L'apprentissage pour Tesseract (« training »).

Pour un schéma général en français, voir http://doc.ubuntu-fr.org/tesseract-ocr

logiciels utilisés :

jTessBoxEditor - pour créer le fichier d'apprentissage
VietOCR.net - une interface Windows pour Tesseract (ne pas installer Tesseract séparément, il est inclus dans VietOCR, l'installer séparément créé des conflits. Si vous l'avez fait, désinstaller Tesseract et supprimez la variable d'environnement dont il se sert).
Ces deux programmes ont été réalisés au départ pour le vietnamien,d'où le nom.

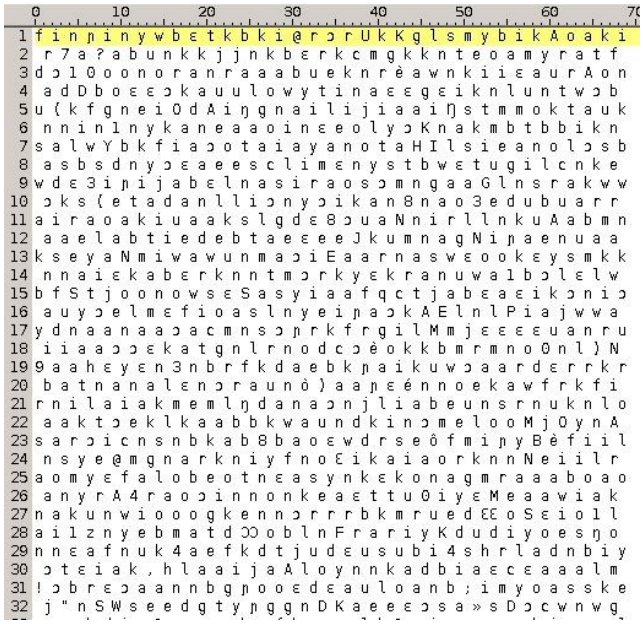
jTessBoxEditor présente 3 onglets « Trainer » (nous l'utiliserons à la fin pour générer le fichier dont a besoin VietOCR (ou Tesseract), « Box Editor » (nous ne l'utiliserons pas ici car nous optons pour la génération automatique d'images), et « TIFF/Box Generator ». Nous commencerons par ce dernier onglet ; vous aurez besoin en entrée d'un texte représentatif et des polices de caractères installées sur l'ordinateur. Si cet ordinateur ne contient pas les polices capables de représenter les caractères Unicode dont vous avez besoin (ça peut être le cas sur Windows XP), il faut envisager de faire les traitements sur un autre ordinateur.

Préparation du texte « représentatif »

: « The quick brown fox jumps over... »
Ce texte sera transformé en une image à l'aide de différentes polices de caractères et d'effets (italiques, gras) : la confrontation des deux est la base de l'apprentissage automatique. Pour améliorer la performance de reconnaissance en situation réelle, il est recommandé d'utiliser une grande taille de police (ici le défaut est 36pt) et de « brouiller » un peu l'image (ajouter du bruit). Il doit y avoir au moins 20 exemplaires des caractères les plus courants (ex : k en bambara) et 5 exemplaires des caractères rares (ex : ŋ) . Ne pas mettre les caractères spéciaux (parenthèses etc) les uns derrière les autres, mais au milieu du texte (comme dans un texte naturel et comme les parenthèses ici) afin que les repères de placement par rapport à la ligne normale des caractères soit bien détectés.

Faut-il mettre un texte naturel (ou plusieurs), ou un texte artificiel mais représentant tous les caractères et leur ponctuation ? La documentation n'est pas claire là-dessus. Je suis en faveur de la première solution, mais mon expérience, et la façon de faire des autres, montrent plutôt des suites de caractères presque aléatoires (en réalité le plus souvent parsemés de mots américains … même pour le vietnamien!!!). Et espacés artificiellement afin que chaque caractère soit isolé (certains caractères peuvent empiéter sur la boîte [« box »] des caractères à gauche et à droite, par exemple un beau « f » en italique), et avec des lignes assez longues pour que le « training » isole bien la « ligne de base » de repère.
J'ai opté pour … le deux ! La première partie, artificielle, devrait utiliser la fréquence d'occurrence dans le Corpus bambara. Soit, pour 1 « ŋ », il faudrait 724 « a », 491 « n », 364

« i », 330 « k »... 219 « ɛ », 141 « ɔ », 28 « ɲ » ; par ailleurs il me semble utile quand même d'inclure les caractères qui ne sont pas dans l'alphabet bambara standard mais qui apparaissent quand même dans les textes, comme « q », « v », « x ». Ainsi que les majuscules, même si elles sont inexistantes dans les textes des années 1970, È, Ò, Ŋ mais aussi toutes les autres, selon leur statistiques dans le Corpus, les lettres accentuées et cédilles, provenant soit de l'utilisation ponctuelle de mots étrangers, soit des restes de l'alphabet ancien en particulier le « ò ». Reste à ne pas négliger toutes les ponctuations et caractères auxiliaires, que j'avais oublié dans un premier essai, cela s'est vu tout de suite dans les résultats, en les mélangeant aléatoirement aux lettres de l'alphabet, et les chiffres. Je ne fait pas des lignes continues de chacun de ces caractères « aaaaaaaaaaa » : en effet l'OCR ne saurait repérer la ligne de base pour les lettres descendantes comme le « j » ou le « g », si la ligne des « jjjjjjjj » de donne pas ce repère, et autres problèmes de positionnement. J'ai choisi de mélanger aléatoirement tous les caractères (des 724 « a » aux quelques ponctuations), soit environ 2 pages pleines de caractères (espacés) en lignes continues de 70 caractères.



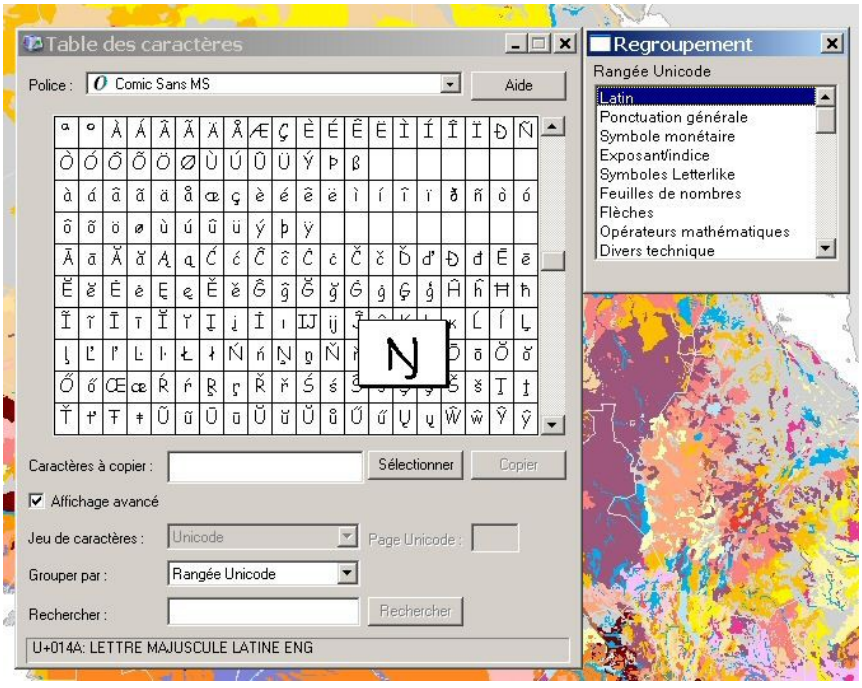
Cette façon de procéder diminue considérablement le nombre de messages d'avertissement « FAILURE could not find a matching box ».
Pour les textes naturels, j'ai opté pour 2 conte et 2 extraits de Kibaru. J'ai intégré les caractères spéciaux à un de ces textes, ajouté quelques ponctuations plus rares comme le « ; », et ajouté quelques majuscules qui ne sont pas dans les textes naturels, mais pour lesquelles on a quand même besoin d'un « apprentissage » (training).
Faites comme vous le sentez mais il est très important que tous les caractères qui doivent être reconnus y soient ; par exemple, pour construire le modèle "OCR bambara avec les tons" j'ai entré - à la main - toutes les configurations possibles des marques diacritiques sur les voyelles et aussi le n et le w, ainsi que de nombreuses situations avec l'article tonal, à distinguer du ton bas.

LIMITATION : Il y a toutefois une limite : le fichier TIFF généré par jTessBoxEditor est assez

gros, veillez à ce qu'il ne dépasse pas les 5 pages (environ 42 mo!) car pendant la procédure d'apprentissage, vous aurez l'erreur "fatale" (mettant fin au traitement) "Java Heap Size" ; soit apparemment un dépassement des possibilités de traitement dans la mémoire de l'ordinateur (testé sur une machine avec 4Go de mémoire).

Choix des polices : il faut s'assurer que ce sont des polices qui contiennent bien les caractères en question. Attention, Windows et les traitements de texte moderne utilisent une technique de substitution, si bien que la police « Comic sans », qui ne contient pas[1] le dessin des lettres ɛ ɔ et ɲ, semblera les avoir : votre traitement de texte utilisera pour ces lettres le dessin qui existe dans une autre police, par exemple Arial (sous Windows XP, la police peut être plus complète sous Windows 8)
Le logiciel d'apprentissage (training), lui, les affichera avec des « carrés » ou affichera des espaces vides : bien surveiller ces accidents et éviter ces polices, ne prendre que les polices qui contiennent tous les caractères Unicode dont vous avez besoin.

On peut le vérifier plus en détail avec l'utilitaire « Table des caractères », en recherchant en fonction du code du caractère, par exemple :
ŋ = U+014b et Ŋ = U+014a existent bien pour « Comic sans » :



1 sous Windows XP. Peut avoir évolué pour les versions ultérieures !

Créer un sous-répertoire : Il va falloir y accumuler tous les fichiers nécessaires. Lui donner de préférence un nom standardisé.
Pour nous ce sera « bam », et pour la version ancienne « bam-old »

Les fichiers demandés par Tesseract : tous les fichiers sont en UTF-8 (sans BOM)

- des couples de fichiers « .tif/.box » : Chaque couple correspond à une police, les deux fichiers portent le même nom, seule l'extension les distingue : 1 fichier avec l'image du texte (.tif), accompagné d'une fichier contenant tous les paramètres de reconnaissance (.box) : par exemple bam-old.arial.exp0.tif et bam-old.arial.exp0.box;
Ces fichiers sont générés dans l'onglet « TIFF/Box Generator » du programme jTessBoxEditor.
1) Input : choisir le fichier du « texte représentatif »
2) Output : indiquer le nom de votre langue. Pour nous ici « bam-old », ou « bam » (attention, cette mention est à préciser également sur la page « Training », les deux doivent être cohérentes !
3) police : ne pas taper le nom dans la boite, choisir à l'aide du bouton à droite de la zone de saisie. Choisir une grande taille de préférence (le défaut de 36pt convient très bien, ne pas changer).
Le nom dans la boite de saisie est le nom du fichier généré : « arial.exp0.tif ». Il n'est pas conseillé d'y toucher, sauf éventuellement la partie qui indique « exp0 » si vous faites plusieurs versions à partir de la même police : versions avec des « textes représentatifs » différents, ou version avec des bruits de fond (« noise ») différents.
→ c'est avec 2) et 3) que seront créés les noms des fichiers de chaque paire .tif/.box
4) anti-aliasing : améliore le dessins des caractères en particulier l'effet d'escalier sur les caractères en italiques. Je coche toujours cette case. Je ne vois pas de raison de ne pas le faire si l'on compte travailler sur des textes imprimés. Si en revanche on compte travailler massivement sur des textes sortis d'imprimantes de mauvaise qualités ou de fax, ou de copies d'écran, il est peut-être intéressant de garder l'effet de pixellisation dit « effet d'escalier ».
5) noise : il s'agit de détériorer un peu l'image générée pour simuler les défauts de qualités existants dans la vie courante (papier de mauvaise qualité, mauvais encrage...). Il est conseillé d'en mettre. J'ai mis 5 (le maximum) partout.
6) Letter tracking : j'ai laissé ces valeurs par défaut, ne sachant pas bien à quoi cela mène.
Finalement : le bouton « Generate » : c'est lui qui créée les paires de fichiers .tif et .box.
Cela prend quelques secondes !... par police (enfin parfois quelques minutes, tout dépend de la taille de l'image générée, donc de la taille du texte fourni)
Il faut ensuite un peu de patience pour balayer toutes les polices que l'on pense retrouver dans les documents à traiter. Si c'est un usage général, il est conseillé d'équilibrer le nombre de polices « serif » et « sans serif », sans oublier une ou deux polices à espacement fixe.
Dans cet essai j'ai utilisé jusqu'à 17 polices :
5 serif : Calibri, Charis SIL, Doulos SIL, Garamond, Times New Roman

8 sans serif : Arial, Century, Comic sans, Dejavu sans, Liberation sans, Lucida sans, Tahoma, Verdana
4 à espacement fixe : Courier new, Lucida console (IBM Letter Gothic), Monospaced, Special Elite (IBM Prestige Elite) - nb : IBM : machines électriques à boules des années 1970. On peut monter jusqu'à 32 polices en théorie. En pratique, il y a une autre limite de 64 fichiers au total. Si vous utilisez 16 polices dans leurs 4 styles : Normal, Bold, Italic, Bold-Italic, vous atteindrez cette limite.

Vous pouvez, mais ça n'est pas nécessaire, vérifier le bon fonctionnement de la reconnaissance en ouvrant le fichier .tif dans le « Box Editor » (onglet du milieu) mais ça n'est pas nécessaire, à ce qu'il semble.

COMBIEN DE POLICES ? Intuitivement on peut s'imaginer que l'apprentissage sera d'autant meilleur que le texte d'apprentissage sera long et varié, et que le nombre de polices sera élevé, de façon à couvrir le plus grand nombre de cas qui peuvent se présenter en pratique. Par expérience directe (ponctuations oubliées), la qualité du texte d'apprentissage est très importante... elle a aussi un gros impact sur les temps de traitements ! En revanche l'expérience est moins concluante avec le nombre et la variété de polices. Faut-il également inclure les 4 styles à chaque fois : Normal, Bold (gras), Italic, Bold Italic ? Rien ne le précise et il n'est pas improbable que le logiciel devrait pouvoir dériver les variantes de style à partir de la police Normale.
Idéalement, on pourrait imaginer fabriquer un fichier de training adapté au texte que l'on veut passer en OCR. Cela peut se justifier pour les très grands textes (« Guerre et Paix »,...)

AUTRES FICHIERS À CRÉER AVANT DE LANCER LE « TRAINER »

Attention : ces fichiers n'ont pas d'extension .txt. Si votre traitement de texte ou éditeur de texte ajoute une extension, vous devrez la supprimer.

- font properties : un fichier contenant la liste des polices et le codage de leur style (italique, gras...) : bam-old.font_properties dont le contenu est une ligne par police/style par exemple

```
        I B F S Fr
arial   0 0 0 0 0 (1ère ligne : la police simple : fichier bam-old.arial.exp0)
ariali  1 0 0 0 0 (la police en italique : fichier bam-old.ariali.exp0)
arialb  0 1 0 0 0 (la police en gras « bold » : fichier bam-old.arialb.exp0)
arialbi 1 1 0 0 0 (la police en gras+italique : fichier bam-old.arialbi.exp0)
times   0 0 0 1 0 (la police simple : fichier bam-old.times.exp0)
timesi  1 0 0 1 0 (la police en italique : fichier bam-old.timesi.exp0)
timesb  0 1 0 1 0 (la police en gras « bold » : fichier bam-old.timesb.exp0)
timesbi 1 1 0 1 0 (la police en gras+italique : fichier bam-old.timesbi.exp0)
```

Explications rapides :

```
<fontname> <italic> <bold> <fixed> <serif> <fraktur>
```

fontname : le nom de la police tel qu'il est écrit dans le fichier .tif (ou .box)

Par exemple pour reprendre l'exemple ci-dessus, « arial ».

italic : 0 si la police n'est pas en italique, 1 si elle est en italique

bold : 0 si la police n'est pas en gras, 1 si elle est en gras

fixed : police à espacement fixe, comme courrier : 1, sinon, 0

serif : 1 si c'est une police serif, c'est-à-dire avec de petits empâtements aux extrémités, comme Times, sinon 0 : par exemple 0 pour Arial, Helvetica, et toutes les polices marques « sans » qui signifie : sans serif.

fraktur : communément appelé « écriture gothique » (gothique allemande) - non utilisé ici.

Il est très important

1) que toutes les polices ayant une paire .tif/.box soit indiquées correctement dans ce fichier. Si vous renoncez à une police supprimez la du répertoire (ou déplacez les fichiers .tif/.box ailleurs) Par contre ce n'est pas grave s'il y a dans ce fichier des noms de polices pour lesquelles il n'y a pas (ou pas encore) de paire .tif/.box

2) que chaque police soit suivie de 5 indications 0 ou 1 (attention aux copiés/collés trop rapides!)

3) si vous avez une situation stable et qui marche bien avec 7/8 polices (ou moins), n'ajoutez des polices qu'avec précaution : vos bons résultats pourraient … se détériorer !

Les autres fichiers ci-dessous bénéficie de moins de documentation. Probablement parce qu'ils ne sont pas utilisés dans l'apprentissage (training), mais seulement lors de l'utilisation de l'OCR, encore que ce point ne soit pas très clair (cf. https://groups.google.com/forum/#!topic/tesseract-ocr/VJXE40iksnI).

Toujours est-il que d'après mes essais, ils sont TRÈS IMPORTANTS pour l'utilisation de l'OCR, les performances et la précision varient grandement en fonction de ces dictionnaires.

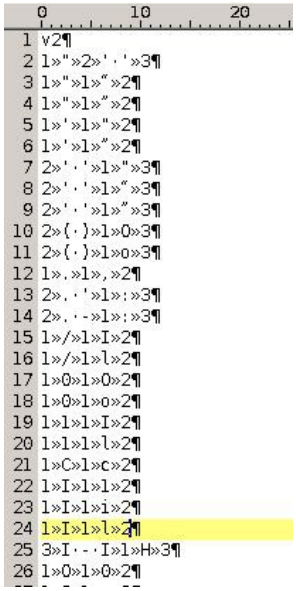
- frequent_words_list : un fichier des mots les plus utilisés : bam-old.frequent_words_list ;
Dans notre cas ce fichier est tiré des 1000 « formes » les plus utilisés dans le Corpus au 15.3.2014, trié mots-uniques (suppression des doublons), soit au total un peu près de 3000 mots si l'on prend leurs formes minuscules, majuscules et capitalisées. Soit un fichier de 17 ko (pour le Bangla dans le lien ci-dessus, le fichier fait 30ko)

- words_list : liste de mots, un fichier dictionnaire : bam_old.words_list ;
Bamadaba (+ yoro, togo, jamu) au 11.3.2014, transformé en bamlatinold pour le bam-old, trié mots-uniques (suppression des doublons), auxquelles ont été ajoutées toutes les formes trouvées dans le Corpus (désambiguïsé), soit environ 18.000 (verbes dérivés, mots composés, etc), sous toutes leurs formes minuscules, majuscules et capitalisées... soit au total plus de 55.000 mots, 562Ko (pour le Bangla 190 Ko, peut être par manque d'accès à un Corpus)

- unicharambigs : un fichier pour tenter de corriger les erreurs d'OCR les plus fréquentes, par exemple un « m » reconnu comme « rri » : bam-old.unicharambigs
Le « Trainer » ne réclame pas ce fichier, mais il sera inclus dans le fichier traineddata, donc il vaut mieux le prévoir avant. Il n'est pas parfaitement clair pour moi si ces fichiers sont uniquement utilisés par le « Trainer », ou s'ils ne sont qu'intégrés dans bam-old.traineddata our faciliter l'OCR plus tard (dans VietOCR), ou les deux.
Exemple simplifié
v2 (première ligne du fichier, n° de version)

```
1 S 1 s 0     (remplacer 1 S majuscule par 1 s minuscule - si ambiguïté (0) )
1 0 1 O 0     (remplacer 1 zéro par 1 O majuscule - si ambiguïté (0) )
1 0 1 o 0     (remplacer 1 zéro par 1 o minuscule - si ambiguïté (0) )
2 ' ' 1  " 1  (remplacer 2 apostrophes par 1 guillemet - toujours (1)  )
2 r n 1  m 1  (remplacer 2 apostrophes par 1 guillemet - toujours (1)  )
```

… plus de détail sur le web.
Attention : chaque groupe est séparé par des tabulations, visualisés ci-dessous par des » , et les espaces par des petits points (ligne 7 entre ' et ')



Le processus fonctionne comme suit, j'imagine : si le mot est absent du dictionnaire, une tentative de substitution est faite d'après ces indications, jusqu'à ce qu'un mot valide soit trouvé.

Deux autres fichiers semblent importants pour une bonne reconnaissance, bien que non obligatoires pour le fonctionnement du « trainer » de jTessBoxEditor :

- **punc** : les règles pour les ponctuations, dans un fichier bam-old.punc_list (qui sera transformé en bam-old.punc-dawg par le programme wordlist2dawg

- **number** : les règles pour faciliter le repérage des nombres, dans un fichier bam-old.number_list (qui sera transformé en bam-old.number-dawg par le programme wordlist2dawg. Par exemple, en français, on trouvera souvent un nombre avant le signe monétaire €, ce qui peut s'écrire :
…€
En revanche, en bambara, on trouvera le nombre après la monnaie, par exemple :
dɔrɔmɛ... La liste des mots les plus fréquents autour des nombres a été tirée du Corpus bambara (Corpus brut) : kilometɛrɛ, san, miliyɔn ...
On a cependant gardé les règles pour le français, les textes pouvant être parfois mixtes, et le risque de conflits entre règles semblant inexistant.

Les modifications de ces dictionnaires doivent être répercutées dans le fichier final bam-old.traineddata. Toutefois on peut éviter de relancer toute la partie initiale du traitement qui concerne l'apprentissage de la graphie des caractères. Il faut pour cela apprendre le fonctionnement de la commande « combine » (à lancer en dehors de jTessBoxEditor, dans une fenêtre DOS) : http://tesseract-ocr.googlecode.com/svn-history/trunk/doc/combine_tessdata.1.html

LE « TRAINER » (L'ENTRAÎNEUR)

Un fois tous les fichiers générés ou préparés, on peut commencer l'apprentissage :
Aller à l'onglet « Trainer » de jTessBoxEditor.

Vérifier les emplacements des fichiers :
- Tesseract executables : normalement rien à changer si vos n'avez rien déplacé depuis l'installation du programme.
- Training data : choisir un fichier (n 'importe lequel, le premier) parmi les fichiers des paires .tif/.box.
- language : taper le nom de votre langage, ici « bam-old »
- bootstrap language : laisser vide (il doit y avoir des usages particuliers que je ne connais pas, jai supposé qu'il s'agissait du language à utiliser par défaut lorsque le language défini ne marche pas, probablement c'est l'anglais)
- dans la liste déroulante qui suit, choisir « Train with existing data » (c'est à dire faire l'apprentissage à l'aide de tous les fichiers qui sont dans le répertoire).

Enfin, cliquer sur le bouton « Run », et s'armer de patience.

Le programme se lance dans des tonnes de calculs, patienter, cela dure environ de 10 à 30 minutes pour un petit essai, et jusqu'à 2h ou plus selon le nombre de fichiers à traiter (tif,box pour chaque police), la taille de ces fichiers (donc la taille du texte d'essai), la puissance de l'ordinateur, la rapidité du disque dur, … et si tout va bien vous récupérez à la fin votre fichier « traineddata » (données résultat de l'apprentissage), pour nous :
bam-old.traineddata
- 16 polices, 24mn de traitement, fichiers résultat 1027 Ko
- texte de 475 mots, 2152 caractères, 1 page tif / police, 11 polices, 19 mn, 1014 ko
- texte de 5817 mots, 15346 caractères, 5 pages tif/police, 12 polices,… 2h40mn de traitement! 1212 Ko

Si vous avez une « erreur fatale » au milieu de ce long traitement, pas de panique
- à l'étape « Shape clustering » qui est assez longue : la cause probable est une erreur dans le fichier font_properties : soit une police manquante (comme indiqué dans le message d'erreur), ou dont le nom est mal indiqué (century au lieu de centurygothic), soit un 0 ou un 1 manquant ou en trop, ou autre parasite inattendu de ce genre.
- à l'étape « MF training » : vérifiez que vous n'avez pas dépassé le limite de 64 paires de fichiers police-style (.tif/.box)
… bien, entendu, il vaut mieux faire les vérifications AVANT, car jTessBoxEditor recommence tout, y compris les analyses des box même si rien n'a changé !!!

La documentation sur ce qui se passe dans cette chaîne de traitement, en particulier « Shape clustering » et « MF training » manque, je ne peux faire l'ombre d'un commentaire là-dessus.

Bonne chance !