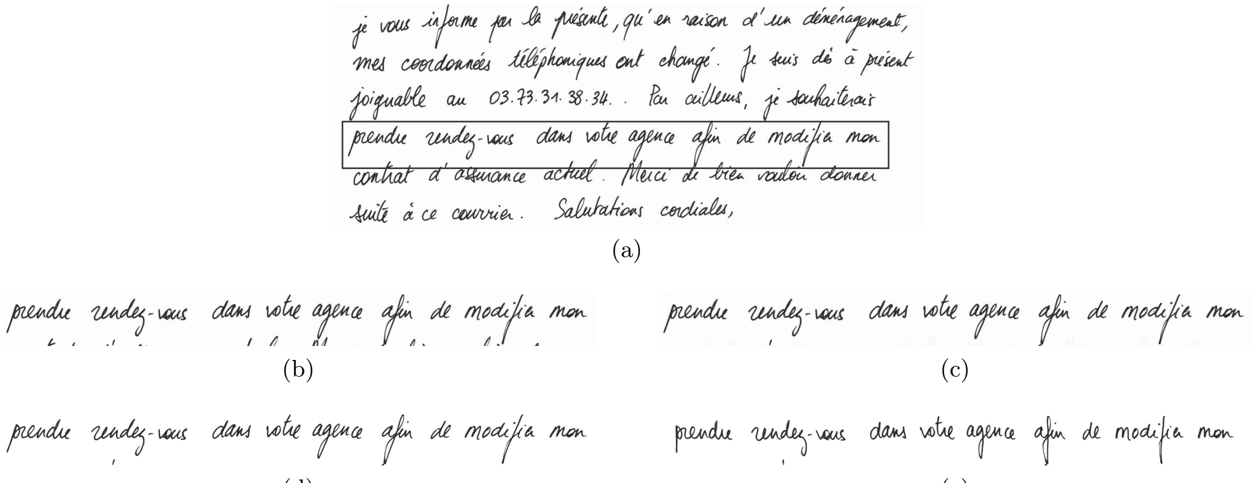
Since writers were not forced to follow guidelines, many text lines are sloped or close to each other. Thus ascenders and descenders appear on line image borders (Figure 1a). Even if those artifacts don't interfere with human reading, they can have a strong impact on some features which are extracted from pictures as it will be explained in subsection (2.1.3).

To filter those artifacts, we have extracted connected components from a binary version of the image. Components are then sorted into three classes given specific criteria: Text body to be kept, descenders and ascenders from other lines to be removed. To identify those noisy components, we have made the assumption that they were in contact with the edge of the image and that their gravity center was peripheral and also that they didn't reach the middle of the image. After their identification, the rejected components are subtracted from the original grayscale image. There have been, admittedly, some false alarm examples, but, as for Rimes data, those combined criteria succeeded mostly in detecting peripheral noise (Figure 1c).

Since we didn't want to binarize images, we have decided to whiten their background in order to remove background noise (Figure 1d). Indeed, some of the features we extract use the grayscale levels and thus background noise can have an influence on those features.

2.1.2 Slant removal

Then the final step of our preprocessing consists of deslanting lines (Figure 1e). Slanted writing can provoke an overlapping between characters and thwart vertical sliding windows analysis. Indeed, models wouldn't be learnt properly on slanted texts since features extracted by the sliding windows would correspond to a mix between two models. Deslanting demands first to find the text baseline. Then slant angle is estimated by studying image vertical projections. Deslanting and baseline extraction are both based on Vinciarelli's work.[12]



je vous informe par la présente, qu'en raison d'un déménagement,
mes coordonnées téléphoniques ont changé. Je suis dès à présent
joignable au 03.73.31.38.34. . Par ailleurs, je souhaiterais
prendre rendez-vous dans votre agence afin de modifier mon
contrat d'assurance actuel. Merci de bien vouloir donner
suite à ce courrier. Salutations cordiales,

(a)

(b) (c)

(d) (e)

Figure 1: Preprocessing steps. (a) Original scanned letter with one coordinates box highlighted. (b) Line image cropped from the letter. (c) Peripheral noise removed. (d) Background whitened. (e) Final image after slant removal.

2.1.3 Feature extraction

After having normalized line images, we need to learn character models from those pictures. Rather than pixel values, we extract features from them. Our work uses features defined by Al-Hajj et al.[13,14] First used on arabic handwriting, they have successfully work on French and English handwriting. Some of those features are statistical (densities, background/foreground transitions) and other are geometrical (local convexity, relative position of gravity centers and baselines). The feature sequence is extracted from gray-level pictures by moving an overlapping sliding window. To cope with different image heights, sliding window subsamples images into a given number of cells. After extraction, all features are derived to obtain dynamic features which are added to the previous ones. Thus we use 56-feature sequences extracted for training and decoding.

2.2 HMM contextual modeling

2.2.1 HMM model

Our system models lines as concatenation of words with spaces in-between and words are represented as a concatenation of its compound character models (Figure 2). Basically, characters are represented as a succession of states with left-right transitions and a self-transition. Each state has a continuous observation density defined as a mixture of N_G gaussian distributions. System parameters are learnt through many iterations of the Baum-Welch algorithm. N_G is gradually incremented between re-estimations. Among numerous parameters, the model include some predominant ones: N_G, the number of states per character model and the number of re-estimations.

2.2.2 Trigraphs models

One of the most challenging property of character recognition is its variability given the context. Indeed, neighboring characters have a strong influence on the shape of a letter. Ligatures and even the shape of the letter can vary widely ((Figure 3) and this phenomenon can affect feature extraction. In an effort to consider the information, our recognition system use a contextual modeling:[7] Rather than model 91 different monographs (different case letters, accentuated letters, digits and special characters), we replace them by trigraphs who add to the central letter its left and right contexts. For example, "t-e+r" and "v-e+n" ((Figure 3) are two different trigraphs sharing the same central letter. Nonetheless, this modelization increases massively the number of models and so the amount of parameters. If we are to model all possible configurations, it leads to $91^3 = 753571$ different trigraphs. Admittedly, only a small part of those appear in current language. We only encounter 9400 different trigraphs in a 11000 words dictionary. Still, available datasets can't be sufficient to learn so many models.

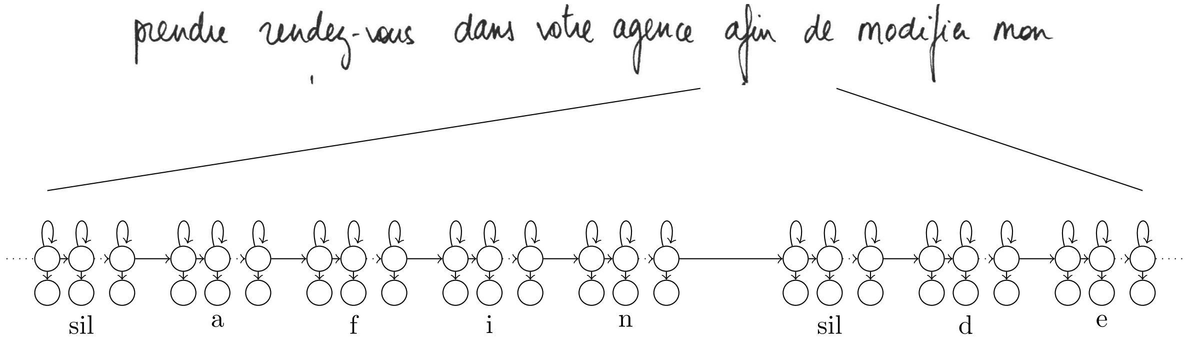


Figure 2: HMM line model: Lines are represented as a concatenation of words, which are themselves character HMMs concatenations - Words are separated with silence models (sil)

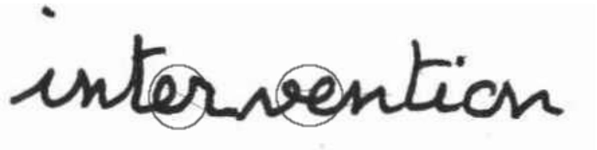


Figure 3: Ligature differences between two 'e' given different contexts

We apply two solutions to reduce the amount of parameters to learn, still following Bianne-Bernard work.[7] First, tying transition matrices between the trigraphs sharing the same central letter reduce the number of models. The second step is to cluster HMMs states using decision trees. Decision trees are here based on questions concerning the right and left context topology (ex: Is the link between the two letters on the lower baseline ?). A decision tree is built for each central letter and the node splitting is decided according to two criteria: cluster minimal occupancy and likelihood maximization. This approach present the advantage of allowing new trigraphs to be introduced in the model. A new trigraph will go down through the decision tree, answering the questions at each level and will find the cluster it belongs to.

The final training system first includes a monograph training phase by Baum-Welch algorithm. Monographs are then replicated into trigraphs, according to the encountered ones in the training database. Following that, trigraph parameters are re-estimated with the same forward-backward algorithm. Then trigraphs states are gathered into clusters. From 75529 trigraphs' states, we finally obtain 2007 different trigraphs with this method. A new phase of parameter re-estimation appears after that. Finally alternating re-estimations and Gaussian mixture incrementations are conducted.

3. CONSTRUCTION OF OUR LANGUAGE MODEL

3.1 Introduction to language modeling

An optical model only based on image analysis, as described in previous section, is challenged by writing quality as errors, noise and irregular writing. In order to tackle this problem, one solution is to introduce a language model which allows to determine the most likely word sequences. It has been introduced in speech recognition since the 80s[15] and more recently in handwriting recognition.[9] The estimation of the word sequence $\hat{W}$ is thus calculated as follows:

$$\hat{W} = \arg\max_{W} P(W|X) = \arg\max_{W} P(X|W)P(W) \tag{1}$$

where the likelihood $P(X|W)$ is determined by an optical model as described in previous section and the prior $P(W)$ of any word sequence $W = (w_1, w_2, ..., w_n)$ is determined by the language model which predicts one word given its context.

The most commonly used approach until now to create a statistical language model is still the n-gram model,[16,17] despite the fact that derived or different approaches exist (Class-based n-gram models,[18] Neural network language models (NNLM)[19]). The n-gram method reduces the context of one word to the $n-1$ previous ones. For each word w_i, its probability is computed given its $n-1$ predecessors $w_{i-n+1}, ..., w_{i-1}$: $P(w_i|w_{i-n+1}, ..., w_{i-1})$

An estimation of this probability is made by counting word sequences on a text corpus (Eq.2):

$$\hat{P}(w_i|w_{i-n+1}, ..., w_{i-1}) = \frac{C(w_{i-n+1}, ..., w_i)}{C(w_{i-n+1}, ..., w_{i-1})} \text{ with } C(.) \text{ being the argument count function} \qquad (2)$$

This approach assumes that the training text n-gram frequencies can model the probabilities of the text to decode. The choice of n, 'history size', is limited by training corpora size. In theory, a 10.000 words 3-grams language model requires to estimate $10.000^3 = 10^{12}$ different trigrams. However, in practice, only a small part of those trigrams appears in current language. But an even smaller part appears in a training corpora. Moreover, many sequences appear very few, which is not statistically significant. In order to tackle this problem, three strategies can be adopted or combined: increase training corpora size, limit n value (classically $n \leq 4$) and use a back-off scheme. The main idea of the back-off strategy is to use the information at the inferior order to model unseen n-grams. If an n-gram sequence has not been seen in the training corpora, its n-1 prefix is more likely to have been observed. Thus the unseen n-gram probability is computed from its n-1 shorter context. By adding such events, probability mass must be redistributed from observed events to unseen ones. This part is referred as discounting or smoothing probabilities. Back-off weight is then calculated from the collected probability mass.

3.2 Building language model for handwritten mail recognition

Building a language model is related to the specific task. Even if large corpora can be available to build general language models, the obtained LM model may be not adapted to the target task as handwritten mail recognition. For instance, a newspaper-based LM would not be likely to model sentences starting with the personal pronoun "Je" (translation: 'I') as found in handwritten mails. Finding an available specific corpus is often an issue, and especially for the French language. We describe here a complete approach to build a language model for French handwritten text line recognition. This includes:

- corpus construction,
- transcription normalization,
- language model parametrization,
- balancing optical and language models

Since no available corpus fits to our specific task (handwritten mail recognition), we choose to build our language model only on available training transcriptions from handwriting database. Seeing that our decoding task only includes text lines, we decide to learn LM rather on lines than on text block transcriptions. Considering the amount of training data and the shortness of lines to decode, a bigram model appears to be best suited for our task. We built it with SRILM Toolkit.[20] Nonetheless, it appears that some bigrams would not be learnt since they are separated by a carriage return. Moreover we want to address the fact that text lines can start with various words. To face these difficulties, we cut mail transcriptions into new line fragments and add them to the LM training corpus. We thus obtain artificial text lines (recuts) as suggested.[10]

As compared to speech language modeling, we introduce punctuation in the language model. Even if those signs are not very often taken into account in WER or WCR rates, they carry much information since text structure relies heavily on them. For example, in most cases, a dot is followed by a grammatical article or a personal pronoun.

Due to the free writing, mail documents present a specificity which is not often encountered on other databases. There are many misspelled errors. In this paper, we try to address syntactical errors. Notably, in Rimes database we noticed that approximately 3% of dictionary entries contained spelling errors. Those errors have two main consequences on the recognition process: Some mistakes from the training database are automatically added to the dictionary and can result in adding errors by the time of decoding. Moreover, misspelled written words can not be decoded if they are not part of the dictionary. A few common spelling errors happen to be as frequent as the correct spelling. Thus, excluding misspelled words from the dictionary is not a satisfying solution. We experiment an approach which consists of gathering several spellings (the correct one and incorrect ones) under the same label in the dictionary. The recognition process will automatically output the true spelling. This approach offers a certain flexibility in the possible spelling of words and can even automatically correct syntactical errors while decoding.

To do so, we add correct spelling to the dictionary when it is missing. Afterwards we correct LM training corpus syntactical errors in order to have matching vocabularies between dictionaries and LMs. Through that procedure, the number of possible spelling (dictionary size) increases while the number of possible outputs decreases (monograms amount).

To cope with unseen bigrams, several discounting strategies exist. In this work, we used Good-Turing discounting method.[21] The unseen event probability is computed as follows:

$$\hat{P}(w_i|w_{i-1}) = \alpha(w_{i-1})\hat{P}(w_i) \text{ if } C(w_{i-1}, w_i) = 0 \tag{3}$$

where $\alpha()$ is the back-off weight.

Grammar probability of a word sequence is then computed from bigrams probability: $P_{grammar}(w_1, .., w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1})$ where $P(w_i|w_{i-1})$ is provided by the language model. To balance language model probabilites weight regarding optical ones, it is possible to introduce a Grammar Scale Factor (GSF): The recognition system can either give more weight to the optical model or to the language (Eq.4). For a given feature sequence X, $\hat{W}$ is the most likely word sequence calculated as follows:

$$\hat{W} = \arg\max_{W} P_{optical}(X|W)P_{grammar}(W)^{GSF} \tag{4}$$

We conduct experiments with various GSF values in Section 4.

In order to evaluate and compare language models' performance, we need to define the adequation between a language model and the text it should help to recognize. Perplexity (PP) is the most commonly used tool. The perplexity of a bigram language model is estimated on a test text as follows:

$$\hat{PP} = 2^{\hat{H}} \text{ where } \hat{H} = \frac{1}{m}\sum_{i=1}^{m} \log p(w_i|w_{i-1}) \text{ with a test text containing m words } (w_1, w_2, ...w_m) \tag{5}$$

Perplexity can be seen as an average estimation of how many different words can follow any given word. The larger the number of possible following words, the higher the perplexity. Of course, perplexity measure is linked to the vocabulary size. So perplexity is only relevant to compare language models sharing the same vocabulary. Furthermore, this measure has limits: A decrease in perplexity does not always conduct to better recognition rates.

4. EXPERIMENTAL RESULTS

4.1 Rimes database

Rimes was created with the funding of French Defense and Research Ministries to evaluate automatic recognition and indexing systems of handwritten letters. The database was collected by asking volunteers to write letters, given scenarios among nine realistic following scenarii: change of personal information, information request, opening and closing account, modification of contract or order, complaint, payment difficulties, reminder letter and damage declaration. The volunteers composed a letter with those pieces of information using their own words. Since its creation in 2006, several word recognition competitions have been conducted and in spring

2011 the first text block recognition competition took place. Participants were given grayscale letters and the coordinate box of each line and the transcription for the training database. The training database contains 1500 letters, for a total amount of 11.329 lines. From this database we used 1370 letters for training (10.318 lines) and 130 for validation (1.011 lines).

4.2 Evaluation tools

The recognition performance can be measured by several tools.[22] Among them, WER (Word Error Rate) appear to be the most commonly employed to compare two sequences of words. WER is defined as the proportion of errors (substitution, insertion and deletion) among the total number of words in the reference text (Eq.6). Since those three kinds of errors are not independent, WER can reach 100%, even when words have been correctly decoded. To address this problem, we also provide the WCR (Word Correct Rate) which does not take into account insertions.

$$WER = \frac{substitutions\ +\ insertions\ +\ deletions}{total\ number\ of\ words};\quad WCR = 1 - \frac{substitutions\ +\ deletions}{total\ number\ of\ words} \tag{6}$$

4.3 Results

Experiments show that recuts have a positive impact on WER (Tab.1). Perplexity increases with corpus size since recuts introduce new bigrams.

Table 1: LM corpus size influence on recognition (LM with discounting; GSF=1)

	LM training corpora	Number of lines	WER	PP
No LM	–	–	51.6%	–
LM	Training transcriptions	15172	50.1%	48.0
	Training + recut transcriptions	43766	49.3%	146.9

We optimize Grammar Scale Factor (GSF) by testing our recognizer on our validation database (Fig.4). We also compare a model with or without discounting probabilities. First, it can be observed that discounting does not improve performance under a certain value of GSF (approximately 10). Indeed discounting probabilities of most seen bigrams weakens language model influence. Then, for higher GSF values, discounting model improves recognition rates since it models unseen bigrams with a sufficient weight, balancing optical probabilities. On our validation base, increasing GSF value 25 seems to reduce WER by 14.3% in absolute value. Lastly, higher GSF values give worser WER. Giving too much importance to the language model has a side effect: The recognizer outputs are often common word successions but far from the actual words.

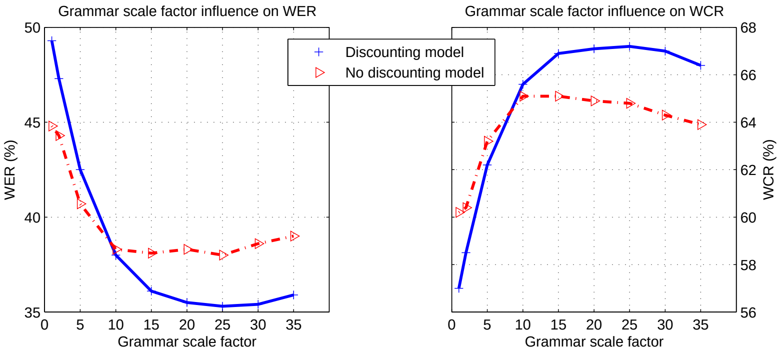


Figure 4: Grammar scale factor influence on recognition performance on validation base - LM built on training database

Correction of syntactical errors leads to a 0.5% decrease in WER (Tab.2). As a positive side effect, we also note a decrease in perplexity which can be explained by the gathering of spellings under common output labels.

To compare perplexity values, we introduce here (Tab.2) a ”normalized perplexity” value[8] which consists of dividing perplexity value by the monograms amount.

Table 2: Influence of error corrections on validation database decoding (LM with discounting; GSF=25)

	Dictionary	Dic. size	Monograms amount	PP	normalized PP	WER
No LM	training	5933	–	–	–	51.6%
	training+validation	6279	–	–	–	49.6%
LM	training	5933	5149	146.9	0.029	35.3%
	training+validation	6279	5457	293.1	0.054	33.5%
	training (corrected)	5999	5028	110.6	0.022	34.8%
	training+validation (corr.)	6352	5319	197.9	0.037	33.1%

4.4 ICDAR 2011 French Handwriting Recognition Competition

One important matter while taking part in a recognition competition is the dictionary choice. Given that no dictionary was provided for ICDAR 2011 French Handwriting Recognition Competition, our dictionary is built from available training transcriptions. In an attempt to address out-of-vocabulary (OOV) question, we add vocabulary from ICDAR 2011 French Word Recognition Competition dictionary. Moreover, we remove hyphenated words and most of the codes because they were not likely to appear in the test. Given that vocabulary (6915 different words), we optimize language model parameters on our validation base. Thus, our language model is built on all the transcriptions given for training with discounting weights and grammar is given a scale weight of 20, regarding both WER and WCR performances. Transcriptions and dictionary were corrected according to the experiments conducted in the previous section. We obtained a $WER = 31.2\%$ and a $WCR = 73.2\%$ on 778 test lines.[23]

5. CONCLUSION

In this paper we have proposed and tested a method of handwritten mail recognition which directly works at line level. First, preprocessing has been adapted to this specific task. Then, our system uses one state-of-the-art contextual HMM-based recognition method. Our contribution consists of building an efficient language model yielding enhanced performances. To build this language model, we only use training transcriptions. By optimizing grammar scale factor and correcting syntaxical errors, we achieve to reduce WER by 16.8% in absolute value as compared to a model without a language model. We have presented our system at the ICDAR 2011 French Handwriting Recognition Competition and it obtained good results, similar to those presented in our experiments.

Future work will consist of processing separately special fields (zip codes, telephone numbers) since they can not be correctly decoded using a closed vocabulary dictionary. Concerning the language modeling, we gathered words under unaccented case-insensitive labels. Although our system already outputs case-sensitive accented words, we could improve recognition performance by building a more accurate language model. Indeed, accents and case have often a decisive grammatical sense in French language and this will be taken into account in our future work.

ACKNOWLEDGMENTS

The authors would like to thank A-L Bianne-Bernard from Télécom ParisTech and Gilles Adda from LIMSI for fruitful discussions.

REFERENCES

1. Steinherz, T., Rivlin, E., and Intrator, N., “Offline cursive script word recognition - a survey,” *International Journal of Document Analysis and Recognition* **2**(2), 90–110 (1999).
2. Plamondon, R. and Srihari, S., “Online and offline handwriting recognition: A comprehensive survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 63–84 (January 2000).

3. Vinciarelli, A., "Online and offline handwriting recognition: A comprehensive survey," *Pattern Recognition* **35**, 1433–1446 (June 2002).
4. Rodríguez-Serrano, J. and Perronnin, F., "Handwritten word-spotting using hidden markov models and universal vocabularies," *Pattern Recognition* **42**(9), 2106–2116 (2009).
5. Grosicki, E. and El-Abed, H., "ICDAR 2009 handwriting recognition competition," in [*ICDAR*], 1398–1402 (2009).
6. El-Yacoubi, A., Gilloux, M., Sabourin, R., and Suen, C.-Y., "An HMM-based approach for off-line unconstrained handwritten modeling and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**(8), 752–760 (1999).
7. Bianne-Bernard, A.-L., Menasri, F., El-Hajj, R., Mokbel, C., Kermorvant, C., and Likforman-Sulem, L., "Dynamic and contextual information in HMM modeling for handwritten word recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **99**(PrePrints) (2011).
8. Marti, U.-V. and Bunke, H., "Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system," *IJPRAI* , 65–90 (2001).
9. Vinciarelli, A., Bengio, S., and Bunke, H., "Offline recognition of unconstrained handwritten texts using HMMs and statistical language models," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**, 709–720 (June 2004).
10. Espana-Boquera, S., Castro-Bleda, M. J., Gorbe-Moya, J., and Zamora-Martinez, F., "Improving offline handwritten text recognition with hybrid HMM/ANN models," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**, 767–779 (2011).
11. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., and Schmidhuber, J., "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31** (May 2009).
12. Vinciarelli, A. and Luettin, J., "A new normalization technique for cursive handwritten words," *Pattern recognition letters* **22**(9), 1043–1050 (2001).
13. Al-Hajj-Mohamad, R., Likforman-Sulem, L., and Mokbel, C., "Arabic handwriting recognition using baseline dependant features and hidden markov modeling," in [*Proceedings of the Eighth International Conference on Document Analysis and Recognition - ICDAR05*], 893–897 (2005).
14. Al-Hajj-Mohamad, R., Likforman-Sulem, L., and Mokbel, C., "Combining slanted-frame classifiers for improved HMM-based arabic handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**, 1165–1177 (2009).
15. Bahl, L., Jelinek, F., and Mercer, R., "A statistical approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **5**, 179–190 (March 1983).
16. Katz, S., "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing* **35**(3), 400–401 (1987).
17. Rabiner, L. and Juang, B.-H., [*Springer Handbook of Speech Processing*], ch. Historical Perspective of the Field of ASR/NLU, 521–537, Springer-Verlag New York, Inc. (2007).
18. Brown, P., DeSouza, P., Mercer, R., Della-Pietra, V., and Lai, J., "Class-based n-gram models of natural language," *Computational Linguistic* **18**(4), 467479 (1992).
19. Bengio, Y., Ducharme, R., and Vincent, P., "A neural probabilistic language model," *Journal of Machine Learning Research* **3**(2), 1137–1155 (2001).
20. Stolcke, A., "Srilm: An extensible language modeling toolkit," in [*Proc. International Conference on Spoken Language Processing*], 901–904 (2002).
21. Good, I., "The population frequencies of species and the estimation of population parameters," *Biometrika* **40**, 237–264 (1953).
22. McCowan, I., Moore, D., Dines, J., Gatica-Perezl, D., Flynn, M., Wellner, P., and Bourlard, H., "On the use of information retrieval measures for speech recognition evaluation," tech. rep., IDIAP (March 2005).
23. Grosicki, E. and El-Abed, H., "ICDAR 2011: French handwriting recognition competition," in [*ICDAR*], (2011).