Philologie numérique : constituer un corpus

Atelier *Humanités numériques*

Jean-Baptiste Camps

École nationale des chartes | Paris, Sciences & Lettres

Casa de Velázquez
Madrid, 9 octobre 2018

Acquisition du texte

Dans la constitution d'un corpus de textes, la première phase est bien sûr l'acquisition du contenu des textes envisagés.

Transcription des témoins, selon les critères scientifiques du projet (transcriptions allographétiques, graphématiques, normalisées ; édition critique ; etc.). Méthode souvent la plus sûre, mais aussi la plus lente ;

“Transcription” assistée par ordinateur en utilisant un algorithme permettant la reconnaissance optique de caractères (*optical character recognition* ou OCR) imprimés, ou la reconnaissance des écritures manuscrites (*handwritten text recognition* ou HTR).

Téléchargement de textes depuis des corpus en lignes, des sites d'édition électronique, des bases de données d'éditeurs, etc.

Reconnaissance des écritures manuscrites

Reconnaissance optique des caractères (imprimés)
Optical character recognition (OCR)

- “problème résolu” de l’informatique ;
- aisé d’obtenir des taux d’erreur caractère (CER) $< 2\%$;
- outils libres : `Tesseract` 4, … ;
- existence de modèles génériques (par langue).

Reconnaissance des écritures manuscrites
Handwritten text recognition (HTR)

- très peu fonctionnel jusqu’à ces dernières années ;
- nouveaux développements : IA (réseaux de neurone récurrents LSTM…) ;
- outils libres : `OCRopy`, …
- modèles spécifiques à entraîner (pour chaque main, écriture,…).

Les étapes

1. traitement des images ;
2. analyse de la mise en page et identification des lignes ;
3. reconnaissance des caractères ;
4. d’éventuels post-traitements, visant à améliorer les résultats.

Plan

Transkribus

- développé par un consortium de recherche européen (projet READ ; Univ. Innsbruck et al.) ;
- financé par la Commission Européenne (Horizon 2020) ;
- permet de charger des images, analyser la mise en page, segmenter...
- opérations réalisées sur les serveurs de Transkribus.

Installer Transkribus

- Se rendre sur : `https://transkribus.eu/Transkribus/` ;
- se connecter (ou créer un compte) et télécharger le logiciel ;
- extraire l’archive ;
- lancer le logiciel (`Transkribus.sh` ou `Transkribus.exe`) ;
- tutoriel : `https://transkribus.eu/wiki/images/7/77/How_to_use_TRANSKRIBUS_-_10_steps.pdf` ;
- wiki : `https://transkribus.eu/wiki/index.php/Main_Page`.

T.P. Transkribus

1. se connecter ;
2. importer un document (choisir un ou deux fol. du dossier `digby_23`) ;
3. lancer l’analyse de la mise en page ;
4. corriger la segmentation
   - régions de texte,
   - zone de ligne,
   - ligne de référence,
   - mots,… ;

   si besoin en
   - étendant le rectangle ;
   - déplaçant les points.

   N.B. : il est possible de choisir les types de zone que l’on veut afficher ou non ;
5. commencer à transcrire ;
6. ajouter quelques balises et compléter quelques métadonnées ;
7. sauvegarder la transcription ;
8. consulter la source ;
9. exporter le document en TEI (attention au “tag abuse” de l’export par défaut).

Plan

1. Un outil “tout en un” : Transkribus

2. Pas-à-pas avec ScanTailor et OCRopy
   - Traitement des images
   - Analyse de la mise en page
   - Reconnaissance des écritures manuscrites

Dans une démarche de reconnaissance des écritures, la qualité des images et de leurs traitements est cruciale.

Besoins :

1. images en 300 DPI ;
2. redressées, débruitées ;
3. binarisées.

Outils : logiciels de traitement d’image, par ex. `ScanTailor`

T.P. ScanTailor

1. Démarrer un nouveau projet ;
2. charger les images du dossier `digby_23` ;
3. suivre les différentes étapes dans le logiciel ;
4. exporter en tiff binarisé 300 DPI.

Analyser la mise en page

Identifier

- zones de texte ;
- décoration ;
- colonnes ;
- lignes ;
- mots ;
- lettres.

Approches

- Sans apprentissage, par ex.
  - `OCRopy` 1 ;
  - Oriflamms (IRHT) ;
- fondée sur des méthodes d’apprentissage (IA), par ex.
  - `OCRopy` 2 ?

Installer OCRopy


```
$ git clone https://github.com/tmbdev/ocropy.git
$ cd ocropy
$ virtualenv ocropus_venv/
$ source ocropus_venv/bin/activate
$ pip install -r requirements.txt
$ python setup.py install
```

Analyser la mise en page avec OCRopy

Depuis la racine du dossier :

```
#Binarisation
$ ./ocropy/ocropus-nlbin tif/* -o book
#Segmentation en lignes
$ ./ocropy/ocropus-gpageseg -n book/*.bin.png^^I
```

Vérifier la qualité du résultat.

Différentes solutions techniques

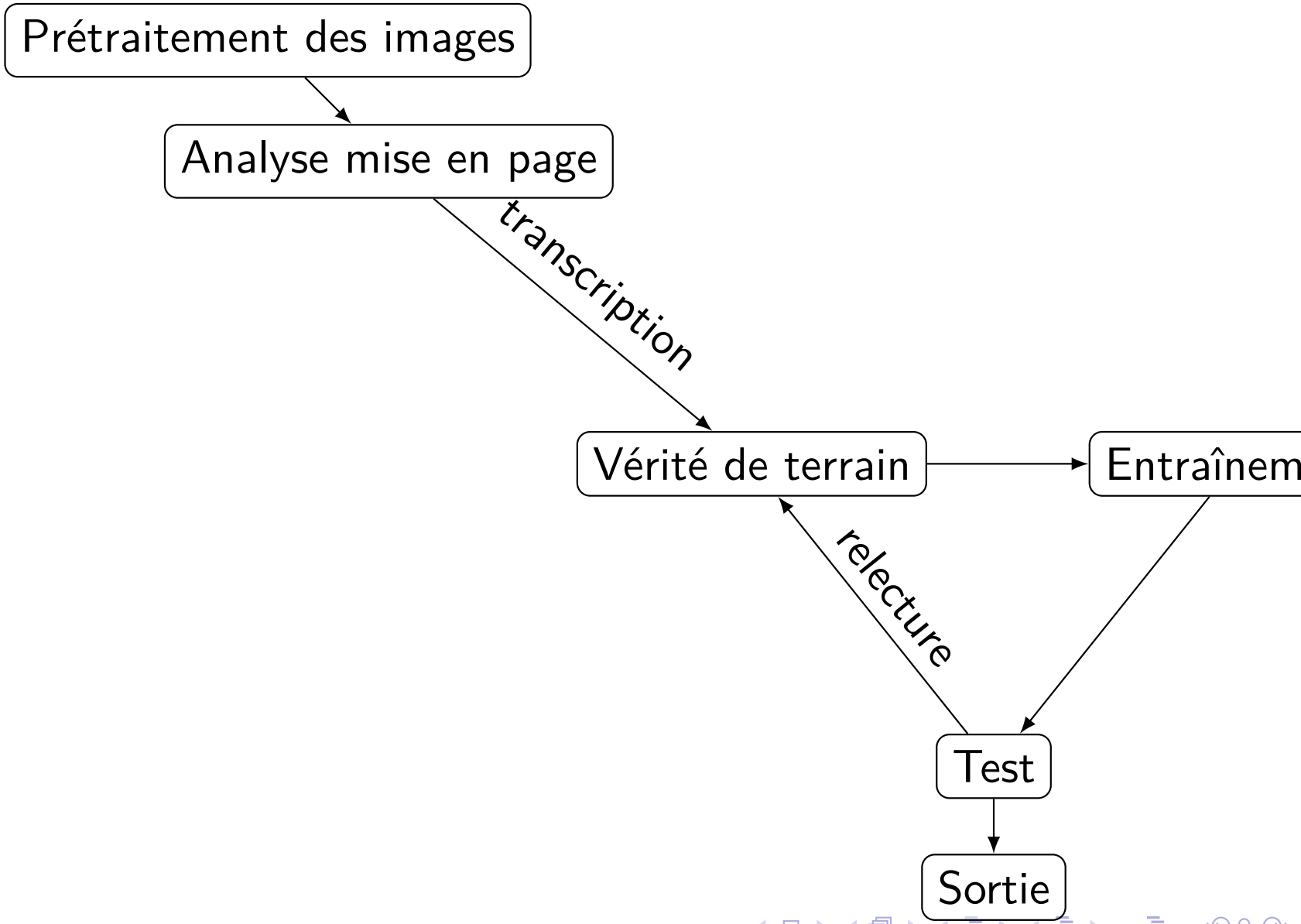
- approches segmentées ou non segmentées ;
- mesures de distance ; méthodes statistiques (chaînes de Markov) ou d’intelligence artificielle (réseaux de neurones convolutifs ou récurrents, LSTM 1D, LSTM 2D, etc.) ;
- outils directement opérationnels ou nécessitant un entraînement.

Ocropy et CLSTM

Outils développés par Thomas M. Breuel (`https://github.com/tmbdev/ocropy` ; `http://github.com/tmbdev/clstm`).

- approche non segmentées ;
- réseaux de neurones récurrents (LSTM) ;
- *open source* et nécessitant l’entraînement d’un modèle.

Ocropy : un apprentissage guidé



Ocropy : un apprentissage guidé

Déclaration de caractères

chars.py

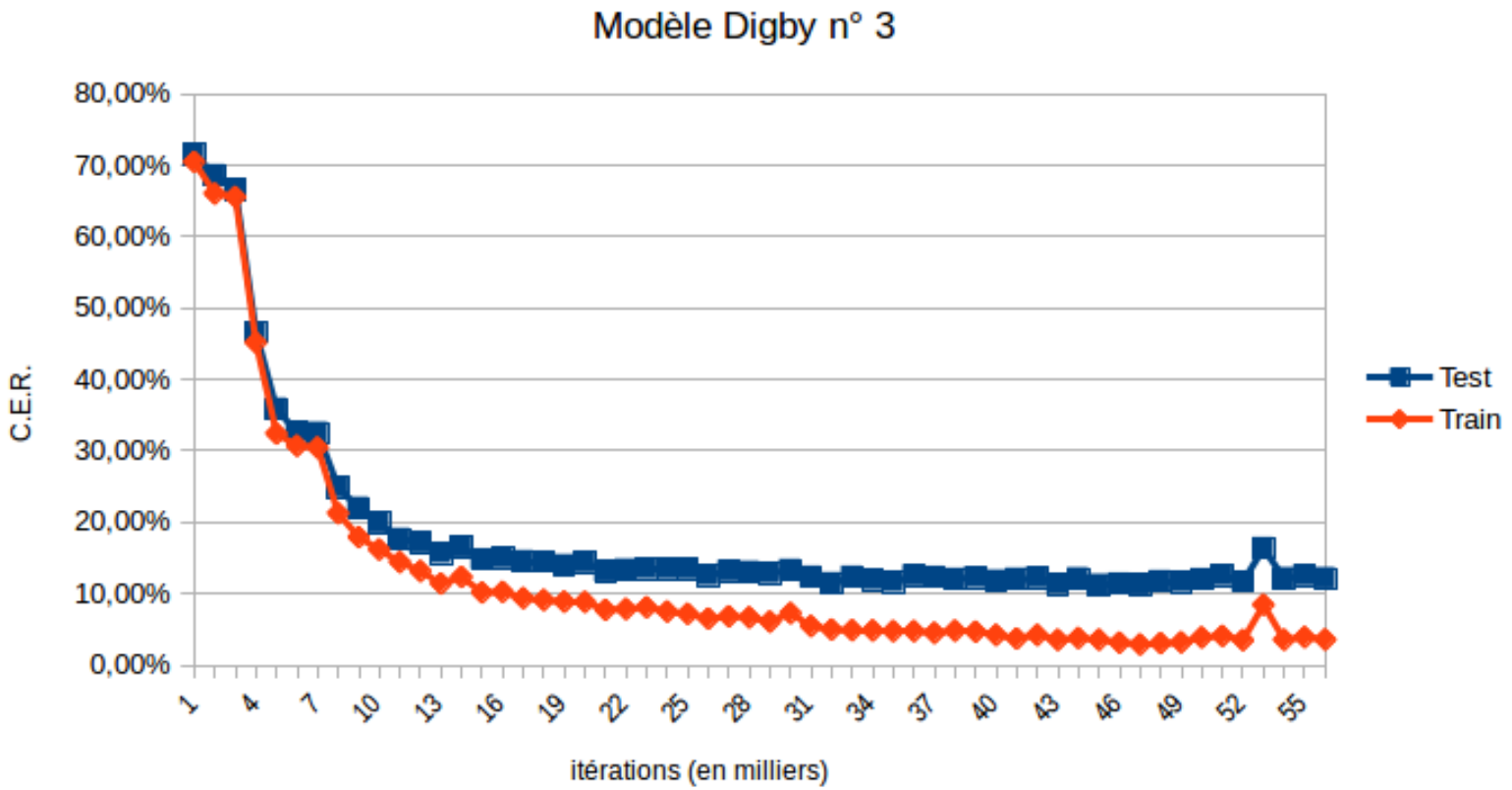
```
# -*- encoding: utf-8 -*-

import re

# common character sets

digits = u"0123456789"
letters = u"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
symbols = ur"""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
ascii = digits+letters+symbols

xsymbols = u"""€¢£»«›‹÷©®†‡°••◦‣¶§÷¡¿▪▫"""
german = u"ÄäÖöÜüß"
french = u"ÀàÂâÆæÇçÉéÈèÊêËëÎîÏïÔôŒœÙùÛûÜüŸÿ"
turkish = u"ĞğŞşıſ"
greek = u"AαBβΓγΔδEεZζHηΘθIιKκΛλMμNνΞξOoΠπPρΣσςTτYυΦφXχΨψΩω"

gbank = 
    u"ㄲㅚㄴ ㅆㅘㅙㄳㄵㄶㄺㄻㄼㄽㄾㄿㅀㅄㅍㅎㄲㄸㅃㅆㅉㅗㅙㅛㅜㅠㅡㄱㄴㄷㄹㅁㅂㅅㄼ ㅖ ㅓ ㅖħħ

medieval = u" ãñõıũȷꝺΛẟẏẽꝛꝑꝓ₂₉ſ⁹"

default = ascii+xsymbols+german+french+gbank+medieval
```

NB : avec CLSTM, cette étape n'est plus nécessaire.

Copier le fichier `chars.py` qui se trouve à la racine du dossier à la place de celui qui se trouve dans `ocrolib`.

Ocropy : un apprentissage guidé

Entraînement sur un ms. (ici Roland d’Oxford)



Ocropy : un apprentissage guidé

Test

```
19 Doctrinal-00010000.pyrnn.gz
20 errors          313
21 missing          84
22 total         14404
23 err           2.173 %
24 errnomiss     1.590 %
25 Doctrinal-00012000.pyrnn.gz
26 errors          285
27 missing          84
28 total         14404
29 err           1.979 %
30 errnomiss     1.395 %
```

```
120 errors          80
121 missing          0
122 total         5741
123 err          1.393 %
124 errnomiss    1.393 %
125 4
126 4       ſ        l̄
127 4
128 3       e̅        -
129 3       e        c
130 3
131 3       t̄        f
132 2       t        c
133 2       ̄        -
134 2       c̄        t
135 0.013934854555
```

Ocropy : un apprentissage guidé

Résultats

- pour un imprimé ancien, des taux d'erreur de l'ordre de 1% sont atteignables...
- pour un ms., des taux inférieurs à 10% sont atteignables avec seulement 400 lignes d'entraînement environ (cas du Digby 23, jusqu'à 7% de CER).

Confusions fréquentes (ms.)


```
digby4/test3-138000.clstm
errors          193
missing           0
total         14767
err           1.307 %
errnomiss     1.307 %
25   s   S
17   _   .
14       _
```

T.P. OCRopy : Préparation des données

1. Essayer d’appliquer un modèle déjà entraîné

```
#Lancer la reconnaissance
$ ./ocropy/ocropus-rpred -n -m
./ocropy/models/digby23-00106000.pyrnn.gz
book/*/*.bin.png
#Extraire le résultat
$ ./ocropy/ocropus-gtedit html -H35
book/*/*.bin.png -o gt.html
```

Pas terrible... Mais comment améliorer le modèle ?

2. Corriger / transcrire

T.P. OCRopy : Entraînement

3. Extraire et entraîner

```
#Extraire
$ ./ocropy/ocropus-gtedit extract gt.html
#Normalisation des caractères
$ for f in book/*/*.gt.txt; do uconv -f utf8 -t utf8
-x nfc -o "${f/gt.txt/gtneu.txt}" "$f"; done
$for f in book/*/*.gtneu.txt; do mv "$f"
"${f/gtneu.txt/gt.txt}"; done
```

Puis placer 90% des lignes corrigées dans un dossier `train` et 10% dans un dossier `test`.
On peut ensuite lancer un entraînement (de zéro ou à partir du modèle précédent, aux choix).

```
#Lancer l'entraînement
$ ./ocropy/ocropus-rtrain -o digby
-d 1 train/*/*.bin.png
```

T.P. OCRopy : Test des résultats

4. Tester les résultats

Une fois que l'entraînement a atteint un niveau satisfaisant, on peut tester la qualité du résultat,

```
# Calcul des erreurs de différents modèles, comparati
for i in *.pyrnn.gz; do
echo "$i" >> modeltest
./ocropy/ocropus-rpred -n -m $i test/*/*.bin.png
./ocropy/ocropus-errs test/*/*.gt.txt 2>>modeltest
done
# Confusions de caractères
./ocropy/ocropus-econf test/*/*.gt.txt
```