Recognition and information extraction in historical handwritten tables: toward understanding early 20^{th} century Paris census[⋆]

Thomas CONSTUM[1], Nicolas KEMPF[1], Thierry PAQUET[1], Pierrick TRANOUEZ[1], Clément CHATELAIN[2], Sandra BREE[3], and François MERVEILLE[4]

[1] LITIS (University of Rouen Normandy), 76000 Rouen, France { thomas.constum , nicolas.kempf, thierry.paquet, pierrick.tranouez }@univ-rouen.fr
[2] LITIS (INSA Rouen Normandy), Rouen, France clement.chatelain@insa-rouen.fr
[3] LARHRA (CNRS), France sandra.bree@msh-lse.fr
[4] Campus Condorcet - The Grand Équipement Documentaire, France francois.merveille@campus-condorcet.fr

Abstract. We aim to build a vast database (up to 9 million individuals) from the handwritten tabular nominal censuses of Paris of 1926, 1931 and 1936, each composed of about 100,000 handwritten simple pages in a tabular format. We created a complete pipeline that goes from the scan of double pages to text prediction while minimizing the need for segmentation labels. We describe how weighted finite state transducers, writer specialization and self-training further improved our results. We also introduce through this communication two annotated datasets for handwriting recognition that are now publicly available, and an open-source toolkit to apply WFST on CTC lattices.

Keywords: handwriting recognition · Document Layout Analysis · self-training · fully convolutional network · table analysis · WFST

1 Introduction

In the digital age, many handwritten corpora containing very interesting information for historians still remain unexploited because it would be too costly and time-consuming to analyze them by hand entirely. The POPP project (Project for the OCRing of the Paris Population census) aims to build a vast database (12 million individuals) from the handwritten tabular nominal censuses of Paris of the years 1926, 1931, 1936 and 1946, each composed of about 100,000 handwritten simple pages in a tabular format (figure 3a shows a double page). In this paper, the first three census are considered since they have the same table structure unlike the 1946 census.

[⋆] Project supported by CollEx-Persée (AAP19_20), with the financial collaboration of the TGIR Progedo and the Grand Equipement Documentaire Campus Condorcet

A similar table structure was used for the population census at the national level in France at the same epoch. This material is a primary source of information for historian demographers to analyze sociological facts, especially how very large cities were formed during the first half of the twentieth century. This fixed structure that span over the entire corpus facilitates the recognition process of the handwritten information. Indeed, every cell contains an expected specific type of information (a name, a date, an address etc.) that can be modeled thanks to a dictionary or a regular expression that is used to drive the recognition process based on beam search Viterbi alignment [14,16,5].

The core of the processing chain is the handwriting recognition module. Even if significant progress have been made these last years thanks to the use of deep neural networks architectures, performance can degrade quickly on heterogeneous data, or heterogeneous writing styles, showing generalization difficulties of such architectures. The size and thoroughness of the training dataset have proved to be a major factor in getting high performance recognition systems [10], thus the interest to generate additional synthetic or augmented data for training the network [22].

Less explored in the handwriting recognition literature, semi-supervised learning offers an interesting framework to exploit large amounts of un-annotated datasets during training. Considering the large amount of data available to us with the three handwritten Paris census, this is a path we decided to explore, more precisely self-training. Evaluation results show a significant performance increase with this framework, reaching nearly mono-writer performance. We analyze the performance in relation to the size of the network architecture.

In this communication, we present the main components of the whole processing chain of the handwritten tabular nominal census and the recognition performance obtained on a subset of the corpus that was manually annotated for training as well as evaluation purposes.The main components are based on deep neural networks for table detection, table line detection and line recognition, whereas syntax-driven handwritten field recognition uses Weighted Finite State Transducers (WFST). Finally, a set of coherency rules that are derived from the fixed structure of the census tables is exploited to fill in missing fields as much as possible. In the end, the extraction results exported through CSV files are made available to the research community of historian demographers. Some components of the processing chain are made freely available to the research community, notably the Kaldi based WFST decoder[5] that has been refactorized for easy integration in a Python programming framework. We also provide free access to the POPP annotated handwritten corpus to the Document Image Analysis research community.[6] This paper is organized as follows.

Section 2 is devoted to the presentation of the corpus. section 3 provides an overview of the processing pipeline. Section 4 is devoted to the pre-processing stage, including table detection, page classification, table dewarping and line table detection. Section 5 is devoted to the handwriting recognition module presen-

[5] https://gitlab.com/projet-popp/sigra/

[6] https://github.com/Shulk97/POPP-datasets/

tation, including the deep NN optical model, self-training and the presentation of the recognition results. Section 6 is devoted to domain knowledge leveraging and content verification, including syntactical and lexicon based language models, decoding , normalization of the recognized fields and logical deduction rules. Section 7 provides some information regarding processing time all along the pipeline.

2 Corpus and ground-truthed datasets

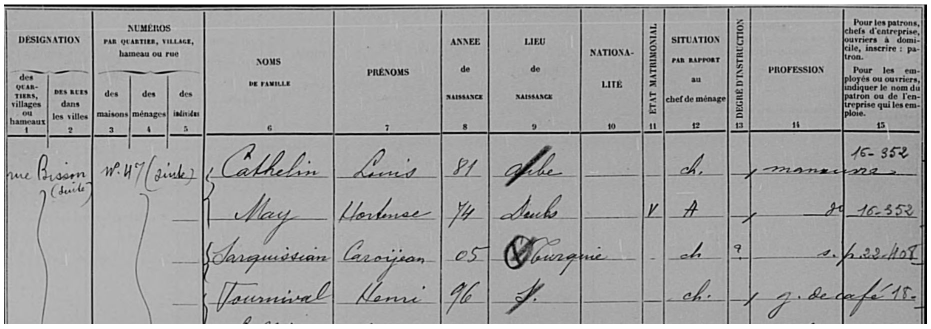
2.1 Presentation of the census

Three census of the population of Paris are considered in this paper: 1926, 1931 and 1936. Each census is divided into 20 boroughs *(arrondissements in French)*, themselves divided into 4 districts each (80 districts in total). Each census consists of approximately 100,000 pages and 3 million lines and we estimate the number of writers for one census between 80 and 500. The raw dataset stored by the Archives of Paris consists in a set of double pages (see figure 3a) made of every pages of the census booklets, including the front and back covers. The pages that are interesting for us are the pages containing tables filled with information that describe each individual. As depicted on figure 3b, these tables contain around 30 lines and each line describes one individual. The tables contain 15 columns. The first 5 columns are actually filled with only 3 types of information: street names, street numbers and household numbers. The 10 other columns contain information about each individual such as name, place of birth and occupation. In this communication, we focus our attention on these 10 columns.

The challenging aspect of this dataset lies in the handwritten nature of the table information. They have been written by the multiple enumerators that were involved in gathering the information when visiting each home. Because of the handwritten aspect of the data, the table layout is often not respected. Sometimes, words are written across two columns and in some cases, some words are written between lines. Figure 1 shows an overview of a page and the challenges that can arise with this dataset. Moreover, since the tables were filled by many writers, some columns were not filled out in the same way. For example, the column *place of birth* is sometimes filled out with the city and the *departement* of birth (French administrative division), with the *departement* only, or with the country of birth only.

2.2 Annotation of two datasets

In order to tune the text recognition model, a first training dataset was necessary. Although the table structure is extremely stable for every page of the census, there is a large variability in the writing styles, background color, and ink type. Therefore, we annotated one double page for each of the 80 districts of the census of 1926, so as to create a generic dataset as representative as possible. Since this dataset already contains a significant diversity of writing styles, there is no need



DÉSIGNATION des quartiers, villages ou hameaux 1	DÉSIGNATION des rues dans les villes 2	NUMÉROS par quartier, village, hameau ou rue: des maisons 3	des ménages 4	des individus 5	NOMS DE FAMILLE 6	PRÉNOMS 7	ANNEE de NAISSANCE 8	LIEU de NAISSANCE 9	NATIONALITÉ 10	ÉTAT MATRIMONIAL 11	SITUATION PAR RAPPORT au chef de ménage 12	DEGRÉ D'INSTRUCTION 13	PROFESSION 14	Pour les patrons, chefs d'entreprise, ouvriers à domicile, inscrire : patron. Pour les employés ou ouvriers, indiquer le nom du patron ou de l'entreprise qui les emploie. 15
	rue Bisson (suite)	N°47 (suite)			Cathelin	Louis	81	Aube			ch.	/	manoeuvre	16-352
					May	Hortense	74	Doubs		V	A	/	d°	16-352
					Sarquissian	Caroijean	05	Turquie		-	ch.	?	s.	p.22.408
					Tournival	Henri	96	S.			ch.	/	g. de café	18-

Fig. 1: Headers and first lines of a table from 1926. The corresponding annotation of the first line is "Cathelain/Louis/81/Aube/¤/¤/ch./¤/manoeuvre/?16-352".

to create year-specific annotated datasets. This first dataset thus contains 160 pages made of 4800 handwritten lines, split into training (80%), validation (10%) and test (10%) sets. The split has been conducted at the double page level so that lines from a given district are located in only one subset. This dataset was manually transcribed line by line using the specific character "/" as a *logical* column separator and the specific character "¤" to indicate empty columns. To describe words written between lines, we defined the symbol "!" and "?" indicating a word respectively written below and above the regular line. These out of line words are not used yet but are annotated for a future use. Figure 1 presents an example of annotation.

Moreover, to conduct some mono-writer experiments, we annotated another dataset consisting of 49 pages (1470 lines) from a single district named *Belleville* located in the 20th *arrondissement* and written by a single writer. Among these 49 pages, 39 pages were used for training, 5 pages were used for validation and the last 5 pages were used for testing.

3 Processing Pipeline

The processing pipeline is depicted on figure 2, below. It is composed of four main stages: pre-processing, handwriting recognition, domain knowledge integration and content verification. The pre-processing stage is devoted to table detection, image dewarping and table row detection. The second stage is devoted to Handwriting recognition applied on the detected rows. Language models are introduced during the third stage to constrain the handwriting recognition stage to allow content extraction and normalization using Weighted Finite State Transducers (WFST). We also apply a rejection rule based on grammars defined for each column. Finally, the stage of content verification introduces a set of coherency rules that are derived from the fixed structure of the census tables to fill in some missing fields as much as possible. In the end, the extraction results exported through CSV files are made available to the research community of historian demographers.

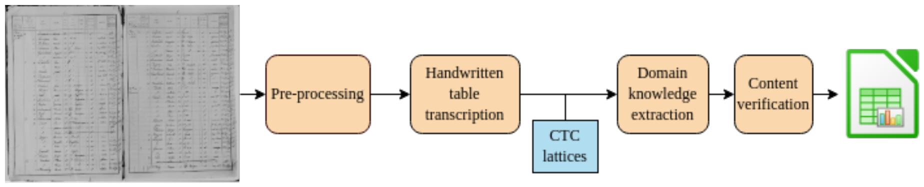


Fig. 2: Organization of the pipeline.

4 Layout analysis and information extraction

4.1 Segmentation and unwarping of tables

The fist step of the processing chain is to detect the two tables in the double page scans. Deep Convolutional Neural Networks (CNN) have proved to be the best pixel-wise predictors for this task[19,15]. The *dhSegment* approach [15] was chosen since it obtains competitive results on Pagenet [25] and its source code is open source[7]. First, we created a dataset made of 80 double pages annotated for page segmentation. The network was trained using the same parameters as in [15] with 90% of the dataset and obtained a mean intersection over union (mIoU) of 0.987 on the remaining 10% of the dataset. We then used the newly trained model to generate new labels and we also annotated a few extra images manually that were found difficult for the model. This new labelled dataset is composed of 223 double pages for training and 40 pages for testing. We finally trained the model on this new dataset and obtained a mIoU of 0.9924 on the test set. Figure 3b shows the segmentation result obtained from the image in figure 3a.

After segmenting the tables, the last step is to use the corners of the tables to dewarp them by applying an affine transformation. The final result of this step is a table image with a null inclination angle. Figure 3c shows the two dewarped tables obtained from the segmentation result of figure 3b. Once dewarped, we can easily crop the irrelevant parts of the image, namely the headers and the 5 first columns. The structure of these columns is indeed not respected as can be seen in figure 1 and they contain address information only.

4.2 Page classification

As mentioned in subsection 2.1, we need to detect the pages with tables and discard the other types of pages. Given the amount of data, we trained a classifier to automatically categorize pages as relevant, irrelevant and badly segmented. For this simple classification task we used the pre-trained MobileNetV2 architecture [20] with a width multiplier of 1. We bootstrapped training the system with 200 pages which were annotated by hand. Then, we generated a larger training dataset by generating predictions on unlabeled pages that were then

[7] https://github.com/dhlab-epfl/dhSegment

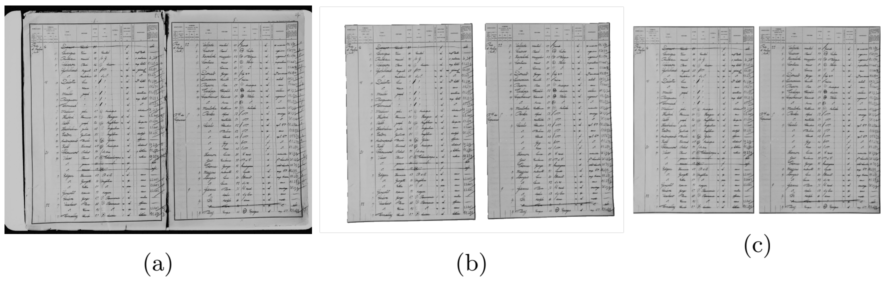


Fig. 3: (a) Example of a double page from the 1926 census. (b) Segmentation results (c) Dewarped tables.

verified manually to obtain a dataset of 1000 annotated pages with 80% of data for training, 10% for validation and 10% for testing. The network was then trained again and we obtained an accuracy of 97.8% on the validation set and 98.5% on the test set. In order to evaluate the performance of the segmentation model, we segmented the 100000 pages of the 1926 Census and then classified each one with the classifier. By counting the number of pages classified as badly segmented by the classifier, we obtained an evaluated ratio of 98% of correctly segmented pages.

4.3 Segmentation of tables into rows

In order to segment our tables into rows, we first used baseline detection using the ARU-Net [12] pre-trained on cBAD. We chose ARU-Net because it has competitive results on cBAD and has the advantage of having the source code and pre-trained weights publicly available. Thanks to the detected baselines, we were then able to reconstruct the table rows by regrouping baselines with similar vertical location. Then, we used this first method to quickly annotate data for segmentation. This time, we trained a dhsegment architecture to segment the baselines of the rows by using 260 annotated pages for training and 32 pages for validation and obtained a mIoU of 0.80 on validation. Though this result could be improved, it is enough to localize the rows by taking the median vertical position of each detected baseline. Finally, to obtain the segmented lines, we simply have to deduce the bounding box from the baseline, knowing that the height of the lines is constant.

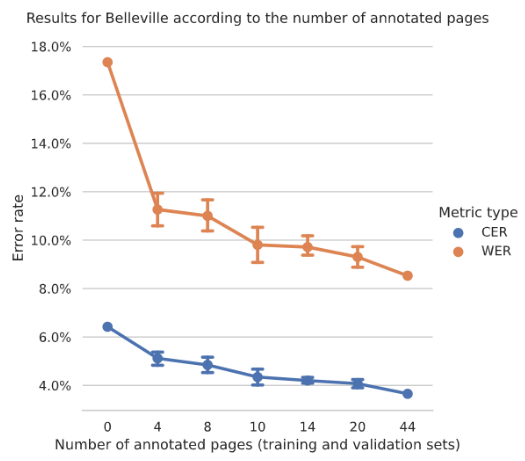
Due to lack of space for writing, the column structure was not always observed. It is very frequent that an information related to one column crosses into the next column. However, by using the column separator symbol that was introduced in the ground truth (subsection 2.2), the HTR model learns to predict the separation into columns while taking as input line images. This avoids the need to segment the rows of the table into columns.

5 Handwriting recognition

5.1 Architecture of the optical model

State of the art neural network architectures for handwritten text recognition are typically made of convolutional layers followed by recurrent layers, either MDLSTM [27] or BLSTM [17,13]. However some recent studies showed very competitive results using fully convolutional networks [31,9,8,30,7]. This kind of architecture has the advantage of being much faster to train than recurrent architectures because it can take full advantage of the parallelization capabilities offered by GPUs. Moreover, several recent publications report on segmentation free approaches [3,4,21,30,7,6] which do not introduce any explicit line segmentation stage, while the network architecture has the ability to be trained at paragraph level, and learn to detect and recognize text lines at the same time, thus avoiding the need for any segmentation ground truth.

In this case, the segmentation is not a difficult step because once the table is dewarped, the lines of text are straight, without slope, parallel and equidistant. Thus, we use in this work a line recognition model. Indeed, paragraph text recognition models can have difficulties to converge especially when we try to modify the width or the depth of the model as we did in subsection 5.3. For our experiments we chose to use the line text architecture described in [7] because of its recurrent-free aspect and also because the source code and training weights on reference datasets are publicly available on Github[8]. This architecture is modular since the encoder of the recognition part can be directly connected to an attention module to perform handwriting recognition on paragraphs.



# pages (lines) training and validation	Number of trainings performed
0 (0)	no fine-tuning
4 (120)	5
8 (240)	5
10 (300)	4
14 (420)	3
20 (600)	2
44 (1320)	1

Fig. 4: Left : Recognition results on the *Belleville* dataset for different numbers of annotated data. Right : Number of trainings performed for each experiment

[8] https://github.com/FactoDeepLearning/VerticalAttentionOCR

5.2 Results of the optical model

We first trained our model on the generic dataset and obtained a CER of 7.08% and a WER of 19.05% on the test set. This first result is correct although the error rate is higher than the state of the art results on datasets like IAM or RIMES with respectively 2.3% [17] and 4.87% [13] of CER on the test set. This indicates that the difficulty represented by this dataset constitutes an interesting challenge for the document analysis community.

Regarding mono-writer recognition, we first evaluated the model trained with the generic dataset on the *Belleville* dataset and obtained 6.42% of CER. Then by specializing the model on the whole *Belleville* dataset we improve the CER to 3.65%, i.e. 43% of relative improvement, which shows the great improvements that can be expected from writer specialization. Then, to evaluate the gain with respect to the number of annotated pages, we also tuned the system using only a fraction of the training and validation sets while testing on the complete test dataset. In order to have more reliable results, we performed cross-validation. For example to evaluate the performance obtained using 13 pages for training and 1 page for validation, we performed 3 trainings using each time randomly chosen pages that had not been used before. The results are summarized in figure 4. With only 4 annotated pages (3 for training and 1 for validation), we reach an average CER of 5.11%. Then, the more we annotate data of a target dataset, the less improvement we get by new annotated lines which confirms the results obtained in [2]. The next challenge to address will be to minimize the number of new lines needed to benefit from writer specialization.

5.3 Self-training

Self-training is a technique related to semi-supervised learning [26] that consists in using a model trained on labelled data, called a teacher, to generate pseudo-labels that are then used for training another model, called a student. This kind of technique is especially useful when a very large amount of unlabelled data is available, which is the case in our project. Self-training has been well explored in the computer vision field [28,29,32], but few publications have investigated self-training for handwriting recognition [23].

In this study, we followed the Noisy Student Training scheme described in [28] which injects noise during the student training. Noise injection makes the student model better than the teacher because its task is harder: it has to reproduce the prediction of the teacher model while having noise applied on the input (using data augmentation) and on the model (using dropout). The student can then become a teacher for another iteration of the process. Student networks can have exactly the same architecture as their teacher, or they can have a wider and deeper architecture, possibly with some recurrent layers or other components.This training process can be repeated until there is no further improvement.

The unlabelled dataset consists in this case of 2.4 million line images selected randomly from the 1926 census. First we trained the initial model (model 0) in a supervised mode on the generic dataset, then we entered the self-training mode for 4 iterations (more iterations did not provide any improvement). Student 1 and 2 have a similar architecture as model 1, while student 3 architecture was 1.5 deeper and 1.25 wider, following the scaling method described in [24]. Indeed, a student 2bis with the initial architecture trained on the predictions of student 2 showed nearly no improvements on the validation set and regressed on the test set. Thus, this model was not used for the next iterations and we used student 3 instead. This new architecture allowed us to further improve the results. This result is particularly interesting because such an architecture is not able to converge using supervised learning on the generic dataset. Finally, we experimented the influence that a BLSTM layer could have on the results. Indeed, this type of layer is most efficient when a large volume of training data is available. Moreover, it is shown in the literature that LSTM layers are very much suited to model contextual information when on top of convolutional layers. Thus, we trained a student 4 using the predictions of student 3. This final model has an architecture C made of the encoder of architecture B and a BLSTM-based decoder. We illustrate this architecture in figure 5. This final architecture brought

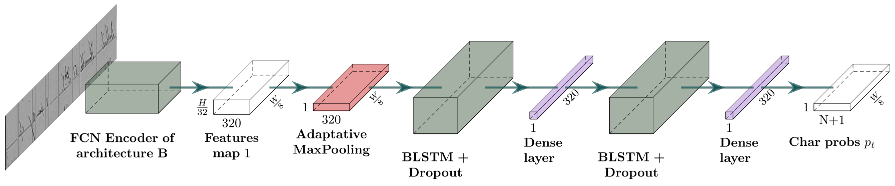


Fig. 5: Schema of architecture C

an improvement of almost 1% of CER on test set, which demonstrates that a BLSTM-based decoder can still be useful compared to a fully convolutional decoder when a large amount of data is available. The table in figure 6 summarizes the results obtained at each experiment.

We also evaluated this self-training process on a single writer dataset. We used the weights of the best model described above to initialize training on the Belleville dataset and then performed inference on the complete unlabelled Belleville district data[9] to generate pseudo-labels. Then we re-trained the model on these pseudo-labels and obtained a CER of 2.66% and a WER of 6.37% on the test set. Compared to the results on the Belleville dataset obtained using supervised learning (CER of 3.65% and a WER of 8.53%), self-training was able to improve the recognition performance using the un-annotated data even if the previous results were already correct. This experiment shows that the noisy student training can also be beneficial for writer specialization.

[9] The complete Belleville census was written by three writers but their writing style are very similar and can be therefore considered as one unique writing style

Model	CER (%) validation	WER (%) validation	CER (%) test	WER (%) test
Initial (Model 0)	6.86	18.66	7.08	19.05
Student 1	6.07	17.12	6.12	17.12
Student 2	5.94	16.80	5.97	16.83
Student 2bis	5.89	16.68	6.02	16.89
Student 3	5.64	15.98	5.43	15.50
Student 4	5.01	14.53	4.52	13.57

Fig. 6: Recognition results of the experiments regarding self-training

6 Leveraging domain knowledge

6.1 Language models

Each column, once located at some beginning and ending positions in the lattice, thanks to the recognition of the column separator symbol ('/') is thereafter decoded using the beam search Viterbi algorithm [14,5] under the constraints of the expected content in this column. As the census tables share the same structure over years, it is possible to model the expected content of each column either with lists of words (in the case of family names, surnames, marital status, position in a household, etc...) or with regular expressions (in the case of date of birth, place of birth, occupation, etc...). We created the dictionaries and formal grammars, using different resources. For example, a list of names and surnames was built using the French National Statistics Institute (INSEE) deceased database of people in France since 1970[10], historical researches [11] and official documents[11] to generate French town names and administrative *départements*'s names. Some country names and foreign towns names were also included as much as possible. Moreover, when necessary, we have created lists of abbreviations used by census agents to compose location names, or occupations. For example, *département* names such as *"Hautes-Pyrénées"* or *"Hautes-Charentes"* can be abbreviated as *"Hte Pyrénées"* or *"H. Charentes"*. Using Finite State Transducers (FST), we combined lists of words and abbreviations in regular expressions to detect every combinations of these elements, and constrain the recognition process of each individual column.

Weighted Finite State Transducers have been proposed for speech recognition [14,16], and are similarly useful for HTR/OCR. However, the freely available libraries that implement WFST and their integration into a recognition engine are not very well documented while they require many steps. Thus, we developed a framework called SIGRA (SImple ctc GRAmmar toolkit) that regroups two modules :

[10] Deceased people database since 1970 (INSEE, in French) : https://www.insee.fr/fr/information/4190491

[11] http://www.toponymiefrancophone.org/divfranco/Bougainville/Liste_generale.aspx?nom=liste_pays

- ”py-ctc-wfst-composer” : a WFST generator for decoding CTC probability lattices
- ”py-ctc-wfst-decoder” : a module that allows to use a WFST to decode CTC lattices

Based on three existing open source components, Thrax[18], OpenFST[1] and Kaldi[16], this framework is user-friendly, with a Python API that implements the required pipeline in RAM instead of writing multiple temporary files on disk, which is time-consuming. This framework is open source and available on Gitlab[12]. Each grammar is thus defined and compiled into a Weighted Finite State Transducers using Thrax and exported using the OpenFST format using the ”py-ctc-wfst-composer” module. Then, the obtained WFSTs are decoded by the ”py-ctc-wfst-decoder” module that uses Kaldi to explore the optical lattice, and find the best recognition path over the lattice.

Finally, by introducing the grammar decoding stage, multiple positive aspects are expected: 1- should the optical model misrecognize a character, lattice decoding may recover the correct character 2- simple but effective *rejection rules* can be implemented when too much discrepancy is detected between the optical best path and the grammar best path. The rejection rules are based on a threshold on the Levenshtein distance between the output string of the grammar and the output string of the optical model. Each threshold has been defined individually for each column. 3- abbreviations can be detected and normalized afterwards to a unique form.

Family names are not effectively rejected by the rejection rule because the list of surnames is huge and a family name with a misrecognized character can often be another valid family name. However, this is not a serious issue because it does not prevent to find individuals with close surnames and thus to remove the ambiguity on the individuals on the basis of other information such as year of birth, place of birth, occupation, marital status etc... Also notice that the recognition hypothesis of the rejected fields are often close to the results even if rejected. This is why they are stored with a specific tag in the final CSV exported files, so that they can be used later on by historians, even if the result was not validated by the system. To illustrate the benefit of using grammars we have introduced four metrics as follows:

- *Field Rejection Rate (FRR)*: ratio of rejected fields over the total number of fields
- *Accepted Field Error Rate (AFER)*: ratio of incorrectly recognized fields among the accepted fields over the total number of accepted fields.
- *OCR-Field Error Rate (OCR FER)*: percentage of erroneous fields at the output of the OCR (no rejection applied).
- *Grammar Field Error Rate (Grammar FER)*: percentage of erroneous fields provided by the grammar (no rejection applied).

Figure 7, shows the performance on test set of the generic dataset regarding these metrics. The rejection strategy is overall effective because the AFER is

[12] https://gitlab.com/users/nkpf/projects/sigra

lower than the OCR-FER and the Grammar-FER. However, the results are very different depending on the column. The lower error rates are obtained on the columns with restricted lexicons such as ”marital status”, ”birth year” or ”nationality” whereas higher error rates are obtained on columns with complex or infinite lexicons such as ”names” or ”occupation”.

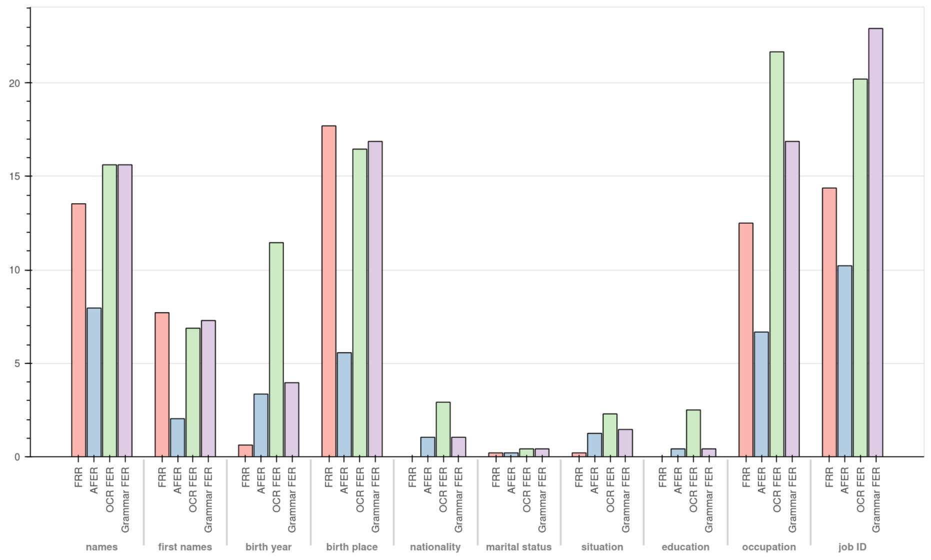


Fig. 7: Field performance metrics on the generic test dataset for each column category

6.2 Normalization and logical deductions

In this project, we wanted not only to obtain the textual content in the census but also to make this content usable by historians. That is why after the text recognition, we apply a step of normalization that standardizes the columns combined with logical deductions. This step is necessary because as explained in subsection 2.1, the columns were not filled in the same way depending on the census agents. This normalization is made possible by the grammars that detect all existing forms for a given term. Beside normalizing the content of the cells, we also applied logical deductions to enrich the database and facilitate the work of the historians. For example, some writers use the Latin formula *ditto* in a cell to indicate that its content is similar to that of the cell above. That is why, we also perform tabular operations such as deducing the corresponding value of each *ditto*.

7 Processing time

In this part, we detail the time of each processing step to process the whole 1926 census, namely 50000 double pages, 100000 simple pages and 3 million lines. Processing times were measured on a machine with an Intel Core I7 CPU and a Nvidia Tesla V100 GPU with 16Gb memory. The total processing time of the whole pipeline amounts to 67 hours approximately. However, since each double page is processed independently, the processing chain may easily be parallelized among several machines. A total of 8.68 hours were spent on page segmentation, 0.52 hours on page classification, 24.16 hours on line segmentation, 22 hours on model predictions, 3 hours on grammar, and 9 hours on normalization and post-processing. Among the different steps of the processing pipeline, the line segmentation step is the most time consuming. Adapting our architecture to perform paragraph-based recognition could thus save a considerable amount of time by eliminating the line segmentation step.

8 Conclusion

We have successfully extracted handwritten alphanumeric information from three censuses of the city of Paris representing a total of 300,000 pages and 9 million individuals. The processing chain relies on state-of-the-art components based on deep networks. The contribution of self-training allowed us to benefit from the unannotated corpus to improve by 2.5% CER (40% of relative improvement) the performance of the optical model. Improvements are expected on this point by exploring alternative strategies to the self-training that has been studied here.

Thanks to weighted state automata it was possible to efficiently constrain and control the recognition process and to reject wrong recognition hypothesis. The knowledge modeling step allowed the integration of external knowledge in the form of demographic and historical databases in order to model the expected contents using regular expressions and dictionaries.

The processing chain could easily process the censuses carried out everywhere in France between the end of the 19^{th} century and the beginning of the 20^{th} century because the same census procedures, the same information, and the same tabular registers were used at that time.

We make available to the community a new labeled manuscript corpus of 6720 lines in total. We also release a flexible library in Python allowing to easily decode recognition lattices under lexical and grammatical constraints by integrating known open source components that have remained difficult to integrate together until now for handwriting recognition.

References

1. Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., Mohri, M.: Openfst: A general and efficient weighted finite-state transducer library. In: Implementation and Application of Automata. pp. 11–23. Springer Berlin Heidelberg (2007)

2. Aradillas, J.C., Murillo-Fuentes, J.J., Olmos, P.M.: Boosting offline handwritten text recognition in historical documents with few labeled lines. arXiv:2012.02544 (2020)
3. Bluche, T.: Joint Line Segmentation and Transcription for End-to-End Handwritten Paragraph Recognition. arXiv:1604.08352 (2016)
4. Bluche, T., Louradour, J., Messina, R.: Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. arXiv:1604.03286 (2016)
5. Chowdhury, S., Garain, U., Chattopadhyay, T.: A Weighted Finite-State Transducer (WFST)-Based Language Model for Online Indic Script Handwriting Recognition. In: International Conference on Document Analysis and Recognition. pp. 599–602 (2011)
6. Coquenet, D., Chatelain, C., Paquet, T.: Handwritten text recognition: from isolated text lines to whole documents. In: ORASIS 2021 (2021)
7. Coquenet, D., Chatelain, C., Paquet, T.: End-to-end Handwritten Paragraph Text Recognition Using a Vertical Attention Network. arXiv:2012.03868 (2020)
8. Coquenet, D., Chatelain, C., Paquet, T.: Recurrence-free unconstrained handwritten text recognition using gated fully convolutional network. International Conference on Frontiers in Handwriting Recognition pp. 19–24 (2020)
9. Coquenet, D., Soullard, Y., Chatelain, C., Paquet, T.: Have Convolutions Already Made Recurrence Obsolete for Unconstrained Handwritten Text Recognition? In: ICDAR Machine Learning Workshop. pp. 65–70. IEEE, Sydney, Australia (2019)
10. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009)
11. Gay, V.: TRF-GIS Communes (1870–1940) Type: dataset
12. Grüning, T., Leifert, G., Strauß, T., Michael, J., Labahn, R.: A Two-Stage Method for Text Line Detection in Historical Documents. International Journal on Document Analysis and Recognition **22**(3), 285–302 (2019)
13. Michael, J., Labahn, R., Grüning, T., Zöllner, J.: Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition. arXiv:1903.07377 (2019)
14. Mohri, M., Pereira, F., Riley, M.: Weighted finite-state transducers in speech recognition. Computer Speech & Language **16**(1), 69–88 (2002)
15. Oliveira, S.A., Seguin, B., Kaplan, F.: dhSegment: A generic deep-learning approach for document segmentation. International Conference on Frontiers in Handwriting Recognition pp. 7–12 (2018)
16. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely, K.: The kaldi speech recognition toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society (2011)
17. Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? In: International Conference on Document Analysis and Recognition (ICDAR). vol. 01, pp. 67–72 (2017)
18. Roark, B., Sproat, R., Allauzen, C., Riley, M., Sorensen, J., Tai, T.: The OpenGrm open-source finite-state grammar software libraries. In: Proceedings of the ACL 2012 System Demonstrations. pp. 61–66. Association for Computational Linguistics, Jeju Island, Korea (Jul 2012)
19. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597 (2015)
20. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv:1801.04381 (2019)

21. Schall, M., Schambach, M.P., Franz, M.O.: Multi-dimensional Connectionist Classification: Reading Text in One Step. In: International Workshop on Document Analysis Systems (DAS). pp. 405–410 (2018)
22. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. Journal of Big Data **6**(1), 1–48 (2019)
23. Stuner, B., Chatelain, C., Paquet, T.: Self-training of blstm with lexicon verification for handwriting recognition. 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) **01**, 633–638 (2017)
24. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv:1905.11946 (2020)
25. Tensmeyer, C., Davis, B., Wigington, C., Lee, I., Barrett, B.: PageNet: Page Boundary Extraction in Historical Handwritten Documents. arXiv:1709.01618 (2017)
26. Thomas, P.: Semi-supervised learning by olivier chapelle, bernhard schölkopf, and alexander zien (review). IEEE Transactions on Neural Networks **20**, 542 (2009)
27. Voigtlaender, P., Doetsch, P., Ney, H.: Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In: International Conference on Frontiers in Handwriting Recognition. pp. 228–233 (2016)
28. Xie, Q., Luong, M.T., Hovy, E., Le, Q.V.: Self-training with noisy student improves imagenet classification. arXiv:1911.04252 (2020)
29. Yalniz, I.Z., Jégou, H., Chen, K., Paluri, M., Mahajan, D.: Billion-scale semi-supervised learning for image classification. arXiv:1905.00546 (2019)
30. Yousef, M., Bishop, T.E.: OrigamiNet: Weakly-Supervised, Segmentation-Free, One-Step, Full Page Text Recognition by learning to unfold. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14698–14707. IEEE (2020)
31. Yousef, M., Hussain, K.F., Mohammed, U.S.: Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. Pattern Recognition **108**, 107482 (2020)
32. Zou, Y., Yu, Z., Liu, X., Kumar, B.V.K.V., Wang, J.: Confidence regularized self-training. arXiv:1908.09822 (2020)