Manuel d'utilisation de Collecte

Philippe Perret

22 avril 2017

Table des matières

1 Le programme `Collecte`

Le programme `Collecte` permet de collecter les informations d'un film afin de produire des séquenciers, des synopsis, de sortir des statistiques, etc., tout fichier pouvant être utile pour les analyses.

1.1 Les trois modes d'utilisation de `Collecte`

— le mode d'aide : `collecte help`,
— le mode de parsing. Pour *parser* les fichiers de collecte
— le mode d'extraction. Pour extraire les fichiers des données collectées, on parle d'*extraction*.

```
$ collecte help[ pdf|html]

$ collecte parse[ options][ chemin/vers/dossier/collecte]

$ collecte extract [options][ chemin/vers/dossier/collecte]
```

Par exemple en ruby :

```
coll = Collecte.new('dossier/collecte')
coll.parse
coll.extract(:all)
```

2 Collecte des informations

La *Collecte* est la première opération à exécuter. Elle consiste à rassembler toutes les informations du film dans un dossier appelé *dossier de collecte*.
Ce dossier contient tous les fichiers de collecte.

2.1 Les fichiers de collecte

On peut trouver dans la dossier de collecte les fichiers suivants :
— le fichier des personnages (-> le fichier des personnages),
— le fichier des brins (-> le fichier des brins),
— le fichier des scènes (-> le fichier des scènes),
— le fichier des métadonnées (-> le fichier des métadonnées).

2.2 Dossier de collecte

Le « dossier de collecte » peut se situer n'importe où sur votre ordinateur. C'est dans ce dossier que seront produit les fichiers de données (Marshal ou PStore) ainsi que le dossier d'extraction où seront placés tous les fichiers extraits.

Sa hiérarchie normale est :

```
.../dossier_collecte/scenes.collecte
                    /personnages.collecte
                    /brins.collecte
                    /data/film.pstore
                    /extraction/...
                               /...
                               /...
```

2.3 Définition de la fin du film

Pour définir la fin du film, il faut impérativement terminer le fichier des scènes par :

```
H:MM:SS FIN
```

Dans le cas contraire, le programme prendra le temps de la dernière scène et lui attribuera arbitrairement une durée de 1 minute.

2.4 Temps réels et temps relatifs

Il faut comprendre que pour le programme il existe deux temps différents, les *temps réels* et les *temps relatifs*.

La divergence entre *temps réel* et *temps relatif* tient au fait que la première scène, dans la collecte, ne commence pas à l'horloge `0:00:00` mais après une amorce du film où sont présentés les producteurs et les diffuseurs.

Les *temps réels*, qui prennent pour référence une première scène qui commence vraiment à `0:00:00`, sont utilisés pour tous les temps des fichiers extraits de la collecte.

Pour la commande `collecte`, ce sont toujours des *temps réels* qu'il faut fournir, pas des *temps relatifs*. Il convient de prendre ces temps dans les fichiers extraits, notamment les séquenciers complets.

2.5 Collecte des scènes

Les scènes du film sont *collectées* dans un fichier du dossier de collecte :

```
<dossier collecte>/scenes.collecte
```

Grâce au bundle TextMate, ce fichier peut être créé à l'aide de la commande `Scènes : créer le fichier`.

2.5.1 Format général d'une scène

Dans le fichier `scenes.collecte`, une scène se présente sous la forme :

```
H:MM:SS LIEU EFFET Décor
Résumé de la scène. brins et notes
Premier paragraphe. brins et notes
Deuxième paragraphe. brins et notes
etc.
```

Par exemple :

```
1:25:52 INT. JOUR Maison de Joe
[PERSO#joe] rentre chez lui. b12 (25)
Joe se gare devant chez lui.
Joe sort de sa voiture et traverse le parking.
Joe rentre chez lui.
```

Ou, pour une scène *en parallèle* dans deux décors différents :

```
1:45:12 INT./EXT. JOUR/NUIT Vaisseau / Base terrestre
Le capitaine [PERSO#anatoli] lance un appel au secours
à la base. b45 (2)
(2) On joue ici sur le contraste jour/nuit.
```

Noter qu'il vaut mieux écrire l'intitulé de la scène en minuscule.
Il sera toujours converti en majuscule pour l'affichage.

Chaque scène doit être **impérativement séparée de la précédente par une ligne vide** :

```
0:00:06 INT. JOUR Maison
Première scène.

0:00:42 EXT. NUIT Jardin
La scène suivante doit être séparée par une ligne.
```

La fin du film se détermine par la ligne :

```
1:12:12 FIN
```

Cette ligne n'est pas comptée pour une scène.

2.5.2 Appartenance de la scène à des brins ou des notes

Les dernières lignes de la scène peuvent permettre de l'introduire dans des brins ou des notes (principalement des brins).

Cette ligne comportera simplement les marques de ces appartenances, avec `b<id brin>` pour les brins ou `n<id note>` pour les notes.

Par exemple, cette scène appartiendra au brin 12 et à la note 4 :

```
0:23:54 INT. JOUR Maison de Joe
[PERSO#joe] rentre chez lui.
b12 n4
```

2.5.3 Points structurels de la scène

Les *points structurels*, ce sont les `pivots`, les `clé de voûte`, le début du dénouement et autres début de la seconde partie de développement.

On peut indiquer la présence de tels points structurels en mettant sur une ligne, plutôt à la fin, la *marque du point structurel* **et seulement cette marque**. Par exemple :

```
0:23:54 INT. JOUR Maison de Joe
[PERSO#joe] rentre chez lui.
STT_CLE_DE_VOUTE
b12 n4
```

Si la scène contient plusieurs points structurels, ils doivent impérativement être spécifiés *sur des lignes différentes*. Par exemple :

```
0:12:54 INT. JOUR Maison de Joe
[PERSO#joe] rentre chez lui.
STT_INC_PERT
STT_INC_DEC
b12 n4
```

Avec le bundle TextMate, il suffit de taper `stt` puis tabulation pour obtenir la liste des tous les points structurels possibles et choisir celui qu'on veut insérer dans la scène.

Cette liste contient :

```
STT_INC_PERT      Incident perturbateur
STT_INC_DEC       Incident déclencheur
STT_PIVOT_1       Premier pivot
STT_DEV_PART1     Première partie de développement
STT_CLE_DE_VOUTE  Clé de voûte
STT_DEV_PART2     Seconde partie de dveloppement
STT_PIVOT_2       Second pivot
STT_CRISE         Crise
STT_DENOUEMENT    Début du dénouement
STT_CLIMAX        Climax
```

Consulter le livre Narration sur la structure pour la définition de ces points.

2.5.4 Création de la scène à l'aide du bundle TextMate

Avec le bundle TextMate, il suffit d'utiliser le snippet `s` pour créer la scène (donc de taper « s » puis la touche tabulation).

En plus de ce snippet, le temps peut être entièrement géré par TextMate, ce qui permet de ne pas s'en soucier. Ouvrir le l'aide du bundle pour apprendre à le faire.

2.6 Collecte et gestion des brins

Un « brin » est comme une catégorie. Il permet de rassembler les scènes sous une même étiquette. Chaque sous-intrigue, dans une histoire, est un *brin*. Mais un *brin* peut aussi concerner un thème, un accessoire, une relation de personnage ou tout autre élément dramatique.

Les *brins* du film sont consignés dans le fichier :

```
<dossier collecte>/brins.collecte
```

2.6.1 Format général d'un brin

Chaque brin, dans le fichier `brins.collecte`, a la forme :

```
INDICE DU BRIN
TITRE DU BRIN
DESCRIPTION DU BRIN
```

Chaque brin doit être séparé de l'autre par une ligne vide.

Par exemple :

```
12
Relation entre Joe et son chien
Ce brin contient toutes les scènes qui concernent la
relation de Joe avec son chien.

13
Utilisation de la casserole
Ce brin contient toutes les scènes qui concerne
l'utilisation qui est faite de la casserole qui
servira finalement au meurtre.
```

Noter que chaque brin doit être séparé d'une ligne.

2.6.2 Création du brin à l'aide du bundle TextMate

À l'aide du bundle TextMate, on peut créer le fichier des brins grâce à la commande `Brins : créer le fichier`.

Ensuite, dans ce fichier, on peut utiliser le snippet `brin` pour créer un nouveau brin qui aura automatiquement le bon index (donc en tapant `brin` puis tabulation).

Pour de plus amples détails, voir l'aide du bundle (en tapant CMD+H **tandis qu'un fichier collecte est actif**).

2.7 Collecte et gestion des personnages

Pour la collecte, les personnages du film sont consignés dans le fichier :

```
<dossier collecte>/personnages.collecte
```

La définition des personnages permet d'utiliser des marques `[PERSO#<id personnage>]` dans les autres fichiers de collecte afin d'associer les *brins*, les scènes ou les beats de scène à des personnages. On gagne aussi énormément de temps en n'ayant pas à écrire le nom du personnage chaque fois.

Grâce à ces marques, par exemple, on peut déterminer le temps de présence d'un personnage à l'écran.

2.7.1 Format général d'une donnée personnage

Chaque personnage, dans le fichier `personnages.collecte`, est consigné sous la forme :

```
PERSONNAGE:<identifiant texte personnage>
  PRENOM:          <prenom du personnage>
  NOM:             <nom du personnage>
  PSEUDO:          <pseudo (si autre que "Prénom Nom")>
  SEXE:            Homme ou Femme
  ANNEE:           <année de naissance>
  FONCTION:        <fonction dans l'histoire>
  DESCRIPTION:     <description du personnage>
  PRENOM_ACTEUR:   <prénom de l'acteur>
  NOM_ACTEUR:      <nom acteur>
```

2.7.2 Création du personnage à l'aide du bundle TextMate

On peut créer le fichier `personnages.collecte`, avec le bundle TextMate, en utilisant la commande `Persos : créer le fichier`.

Ensuite, dans ce fichier, il suffit d'utiliser le snippet `perso` pour créer facilement un nouveau personnage (donc de taper `perso` puis la touche tabulation).

Dans le fichier de collecte des scènes, il suffit de taper `p` puis tabulation pour choisir un personnage très facilement.

Pour de plus amples détails sur l'utilisation des personnages, voir l'aide du bundle (en tapant CMD+H **tandis qu'un fichier collecte est actif**).

2.8 Fichier des métadonnées

Un fichier de métadonnées peut être créé à la racine du dossier de collecte :

```
<dossier collecte>/metadata.json
```

Ce fichier permet de définir les données suivantes :

```
{
  /* Le titre du film */
  "titre": "<le titre du film>",
  /* ID du film */
  "id": "<identifiant filmodico",
  /* Les auteurs de la collect */
  "auteurs": ["Auteur 1", "Auteur 2"/* , etc. */],
  /* Date de debut de la collecte */
  "date_debut": "JJ/MM/AAAA",
  /* Date de fin de la collecte */
  "date_fin": "JJ/MM/AAAA"
}
```

2.8.1 Définition du titre du film

Définir le titre, dans les métadonnées, à l'aide de la propriété `titre`.

2.8.2 Définition de l'identifiant du film

Définir l'identifiant Filmodico, dans les métadonnées, à l'aide de la propriété `id`.

2.8.3 Dates de début et de fin de collecte

Définir la date de début de collecte avec la propriété `date_debut` dans les métadonnées, et la date de fin de collecte, si la collecte est terminée, à l'aide de la propriété `date_fin`.

La valeur est une date sous la forme `JJ/MM/AAAA`. Par exemple `14/04/2017`.

2.8.4 Définition des auteurs de la collecte

Définir le titre, dans les métadonnées, à l'aide de la propriété `auteurs`.

2.8.5 Définition de la structure

La structure du film peut être définie dans le fichier des métadonnées, grâce à la donnée `structure` :

```
// Dans le fichier metadata.json
{
  ...
  "structure":{
    "pivot_1":              "0:21:45", // (*)
    "incident_perturbateur":"5:03",    // (*)
    "incident_declencheur": "12:20",   // (*)
    "developpement":        "0:25:00", // (*)
    "cle_de_voute":         "0:59:30", // (*)
    "crise":                "1:17:00", // (*)
    "pivot_2":              "1:24:12", // (*)
    "denouement":           "1:29:10", // (*)
    "climax":               "1:40:12"  // (*)
  }
}
```

(*) Il s'agit des temps réels (cf. l'explication sur les temps réels et relatifs). Ils peuvent être donc définis après une première extraction.

Ces temps permettent :
— de définir dans le fichier statistique les écarts par rapport au *Paradigme de Field Augmenté*.
— de séparer les actes dans le fichier résumé et le fichier synopsis.

3 Parsing de la collecte

Le *parsing* est l'opération qui consiste à transformer les données collectées dans l'étape précédente en données utilisables pour créer des fichiers d'analyse, des statistiques, etc.

Pour parser un dossier de collecte, le plus simple est d'utiliser dans le Terminal la commande :

```
collecte parse mon/dossier/collecte
```

Ou de le faire par ruby :

```
coll = Collecte.new('mon/dossier/collecte')
coll.parse
```

Pour tout savoir sur la commande collecte.

4 Extraction des informations du film

L'*extraction* est la troisième opération de la collecte, et son but ultime. Elle permet d'obtenir des séquenciers, des fichiers brins et même des statistiques sur le film.

On peut extraire :
— un résumé (ou *des* résumés partiels),
— un synopsis (ou *des* synopsis partiels),
— un séquencier (ou *des* séquenciers partiels),
— les séquenciers de chaque brin défini (complet ou partiel),
— les séquenciers de chaque brin automatique personnage (complet ou partiel),
— les séquenciers de chaque brin automatique des relations de personnages (complet ou partiel)

*() On obtient une sortie *partielle* simplement en définissant le début ou la fin de l'extrait à l'aide des options `:from_time` et `:to_time`. Cf. Définir les temps de début et de fin.

4.1 Le fichier résumé

Le *fichier résumé* présente simplement la liste des résumés de scènes assemblés bout à bout.

En ligne de commande :

```
// forme longue
$ collecte extract --resume

// forme courte
$ collecte extract -res -o=text
```

Si aucun format (`-o`) n'est défini, la sortie sera en HTML.

En programmation ruby :

```
coll = Collecte.new('mon/dossier/collecte')
coll.extract(:resume, {format: :text})
```

Si aucun format n'est défini, la sortie sera en HTML.

Si la structure est définie (cf. définition de la structure), des retours à la ligne séparent les actes pour rendre le texte plus clair.

4.2 Le fichier synopsis

Le *fichier synopsis* présente simplement le synopsis du film, c'es-à-dire les résumés des scènes assemblés ou leurs *paragraphes* (leur *beats*) s'ils sont définis.

En ligne de commande :

```
// forme longue
$ collecte extract --synopsis

// forme courte
$ collecte extract -syn
```

En programmation ruby :

```
coll = Collecte.new('chemin/vers/dossier/collecte')
coll.extract(as: :synopsis, no_horloge: true)
```

Par défaut, une petite horloge indiquant le temps est placée dans la marge gauche du synopsis. On peut la supprimer en ajoutant comme ci-dessus l'option `:no_horloge` à `true`.

Si la structure est définie (cf. définition de la structure), des doubles retours à la ligne séparent les actes.

4.3 Les fichiers séquencier

Les *fichiers séquenciers* sont certainement les fichiers les plus extraits pour les analyses, notamment sous leur forme de fichiers brins.

En ligne de commande

```
// forme longue
$ collecte extract --sequencier

// forme courte
$ collecte extract -seq
```

En programmation ruby

```
coll = Collecte.new('ma/collecte')
coll.extract(
  as: :sequencier,
  format:   :html # par defaut
)
```

Voir toutes les options utilisables avec cette commande.

4.4 Les fichiers brins

Les fichiers « brins » sont des séquenciers (liste de scènes) dont on a filtré les scènes appartenant au brin en question.

Ces brins doivent être définis dans le fichier des brins.

Il existe deux types de brin particuliers, qui sont construits automatiquement, ce sont les brins personnages et les [brins relation de personnages]. S'y reporter pour le détail.

4.4.1 Désignation des brins

Les brins à extraire sont désignés de façon textuelle dans les options. Par exemple :

```
$ collecte extract -brins=2
```

```
$ collecte extract -brins=2+4

$ collecte extract -brins=2,(4+12)
```

Ou en ruby :

```
coll = Collecte.new('dossier/collecte')
coll.extract(
  as: :brins,
  brins:  '2'
)

coll.extract(
  as: :brins,
  brins:  '2+4'
)

coll.extract(
  as: :brins,
  brins:  '2,(4+12)'
)
```

Pour un brin seul, on indique seulement son identifiant dans le [fichier brins].

Si on veut toutes les scènes qui appartiennent à deux brins (ou plus), on utilise le signe `+`. Noter que dans ce cas les scènes retenues sont celles qui appartiennent à TOUS les brins désignés.

Par exemple :

```
$ collecte extract -b=2+4
```

La commande précédente crée un séquencier-brin qui contiendra toutes les scènes qui appartiennent en même temps au brin 2 ET au brin 4. Elles doivent *impérativement* appartenir aux deux brins.

Si on veut exprimer la condition `ou`, on utilise une virgule. Par exemple :

```
$ collecte extract -b=2,4
```

La commande précédente crée un séquencier-brin qui contiendra toutes les scènes du brin 2 et toutes les scènes du brin 4 (entrelacées).

On peut créer une description complexe à l'aide des parenthèses :

```
$ collecte extract -b=2+(4,12)+59
```

La commande précédente crée un séquencier-brin des scènes qui répondent aux conditions : appartenir impérativement au brin 2, ET appartenir impérativement au brin 59, ET appartenir impérativement au brin 4 OU au brin 12.

Avec un + dans la parenthèse :

```
$ collecte extract -b=2,(4+5)
```

Crée un séquencier-brin des scènes qui appartiennent au brin 2 ou les scènes qui appartiennent au brin 4 ET au brin 5.

4.5 Les brins personnages

Pour sortir les brins personnages, c'est-à-dire une liste des scènes du ou des personnages, on utilise :

En ligne de commande

```
// On doit se trouver dans le dossier de collecte
$ collecte extract --all_personnages
// => les brins de chaque personnage

$ collecte extract --personnage=<id personnage>
// => extrait le brin du personnage en question

$ collecte extract -p=<id personnage>
// => idem

$ collecte extract -p=<id_perso_1>,<id_perso_2>,...
// => Extrait les brins de chaque personnage
//    spécifié. Attention, ne pas mettre d'espace
```

Les identifiants de personnage, ci-dessus, correspondent à ceux définis dans le fichier des personnages.

En ruby

```
coll = Collecte.new('mon/dossier/collecte')
coll.extract(:all_personnages, format: :html)
# => Extrait tous les personnages (un fichier HTML par
#    perso)

coll.extract(
  as: :brin_personnage, personnages: :all
```

```
)
# idem, tous les brins

coll.extract(
  as: :brin_personnage, personnage: ID_Perso
)
# => Extrait seulement le brin du personnage ID_Perso

coll.extract(
  as: :brin_personnage,
  personnages: [Idperso1, Idperso2, ....]
)
# => Extrait les brins de chaque personnage de la liste.
```

Les identifiants de personnages sont ceux définis dans le fichier des personnages.

4.6 Le fichier Statistiques

Le *fichier statistiques* présente des statistiques sur le film, telles que le nombre de scènes, la durée moyenne par scène, le nombre de personnages, de décors, etc.

En ligne de commande

```
// format long
$ collecte extract --statistiques --from=0:00 --to=1:00:00

// format court
$collecte extract -stats -f=0:0 -t=1:0:0
```

Note : la commande ci-dessus permet de ne sortir les statistiques que pour une période de temps.

En programmation ruby

```
coll = Collecte.new('ma/collecte')
coll.extract(
  as:         :statistiques,
  from_time:  "1:00",
  to_time:    "1:0:0"
)
```

Si la structure est définie (cf. définition de la structure), le fichier calcule aussi les écarts ou les concordances avec le *paradigme de Field augmenté*.

5 La commande `collecte`

La commande `collecte` permet de *parser* et d'*extraire* tous les types de fichiers possible pour les analyses, même le fichier statistiques.

Cette commande fonctionne, à la base, avec la spécification de l'opération a exécuter en premier argument :

```
$  collecte [operation]
```

Il existe trois opérations principales :

```
help      Pour obtenir l'aide du programme (ce manuel)
parse     Pour parser les fichiers collecte
extract   Pour extraire tout type de fichier des données
          de collecte.
```

En conséquence :

```
$ collecte help pdf
// => ouvre ce manuel en version PDF
$ collecte help
// => ouvre ce manuel en version HTML

$ collecte parse
// => parse tous les fichiers collecte du dossier
//    dans lequel on se trouve

$ collecte extract --all
// => Extrait tous les fichiers possibles à partir
//    des données  de collecte fournies, en format
//    HTML.
```

Ces trois commandes s'utilisent avec des options ou des arguments que nous allons détailler. Mais pour commencer, il faut installer cette commande manuellement.

5.1 Installation de la commande `collecte`

— Charger l'application `Collecte.app` dans le dossier Applications si nécessaire,
— Créer un lien symbolique de la commande dans `/usr/local/bin` pour pouvoir l'utiliser en ligne de commande.

Pour créer le lien symbolique, utiliser dans l'application Terminal :

```
$ sudo ln -s /Application/Collecte.app/Contents/MacOS/Collecte /usr/local/
```

Note : `sudo` n'est pas nécessaire si vous êtes en root.

5.2 Opération `collecte help`

Comme nous venons de le voir, cette opération permet d'ouvrir le présent manuel, soit en version PDF (ajouter `pdf`) soit en version HTML (ne rien ajouter ou ajouter `html`).

5.3 Opération `collecte parse`

Cette opération permet de parser tous les fichiers de collecte en vue de leur utilisation pour produire les fichiers d'analyse.

Le résultat de ce parsing est un fichier `film.pstore` ou `film.msh` qui contient toutes les données du film.

Les arguments possibles pour le parse sont :

```
$ collecte parse
// => Parse tous les fichiers existants dans le dossier
//    courant.

$ collecte parse mon/dossier/collecte
// => Parse tous les fichiers du dossier
//    ‘mon/dossier/collecte‘

$ collecte parse -p mon/dossier/collecte
// => Parse seulement le fichier des personnages du
//    dossier spécifié.

$ collecte parse -b mon/dossier/collecte
// => Parse seulement le fichier des brins du dossier
//    spécifié.
```

Mais la collecte étant rapide, même pour un fichier de scènes, il vaut mieux tout collecter chaque fois.

5.4 Opération `collecte extract`

C'est l'opération d'extraction des données qui permet de produire tous les fichiers nécessaires pour une analyse.

Cette opération connait le plus d'options, afin de pouvoir produire exactement ce qu'il faut.

5.5 Emplacement des fichiers produits

Les fichiers extraits de la collecte sont placés dans un dossier `extraction` dans le dossier de la collecte.

Leur nom dépendra des différentes options choisies. Cf. nom des fichiers extraits.

6 Les différentes options de la command `collecte extract`

6.1 Extraire tous les fichiers

On peut, à l'aide d'une unique commande, extraire tous les fichiers d'analyse, c'est-à-dire :

— le séquencier complet,
— le résumé complet,
— le synopsis complet avec horloge,
— tous les séquenciers des brins (appelés « fichiers brins »),
— le fichier statistiques.

Options en ligne de commande

```
--all
-a
```

Par exemple :

```
collecte extract --all
ou
collecte extract -a
```

Noter que si le parsing n'a pas eu encore lieu il sera exécuté avant l'extraction.

Options en ruby

```
coll = Collecte.new('dossier/collecte')
coll.extract(:all)
```

Le format de sortie sera HTML. Pour un autre format (`:text` ou `:xml`, mettre en second argument `format: :<format>`).

6.2 Extraire le résumé

Le fichier « résumé » est un fichier qui ne reprent que les résumés des scènes et les assemble l'un au bout de l'autre.

Options ligne de commande

```
// forme longue
$ collecte extract --resume

// forme abregee
$ collecte extract -res
```

Options en ruby

```
coll = Collecte.new('dossier/collecte')
coll.extract(
  as:   :resume
)
```

Si les actes sont définis (cf. définition des actes), une séparation est ajoutée à chaque changement d'acte.

6.3 Extraire le synopsis

Options ligne de commande

```
// forme longue
$ collecte extract --synopsis

// forme abregee
$ collecte extract -syn
```

Programmation ruby

```
coll = Collecte.new('dossier/collecte')
coll.extract(
  as: :synopsis
)
```

Autres options

```
:no_horloge     Supprime les horloges si true

// Commande
$ collect extract -syn --no_horloge

// Ruby
coll.extract(
  as:         :synopsis,
  no_horloge: true
)
```

Note : comme pour les résumés, si les actes sont définis (cf. définition des actes), une séparation est faite suivant les différentes parties.

6.4 Extraire des séquenciers

Pour produire des séquenciers, c'est-à-dire des fichiers comportant les scènes et les intitulés, utiliser l'option `-sequencier` ou `-seq`.

Options lignes de commande

```
--sequencier
-seq
```

Par exemple, pour produire un séquencier à partir de la 10 minute réelle :

```
$ collecte extract -seq -f=10:00
```

Options en ruby

```
as: :sequencier
```

Exemple :

```
Coll = Collecte.new('mon/dossier/collecte')
Coll.parse # si necessaire
Coll.extract(
  as: :sequencier,
  from_time: '10:0'
)
```

Par défaut, le format sera du HTML.

6.5 Extraire un ou des brins

Rappel : un « brin » est en fait un séquencier qui se concentre sur un élément particulier de la narration. Dans le fichier de collecte des scènes, les scènes sont mises dans des brins à l'aide des marques `b<indice du brin>`. Cf. la gestion des brins.

Options ligne de commande

```
// forme longue
$ collecte extract --brins=<designation>

// forme abregee
$ collecte extract -b=<designation>
```

Pour produire tous les brins, on utilise `-brins` seul, sans argument :

```
$ collecte extract --brins
```

Cf. la désignation des brins.

Programmation ruby

```
coll = Collecte.new('dossier/collecte')
coll.extract(
  as:     :brins,
  brins:  '<designation>'
)
```

Cf. la désignation des brins pour voir comment désigner les brins à voir.

Exemple :

```
coll = Collecte.new('dossier/collecte')
coll.extract(
  as:     :brins,
  brins:  '2+(4,12)'
)
```

6.6 Extraire le fichier statistiques {#extraire statistiques}

Le fichier *statistiques* contient des données statistiques sur le film, telles que le nombre de scènes, leur durée moyenne, les temps de présence des personnages, etc.

Options en ligne de commande

```
--statistiques ou -stats
```

Par exemple :

```
$ collecte extract -stats
```

Programmation ruby

```
coll = Collecte.new('dossier/collecte')
coll.extract(as: :statistiques)
```

On peut tout à fait obtenir les statistiques seulement pour une portion de temps, ou pour un brin en particulier, en définissant ces paramètres.

6.7 Définir le temps de début et le temps de fin

Pour tous les fichiers extraits, on peut définir le temps de départ à l'aide des options `from` et `to` qui doivent définir une horloge `H:MM:SS`.

Options ligne de commande

```
--from=<horloge>(*)  Depuis ce temps réel (**)
-f=<horloge>

--to=<horloge>       À ce temps réel (**)
--t=<horloge>
```

(*) Voir le formatage des horloges.
(**) Voir l'explication sur les temps réels et relatifs.

Exemple :

```
$ collecte extract --all --from=0:12 --to=1:12:00
```

Avec la commande précédente, tous les fichiers seront extraits, entre la 12ème minute et 1 heure 12. On peut utiliser aussi la formule d'option courte :

```
$ collecte extract -a -f=0:12 -t=1:12:00
```

Options en ruby

```
:from_time  => "<horloge>"
:to_time    => "<horloge>"
```

Exemple :

```
coll = Collection.new('mon/dossier/collecte')
coll.extract(
  format:     :html,
  as:         :sequencier,
  from_time:  '0:12',
  to_time:    '1:25:56'
)
```

6.8 Définir le format de sortie

Pour le moment, la sortie la plus sûre est en `HTML` (format par défaut), mais elle le sera plus tard en `XML` et en `TEXT`.

Options en ligne de commande

```
--html  ou  --output_format=html  ou -o=html
--xml   ou  --output_format=xml   ou -o=xml
--text  ou  --output_format=text  ou -o=text
```

Par exemple :

```
$ collecte extract --all --xml
ou
$ collecte extract -a -o=xml
```

Programmation ruby

```
coll = Collecte.new('dossier/collecte')
coll.extract(:all, format: :xml)
```

6.9 Options de l'extraction

6.9.1 Option : ne pas afficher la timeline

Par défaut, lorsque l'on produit un séquencier en `html`, une *timeline* est écrite en haut du document, qui permet de visualiser l'emplacement des scènes.

Cette timeline peut être supprimée à l'extraction avec l'option `-no_timeline` ou `-ntl`.

Par exemple, en ligne de commande :

```
$ collecte extract -seq --no_timeline
```

En ruby :

```
coll = Collecte.new('dossier/collecte')
coll.extract(
  as: :sequencier,
  no_timeline: true
)
```

6.9.2 Option : suggérer la structure

Au moment de l'analyse, avant le placement des [points structurels], il peut être intéressant de demander à l'extraction de suggérer les positions de ces points structurels (mettre en relief les quarts-temps, les tiers-temps, etc.).

Pour ce faire, on utilise l'option `-suggest_structure` ou `-stt`.

Par exemple, en ligne de commande :

```
$ collecte extract --sequencier --suggest_structure
ou
$ collecte extract -seq -stt
```

Ou en ruby :

```
coll = Collecte.new('dossier/collecte')
coll.extract(
  as: :sequencier,
  suggest_structure: true
)
```

6.9.3 Option : forcer le parsing avant l'extraction

Parfois des données sont modifiées dans les fichiers de collecte. Mais si un *parsing* a déjà eu lieu, les nouvelles données ne seront pas prises en compte. Pour qu'elles le soient, il faut *forcer à novueau le parsing*. On utilise pour ça l'option `force_parsing`.

Options en ligne de commande

```
--force_parsing
-fp
```

Par exemple :

```
$ collecte extract -a -fp
// ou
$ collecte extract --all --force_parsing
```

La commande précédente extrait tous les fichier (`-a`) en forçant le parsing des données (`-fp`)

Option en ruby

```
:force_parsing
```

Par exemple :

```
coll = Collecte.new('mon/dossier/collecte')
coll.extract(:all, force_parsing: true)

# ou

coll.extract(
  as:             :sequencier,
  format:         :xml,
  force_parsing:  true
)
```

6.9.4 Option : mettre une horloge au synopsis

On peut ajouter une horloge en regard des scènes du synopsis, pour les situer dans le temps, à l'aide de l'option `-horloge` ou `-h`.

Par exemple, en ligne de commande :

```
$ collecte extract --synopsis --horloge
ou
$ collecte extract -syn -h
```

Ou en ruby :

```
coll = Collecte.new('dossier/collecte')
coll.extract(
  as:       :synopsis,
  horloge:  true
)
```

6.9.5 Option : ouvrir le fichier extrait

On peut ouvrir directement le fichier produit, en fin d'extraction, à l'aide de l'option `-open_file` ou `-open`.

Par exemple :

```
$ collecte extract --brins=2 --open_file
ou
$ collecte extract -b=2 -open
```

Ou en ruby :

```
coll = Collecte.new('dossier/collecte')
coll.extract(
  as:          :brin,
  brins:       '2',
  open_file:   true
)
```

6.9.6 Option : débugger

On peut ouvrir le fichier log en fin d'opération (parsing, extraction) à l'aide de l'option `-debug` ou `-d`.

Par exemple, en ligne de programme :

```
$ collecte parse --debug
ou
$ collecte parse -d
```

Ou, en ruby :

```
coll = Collecte.new('dossier/collecte')
coll.parse(debug: true)
```

7 Annexes

7.1 Formatage des horloges

Les horloges s'expriment par :

```
H:MM:SS
```

Par exemple :

```
12:30

# Pour «~12 minutes et 30 secondes~»

1:25:03

# Pour «~une heure vingt-cing minutes et trois
# secondes~»
```

Les zéros ne sont pas obligatoires, on peut donc tout à fait écrire :

```
1:2:5

# Pour « une heure, deux minutes et cinq secondes~»
```

7.2 Nom et titre des fichiers extraits

Le nom des fichiers extraits (qui se trouvent dans le dossier de la collecte) dépend des différentes options de collecte.

Un fichier complet, c'est-à-dire partant du premier temps et allant jusqu'au dernier, porte le titre « complet » et son nom de fichier est précédé de `full_`. Par exemple :

```
full_sequencier.html
full_brin_1.html
full_personnage_jan.html
```

Un fichier qui ne contient qu'une partie du temps du film porte dans son titre « partiel » (p.e. « Séquencier partiel ») et son nom de fichier contient le temps de départ et de fin. Par exemple :

```
sequencier_210_3600.html
```

7.3 Le bundle TextMate

Le bundle TextMate permet de procéder avec beaucoup plus de souplesse et de facilité à la collecte des scènes, des personnages et des brins du film.

Cf. aussi l'outil de collecte en ligne.

TODO : À DÉVELOPPER

7.4 Outil de collecte en ligne

TODO : À DÉVELOPPER