# Guide de l'utilisateur de TEXTABLE v1.4



## Contenu

# Figures

# 1. Introduction

Comme tout type de donnée, le texte se prête à un traitement informatique et statistique. Ainsi, pour autant qu'on dispose des sources appropriées, il est possible d'étudier la variation de quantités mesurables dans des textes (telles que fréquences et mesures de complexité) en fonction de l'auteur, du genre, de l'époque, etc. Ces variations peuvent également être visualisées par le biais de diverses représentations graphiques, ou encore être exploitées pour établir des catégories de textes.

Il existe plusieurs logiciels libres ou commerciaux permettant à des utilisateurs non spécialistes d'appliquer des traitements statistiques. De façon très générale, ces logiciels prennent en entrée des données *tabulées*, où les lignes représentent les *individus* qui composent un échantillon, et les colonnes des *variables* qui décrivent les individus. La figure 1 ci-dessous présente un exemple typique d'une telle représentation:

```
sepal_len  sepal_wid   petal_len   petal_wid   type
5.1        3.5         1.4         0.2         Iris-setosa
4.9        3.0         1.4         0.2         Iris-setosa
4.7        3.2         1.3         0.2         Iris-setosa
4.6        3.1         1.5         0.2         Iris-setosa
5.0        3.6         1.4         0.2         Iris-setosa
...
```

**Figure 1: Extrait d'un fichier de données tabulées.**

L'une des principales difficultés posées par le traitement statistique des données *textuelles* est précisément la contrainte de passer de la représentation informatique du texte comme chaîne de caractères à une représentation tabulée, où les lignes correspondent typiquement à des parties du texte et les colonnes aux quantités mesurées dans ces parties (fréquences, mesures de complexités, etc.).

Sans être particulièrement complexe d'un point de vue conceptuel, la conversion du texte en table implique en général des opérations trop fastidieuses pour être effectuées manuellement. Des langages de scriptage comme Perl ou Python sont parfaitement adaptés à ce type de traitement, mais leur utilisation requiert des compétences spécifiques en programmation.

TEXTABLE v1.4 est conçu pour permettre à des utilisateurs non programmeurs de construire des données tabulées à partir de données textuelles, par le biais d'une interface à la fois flexible et intuitive. Il offre en particulier les fonctionnalités suivantes:

- importation de sources textuelles diverses
- application d'opérations de recodage systématique
- application de traitements analytiques tels que segmentation et annotation des unités
- sélection manuelle, automatique ou aléatoire de sous-ensembles d'unités textuelles
- calcul d'indices numériques tels que fréquences et mesures de complexité
- exportation des résultats sous forme de tableaux de données appropriés pour un traitement ultérieur par un logiciel statistique

# 2. Distribution

TEXTABLE v1.4 est développé en Python et distribué sous licence open source (GNU GPL v3). TEXTABLE v1.4 inclut une copie de la librairie Python LTTL v1.4 (*LangTech Text Library*), elle aussi distribuée sous licence GNU GPL v3. TEXTABLE v1.4 et LTTL v1.4 s'exécutent sur Windows et MacOS X.[1]

TEXTABLE v1.4 est une extension du logiciel Orange Canvas (http://orange.biolab.si). Orange Canvas est lui-même basé sur Python (http://python.org), et tous deux sont distribués sous licence open source et en version Windows, Mac ou Linux.

# 3. Installation

Python v2.7 et Orange Canvas v2.7 (ou plus récent[2]) doivent impérativement être installés *avant* TEXTABLE v1.4. En principe, les versions récentes d'Orange Canvas incluent d'office la version 2.7 de Python). Après l'installation, TEXTABLE v1.4 apparaît sous la forme d'un onglet supplémentaire dans Orange Canvas (voir figure 8 ci-dessous); les fichiers source et autres documents associés sont alors accessibles dans votre distribution d'Orange, dans le dossier */OrangeWidgets/Textable*.

## *3.1 Windows*

1. Sur la page *Download* du site d'Orange Canvas, téléchargez l'installeur du logiciel en suivant le lien *Full package* pour une première installation ou *Pure Orange* pour une mise à jour.
2. Exécutez l'installeur d'Orange Canvas et cliquez **OK** à chaque étape (y compris l'installation de modules Python).
3. Lancez Orange Canvas puis sélectionnez le menu **Options > Add-ons...** Dans la fenêtre qui s'ouvre, cliquez sur **Refresh list**, cochez la case *Orange-Textable* puis cliquez sur **OK** (deux fois).

Si l'étape 3 s'est exécutée correctement, l'onglet TEXTABLE apparaît dans la liste à gauche de la fenêtre d'Orange Canvas après avoir quitté et relancé l'application.

*Seulement si l'étape 3 ne s'est pas exécutée correctement:*

4. Rendez-vous sur PyPI pour télécharger l'installeur de TEXTABLE v1.4 pour Windows (*MS Windows installer*, fichier *.exe*).
5. Exécutez l'installeur de TEXTABLE v1.4 et cliquez **OK** à chaque étape.

## *3.2 MacOS X*

1. Sur la page *Download* du site d'Orange Canvas, téléchargez l'image disque du logiciel en suivant le lien *Bundle*.
2. Dans la fenêtre qui s'ouvre au terme du téléchargement, déplacez l'icône d'Orange Canvas vers celle du dossier *Applications*.
3. Lancez Orange Canvas puis sélectionnez le menu **Options > Add-ons...** Dans la fenêtre qui s'ouvre, cliquez sur **Refresh list**, cochez la case *Orange-Textable* puis cliquez sur **OK** (deux fois).

Si l'étape 3 s'est exécutée correctement, l'onglet TEXTABLE apparaît dans la liste à gauche de la fenêtre d'Orange Canvas après avoir quitté et relancé l'application.

*Seulement si l'étape 3 ne s'est pas exécutée correctement:*

---

[1] Bien que la compatibilité avec Linux soit vraisemblable, elle n'a pas été spécifiquement testée.
[2] TEXTABLE v1.4 n'est pas compatible avec les versions d'Orange Canvas antérieures à 2.7.

4. Rendez-vous sur PyPI pour télécharger la distribution source de TEXTABLE v1.4 (fichier *.tar.gz*).
5. Décompressez l'archive puis ouvrez un *Terminal* et naviguez jusque dans l'archive décompressée (voir ci-dessous pour plus de détails sur cette étape). Saisissez alors la commande suivante:

   ```
   python setup.py install
   ```

   NB: si ce processus échoue, il est parfois possible de corriger le problème en remplaçant la commande précédente par celle-ci:

   ```
   /Applications/Orange.app/Contents/MacOS/python setup.py install
   ```

*En cas de difficulté pour "ouvrir un Terminal et naviguer jusque dans l'archive décompressée..." :*

a) Glissez-déposez sur le bureau le dossier *Orange-Textable-X* (ou *X* est le numéro de version, p.ex. "1.4") qui se trouve dans l'archive téléchargée.

b) Dans *Finder > Applications > Utilitaires*, double-cliquez sur *Terminal*.

c) Dans le terminal, saisissez exactement la commande

   ```
   cd Desktop/Orange-Textable-X
   ```

   (où *X* est toujours le numéro de version).

d) Saisissez alors la commande

   ```
   python setup.py install
   ```

   (ou, si nécessaire, la commande alternative indiquée ci-dessus).

# 4. Introduction à l'utilisation d'Orange Canvas

L'utilisation de TEXTABLE v1.4 implique une compréhension des bases du fonctionnement d'Orange Canvas. Les sections suivantes proposent une brève introduction à ces éléments.

## *4.1 Généralités*

Orange Canvas est un logiciel d'analyse de données dont la particularité est d'offrir une interface de *programmation visuelle*: il permet en effet d'émuler certaines fonctionnalités d'un langage de programmation en manipulant des éléments graphiques plutôt que des commandes sous forme de code. En particulier, le logiciel permet de construire graphiquement des chaînes de traitement (*schémas*), en arrangeant des unités (*widgets*) et en les reliant entre elles par des connexions.

La figure 2 ci-dessous présente la fenêtre principale d'Orange Canvas (v2.7) et illustre son fonctionnement au moyen d'une chaîne de traitement simple. Notez que la fenêtre est divisée en plusieurs parties:

- La barre de menus est surtout utile pour ouvrir et sauver des schémas dans différents formats (menu **File**).
- A gauche de la fenêtre, les onglets **Textable**, **Data**, **Visualize**, etc. constituent un catalogue de widgets classés par type de fonctionnalité. Cliquer sur un onglet permet de l'activer et voir les widgets (voir section suivante) qui le composent, comme c'est le cas de **Visualize** dans cet exemple. Notez que chaque onglet est associé à une couleur spécifique, qui se retrouve sur les instances de widgets correspondantes.

- Les boutons en bas à gauche donnent accès à diverses fonctions d'édition de schémas notamment (c'est au moyen de ces boutons qu'ont été créées les zones de texte et les flèches apparaissant sur la figure).

- Toute la partie droite de la fenêtre constitue le *canevas*, soit l'espace réservé pour la création de schémas de traitement de données.

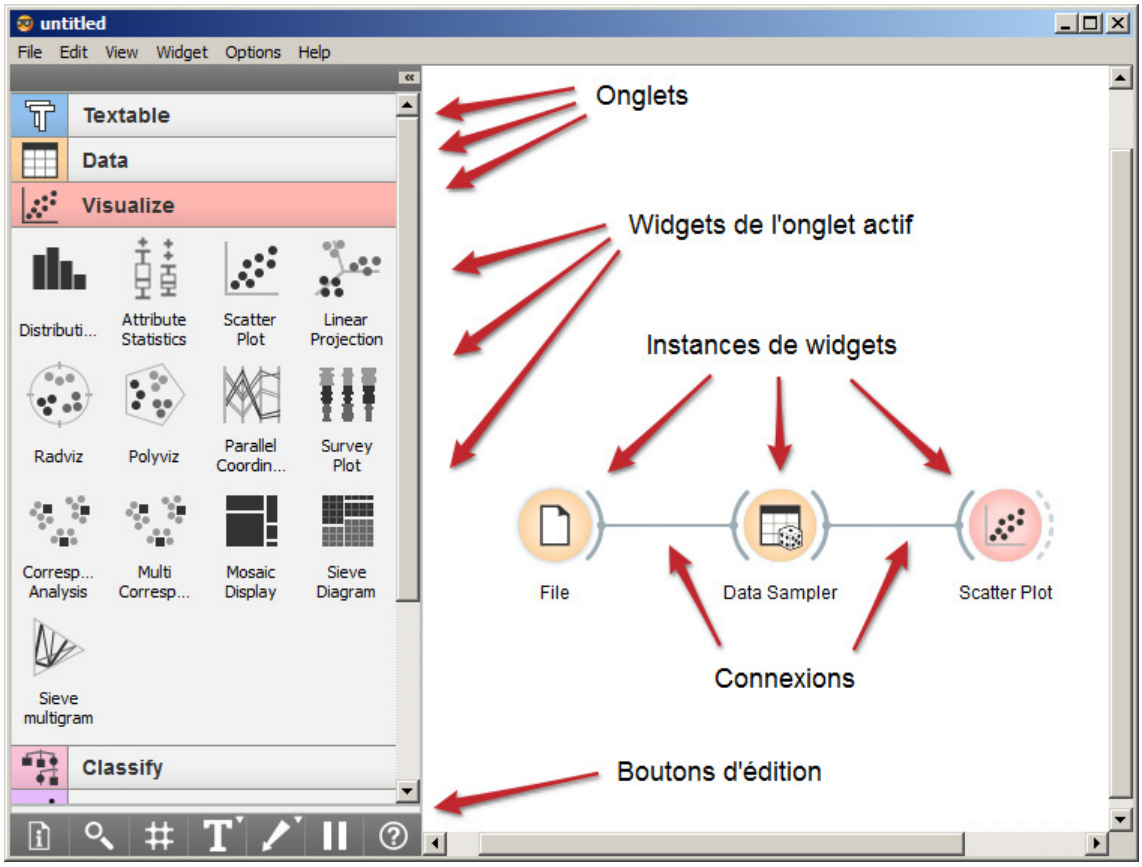



**Figure 2: La fenêtre principale d'Orange Canvas (v2.7), présentant un exemple de schéma simple.**

## *4.2 Widgets, connexions et flux de données*

Le schéma présenté sur la figure 2 comporte trois instances de widgets (**File**, **Data Sampler** et **Scatter Plot**) reliées par deux connexions. En général, un widget représente un traitement effectué sur des données entrantes pour produire d'autres données en sortie. Le flux des données dans le système est déterminé par les connexions entre les instances de widgets.

Les données sur lesquelles opère une instance sont normalement définies par ses *connexions entrantes*; par convention, les connexions entrantes sont toujours reliées au côté gauche du widget. Certains widgets n'admettent pas de connexion entrantes – c'est typiquement le cas des widgets permettant l'*importation* de données dans le système, comme le widget **File** dans notre exemple.

Après avoir effectué le traitement qui lui est propre, un widget émet le résultat par le biais de ses *connections sortantes*, conventionnellement reliées à son côté droit. Ainsi, le widget **File** envoie des données en direction du widget **Data Sampler**, qui lui-même émet des données à l'intention du widget **Scatter Plot**. Certains widgets n'admettent pas de connexions sortantes, en particulier les widgets spécialisés dans l'*exportation* des données.

L'orientation des connexions entrantes et sortantes induit un sens naturel pour la construction des schémas: le plus souvent, les widgets d'importation (ou de génération) de données sont placés à

gauche du schéma, les widgets d'exportation (ou de visualisation) à droite, et les données circulent dans le schéma de la gauche vers la droite.

## *4.3 Types de données et connexions multiples*

Les connexions entrantes et sortantes de chaque widget sont *typées*: elles ne peuvent recevoir et émettre que certains *types* de données spécifiques. Par exemple, le widget **File** émet des données de type *ExampleTable*, un format de tableau partagé par de nombreux widgets au sein d'Orange Canvas. A ce titre, le widget **File** ne peut être connecté qu'avec des widgets acceptant le type **ExampleTable** en entrée, ce qui est le cas de **Data Sampler** dans notre exemple.

En général, les widgets admettent un nombre illimité de connexions sortantes. Ainsi, la figure 3 ci-dessous montre que le widget **Data Sampler** peut être connecté simultanément à plusieurs widgets. Par ailleurs, **Data Sampler** fait partie des widgets qui peuvent émettre des données *distinctes* (et potentiellement de types distincts) sur des connexions différentes; le *nom* écrit en bleu sur chaque connexion permet alors de les différencier (dans l'exemple, les mêmes données sont émises vers **Save** et **Scatter Plot**, et d'autres données vers **Data Table**). Les widgets de ce genre peuvent définir des types de données émis par défaut, ou s'en remettre entièrement à l'utilisateur pour déterminer quel type émettre sur quelle connexion (voir section 4.5 ci-dessous).

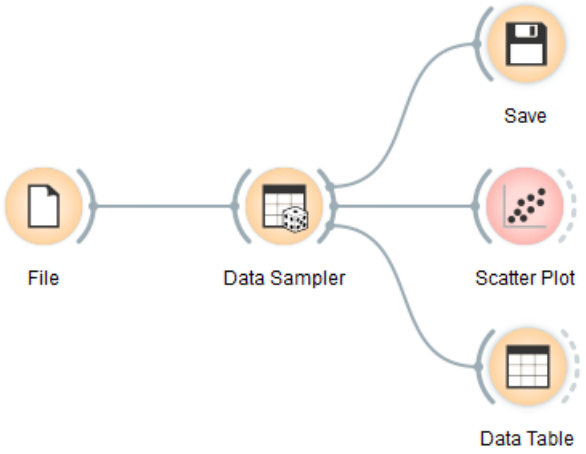



**Figure 3: Connexions sortantes multiples.**

Certains widgets n'admettent qu'une seule connexion entrante. Supposons que *A* soit un tel widget, et qu'il soit connecté avec un widget *B* (*B* étant dans ce cas le widget émetteur). Si une nouvelle connexion est créée depuis un autre widget *C* vers le widget *A*, alors la nouvelle connexion ($C \rightarrow A$) annule et remplace l'ancienne ($B \rightarrow A$). D'autres widgets admettent plusieurs connexions entrantes simultanées. Selon les cas, ces connexions peuvent être du même type ou de plusieurs types distincts.

Pour connaître le ou les types de données qu'un widget peut recevoir ou émettre, il suffit de placer le curseur sur l'icône du widget en question dans l'onglet correspondant. La figure 4 ci-dessous illustre ce mécanisme avec le widget **Data Sampler** (onglet **Data**): celui-ci prend en entrée et émet en sortie des données de type *ExampleTable*. L'affichage n'indique pas si plusieurs connexions entrantes sont admises, mais précise que le widget peut émettre des données (de même type) sur deux connexions distinctes. Pour interpréter cette information, il est utile de se référer à la *description* succincte du widget: celui-ci permet de sélectionner un sous-ensemble des données entrantes. Les deux connexions sortantes correspondent ainsi aux données sélectionnées (nommées *Data Sample*) et aux données restantes (*Remaining Data*).

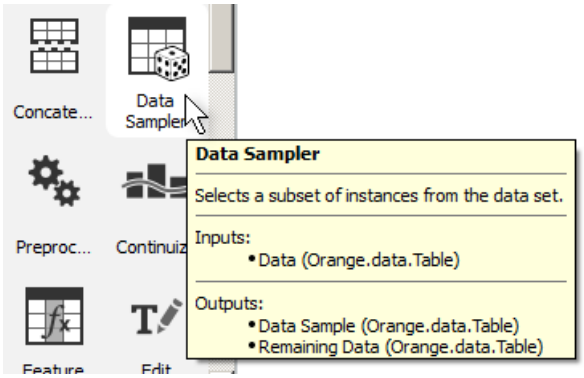



**Figure 4: Déterminer le type de données qu'un widget peut recevoir (*Inputs*) et émettre (*Outputs*).**

## *4.4 Configuration des widgets*

En général, le traitement effectué par un widget est paramétrable par l'utilisateur. Pour accéder à l'interface d'un widget donné, il suffit de double-cliquer sur le widget dans le canevas. Dans notre exemple, le widget **File** affiche alors l'interface représentée sur la figure 5 ci-dessous.

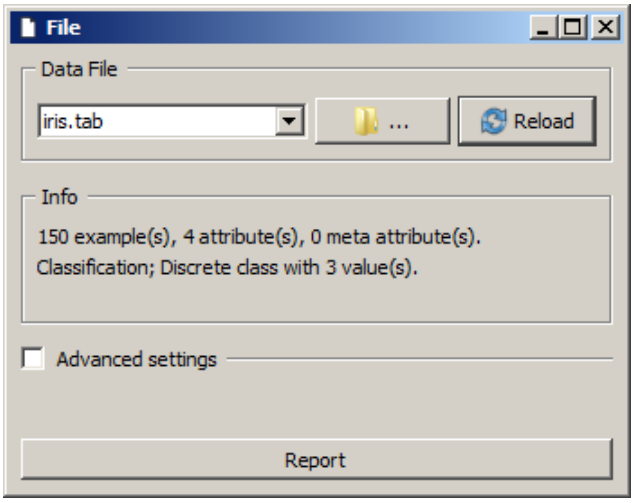



**Figure 5: Interface du widget *File*.**

Cette interface est particulièrement simple. La zone supérieure permet de sélectionner le fichier à ouvrir (en l'occurrence *iris.tab*, cf. Figure 1) et éventuellement de le réimporter pour tenir compte de modifications faites hors du logiciel (**Reload**). La partie centrale renseigne l'utilisateur sur certaines caractéristiques des données importées, telles que le nombre d'individus et de variables. Une case à cocher permet d'accéder à des réglages avancés et le bouton **Report** déclenche l'ouverture d'une fenêtre séparée pour l'affichage d'informations plus détaillées sur les données importées.

Nous verrons dans la suite de ce document d'autres exemples qui illustrent la diversité des interfaces de widgets dans Orange Canvas – diversité qui témoigne de la variété considérable des fonctionnalités offertes par le logiciel.

## *4.5 Création et édition de schémas*

Construire un schéma dans Orange Canvas implique de créer des copies (ou *instances*) de widgets, les positionner sur le canevas, les configurer, et créer des connexions entre eux. La façon la plus simple de créer une instance de widget est de cliquer sur son icône dans l'onglet correspondant (par exemple l'onglet **Data** pour le widget **File**); le même résultat peut être obtenu par "glisser-déposer" depuis les icônes dans les onglets jusque sur le canevas.

Il est possible de créer ainsi plusieurs copies d'un même widget. Chaque instance peut être positionnée (par glisser-déposer) et configurée séparément en double-cliquant sur son icône sur le canevas.

La création d'une connexion entre deux instances s'effectue également par glisser-déposer, depuis l'arc de cercle grisé sur le côté droit de l'instance émettrice jusqu'à celui situé à gauche de l'instance réceptrice, comme illustré sur la figure 6 ci-dessous.

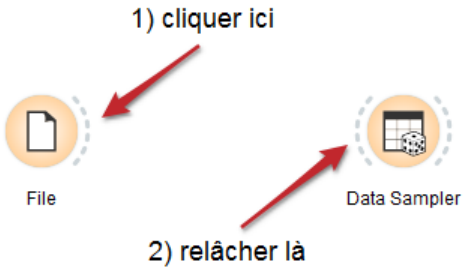



**Figure 6: Création d'une connexion.**

Lorsqu'une connexion est créée entre deux instances de widgets ayant plusieurs connexions sortantes et/ou entrantes distinctes, Orange Canvas affiche le dialogue représenté sur la figure 7 ci-dessous. Cette interface permet de créer une connexion par glisser-déposer depuis l'un des carrés bleus de gauche jusqu'à l'un des carrés de droite; pour supprimer une connexion, il suffit de cliquer sur la ligne grisée qui la représente.

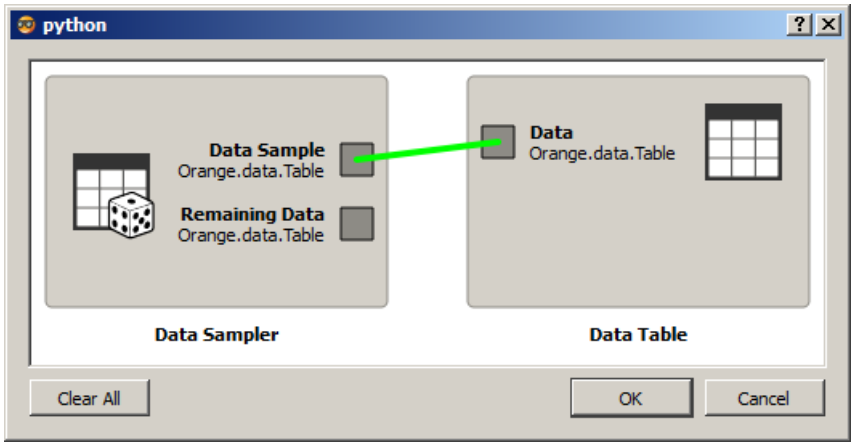



**Figure 7: Sélection des connexions sortantes et/ou entrantes.**

Instances de widgets et connexions peuvent être supprimées par le biais d'un clic-droit sur l'élément concerné puis de la sélection de **Remove** dans le menu contextuel (la suppression d'une instance entraîne celle de toutes ses connexions entrantes et sortantes). Le menu contextuel permet également de renommer les instances (**Rename**) et d'accéder à la documentation détaillée du widget (**Help**) si celle-ci est disponible.

# 5. Utilisation de TEXTABLE v1.4

TEXTABLE v1.4 est une extension d'Orange Canvas qui se compose d'un ensemble de widgets (voir figure 8 ci-dessous) permettant d'importer des données textuelles dans le logiciel et de les convertir en données tabulées. Les sections suivantes introduisent les principaux éléments de l'utilisation de ces widgets.

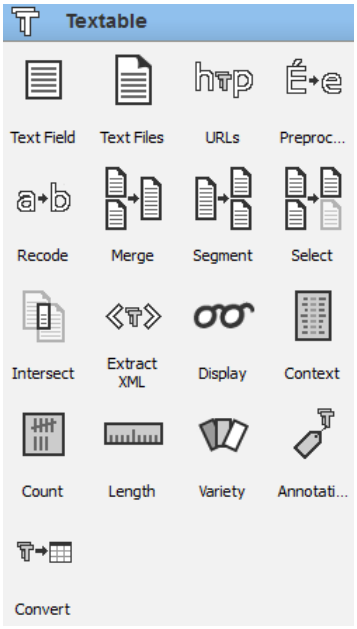



**Figure 8: L'onglet TEXTABLE (v1.4).**

## *5.1 Segmentations, segments, annotations et labels*

Les données qui circulent entre les widgets de TEXTABLE v1.4 ne sont pas, comme on pourrait s'y attendre, des chaînes de caractères. Le plus souvent, ces données sont des *segmentations*, soit des *listes de segments annotés*. Une bonne compréhension du concept de segmentation est essentielle pour l'utilisation de TEXTABLE v1.4, et la façon la plus simple d'y parvenir est d'en considérer quelques exemples.

Soit la chaîne de 17 caractères suivante:

*un exemple simple*

Chaque nombre entier compris entre 1 et 17 définit une *position* dans cette chaîne. Un *segment* peut alors être défini comme une paire d'entiers correspondants à une position initiale et une position finale. Ainsi, le segment (1, 2) correspond à la sous-chaîne *un*, (4, 10) à la sous-chaîne *exemple*, et (1, 17) à la chaîne toute entière. Dans la suite, nous dirons par exemple que *un* est le *contenu* du segment (1, 2).

Dans sa forme la plus simple, une segmentation est définie comme une liste de segments. Par exemple, la liste ((1, 2), (4, 10), (12, 17)) décrit la segmentation en mots de la chaîne considérée. Plusieurs segmentations différentes peuvent être associées à une même chaîne, comme l'illustrent les exemples suivants:

- segmentation en caractères ((1, 1), (2, 2), (3, 3), ..., (16, 16), (17, 17))
- segmentation en paires (*bigrammes*) de caractères ((1, 2), (2, 3), (3, 4), ..., (15, 16), (16, 17))
- segmentation "en chaîne" ((1, 17))
- segmentation "fantaisie" ((2, 11), (1, 4), (1, 8), (7, 8), (5, 17))
- etc.

Par ailleurs, chaque segment de chaque segmentation peut lui-même être associé à une ou plusieurs *annotations* différentes. En général, une annotation est une paire (*clé*: *valeur*), où la clé est une chaîne de caractères et la valeur un nombre ou une chaîne de caractères. La valeur sert à stocker une information arbitraire au sujet d'un segment, information accessible par le biais de la clé qui lui est associée.

Une utilisation typique des annotations est d'indiquer l'appartenance des segments à des catégories, par exemple *consonne*/*voyelle* pour un caractère, ou *nom*/*verbe*/*article*/... (ou plus simplement *lexical*/*grammatical*) pour un mot. Dans notre exemple, le segment (1,2) pourrait ainsi être associé à une annotation (*partie du discours*: *article*), ou encore (*type de mot*: *grammatical*), ce que nous noterons par convention (1, 2, {(*partie du discours*: *article*), (*type de mot*: *grammatical*)}).[3]

Un exemple complet de segmentation de notre chaîne pourrait donc être:

(
  ( 1, 2, { (*partie du discours*: *article*), (*type de mot*: *grammatical*) } ),
  ( 4, 10, { (*partie du discours*: *nom*), (*type de mot*: *lexical*) } ),
  ( 12, 17, { (*partie du discours*: *adjectif*), (*type de mot*: *lexical*) } )
)

Il est à noter que rien ne contraint chaque segment d'une segmentation à posséder exactement les mêmes clés d'annotation que les autres comme c'est le cas dans cet exemple. La segmentation suivante est donc tout aussi acceptable que la précédente, quoique sans doute moins utile:

((1, 2, {(*partie du discours*: *article*)}), (4, 10), (12, 17, {(*type de mot*: *lexical*)}))

Les exemples présentés ci-dessus sont suffisants pour comprendre la notion de segmentation telle qu'elle est utilisée dans le cadre de TEXTABLE v1.4. Au sens strict, toutefois, celle-ci est un peu plus générale. En effet, le logiciel permet de traiter plusieurs chaînes distinctes, provenant par exemple de fichiers distincts, et une segmentation peut contenir des segments appartenant à des chaînes différentes. Pour cette raison, chaque segment spécifie (en plus de sa position initiale, sa position finale et ses annotations) la chaîne à laquelle il est rattaché.

La dernière caractéristique fondamentale d'une segmentation dans TEXTABLE v1.4 est d'être associée à un *label*. Un label est simplement une chaîne de caractères choisie par l'utilisateur et utilisée pour identifier une segmentation ou un ensemble de segmentations dans le cadre du logiciel.

## *5.2 Types de données et catégories de widgets*

TEXTABLE v1.4 introduit principalement deux nouveaux types de données dans Orange Canvas: *Segmentation* et *Table*. Le type *Segmentation* correspond à une segmentation telle que définie dans la section précédente. Le type *Table* est un format de représentation de tableau permettant de stocker des chaînes Unicode (http://unicode.org) – cet encodage étant mal supporté par le type *ExampleTable* propre à Orange Canvas.

Les widgets qui composent TEXTABLE v1.4 peuvent être répartis en 4 catégories principales en fonction du type de données qu'ils admettent en entrée et émettent en sortie:

**Widgets d'importation de textes (∅ → *Segmentation*).** Ces widgets n'admettent pas de connexions entrantes et émettent des données de type *Segmentation*. Leur fonction est d'importer des données textuelles dans Orange Canvas, à partir d'une saisie au clavier (**Text Field**), de fichiers (**Text Files**) ou de données internet (**URLs**).

**Widgets de traitement de segmentations (*Segmentation* → *Segmentation*).** Cette catégorie comprend le plus grand nombre de widgets. Ils ont en commun de prendre des données de type *Segmentation* en entrée et produire des données du même type en sortie. Certains de ces widgets

[3] En d'autres termes, un segment est une liste de 3 éléments (*position initiale*, *position finale*, *annotations*), où chaque élément de l'ensemble d'annotations est une paire (*clé*: *valeur*); nous admettrons ici que la notation (*position initiale*, *position finale*) vue précédemment pour les segments est un raccourci pour (*position initiale*, *position finale*, ∅), soit le cas d'un segment non annoté.

(**Preprocess** et **Recode**) ont pour fonction de générer des données textuelles modifiées; d'autres (**Merge**, **Segment**, **Select**, **Intersect** et **Extract XML**) ne génèrent pas de nouvelles données textuelles mais uniquement de nouvelles *segmentations*; le widget **Display**, enfin, sert à visualiser les détails d'une segmentation (contenu et adresse des segments, ainsi que leurs éventuelles annotations).

**Widgets de création de tables (*Segmentation* → *Table*).** Ces widgets prennent des données de type *Segmentation* en entrée et produisent des données type *Table* en sortie. Ce sont donc ces widgets qui se chargent de la conversion des données textuelles en données tabulées, par le biais d'opérations de comptage (**Count**), de calculs de longueur (**Length**) et de diversité (**Variety**), ou encore par l'exploitation des annotations associées aux segments (**Annotation**); le widget **Context**, enfin, permet de représenter sous forme de table des concordances et des listes de collocations.

**Widget de conversion/exportation de tables (*Table* → *ExampleTable*).** Cette catégorie ne contient qu'un seul widget, **Convert**, qui admet des données de type *Table* en entrée et les convertit en données de type *ExampleTable* pour un traitement ultérieur avec les widgets standard d'Orange Canvas. Il permet en outre d'appliquer diverses transformations standard (tri, normalisation, etc.), et d'exporter les données tabulées vers des fichiers textes.

Les sections suivantes introduisent successivement les widgets appartenant à chacune de ces catégories.

## *5.3 Widgets d'importation de textes*

Les widgets de cette catégorie ont la particularité de ne pas admettre de connexions entrantes, c'est pourquoi ils apparaissent typiquement à l'extrême gauche d'un schéma. Ils importent des chaînes de caractères en mémoire et transmettent aux autres widgets des segmentations de ces chaînes. Tout widget de ce type importe au moins une chaîne de caractère et émet au moins une segmentation composée d'un segment unique recouvrant l'ensemble de la chaîne.

### 5.3.1 Widget *Text Field*

Ce widget permet à l'utilisateur d'importer dans le logiciel des données saisies au clavier. Il émet une segmentation contenant un segment (non annoté) unique couvrant l'ensemble de la chaîne.

L'interface du widget est divisée en trois zones (voir figure 9 ci-dessous). La partie supérieure est un champ de texte éditable par l'utilisateur. Les fonctions standard d'édition (copier, coller, annuler, etc.) sont accessibles par un clic-droit sur la surface du champ.

La section **Options** permet de définir le label de la segmentation sortante (**Output segmentation label**), en l'occurrence *exemple*.

La section **Info** est commune à la plupart des widgets de TEXTABLE v1.4. Elle renseigne généralement l'utilisateur sur les données d'entrée et/ou de sortie ainsi que sur le statut actuel du traitement. Dans le cas du widget **Text Field**, elle indique la longueur en caractères de la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (en particulier si aucun texte n'a été saisi dans le champ). Dans l'exemple, le segment unique contenu dans la segmentation sortante couvre 17 caractères.

Le bouton **Send** et la case à cocher **Send automatically** sont également partagés par la plupart des widgets de TEXTABLE v1.4. Le bouton a pour fonction de déclencher l'envoi de données, en

l'occurrence une segmentation, sur la ou les connexions sortantes. Lorsqu'elle est sélectionnée, la case à cocher désactive le bouton et le widget tente d'émettre automatiquement une segmentation à chaque modification de son interface (édition du texte ou modification du label).

Il est à noter que le contenu du champ de texte est systématiquement converti en Unicode; par ailleurs, il est soumis à la technique de décomposition-recomposition canonique Unicode[4], si bien que des séquences Unicode telles que *LATIN SMALL LETTER C* (U+0063) + *COMBINING CEDILLA* (U+0327) sont systématiquement remplacées par l'équivalent unitaire *LATIN SMALL LETTER C WITH CEDILLA* (U+00C7).

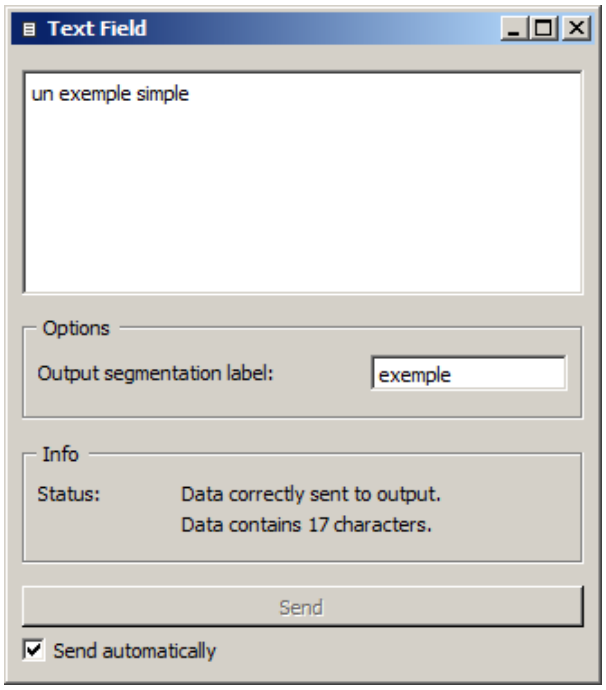



**Figure 9: Interface du widget *Text Field*.**

### 5.3.2 Widget *Text Files*

Ce widget a pour fonction d'importer le contenu d'un ou plusieurs fichiers texte dans Orange Canvas. Il émet en sortie une segmentation contenant un segment (potentiellement annoté) pour chaque fichier importé. Comme dans le cas du widget **Text Field**, le contenu textuel importé est systématiquement converti en Unicode (à partir de l'encodage défini par l'utilisateur) et soumis à la technique de décomposition-recomposition canonique Unicode (voir section 5.3.1 ci-dessus).

Comme la plupart des widgets de TEXTABLE v1.4, l'interface de **Text files** se décline en deux versions, selon que la case à cocher **Advanced Settings** est sélectionnée ou non. En général, l'interface de base donne accès à un nombre restreint de fonctionnalités jugées fondamentales, et la version avancée offre un jeu d'options plus riche et plus complexe. Ce partitionnement a surtout une vocation pédagogique: il permet à l'utilisateur débutant de se concentrer sur les aspects les plus importants de chaque widget sans pour autant restreindre la gamme de fonctionnalités accessibles à l'utilisateur confirmé.

[4] Voir http://unicode.org/reports/tr15

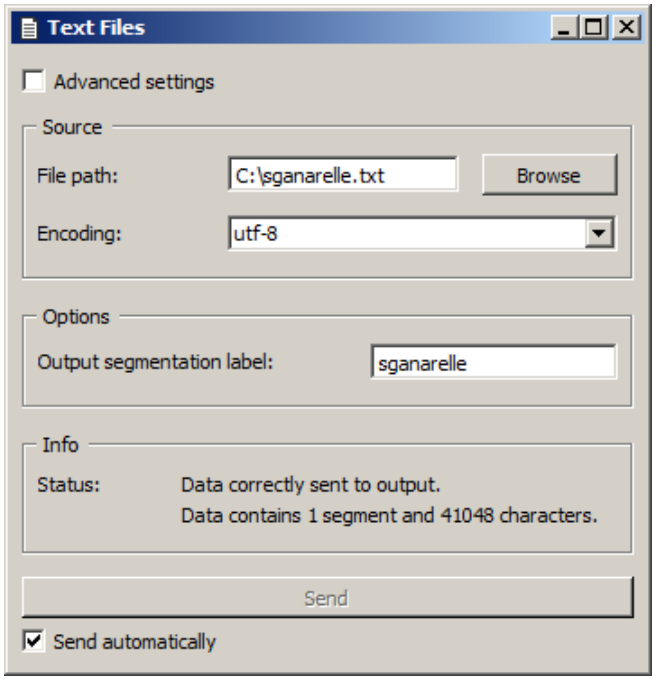



**Figure 10: Widget *Text Files* (interface de base).**

### *Interface de base*

Dans sa version de base (voir figure 10 ci-dessus), le widget **Text Files** est limité à l'importation d'un fichier unique. L'interface comprend une section **Source** permettant de sélectionner le fichier d'entrée. Le bouton **Browse** ouvre un dialogue d'ouverture de fichier; le cas échéant, le fichier sélectionné apparaît dans le champ de texte **File path** (il peut également être directement saisi au clavier). Le menu déroulant **Encoding** permet en outre de spécifier l'encodage du fichier.

La section **Options** permet de définir le label de la segmentation sortante (**Output segmentation label**).

La section **Info** indique le nombre de caractères dans le segment unique contenu dans la segmentation sortante, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, problème d'encodage, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Text Field** (voir section 5.3.1 ci-dessus).

### *Interface avancée*

La version avancée du widget **Text Files** permet d'importer plusieurs fichiers dans un ordre déterminé; chaque fichier peut par ailleurs être associé à un encodage distinct et des annotations spécifiques. La segmentation émise en sortie contient un segment pour chaque fichier importé.

La section **Sources** de l'interface avancée (voir figure 11 ci-dessous) permet de sélectionner le ou les fichier d'entrée ainsi que leur encodage, déterminer leur ordre d'apparition dans la segmentation émise, et leur assigner une éventuelle annotation. La liste des fichiers importés apparaît en haut de la fenêtre, et les colonnes de cette liste indiquent (a) le nom de chaque fichier, (b) l'annotation correspondante (le cas échéant), et (c) l'encodage qui lui est associé.

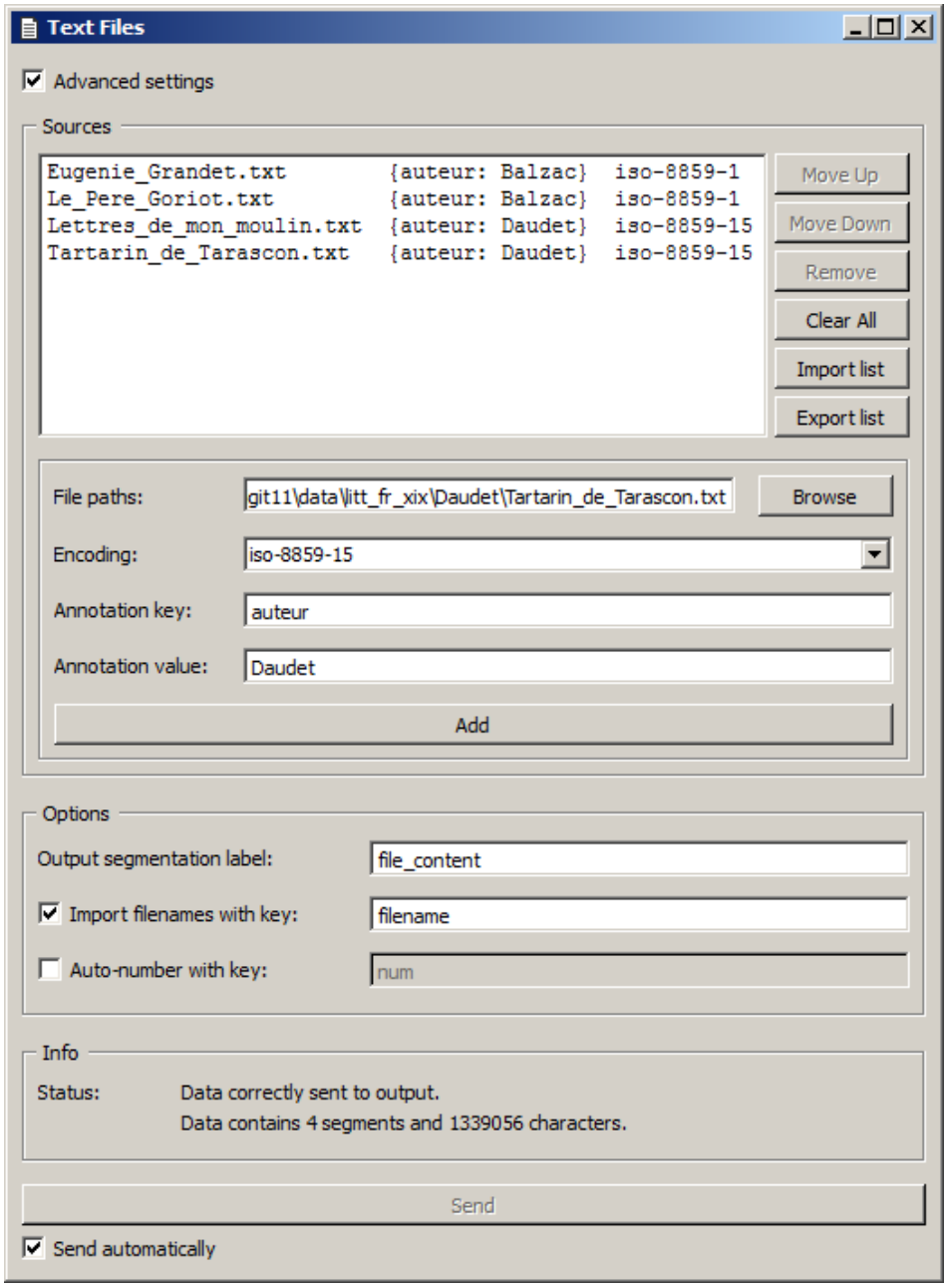



**Figure 11: Widget *Text Files* (interface avancée).**

Sur la figure 11, on voit ainsi que quatre fichiers sont importés et que tous sont pourvus d'une annotation dont la clé est *auteur*. Les deux premiers associent à cette clé la valeur *Balzac* et sont encodés en *iso-8859-1*; les deux derniers ont la valeur *Daudet* et sont encodés en *iso-8859-15*.

Les premiers boutons à droite de la liste des fichiers importés permettent de modifier leur ordre d'apparition dans la segmentation produite en sortie (**Move Up** et **Move Down**), de supprimer un fichier de la liste (**Remove**) ou de la vider entièrement (**Clear All**). A l'exception de **Clear All**, tous ces boutons requièrent de sélectionner une entrée de la liste au préalable. **Import list** permet d'importer une liste de fichiers en format JSON (voir annexe A) et de les ajouter aux sources déjà sélectionnées. **Export list** permet à l'inverse d'exporter la liste des sources dans un fichier en format JSON.

La partie restante de la section **Sources** permet d'ajouter de nouveaux fichiers dans la liste. La façon la plus simple de le faire est de cliquer d'abord sur le bouton **Browse**, qui ouvre un dialogue de

sélection de fichier. Après avoir sélectionné un ou plusieurs fichiers[5] dans ce dialogue et validé ce choix en cliquant sur **Ouvrir**, les fichiers apparaissent dans le champ de texte **File paths** et peuvent être ajoutés à la liste en cliquant sur le bouton **Add**. Il est aussi possible de saisir le chemin complet des fichiers directement dans le champ de texte, en séparant les chemins correspondant aux fichiers successifs par la chaîne " / " (*espace* + *slash* + *espace*).

Avant d'ajouter un ou des fichiers dans la liste en cliquant sur **Add**, il est possible de sélectionner leur encodage (**Encoding**), et de leur assigner une annotation en spécifiant sa clé dans le champ **Annotation key** et la valeur correspondante dans le champ **Annotation value**. Ces trois paramètres (encodage, clé et valeur) seront appliqués à chacun des fichiers apparaissant dans le champ **File paths** au moment de leur ajout dans la liste avec **Add**.

La section **Options** permet de spécifier le label affecté à la segmentation produite en sortie (**Output segmentation label**). La case à cocher **Import filenames with key** permet de créer pour chaque fichier importé une annotation dont la valeur est le nom du fichier (tel qu'affiché dans la liste) et dont la clé est spécifiée par l'utilisateur dans le champ de texte à droite de la case à cocher. De façon similaire, la case **Auto-number with key** permet de numéroter automatiquement les fichiers importés et d'associer le numéro à la clé d'annotation spécifiée dans le champ de texte à droite.

Sur la figure 11, on a ainsi décidé d'assigner le label *littérature_XIXe* à la segmentation produite, et d'associer le nom de chaque fichier à la clé d'annotation *filename*. En revanche l'option d'auto-numérotation n'a pas été activée.

La section **Info** indique la longueur en caractères de la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de fichier sélectionné, problème d'encodage, etc.). Dans l'exemple, les 4 segments correspondant aux fichiers importés totalisent ainsi 1'339'056 caractères.

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Text Field** (voir section 5.3.1 ci-dessus).

### 5.3.3 Widget *URLs*

Ce widget permet d'importer le contenu associé à une ou plusieurs adresses internet (URLs) dans Orange Canvas. Il émet en sortie une segmentation contenant un segment (potentiellement annoté) pour chaque URL dont le contenu est importé. Comme dans le cas du widget **Text Field**, le contenu textuel importé est systématiquement converti en Unicode (à partir de l'encodage défini par l'utilisateur) et soumis à la technique de décomposition-recomposition canonique Unicode (voir section 5.3.1 ci-dessus).

L'interface du widget est une réplique presque conforme de celle du widget **Text Files** (voir section 5.3.2 ci-dessus) et seules les différences sont documentées ici (voir aussi annexe A).

#### *Interface de base*

Dans sa version de base (voir figure 12 ci-dessous), le widget **URLs** est limité à l'importation du contenu d'une URL unique. L'URL est spécifiée par le biais du champ du même nom dans la section **Source**.

[5] Pour sélectionner plusieurs fichiers, il est possible d'utiliser *shift+clic* ou *ctrl+clic*.

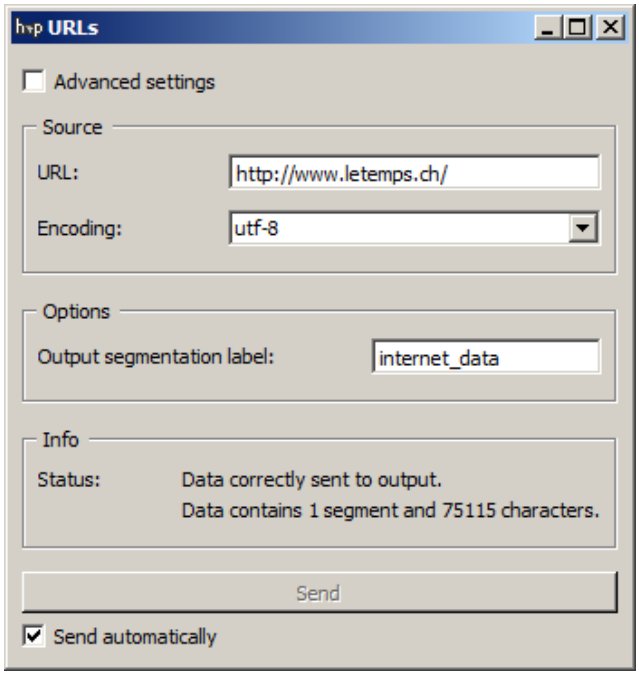



**Figure 12: Widget *URLs* (interface de base).**

### *Interface avancée*

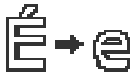
La version avancée du widget **URLs** permet d'importer le contenu de plusieurs URLs dans un ordre déterminé; chaque URL peut par ailleurs être associé à un encodage distinct et des annotations spécifiques. La segmentation émise en sortie contient un segment pour chaque URL importée.

Pour importer le contenu d'une ou plusieurs URLs, celles-ci doivent être saisies dans le champ du même nom dans la section **Sources** de l'interface avancée (voir figure 13 ci-dessous), avant de cliquer sur **Add**. Le cas échéant, les URLs successives doivent être séparées par la chaîne " / " (*espace* + *slash* + *espace*).

## *5.4 Widgets de traitement de segmentations*

Les widgets de cette catégorie prennent des segmentations en entrée et émettent des segmentations en sortie. Certains d'entre eux assument des fonctions de recodage (**Preprocess**, **Recode**), c'est-à-dire qu'ils génèrent des données textuelles modifiées. D'autres ne génèrent pas de nouvelles données textuelles mais uniquement de nouvelles segmentations (**Merge**, **Segment**, **Select**, **Intersect** et **Extract XML**). Le widget **Display**, enfin, permet de visualiser les détails d'une segmentation.

### 5.4.1 Widget *Preprocess*



Ce widget prend une segmentation en entrée, crée une copie modifiée du contenu de cette segmentation, et renvoie en sortie une nouvelle segmentation correspondant aux données modifiées. Les modifications possibles sont la modification de la casse (minuscule/majuscule) et le remplacement des caractères accentués par leurs équivalents non accentués.

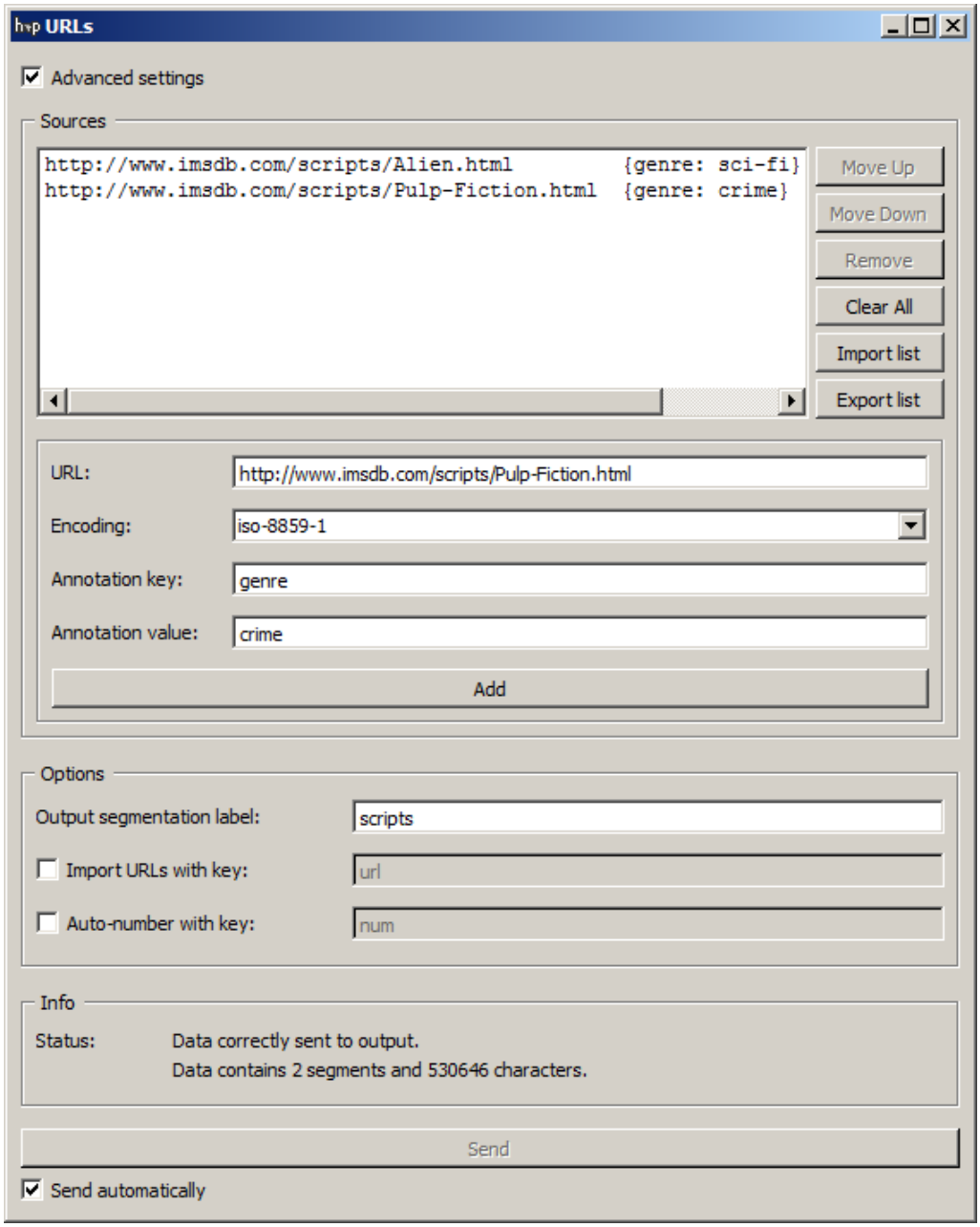



**Figure 13: Widget *URLs* (interface avancée).**

Il est à noter que le widget **Preprocess** crée une copie de chaque segment modifié en mémoire, ce qui peut entraîner un dépassement de la capacité de mémoire vive de la machine; par ailleurs, il ne peut s'appliquer qu'à des segmentations dépourvues de chevauchements, c'est-à-dire telles qu'aucune partie du texte n'est couverte par plus d'un segment.

Dans la section **Preprocessing** de l'interface du widget (voir figure 14 ci-dessous), la case à cocher **Transform case** déclenche la modification systématique de la casse: sélectionner **to lower** pour convertir tous les caractères en minuscules et **to upper** pour les convertir en majuscules. La case **Remove accents** contrôle le remplacement des caractères accentués par leurs équivalents non accentués (par exemple *é* → *e*, *ç* → *c, etc.*).

La section **Options** permet de définir le label de la segmentation produite en sortie (**Output segmentation label**). La case **Copy annotations** copie toutes les annotations de la segmentation

d'entrée vers la segmentation de sortie; elle n'est accessible que lorsque la case **Advanced settings** est sélectionnée (dans le cas contraire, les annotations sont copiées par défaut).

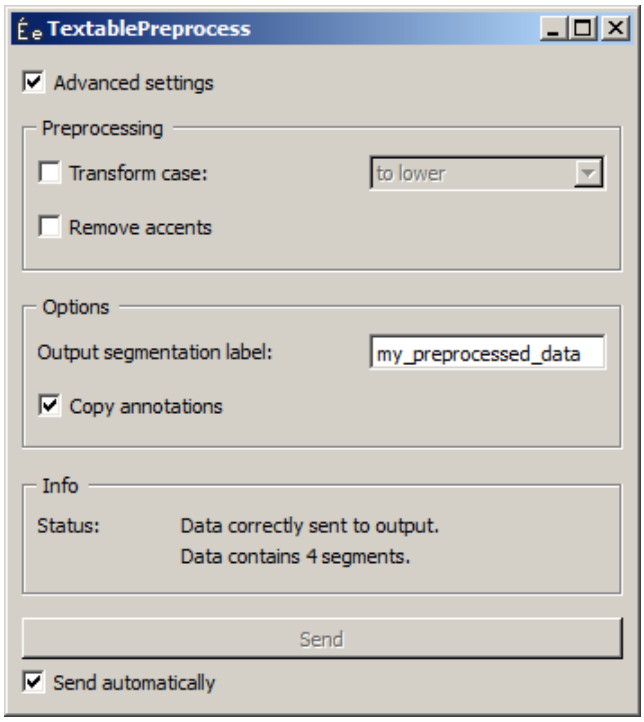
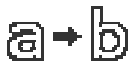



**Figure 14: Widget *Preprocess*.**

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, chevauchements dans la segmentation d'entrée, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent essentiellement de la même façon que dans le widget **Text Field** (voir section 5.3.1 ci-dessus). La seule différence est que si la case à cocher est sélectionnée, le widget envoie automatiquement une segmentation également lorsque ses données entrantes sont modifiées (par suppression ou ajout d'une connexion, ou parce que des données modifiées sont reçues sur une connexion existante).

### 5.4.2 Widget *Recode*



Ce widget prend une segmentation en entrée, crée une copie modifiée du contenu de cette segmentation, et renvoie en sortie une nouvelle segmentation correspondant aux données modifiées. Les modifications appliquées sont définies par des *substitutions*, soit des paires composées d'une expression régulière (servant à identifier les portions de texte à modifier) et d'une chaîne de remplacement.

Il est possible de "capturer" des portions de texte au moyen des parenthèses apparaissant dans les expressions régulières, afin de les insérer dans les chaînes de remplacement, où les séquences *&1*, *&2*, etc. correspondent aux paires de parenthèses successives (numérotées sur la base de la position des parenthèses ouvrantes).

De même que le widget **Preprocess** (voir section 5.4.1 ci-dessus), le widget **Recode** crée une copie de chaque segment modifié en mémoire, ce qui peut entraîner un dépassement de la capacité de mémoire vive de la machine; par ailleurs, les deux widgets ne peuvent s'appliquer qu'à des segmentation dépourvues de chevauchements, c'est-à-dire telles qu'aucune partie du texte n'est couverte par plus d'un segment.

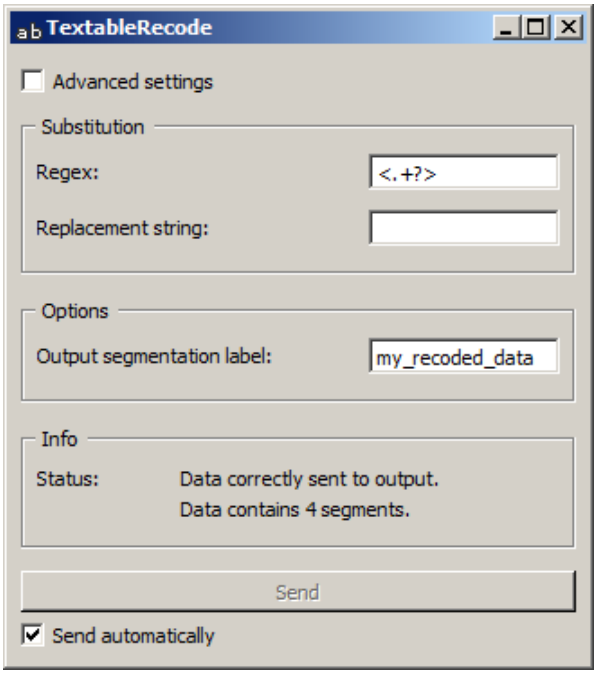



**Figure 15: Widget *Recode* (interface de base).**

## *Interface de base*

La version de base du widget est limitée à l'application d'une substitution unique. La section **Substitution** (voir figure 15 ci-dessus) permet de spécifier l'expression régulière (**Regex**) et la chaîne de remplacement (**Replacement string**) correspondante. Si la chaîne de remplacement est laissée vide, les portions de textes identifiées par l'expression régulière seront simplement supprimées; c'est le cas dans l'exemple de la figure 15, qui aboutit à la suppression des balises XML/HTML. [6]

La section **Options** permet de définir le label de la segmentation sortante (**Output segmentation label**). Les annotations de chaque segment entrant sont systématiquement copiées dans les segments sortants correspondants (voir *Interface avancée* ci-dessous, option **Copy annotations**).

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, chevauchements dans la segmentation d'entrée, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

## *Interface avancée*

Dans sa version avancée, le widget **Recode** permet de définir plusieurs substitutions et de déterminer l'ordre dans lequel elles doivent être appliquées successivement à chaque segment de la segmentation entrante.

[6] Pour plus de détails concernant la syntaxe des expressions régulières, voir la documentation de Python (http://docs.python.org/library/re.html). Notez que l'option *-u* (*Unicode dependent*) est activée par défaut.

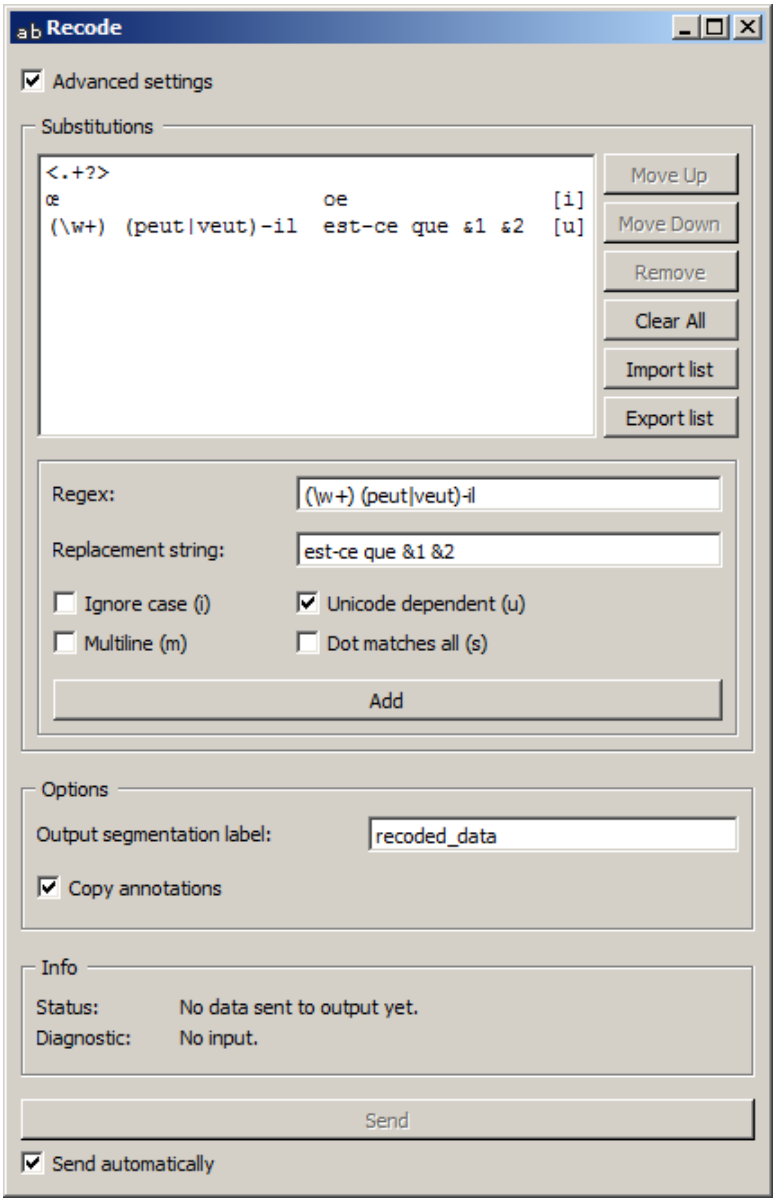



**Figure 16: Widget *Recode* (interface avancée).**

L'interface avancée (voir figure 16 ci-dessus) présente des similarités avec celle du widget **Text Files** (voir section 5.3.2 ci-dessus). La section **Substitutions** permet de définir les substitutions appliquées à chaque segment entrant successif et de déterminer leur ordre d'application. Dans la liste qui apparaît en haut de la fenêtre, chaque ligne spécifie une substitution, et les colonnes indiquent pour chaque substitution (a) l'expression régulière correspondante, (b) la chaîne de remplacement (qui peut être vide), et (c) les options associées à l'expression régulière.[7]

Sur la figure 16, on voit ainsi que trois substitutions ont été définies. La première supprime les balises xml/html (elle les remplace par la chaîne vide). La seconde remplace la ligature *œ* (ou *Œ*, puisque l'option **Ignore case** est sélectionnée, comme indiqué par la lettre *i* entre crochets carrés) par la séquence *oe*.

La dernière substitution est plus complexe: l'expression régulière identifie des séquences composées d'un mot (*\w+*) suivi par un espace, puis par *peut-il* ou *veut-il*; ces séquences sont alors remplacées

[7] Pour plus de détails concernant l'effet des options *i*, *u*, *m* et *s*, voir la documentation de Python (http://docs.python.org/library/re.html).

par *est-ce que* suivi par un espace, puis par le mot identifié avec *\w+*, un espace, et enfin la forme verbale (*peut* ou *veut*) présente dans la chaîne originale. Cette substitution illustre la possibilité de "capturer" des portions de texte au moyen des parenthèses apparaissant dans l'expression régulière.

Pour prendre un exemple concret, l'application successive de ces trois substitutions à la chaîne

> *<exemple>Bob peut-il se fier à son cœur?</exemple>*

produira tour à tour les versions modifiées

> *Bob peut-il se fier à son cœur?*
> *Bob peut-il se fier à son coeur?*
> *est-ce que Bob peut se fier à son coeur?*

Les premiers boutons à droite de la liste des substitutions permettent de modifier l'ordre dans lequel elles sont successivement appliquées à chaque segment de la segmentation entrante (**Move Up** et **Move Down**), de supprimer une substitution de la liste (**Remove**) ou de la vider entièrement (**Clear All**). A l'exception de **Clear All**, tous ces boutons requièrent de sélectionner une entrée de la liste au préalable. **Import list** permet d'importer une liste de substitutions en format JSON (voir annexe A) et de les ajouter aux sources déjà sélectionnées. **Export list** permet à l'inverse d'exporter la liste des substitutions dans un fichier en format JSON.

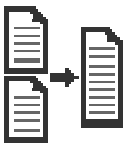
La partie restante de la section **Substitutions** permet d'ajouter de nouvelles substitutions dans la liste. Pour définir une nouvelle substitution, il faut spécifier l'expression régulière (**Regex**) et la chaîne de remplacement (**Replacement string**) correspondantes; cette dernière peut être laissée vide, auquel cas les portions de textes identifiées par l'expression régulière seront simplement supprimées. Les cases à cocher **Ignore case (i)**, **Unicode dependent (u)**, **Multiline (m)** et **Dot matches all (s)** contrôlent l'application des options correspondantes à l'expression régulière. L'ajout de la nouvelle substitution à la liste s'effectue finalement en cliquant sur le bouton **Add**.

La section **Options** permet de définir le label de la segmentation produite en sortie (**Output segmentation label**). La case à cocher **Copy annotations** copie toutes les annotations de la segmentation d'entrée vers la segmentation de sortie.

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, chevauchements dans la segmentation d'entrée, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

### 5.4.3 Widget *Merge*



Ce widget prend plusieurs segmentations en entrée, copie successivement chaque segment de chaque segmentation entrante pour former une nouvelle segmentation, et renvoie cette segmentation sur ses connexions sortantes.

La section **Ordering** de l'interface du widget (voir figure 17 ci-dessus) permet de sélectionner l'ordre dans lequel les segmentations entrantes sont placées pour former la segmentation sortante fusionnée. Le label de chaque segmentation entrante apparaît sur une ligne de la liste et peut être sélectionné puis déplacé en cliquant sur les boutons **Move Up** et **Move Down**.

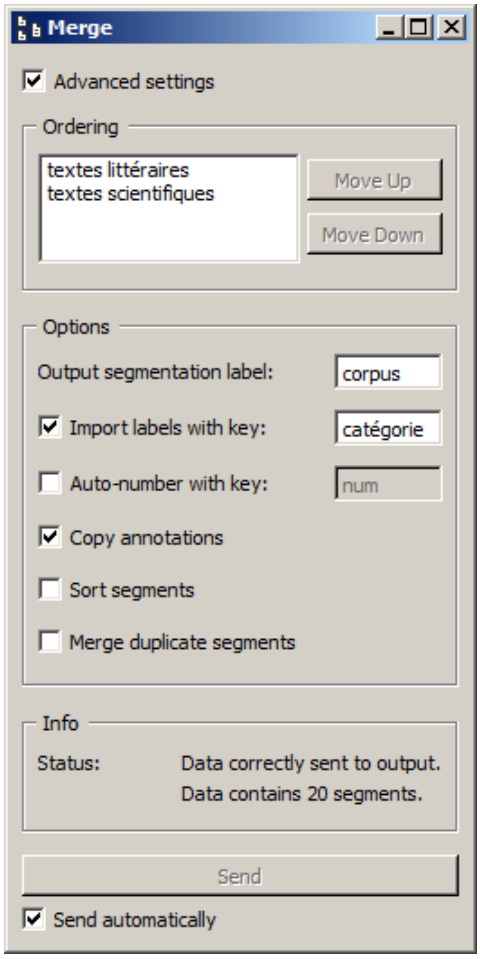



**Figure 17: Widget *Merge*.**

La section **Options** permet de spécifier le label affecté à la segmentation produite en sortie (**Output segmentation label**). La case à cocher **Import labels with key** permet de créer pour chaque segmentation entrante une annotation dont la valeur est le label de la segmentation (tel qu'affiché dans la liste) et dont la clé est spécifiée par l'utilisateur dans le champ de texte à droite de la case à cocher. De façon similaire, la case **Auto-number with key** permet de numéroter automatiquement les segmentations entrantes et d'associer le numéro à la clé d'annotation spécifiée dans le champ de texte à droite. La case à cocher **Copy annotations** copie toutes les annotations des segmentations entrantes vers la segmentation émise.

Les deux derniers éléments de la section **Options** influencent l'ordre des segments contenus dans la segmentation sortante ainsi que leur nombre. La case à cocher **Sort segments** permet de trier les segments sur la base de leur adresse (numéro de chaîne d'abord, puis position initiale, et enfin position finale); cette option est utile typiquement pour réarranger dans l'ordre du texte des segments appartenant à des segmentations différente d'un même texte. La case à cocher **Merge duplicate segments** permet de fusionner en un seul segment des segments distincts mais dont l'adresse est la même; les annotations associées aux segments fusionnés sont toutes copiées dans le segment unique résultant.[8]

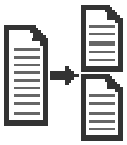
[8] Dans le cas où les segments fusionnés ont des valeurs distinctes pour la même clé d'annotation, seule la valeur du dernier segment (dans l'ordre de la segmentation sortante avant fusion) sera conservée.

Lorsque la case à cocher **Advanced settings** n'est pas sélectionnée, seules les options **Output segmentation label** et **Import labels with key** sont accessibles. Dans ce cas, l'auto-numérotation est desactivée, les annotations copiées par défaut, et les segments triés par adresse mais non fusionnés.

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, pas de label sélectionné pour la segmentation sortante, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

### 5.4.4 Widget *Segment*



Ce widget prend une segmentation en entrée et crée une nouvelle segmentation en subdivisant chaque segment d'origine en une série de nouveaux segments. Par défaut, il opère sur la base d'une description de la forme des nouveaux segments (au moyen d'expressions régulières); alternativement, il peut aussi fonctionner à partir d'une description des séparateurs qui apparaissent entre les segments. Il permet également de créer des annotations pour les segments produits.

De même qu'avec le widget **Recode** (voir section 5.4.2 ci-dessus), il est possible de "capturer" des portions de texte au moyen des parenthèses apparaissant dans une expression régulière, en l'occurrence pour les recopier dans la clé d'annotation et/ou la valeur associée; on utilisera pour cela les séquences *&1*, *&2*, etc. correspondant aux paires de parenthèses successives (numérotées sur la base de la position des parenthèses ouvrantes) de l'expression régulière.[9]

#### *Interface de base*

La version de base du widget est limitée à l'application d'une expression régulière unique, spécifiée dans la section **Regex** (voir figure 18 ci-dessous). L'expression donnée en exemple (.) crée un segment pour chaque caractère de chaque segment entrant.[10]

La section **Options** permet de définir le label de la segmentation sortante (**Output segmentation label**). Les annotations de chaque segment entrant sont systématiquement copiées dans les segments sortants correspondants (voir *Interface avancée* ci-dessous, option **Import annotations**).

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, chevauchements dans la segmentation d'entrée, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

---

[9] Cette possibilité ne s'applique pas lorsque le widget est configuré pour identifier les séparateurs plutôt que les segments eux-mêmes (**Mode: Split**, voir *Interface avancée* ci-dessous).

[10] Il est à noter que l'option *-u* (*Unicode dependent*) est activée par défaut (voir la documentation de Python, http://docs.python.org/library/re.html).

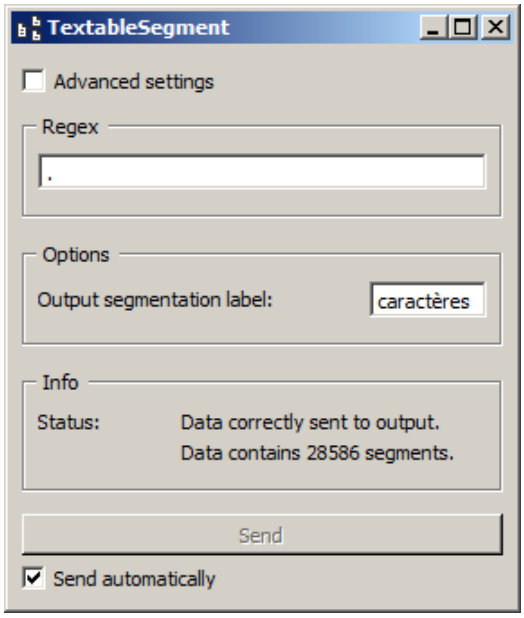



**Figure 18: Widget *Segment* (interface de base).**

### *Interface avancée*

Dans sa version avancée, le widget permet de définir plusieurs expressions régulières et de déterminer l'ordre dans lequel elles doivent être appliquées successivement à chaque segment de la segmentation entrante. Il permet également de spécifier si une expression régulière donnée décrit la forme des segments visés (mode **Tokenize**) ou plutôt celle des séparateurs entre ces segments (mode **Split**[11]).

L'interface avancée (voir figure 19 ci-dessous) présente des similarités avec celle du widget **Recode** (voir section 5.4.2 ci-dessus). La section **Regexes** permet de définir les expressions régulières appliquées successivement à chaque segment de la segmentation entrante et de déterminer leur ordre d'application. Dans la liste qui apparaît en haut de la fenêtre, les colonnes indiquent (a) le mode associé à cette expression régulière, soit *t* pour **Tokenize**, choix par défaut, ou *s* pour **Split**, (b) l'expression proprement dite, (c) l'annotation correspondante (le cas échéant), et (d) les options associées à cette expression.

Sur la figure 19, on voit ainsi que quatre expressions régulières ont été définies, toutes avec le mode *Tokenize*; chacune identifie un type de caractère dans la segmentation entrante et lui assigne une annotation dont la clé est *type*. Les classes de caractères identifiées par les quatre expressions ne sont pas mutuellement exclusives, mais après les avoir successivement appliquées, le widget applique automatiquement un tri des segments et fusionne ceux dont l'adresse est identique – exactement comme les options **Sort segments** et **Merge duplicate segments** du widget **Merge** (voir section 5.4.3 ci-dessus, et note 8 en particulier). Au final, chaque caractère appartient ainsi à un segment unique, dont la valeur pour la clé d'annotation *type* est la dernière qui lui a été affectée selon l'ordre d'application des expressions régulières.

La première des quatre expressions crée un segment pour chaque caractère (.) et lui assigne l'annotation *autre*. La seconde crée un segment pour chaque caractère *alphanumérique* (\w), et lui assigne l'annotation *voyelle*. Les deux dernières identifient respectivement les consonnes et les chiffres et les annotent comme tels. Pour illustrer le mécanisme évoqué au paragraphe précédent, on peut noter qu'avant le tri des segments et la fusion des duplicats, chaque consonne de la

[11] NB: en mode **Split**, les éventuels segments vides compris résultant de la segmentation sont supprimés.

segmentation entrante est associée à trois segments dont les valeurs pour la clé d'annotation *type* sont (dans l'ordre) *autre*, *voyelle* et *consonne*; après le tri et la fusion, seule la dernière de ces valeurs sera conservée.

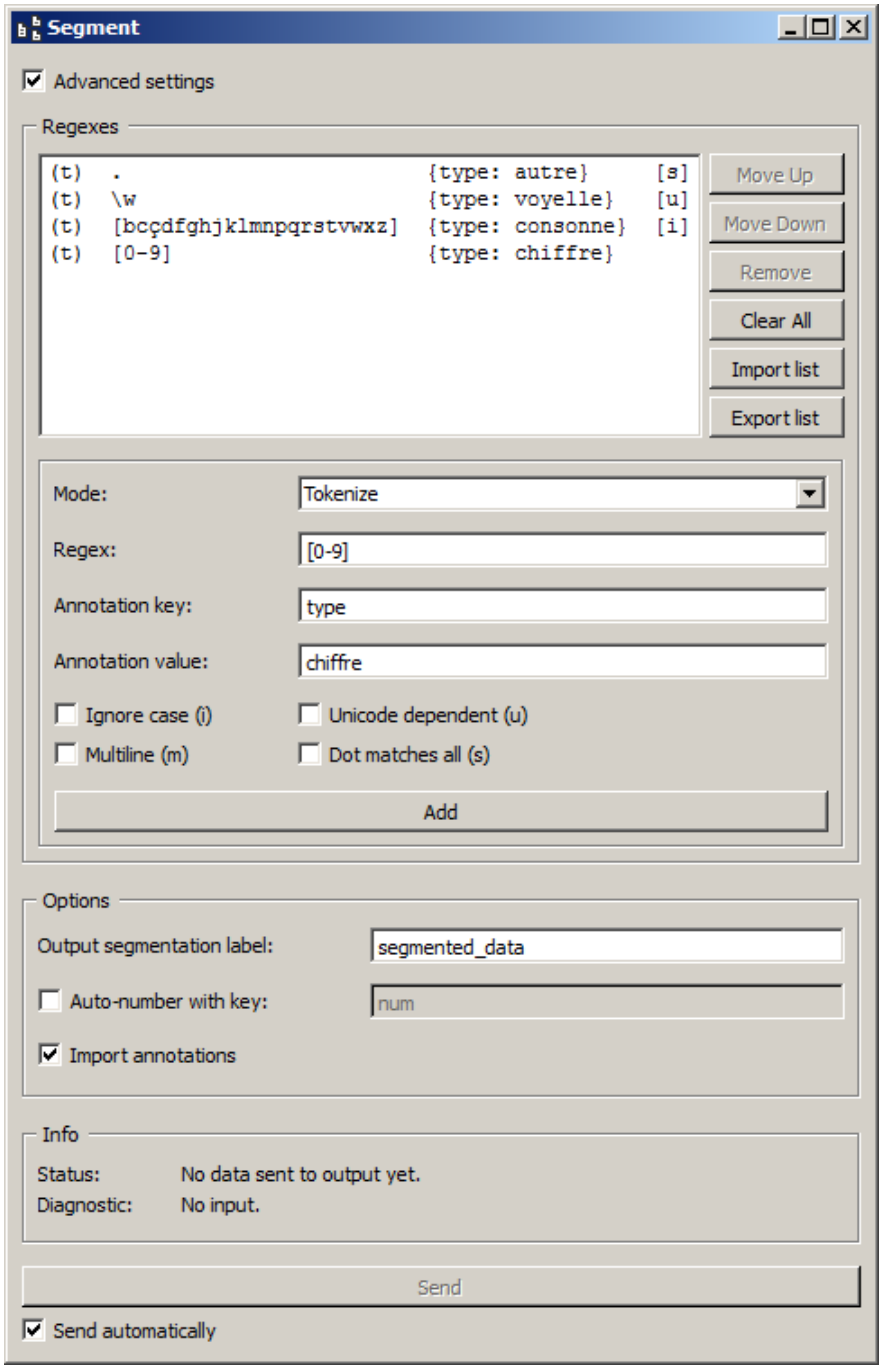



**Figure 19: Widget *Segment* (interface avancée).**

Les permiers boutons à droite de la liste des expressions régulières permettent de modifier l'ordre dans lequel elles sont successivement appliquées à chaque segment de la segmentation entrante (**Move Up** et **Move Down**), de supprimer une expression de la liste (**Remove**) ou de la vider entièrement (**Clear All**). A l'exception de **Clear All**, tous ces boutons requièrent de sélectionner une entrée de la liste au préalable. **Import list** permet d'importer une liste de substitutions en format JSON (voir annexe A) et de les ajouter aux sources déjà sélectionnées. **Export list** permet à l'inverse d'exporter la liste des substitutions dans un fichier en format JSON.

La partie restante de la section **Regexes** permet d'ajouter de nouvelles expressions régulières dans la liste. A cet effet, il faut spécifier l'expression régulière (**Regex**) et, optionnellement, la clé d'annotation (**Annotation key**) et la valeur (**Annotation value**) correspondantes. Les cases à cocher **Ignore case (i)**, **Unicode dependent (u)**, **Multiline (m)** et **Dot matches all (s)** contrôlent l'application des options correspondantes à l'expression régulière. L'ajout de la nouvelle expression à la liste s'effectue finalement en cliquant sur le bouton **Add**.

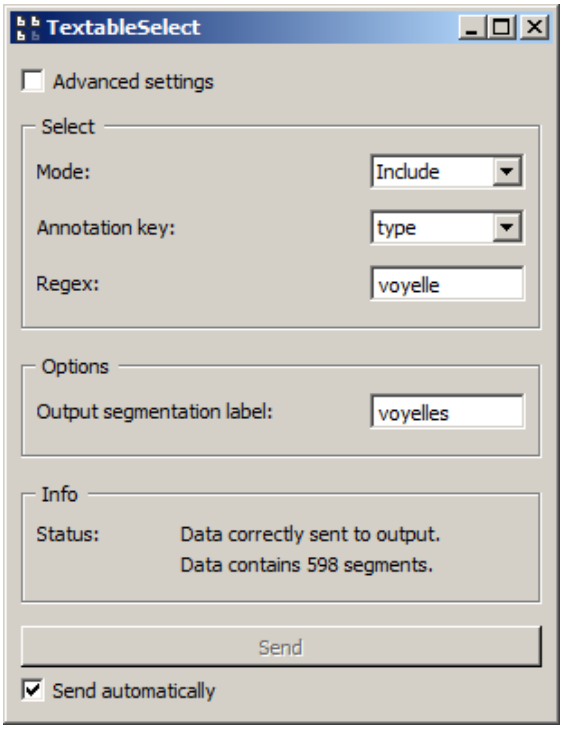


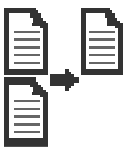

**Figure 20: Widget *Select* (interface de base).**

La section **Options** permet de spécifier le label affecté à la segmentation produite en sortie (**Output segmentation label**). La case à cocher **Auto-number with key** permet de numéroter automatiquement les segments de la segmentation sortante et d'associer le numéro à la clé d'annotation spécifiée dans le champ de texte à droite. La case à cocher **Import annotations** copie dans chaque segment de la segmentation sortante toutes les annotations associées au segment correspondant dans la segmentation entrante.

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, aucune expression régulière spécifiée, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

### 5.4.5 Widget *Select*



Ce widget prend une segmentation en entrée et crée une nouvelle segmentation sur la base d'une partie des segments entrants. La sélection des segments peut être basée sur leur contenu, leurs annotations ou leur fréquence; elle peut également être aléatoire. Quelle que soit la méthode

utilisée, le widget émet sur une seconde connexion sortante (non sélectionnée par défaut) une segmentation contenant les segments *non* sélectionnés.

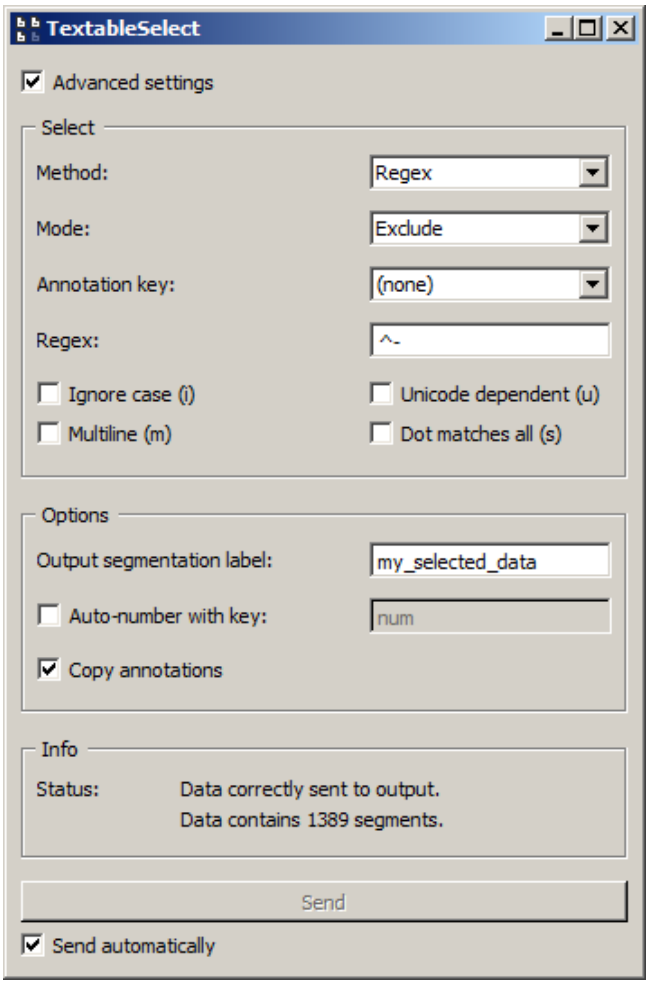



**Figure 21: Widget *Select* (interface avancée, méthode *Regex*).**

## *Interface de base*

La version de base du widget (voir figure 20 ci-dessus) est limitée à la sélection de segments sur la base de leur correspondance avec une expression régulière (voir *Interface avancée* ci-dessous, méthode **Regex**). Les différences avec l'interface avancée sont les suivantes: (i) les options des expressions régulières ne sont pas accessibles (*-u*, *Unicode dependent*, est toutefois activée par défaut); (ii) l'auto-numérotation est desactivée; et (iii) la copie des annotations est effectuée par défaut.

## *Interface avancée*

Dans sa version avancée, la section **Select** de l'interface du widget se décline en trois versions selon la valeur choisie pour le menu déroulant **Method** (voir figures 21 à 23 ci-dessous).

### A) Method: Regex

Cette méthode consiste à sélectionner les segments de la segmentation entrante en fonction de la correspondance de leur contenu ou de leurs annotations avec une expression régulière. Le menu déroulant **Mode** (voir figure 21 ci-dessous) permet de spécifier si les segments correspondant à l'expression régulière doivent être sélectionnés (**Include**) ou non (**Exclude**), auquel cas ce sont les segments *ne* correspondant *pas* à l'expression qui seront sélectionnés.

Le menu déroulant **Annotation key** permet de choisir une clé d'annotation de la segmentation entrante; le cas échéant, ce sont les segments dont la valeur d'annotation pour cette clé correspond à l'expression régulière qui seront sélectionnés (ou non). Si la valeur **(none)** est sélectionnée, c'est le *contenu* des segments qui sera soumis à l'expression régulière.

Le champ **Regex** sert à spécifier l'expression régulière utilisée pour la sélection des segments, et les cases à cocher **Ignore case (i)**, **Unicode dependent (u)**, **Multiline (m)** et **Dot matches all (s)** contrôlent l'application des options correspondantes à cette expression.

Dans l'exemple de la figure 21 ci-dessous, le widget est configuré pour exclure (**Mode: Exclude**) de la segmentation entrante les segments dont le contenu (**Annotation key: (none)**) commence par un tiret (**Regex:** ^-).

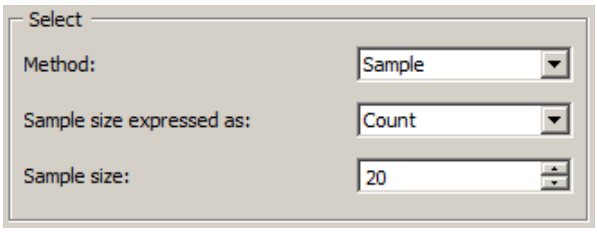



**Figure 22: Widget *Select* (interface avancée, méthode *Sample*).**

### B) Method: Sample

Cette méthode (voir figure 22 ci-dessus) consiste à sélectionner les segments de la segmentation entrante par le biais d'un processus d'échantillonnage aléatoire, tel que tous les segments d'entrée ont une probabilité égale d'être retenus ou non.

Le menu déroulant **Sample size expressed as** permet de choisir la façon d'exprimer la taille souhaitée pour l'échantillon. Si la valeur **Count** est sélectionnée, comme sur la figure 22, la taille de l'échantillon sera exprimée directement en nombre de segments (**Sample size**). Si la valeur **Proportion** est sélectionnée, la taille sera exprimée en pourcentage du nombre de segments entrants (**Sampling rate (%)**).

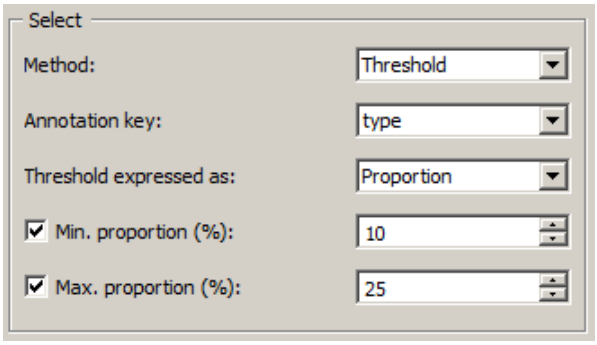



**Figure 23: Widget *Select* (interface avancée, méthode *Threshold*).**

### C) Method: Threshold

Cette méthode consiste à ne retenir de la segmentation entrante que les segments dont le contenu (ou la valeur d'annotation pour une clé donnée) a une fréquence comprise entre des bornes données.

Le menu déroulant **Annotation key** (voir figure 23 ci-dessus) permet de sélectionner une clé d'annotation de la segmentation entrante; le cas échéant, c'est la fréquence des valeurs d'annotation associées à cette clé qui conditionnera l'inclusion des segments entrants. Si la valeur **(none)** est sélectionnée, c'est la fréquence du *contenu* des segments qui sera déterminante.

Le menu déroulant **Threshold expressed as** permet de choisir la façon d'exprimer les seuils de fréquence minimaux et maximaux. Si la valeur **Count** est sélectionnée, les seuils seront exprimés en fréquences absolues (**Min./Max. count**). Si la valeur **Proportion** est sélectionnée, comme sur la figure 23, les seuils seront exprimés en fréquence relative (**Min./Max. proportion (%)**). Pour les deux valeurs (minimum et maximum), le seuillage n'est effectué que si la case correspondante est cochée.

Dans l'exemple de la figure 23, le widget est configuré pour ne retenir que les segments dont la valeur d'annotation pour la clé *type* (**Annotation key**) a une fréquence relative (**Threshold expressed as: Proportion**) comprise entre 10% (**Min. proportion (%)**) et 25% (**Max. proportion (%)**) dans la segmentation entrante.

Les éléments de la section **Options** de l'interface du widget sont communs aux trois méthodes de sélection présentés ci-dessus. Le champ **Output segmentation label** permet de spécifier le label affecté à la segmentation produite en sortie.[12] La case à cocher **Auto-number with key** permet de numéroter automatiquement les segments de la segmentation sortante et d'associer le numéro à la clé d'annotation spécifiée dans le champ de texte à droite. La case à cocher **Copy annotations** copie toutes les annotations de la segmentation entrante vers la segmentation émise.

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, aucun segment entrant sélectionné, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

### 5.4.6 Widget *Intersect*

Ce widget prend plusieurs segmentations en entrée et sélectionne les segments d'une d'entre elles (segmentation "source") sur la base des segments présents dans une autre (segmentation "filtre"). Il émet sur une seconde connexion sortante (non sélectionnée par défaut) une segmentation contenant les segments *non* sélectionnés.

La section **Intersect** de l'interface du widget (voir figure 24 ci-dessous) permet de spécifier si les segments de la segmentation source qui correspondent à un type présent dans la segmentation filtre doivent être inclus (**Mode: Include**) dans la segmentation sortante ou exclus (**Mode: Exclude**) de celle-ci. Cette section sert également à sélectionner parmi les segmentations entrantes la segmentation source (**Source segmentation**) et la segmentation filtre (**Filter segmentation**).[13]

Le menu déroulant **Source annotation key** permet de sélectionner une clé d'annotation de la segmentation source; le cas échéant, ce sont les segments dont la valeur d'annotation pour cette clé correspond à un type présent dans la segmentation filtre qui seront in-/exclus. Si la valeur **(none)** est sélectionnée, c'est le *contenu* des segments qui sera déterminant. De façon analogue, si la valeur d'une clé d'annotation de la segmentation filtre est sélectionnée dans le menu déroulant **Filter annotation key** (accessible uniquement lorsque la case à cocher **Advanced Settings** est

[12] Il s'agit ici de la segmentation contenant les segments sélectionnés et émise sur le canal par défaut; la segmentation contenant les segments restants reçoit le même label précédé par le préfixe "NEG_".

[13] Il est à noter que l'interface permet en principe d'utiliser la même segmentation comme source et comme filtre, ce qui ne peut avoir de sens que si des valeurs différentes sont sélectionnées dans les menus **Source annotation key** et **Filter annotation key**.

sélectionnée), ce sont les types de valeurs d'annotation correspondantes (plutôt que de *contenu* des segments) qui conditionneront l'in-/exclusion des segments de la segmentation source.

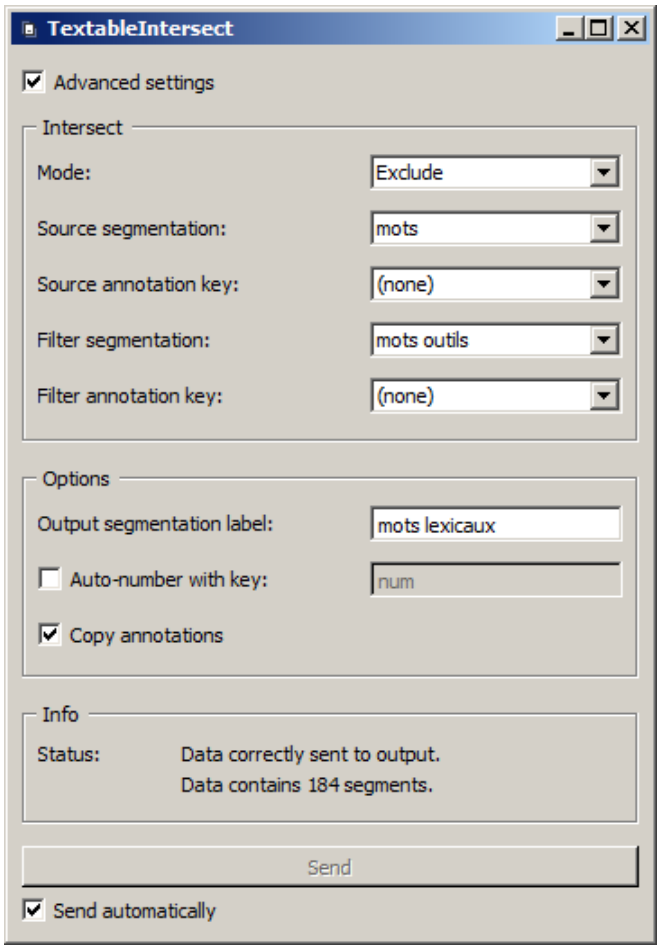



**Figure 24: Widget *Intersect*.**

Ainsi, sur la figure 24 ci-dessus, le widget reçoit en entrée deux segmentations. La première (**Source segmentation**), dont le label est *mots*, est le résultat de la segmentation d'un texte en mots, telle qu'effectuée avec le widget **Segment** (voir section 5.4.4 ci-dessus). La seconde (**Filter segmentation**), dont le label est *mots-outils*, est le fruit de la segmentation en mots d'une liste de mots dits "outils" (articles, pronoms, prépositions, etc.) – typiquement utilisée en recherche d'information pour exclure des requêtes ces mots jugés non pertinents.

Comme les menus déroulants **Source annotation key** et **Filter annotation key** sont réglés sur la valeur **(none)**, c'est le contenu des segments des deux segmentations qui détermine la suite des opérations plutôt que les valeurs d'une clé d'annotation donnée. Concrètement, les segments de la segmentation source (soit les mots du texte) dont le contenu correspond à celui d'un segment de la segmentation filtre (soit à un mot-outil) seront exclus (**Mode: Exclude**) de la segmentation sortante. Par contraste, choisir la valeur **Include** aboutirait à n'inclure en sortie que les mots-outils du texte.

La section **Options** permet de spécifier le label affecté à la segmentation produite en sortie (**Output segmentation label)**.[14] La case à cocher **Auto-number with key** permet de numéroter automatiquement les segments de la segmentation sortante et d'associer le numéro à la clé d'annotation spécifiée dans le champ de texte à droite. La case à cocher **Copy annotations** copie

[14] Il s'agit ici de la segmentation contenant les segments sélectionnés et émise sur le canal par défaut; la segmentation contenant les segments restants reçoit le même label précédé par le préfixe "NEG_".

toutes les annotations de la segmentation entrante vers la segmentation émise. Par défaut, lorsque la case **Advanced settings** n'est pas sélectionnée, l'auto-numérotation est désactivée et la copie des annotations systématiquement effectuée.

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, aucun segment entrant sélectionné, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

### 5.4.7 Widget *Extract XML*

Ce widget prend une segmentation en entrée, recherche dans son contenu des portions correspondant à un type d'élément XML spécifique et crée un nouveau segment pour chaque occurrence de cet élément. Il est à noter que si une occurrence donnée est répartie sur plusieurs segments de la segmentation entrante, elle donnera lieu à la création d'autant de segments dans la segmentation sortante.

Tous les attributs des éléments extraits sont automatiquement convertis en annotations dans la segmentation sortante. Par exemple, l'extraction de l'élément * * dans le fragment suivant:

* Saperlipopette! *

aboutira à la création d'un segment dont le contenu est *Saperlipopette!* et dont la valeur d'annotation pour la clé *type* est *interjection*.

Ce widget constitue ainsi le moyen le plus simple et le plus flexible d'importer dans TEXTABLE v1.4 une segmentation et des annotations arbitraires spécifiées par l'utilisateur pour un texte donné. Mentionnons toutefois la limitation suivante: le widget supprime automatiquement tout segment de longueur nulle dans la segmentation sortante. Cela a notamment pour conséquence qu'il n'est pas possible d'importer des éléments XML vides (qu'ils soient de la forme *<element></element>* ou *<element/>*).

#### *Interface de base*

Dans l'interface de base du widget (voir figure 25 ci-dessous), la section **XML Extraction** permet de spécifier l'élément XML à extraire (**XML element**). Le widget ne permet en effet d'extraire qu'un seul type d'élément à la fois; en revanche, il extrait toutes les occurrences de cet élément, y compris celles qui sont emboîtées dans d'autres occurrences du même type, le cas échéant.

La case à cocher **Remove markup** permet de déclencher la suppression du balisage interne. Le cas échéant, les éventuelles balises apparaissant à l'intérieur des éléments XML extraits seront exclues de la segmentation sortante. Une conséquence importante de l'usage de cette option est que les éléments extraits seront potentiellement décomposés en plusieurs segments correspondant aux portions de leur contenu que séparent les balises internes omises (voir *Interface avancée* ci-dessous pour un exemple de ce mécanisme[15]).

[15] En comparaison avec l'interface avancée, il faut également noter que dans l'interface de base, les options **Prioritize shallow attributes** et **Merge duplicate segments** sont désactivées par défaut.

La section **Options** se limite au choix du label de la segmentation sortante (**Output segmentation label**). Par défaut, les annotations des segments entrants sont copiés dans les segments sortants.

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, aucun segment créé en sortie, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

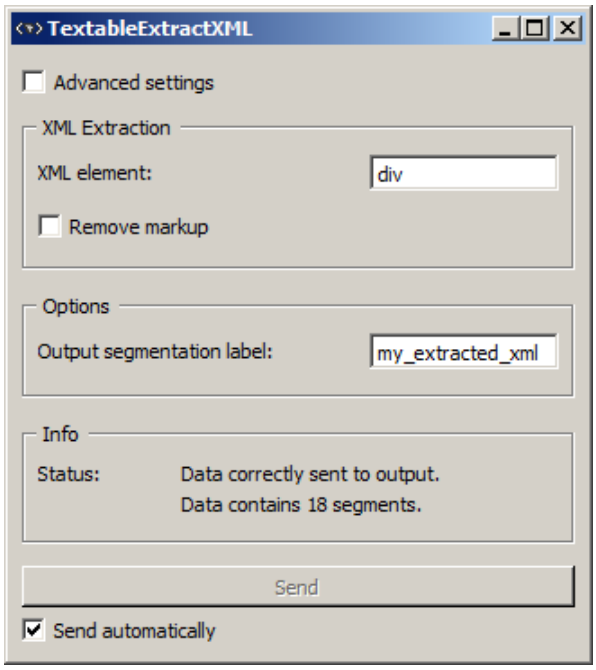



**Figure 25: Widget *Extract XML* (interface de base).**

### *Interface avancée*

La section **XML Extraction** de l'interface du widget (voir figure 26 ci-dessous) permet de configurer l'extraction des éléments XML. Le champ **XML element** permet d'indiquer le type d'élément XML qui doit être recherché. La case à cocher **Import element with key** permet d'assigner à chaque segment produit en sortie une annotation dont la clé est le texte contenu dans le champ immédiatement à droite et dont la valeur est le nom de l'élément XML extrait par le widget.

Si la case à cocher **Remove markup** est sélectionnée, les éventuelles balises apparaissant à l'intérieur des éléments XML extraits seront exclues de la segmentation sortante. Une conséquence importante de l'usage de cette option est que les éléments extraits seront potentiellement décomposés en plusieurs segments correspondant aux portions de leur contenu que séparent les balises internes omises. Par exemple, dans le cas du fragment suivant:

*<text>un <keyword>fragment</keyword> de code XML</text>*

l'extraction de l'élément *<text>* donnera lieu à la création de trois segments:

*un*
*fragment*
*de code XML*

Si en revanche l'option **Remove markup** n'est pas sélectionnée, un segment unique sera créé:

*un <keyword>fragment</keyword> de code XML*

La case à cocher **Prioritize shallow attributes** détermine le comportement du widget dans le cas très particulier où (a) des éléments du type extrait sont (exactement) emboîtés les uns dans les autres, (b) ils possèdent des valeurs différentes pour le même attribut, (c) l'option **Remove markup** est sélectionnée et (d) l'option **Merge duplicate segments** (section **Options**, voir ci-dessous) également. Ce pourrait être le cas de l'extraction de l'élément * * dans le fragment suivant, par exemple:

* deux éléments exactement emboités *

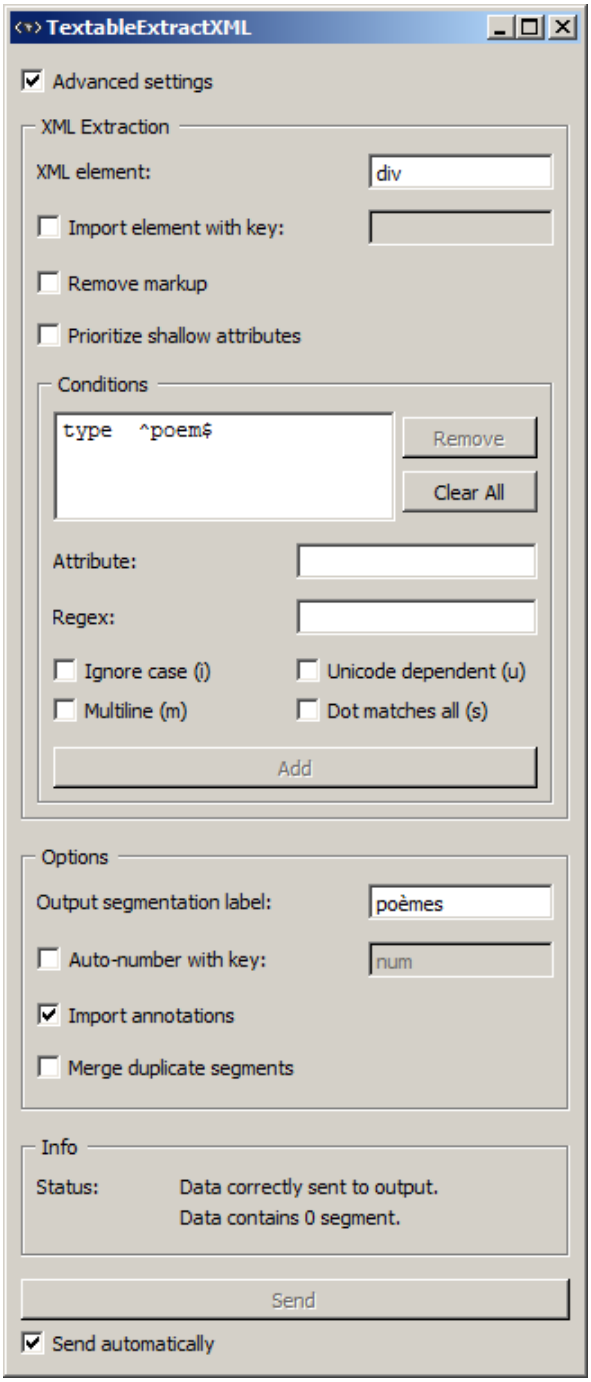



**Figure 26: Widget *Extract XML* (interface avancée).**

Dans une telle situation, le widget créera d'abord deux segments possédant exactement la même adresse (puisque les balises internes sont supprimées avec **Remove markup**), puis sous l'effet de **Merge duplicate segments** (voir ci-dessous), il cherchera à les fusionner en un seul. Il ne pourra alors

conserver qu'une seule des valeurs d'annotation concurrentes *A* et *B* pour la clé d'annotation *type*; par défaut il s'agira de la valeur associée à l'élément le plus proche de la racine dans l'arborescence XML, en l'occurrence la valeur *A*. Si en revanche l'option **Prioritize shallow attributes** est sélectionnée, c'est la valeur de l'élément le plus proche de la "surface" qui sera conservée, soit *B* dans cet exemple.

La sous-section **Conditions** incluse dans la section **XML Extraction** permet de restreindre l'extraction en spécifiant des conditions portant sur les attributs des éléments extraits. Ces conditions sont exprimées sous la forme d'expressions régulières que doivent satisfaire les valeurs d'attributs donnés. Dans la liste qui apparaît en haut de cette sous-section, les colonnes indiquent (a) l'attribut concerné, (b) l'expression régulière correspondante, et (c) les options associées à cette expression.[16]

Dans la figure 26 ci-dessus, on a ainsi restreint l'extraction aux seuls éléments * * possédant un attribut *type* dont la valeur est *poem*. Si plusieurs conditions étaient définies, elles devraient toutes être satisfaites pour qu'un élément donné soit extrait. Les boutons à droite permettent de supprimer la condition sélectionnée (**Remove**) ou de vider entièrement la liste (**Clear All**).

La partie restante de la sous-section **Conditions** permet d'ajouter de nouvelles conditions dans la liste. A cet effet, il faut spécifier l'attribut concerné (**Attribute**) et l'expression régulière correspondante (**Regex**). Les cases à cocher **Ignore case (i)**, **Unicode dependent (u)**, **Multiline (m)** et **Dot matches all (s)** contrôlent l'application des options correspondantes à l'expression régulière. L'ajout de la nouvelle condition à la liste s'effectue finalement en cliquant sur le bouton **Add**.

La section **Options** permet de spécifier le label affecté à la segmentation produite en sortie (**Output segmentation label**). La case à cocher **Auto-number with key** permet de numéroter automatiquement les segments de la segmentation sortante et d'associer le numéro à la clé d'annotation spécifiée dans le champ de texte à droite. La case à cocher **Import annotations** copie dans chaque segment sortant toutes les annotations associées au segment correspondant dans la segmentation entrante. La case **Merge duplicate segments** permet de fusionner en un seul segment des segments distincts mais dont l'adresse est la même; les annotations associées aux segments fusionnés sont toutes copiées dans le segment unique résultant.[17]

La section **Info** indique le nombre de segments dans la segmentation produite en sortie le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée, aucun segment créé en sortie, etc.).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

### 5.4.8 Widget *Display*

Ce widget prend une segmentation en entrée et affiche à l'écran le contenu et les annotations des segments qui la composent. Il permet en outre d'exporter ces informations dans un fichier texte. Par ailleurs, il fait suivre la segmentation sans modification sur ses connexions sortantes.[18]

[16] Voir la documentation de Python (http://docs.python.org/library/re.html).
[17] Voir note 8 ci-dessus.
[18] Il renvoie aussi, sur un second canal non sélectionné par défaut, une segmentation contenant l'entier de la chaîne affichée dans un unique segment.

**Display** joue un rôle essentiel dans la construction de schémas: il constitue le meilleur moyen de vérifier que le paramétrage des widgets de traitement de segmentations présentés ci-dessus aboutit bien au résultat voulu en termes de création ou modification de segments et d'annotations.

Il est à noter que pour de longues segmentations, le widget peut paraître bloqué pendant un certain temps après le déroulement de la barre de progression – un problème lié à la librairie d'interface graphique sur laquelle repose Orange Canvas. A moins d'un dépassement de la capacité de mémoire vive de la machine, le problème se règle normalement de lui-même après quelques instants.

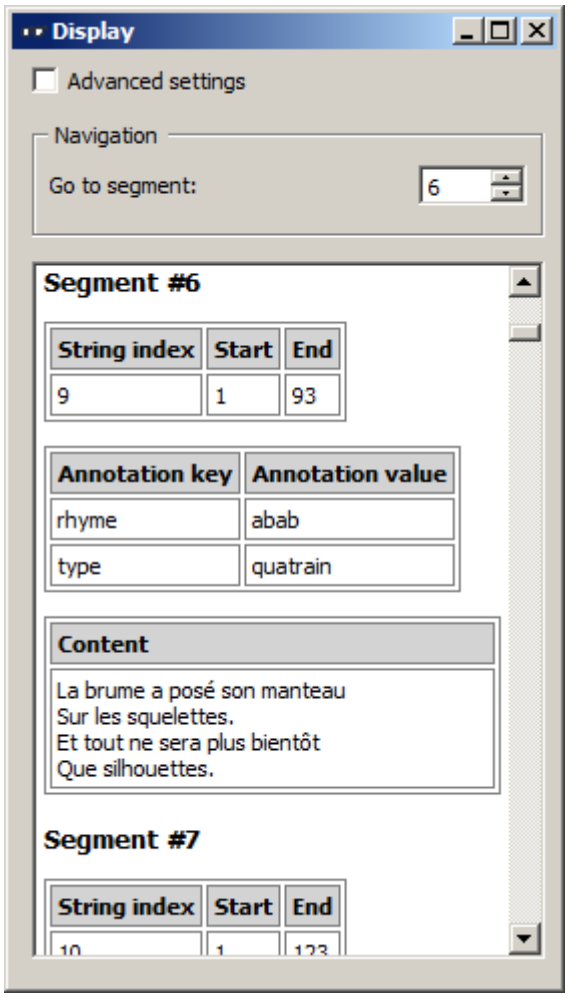



**Figure 27: Widget *Display* (interface de base).**

## *Interface de base*

Dans sa version de base, le widget formate la segmentation entrante en HTML et affiche pour chaque segment son numéro, son adresse complète (numéro de chaîne, positions initiale et finale) ainsi que ses annotations (voir figure 27 ci-dessus). La section **Navigation** permet d'afficher directement un segment particulier au moyen du contrôle **Go to segment**.

On peut relever que l'interface de base de **Display** est plus dépouillée que celle des autres widgets de TEXTABLE v1.4: elle n'inclut ni section **Info**, ni bouton **Send** et case à cocher **Send automatically**. Ce qui motive cette conception est la volonté de mettre l'accent sur la fonctionnalité fondamentale de visualisation du contenu et des annotations de la segmentation entrante – unique raison de l'utilisation de **Display** dans la plupart des cas. Dans ce contexte, l'envoi des données sur les connexions sortantes est activé par défaut.

### *Interface avancée*

L'interface avancée du widget (voir figure 28 ci-dessous) rétablit la section **Info**, qui indique le nombre de segments dans la segmentation reçue et émise le cas échéant, ou les raisons pour lesquelles aucune segmentation n'est émise (par exemple pas de données en entrée). Elle inclut également le bouton **Send** et la case à cocher **Send automatically** traditionnels, qui fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

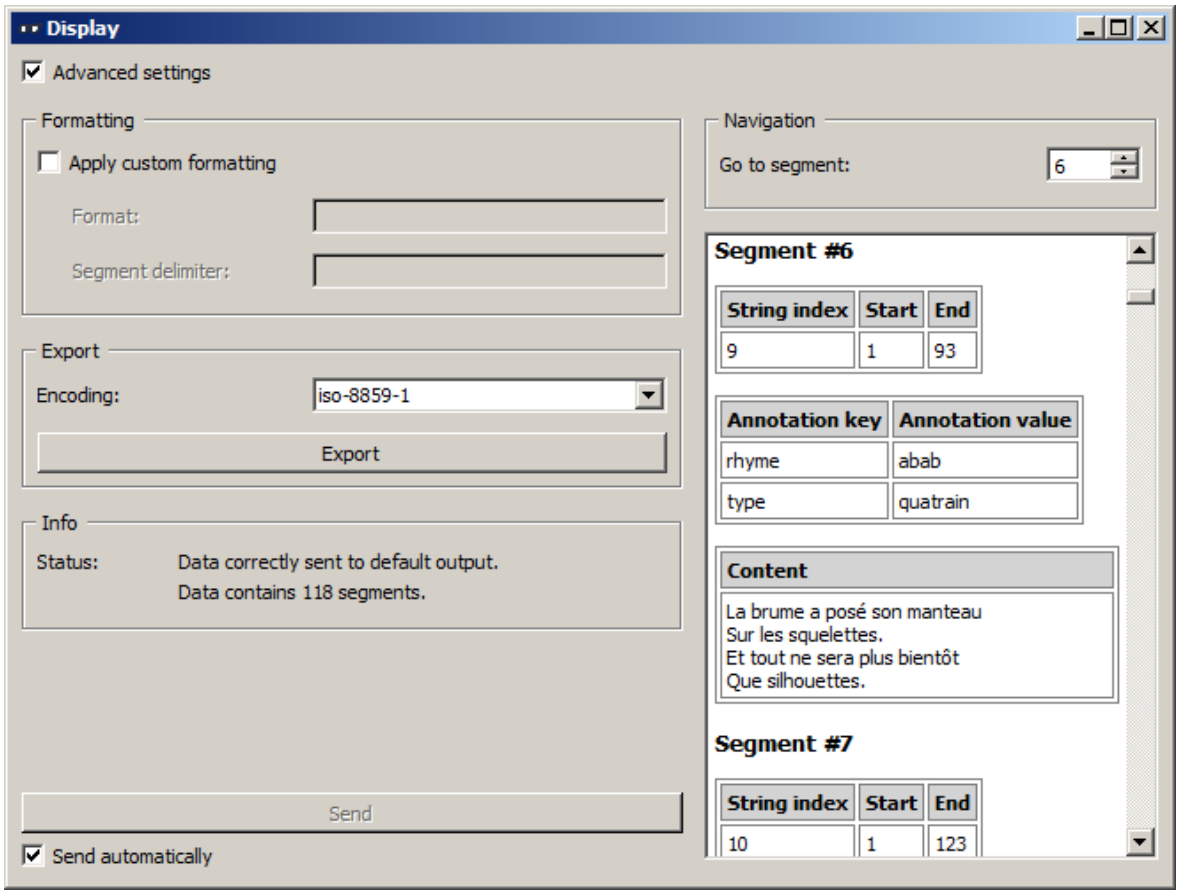



**Figure 28: Widget *Display* (interface avancée).**

La case **Apply custom formatting** permet de produire un rendu personnalisé. Dans ce mode, le formatage de chaque segment est déterminé par la chaîne saisie dans le champ **Format**. Celle-ci peut contenir du texte qui sera reproduit tel quel dans la sortie formatée, ainsi que des références à des variables à insérer dans l'output. Ces références prennent la forme générale suivante:

*%(nom_de_variable)format*

où *nom_de_variable* désigne la variable à insérer et *format* le format désiré pour cette variable. Pour une utilisation de base, il suffit de savoir que le code de format *s* désigne une chaîne de caractères et *i* un nombre entier.[19] Si le nom de variable est l'une des chaînes prédéfinies suivantes, il sera interprété comme indiqué dans la colonne de droite:[20]

[19] Pour plus de détails sur la syntaxe des codes de format, voir la documentation de Python (http://docs.python.org/library/stdtypes.html#string-formatting).

[20] De façon générale, les chaînes prédéfinies dans le cadre de TEXTABLE v1.4 ont en commun de commencer et se terminer par deux caractères *underscore* (_); il est vivement recommandé d'éviter cette forme pour tous les noms fournis par l'utilisateur (en particulier pour les labels de segmentation, ainsi que les clés et valeurs d'annotation).

| | |
|---|---|
| *__content__* | contenu du segment |
| *__num__* | numéro du segment |
| *__str_index__* | numéro de chaîne |
| *__str_index_raw__* | numéro de chaîne indexé à partir de 0 |
| *__start__* | position initiale |
| *__start_raw__* | position initiale indexée à partir de 0 |
| *__end__* | position finale |

Si au contraire le nom de variable n'est pas compris dans cette liste, le programme l'interprétera comme une clé d'annotation et tentera d'afficher la valeur correspondante (ou la chaîne *__none__* si cette clé n'est pas définie pour le segment considéré).

La chaîne saisie dans le champ **Segment delimiter** sera insérée entre chaque segment de la segmentation formatée. Utilisez la séquence *\n* pour un retour à la ligne et *\t* pour une tabulation.

Par exemple, soit la chaîne *un exemple simple*, segmentée en caractères et annotée comme dans l'exemple de la figure 19 ci-dessus. En saisissant la chaîne *%(__num__)i\t%(__content__)s\t%(type)s* sous **Format** et *\n* sous **Segment delimiter**, on obtient le formatage suivant:

*1 u voyelle*
*2 n consonne*
*3 autre*
*4 e voyelle*
*5 x consonne*
*6 e voyelle*
*7 m consonne*
*8 p consonne*
*9 l consonne*
*10 e voyelle*
*11 autre*
*12 s consonne*
*13 i voyelle*
*14 m consonne*
*15 p consonne*
*16 l consonne*
*17 e voyelle*

La section **Export** de l'interface du widget permet également d'exporter la segmentation formatée (version HTML standard ou personnalisée) dans un fichier. Il faut pour cela choisir un encodage (**Encoding**) puis cliquer sur **Export** pour ouvrir un dialogue de sélection de fichier.

Lorsque l'option **Apply custom formatting** n'est pas sélectionnée, la section **Navigation** est activée et permet d'afficher directement un segment particulier au moyen du contrôle **Go to segment**.

## *5.5 Widgets de création de tables*

Les widgets de cette catégorie prennent des segmentations en entrée et produisent des données tabulées en sortie. La conversion s'effectue par le biais d'opérations de comptage (**Count**), de mesures de longueur (**Length**) et de diversité (**Variety**), ou encore par l'exploitation des annotations associées aux segments (**Annotation**); le widget **Context**, enfin, permet de représenter sous forme de table des concordances et des listes de collocations.

### 5.5.1 Widget *Count*

Ce widget prend une ou plusieurs segmentations en entrée, compte la fréquence des segments définis par l'une des segmentations (éventuellement à l'intérieur des segments définis par une autre), et renvoie le résultat du comptage sous forme de *table de contingence* (ou *matrice de cooccurrences*, ou encore *matrice documents–termes*).

Les tables de contingence produites par ce widget sont de type *PivotCrosstab*, un sous-type du format général *Table* (voir sections 5.2 ci-dessus et surtout 5.6 ci-dessous). Dans une telle table, chaque colonne correspond à un type d'*unité*, chaque ligne correspond à un type de *contexte*, et la cellule à l'intersection d'une colonne et d'une ligne données contient l'*effectif* (ou *fréquence absolue*, ou encore *nombre d'occurrences*) de ce type d'unité dans ce type de contexte.

Pour prendre un exemple simple, supposons qu'on dispose de deux segmentations de la chaîne *un exemple simple*: [21]

A) label = *mots*

(
  ( 1, 2, { (*partie du discours*: *article*), (*type de mot*: *grammatical*) } ),
  ( 4, 10, { (*partie du discours*: *nom*), (*type de mot*: *lexical*) } ),
  ( 12, 17, { (*partie du discours*: *adjectif*), (*type de mot*: *lexical*) } )
)

B) label = *lettres* (extrait)

(
  ( 1, 1, { (*type de lettre*: *voyelle*) } ),
  ( 2, 2, { (*type de lettre*: *consonne*) } ),
  ( 4, 4, { (*type de lettre*: *voyelle*) } ),
  ...,
  ( 16, 16, { (*type de lettre*: *consonne*) } ),
  ( 17, 17, { (*type de lettre*: *voyelle*) } )
)

Typiquement, on pourrait définir les types d'unités sur la base du contenu des segments de la segmentation *lettres*, et les types de contextes sur la base du contenu des segments de la segmentation *mots*. Le comptage de ces types d'unités dans ces types de contextes produirait alors la table de contingence suivante:[22]

| *__context__* | *u* | *n* | *e* | *x* | *m* | *p* | *l* | *s* | *i* |
|---|---|---|---|---|---|---|---|---|---|
| *un* | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *exemple* | 0 | 0 | 3 | 1 | 1 | 1 | 1 | 0 | 0 |
| *simple* | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

[21] La première de ces segmentations a été donnée en exemple dans la section 5.1 ci-dessus, et l'on adopte ici les mêmes conventions de notation; en particulier, on n'indique pas le numéro de la chaîne associée à chaque segment mais uniquement ses positions initiale et finale.

[22] L'en-tête de la première colonne, *__context__*, est un nom prédéfini (voir note 19 ci-dessus).

Alternativement, on pourrait préférer compter les *valeurs d'annotation* (plutôt que le contenu) des unités et/ou des contextes. Par exemple, en définissant les unités sur la base des annotations associées à la clé *type de lettre* dans la segmentation *lettres*, et les contextes sur la base des annotations associées à la clé *type de mot* dans la segmentation *mots*, on obtiendrait la table suivante:

| __context__ | voyelle | consonne |
|---|---|---|
| grammatical | 1 | 1 |
| lexical | 5 | 8 |

Le jeu du croisement des segmentations et des clés d'annotation constitue un mécanisme extrêmement flexible qui permet de produire facilement une large variété de tables de contingence. Dans ce contexte, c'est à l'utilisateur que revient la responsabilité de fournir une définition cohérente des unités et contextes. De façon générale, pour que soit comptée une occurrence d'une unité donnée dans un contexte donné, il faut (a) que le segment correspondant à l'unité et celui correspondant au contexte soient associés à la même chaîne, (b) que la position initiale du segment "unité" dans la chaîne soit supérieure ou égale à celle du segment "contexte", et (c) inversement que la position finale de l'unité soit inférieure ou égale à celle du contexte. En bref, l'unité doit être *contenue* dans le contexte.

Un cas limite d'utilisation rendu possible par ce mode de fonctionnement consiste à définir unités et contextes sur la base de la même segmentation. En effet, puisque tout segment est contenu dans lui-même, rien n'empêche d'utiliser par exemple la segmentation *mots* pour définir les unités à partir de la clé *partie du discours* et les contextes à partir de la clé *type de mot*:

| __context__ | article | nom | verbe |
|---|---|---|---|
| grammatical | 1 | 0 | 0 |
| lexical | 0 | 1 | 1 |

Toujours en se limitant à une seule segmentation, TEXTABLE v1.4 propose deux autres façons de définir les contextes. La première repose sur la notion d'une "fenêtre" de *n* segments que l'on fait "coulisser" progressivement du début à la fin de la segmentation. Dans notre exemple, en appliquant ce principe à la segmentation *lettres* et en fixant la taille de la fenêtre à 11 segments, on définit ainsi les contextes suivants:

1. *un exemple si*
2. *n exemple sim*
3. *exemple simp*
4. *xemple simpl*
5. *emple simple*

En définissant par ailleurs les unités sur la base des annotations *type de lettre* par exemple, on obtient alors les décomptes suivants (où les contextes sont représentés par leurs positions successives):

| __context__ | voyelle | consonne |
|---|---|---|
| 1 | 5 | 6 |
| 2 | 4 | 7 |
| 3 | 4 | 7 |
| 4 | 3 | 8 |
| 5 | 4 | 7 |

Le dernier mode de spécification des contextes que propose TEXTABLE v1.4 et n'impliquant qu'une seule segmentation consiste à définir les contextes comme les *n* segments immédiatement à gauche et/ou à droite de chaque segment. Par exemple, toujours à partir de la seule segmentation *lettres* et des annotations *type de lettre*, définir les contextes sur la base des deux segments immédiatement à gauche de chaque segment ainsi que du segment immédiatement à droite aboutit à la table de contingence suivante (où le symbole '+' sépare les segments successifs du contexte et le symbole underscore '_' sépare les parties gauche et droite du contexte):

| __context__ | voyelle | consonne |
|---|---|---|
| voyelle+consonne_consonne | 2 | 3 |
| consonne+voyelle_voyelle | 2 | 0 |
| consonne+voyelle_consonne | 2 | 0 |
| consonne+consonne_voyelle | 2 | 0 |
| consonne+consonne_consonne | 0 | 1 |

Une telle table nous indique notamment que dans un contexte composé, à gauche, d'une séquence *voyelle+consonne* et, à droite, d'une consonne (par exemple *ex_m* ou *em_l*), on a observé 2 fois une voyelle et 3 fois une consonne. Un cas particulier de ce type de table est celui de la *matrice de transition* qui définit une *chaîne de Markov*, où l'on ne considère que le contexte gauche des segments:

| __context__ | voyelle | consonne |
|---|---|---|
| voyelle_ | 0 | 5 |
| consonne_ | 5 | 4 |

Notons encore que la spécification des contextes, contrairement à celle des unités, est optionnelle. En effet, il est toujours possible de compter globalement la fréquence des unités d'une segmentation et produire ainsi une table ne possédant qu'une seule ligne correspondant à l'ensemble de la segmentation concernée (soit *lettres*, dans l'exemple suivant):

| __context__ | u | n | e | x | m | p | l | s | i |
|---|---|---|---|---|---|---|---|---|---|
| lettres | 1 | 1 | 4 | 1 | 2 | 2 | 2 | 1 | 1 |

Enfin, dans tous les cas de figures envisagés ci-dessus, on pourrait aussi s'intéresser à la fréquence des *séquences* de 2, 3, ..., *n* segments (ou *n–grammes*) plutôt qu'à celle des segments isolés:

| __context__ | un | ne | ex | xe | em | mp | pl | le | es | si | im |
|---|---|---|---|---|---|---|---|---|---|---|---|
| lettres | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 |

Après avoir ainsi brossé le panorama des types de tables de contingence que peut produire le widget **Count**, nous pouvons nous pencher sur son interface (voir figures 29 à 32 ci-dessous). Celle-ci contient deux sections séparées pour la définition des unités (**Units**) et des contextes (**Contexts**).

Dans la section **Units**, le menu déroulant **Segmentation** permet de sélectionner parmi les segmentations entrantes celle dont les segments feront l'objet du comptage. Le menu **Annotation key** affiche les éventuelles clés d'annotation associées à la segmentation choisie; si l'une de ces clés est sélectionnée, ce sont les valeurs d'annotation correspondantes qui seront comptées; si en revanche la valeur **(none)** est sélectionnée, ce sera le *contenu* des segments. Le menu déroulant **Sequence length** permet d'indiquer s'il faut compter les segments isolés ou les *n–grammes* de segments; dans ce dernier cas, la chaîne (optionnelle) spécifiée dans le champ de texte **Intra-**

**sequence delimiter** sera utilisée pour séparer le contenu ou la valeur d'annotation correspondant à chaque segment dans les en-têtes de colonne.[23]

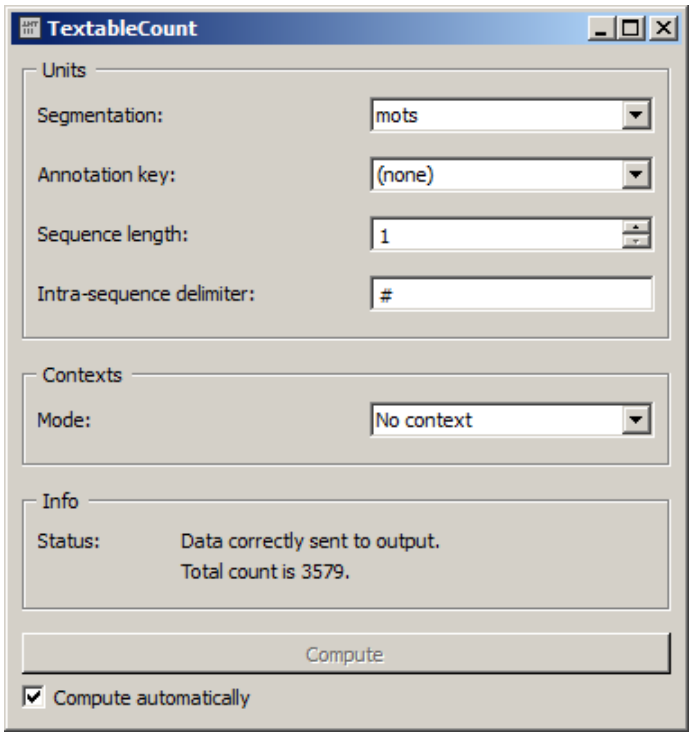



**Figure 29: Widget *Count* (mode *No context*).**

La section **Contexts** se décline en plusieurs variantes, selon la valeur sélectionnée dans le menu déroulant **Mode**. Celui-ci permet de choisir entre les façons de définir les contextes décrites plus haut. Le mode **No context** (voir figure 29 ci-dessus) correspond au cas où le décompte des unités est effectué de façon globale dans l'ensemble de la segmentation spécifiée dans la section **Units** (à laquelle nous référerons ci-dessous par le terme de *segmentation des unités*).

Le mode **Sliding window** (voir figure 30 ci-dessous) implémente la notion de "fenêtre coulissante" introduite plus haut. Il permet typiquement d'observer l'évolution des fréquences au fil de la segmentation des unités. Le seul paramètre est la taille de la fenêtre (en nombre de segments), fixée au moyen du curseur **Window size**.

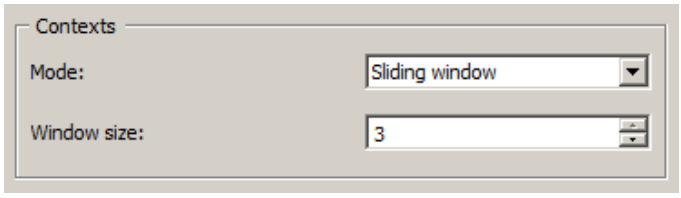



**Figure 30: Widget *Count* (mode *Sliding window*).**

[23] La même chaîne de caractères sera insérée, le cas échéant, entre les segments successifs constituant le contexte gauche et/ou droit si le mode **Left–right neighborhood** est sélectionné.

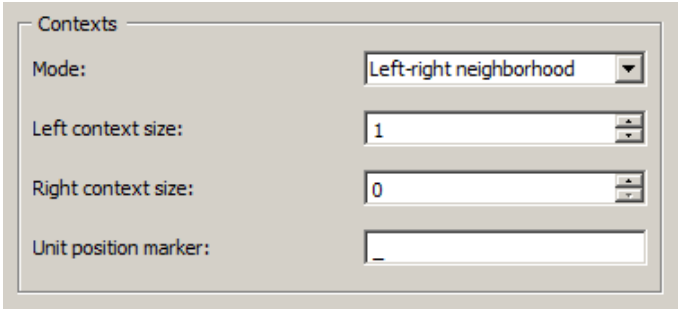



**Figure 31: Widget *Count* (mode *Left–Right neighborhood*).**

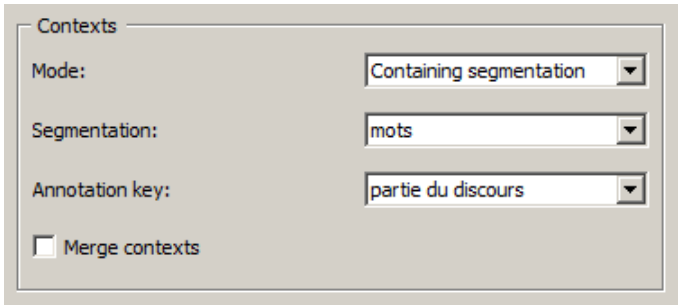



**Figure 32: Widget *Count* (mode *Containing segmentation*).**

Le mode **Left–right neighborhood** (voir figure 31 ci-dessus) permet de définir les types de contextes comme les *n* segments immédiatement à gauche et/ou à droite de chaque segment; c'est ce mode qui permet notamment de construire la matrice de transition d'une chaîne de Markov. Les paramètres **Left context size** et **Right context size** déterminent le nombre de segments pris en considération dans chaque partie du contexte. Le champ de texte **Unit position marker** permet de spécifier la chaîne de caractères (potentiellement vide) à insérer entre les partie gauche et droite du contexte dans les en-têtes de lignes.

Enfin, le mode **Containing segmentation** (voir figure 32 ci-dessus) correspond au cas où les contextes sont définis par les types de segments apparaissant dans une segmentation qui peut être celle des unités ou une autre. Cette segmentation, que nous appellerons par analogie *segmentation des contextes*, est sélectionnée parmi les segmentations entrantes au moyen du menu déroulant **Segmentation**. Le menu **Annotation key** affiche les éventuelles clés d'annotation associées à la segmentation des contextes; si l'une de ces clés est sélectionnée, ce sont les types de valeurs d'annotation correspondantes qui constitueront les en-têtes des lignes; si en revanche la valeur **(none)** est sélectionnée, c'est le *contenu* des segments qui sera utilisé. La case à cocher **Merge contexts** permet de compter les unités de façon globale dans l'ensemble de la segmentation des contextes.

La section **Info** indique la somme des fréquences dans la table produite en sortie le cas échéant, ou la raison pour laquelle aucune table n'est émise (pas de données en entrée ou fréquence totale nulle).

Le bouton **Compute** et la case à cocher **Compute automatically** fonctionnent de la même façon que **Send** et **Send automatically** dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

### 5.5.2 Widget *Length*

Ce widget prend une ou plusieurs segmentations en entrée, mesure la longueur d'une des segmentations (éventuellement à l'intérieur des segments définis par une autre), et renvoie le résultat sous forme de table. Il permet également de calculer la longueur moyenne des segments d'une segmentation sur la base des unités définies par une autre segmentation.

Les tables produites par le widget **Length** possèdent au moins 2 colonnes, et au plus 4. La première colonne contient les en-têtes correspondant aux contextes – qui sont définis essentiellement de la même façon qu'avec le widget **Count** (voir section 5.5.1 ci-dessus). La deuxième colonne fournit les indications de longueur (auquel cas son en-tête est *__length__*) ou de longueur moyenne (en-tête *__length_average__*). Dans ce dernier cas, la troisième colonne peut contenir les écarts-types correspondants si leur affichage est requis par l'utilisateur (en-tête *__length_std_deviation__*), et la dernière colonne indiquera le nombre d'éléments sur lequel le calcul de moyenne est effectué (en-tête *__length_count__*).

Reprenons l'exemple des segmentations *mots* et *lettres* de la chaîne *un exemple simple* (voir section 5.5.1 ci-dessus). On considère ici essentiellement deux configurations de base. La première est celle où l'on s'intéresse simplement à la longueur d'une segmentation donnée, par exemple *lettres*:

| *__context__* | *__length__* |
|---|---|
| *lettres* | 15 |

Pour fixer la terminologie, nous désignerons par le terme d'*unités de mesure* les segments dont le décompte est interprété comme une mesure de longueur, soit en l'occurrence les segments de la segmentation *lettres*.

La seconde configuration de base est celle où l'on souhaite connaître la longueur *moyenne* des segments d'une segmentation, par exemple *mots*, en termes d'unités de mesure appartenant à une autre segmentation (*lettres* en l'occurrence):

| *__context__* | *__length_average__* | *__length_std_deviation__* | *__length_count__* |
|---|---|---|---|
| *mots* | 5 | 2.16024684906 | 3 |

Dans ce cas, nous appellerons *unités de moyennisation* les segments dont la longueur est mesurée et moyennisée. Notons que le calcul de la longueur moyenne présuppose qu'au moins une unité de mesure est *contenue* dans une unité de moyennisation, au sens défini dans la discussion sur les relations entre unités et contextes dans la description du widget **Count** (voir section 5.5.1 ci-dessus).

Ces deux configurations élémentaires (mesure de longueur et calcul de longueur moyenne) peuvent ensuite être combinées avec deux modes de spécification des *contextes* – donc des lignes de la table. Le premier mode consiste à définir les contextes sur la base des contenus ou annotations d'une segmentation donnée; par exemple, voici la longueur des segments de *mots* (contextes) en terme de ceux de *lettres* (unités de mesure):

| *__context__* | *__length__* |
|---|---|
| *un* | 2 |
| *exemple* | 7 |
| *simple* | 6 |

Il est à noter que, de façon analogue à ce qui se passe avec le widget **Count**, ce sont les *types* de segments qui constituent les en-têtes de lignes, comme l'illustre l'exemple suivant, où les mêmes segmentations sont utilisées mais les contextes sont définis par les valeurs d'annotation associées à la clé *type de mot*:

| __context__ | __length__ |
|---|---|
| grammatical | 2 |
| lexical | 13 |

Le calcul de la longueur moyenne est également applicable lorsque les contextes sont définis sur la base d'une segmentation. Dans ce cas, on utilisera généralement trois segmentations différentes pour définir les unités de mesure, les unités de moyennisation, et les contextes; par exemple, il pourrait s'agir de calculer la longueur moyenne des mots (en nombre de lettres) dans des textes différents. Pour rester dans le cadre de notre exemple basé sur deux segmentations seulement, on peut exploiter le fait que tout segment est contenu dans lui-même et calculer la longueur moyenne des mots (en nombre de lettres) en fonction des annotations *type de mot* (autrement dit on utilise ici une seule et même segmentation pour déterminer les contextes et les unités de moyennisation):

| __context__ | __length_average__ | __length_std_deviation__ | __length_count__ |
|---|---|---|---|
| grammatical | 2.0 | 0.0 | 1 |
| lexical | 6.5 | 0.5 | 2 |

Le second mode de spécification des contextes repose sur le concept (déjà introduit dans la description du widget **Count**, voir section 5.1.2 ci-dessus) d'une "fenêtre" de *n* segments que l'on fait "coulisser" progressivement du début à la fin de la segmentation. Par exemple, en fixant la taille de la fenêtre à 2 segments, on peut examiner la longueur moyenne des mots (en nombre de lettres) dans les bigrammes successifs de la segmentation *mots* (identifiés par leur position):

| __context__ | __length_average__ | __length_std_deviation__ | __length_count__ |
|---|---|---|---|
| 1 | 4.5 | 2.5 | 2 |
| 2 | 6.5 | 0.5 | 2 |

Par construction, chaque cellule de la colonne *__length_count__* contiendra alors la même valeur, soit la taille de la fenêtre. Sur la base de cette observation, il est assez facile de se convaincre que ce dernier mode de spécification des contextes n'a de sens que lorsqu'on s'intéresse à l'évolution d'une longueur *moyenne* au fil d'une segmentation.

Nous passons maintenant à la présentation de l'interface du widget (voir figure 33 ci-dessous). Celle-ci contient trois sections séparées pour la spécification des unités de mesure (**Units**), des unités de moyennisation (**Averaging**), et des contextes (**Contexts**).

La section **Units** ne contient qu'un seul menu déroulant (**Segmentation**) servant à sélectionner parmi les segmentations entrantes celle dont les segments fourniront les unités de mesure.

Dans la section **Averaging**, la case à cocher **Average over segmentation** déclenche le calcul de la longueur moyenne. Le cas échéant, le menu déroulant immédiatement à droite permet de sélectionner la segmentation dont les segments constitueront les unités de moyennisation. La case à cocher **Compute standard deviation** permet de calculer, outre la longueur moyenne, son écart-type. Il est à noter que pour de grandes segmentations, cette option est susceptible de rallonger spectaculairement le temps de calcul.

Comme pour le widget **Count** (voir section 5.5.1 ci-dessus), la section **Contexts** se décline en plusieurs variantes selon la valeur sélectionnée dans le menu déroulant **Mode**. Celui-ci permet de choisir entre les modes de spécification des contextes décrits plus haut. Le mode **No context** correspond au cas où la mesure de longueur ou le calcul de longueur moyenne sont appliqués globalement à l'ensemble de la segmentation définissant les unités de mesure (spécifiée dans la section **Units**).

Le mode **Sliding window** (identique à celui du widget **Count**, voir figure 30 ci-dessus) implémente la notion de "fenêtre coulissante" introduite plus haut. Il permet d'observer l'évolution des longueurs moyennes au fil de la segmentation des unités de moyennisation. Le seul paramètre est la taille de la fenêtre (en nombre de segments), fixée au moyen du curseur **Window size**.

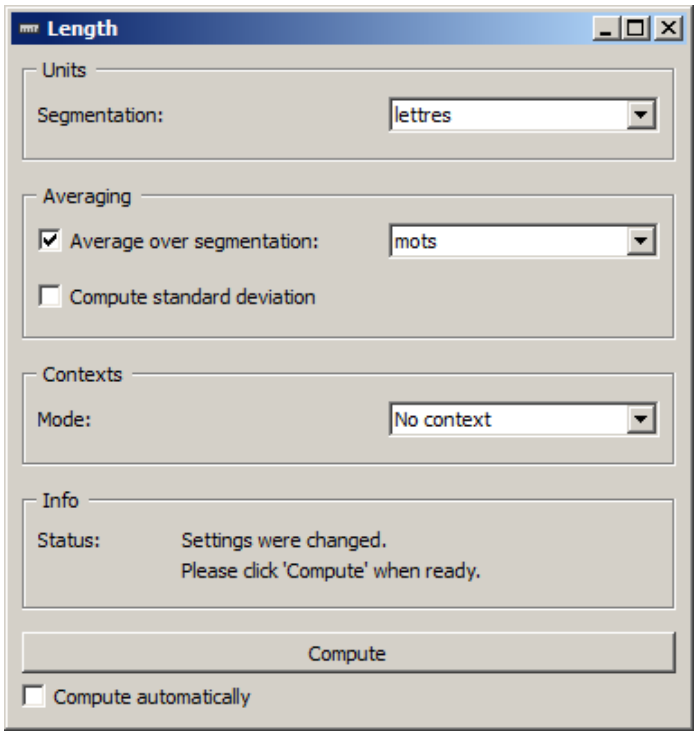



**Figure 33: Widget *Length* (mode *No context*).**

Enfin, le mode **Containing segmentation** (identique à celui du widget **Count**, voir figure 32 ci-dessus) correspond au cas où les contextes sont définis par les types de segments apparaissant dans une segmentation (qui sera le plus souvent distincte des segmentations fournissant les unités de mesure et les unités de moyennisation). Cette segmentation est sélectionnée parmi les segmentations entrantes au moyen du menu déroulant **Segmentation**. Le menu **Annotation key** affiche les éventuelles clés d'annotation associées à la segmentation sélectionnée; si l'une de ces clés est sélectionnée, ce sont les types de valeurs d'annotation correspondantes qui constitueront les en-têtes des lignes; si en revanche la valeur **(none)** est sélectionnée, c'est le *contenu* des segments qui sera utilisé. La case à cocher **Merge contexts** permet de mesurer la longueur ou calculer la longueur moyenne de façon globale dans l'ensemble de la segmentation définissant les contextes.

La section **Info** indique si une table a pu être correctement produite en sortie le cas échéant, ou la raison pour laquelle aucune table n'est émise (pas de données en entrée).

Le bouton **Compute** et la case à cocher **Compute automatically** fonctionnent de la même façon que dans le widget **Count** (voir section 5.5.1 ci-dessus).

### 5.5.3 Widget *Variety*

Ce widget prend une ou plusieurs segmentations en entrée, mesure la variété des segments d'une des segmentations (éventuellement à l'intérieur des segments définis par une autre), et renvoie le résultat sous forme de table; il permet également de calculer la variété moyenne par catégorie (sur la base des valeurs d'annotation des segments). Pour rendre ces deux mesures moins dépendantes de la longueur de la segmentation, il est possible de calculer leur valeur moyenne sur des sous-échantillons de taille fixée.

Les tables produites par le widget **Variety** possèdent au moins 2 colonnes, et au plus 4. La première colonne contient les en-têtes correspondant aux contextes – qui sont définis essentiellement de la même façon qu'avec le widget **Count** (voir section 5.5.1 ci-dessus). La deuxième colonne fournit les mesure de variété et son en-tête est *__variety__*, à moins qu'un rééchantillonnage ait été appliqué (auquel cas l'en-tête sera *__variety_average__*). Dans ce dernier cas, la troisième colonne contiendra les écarts-types correspondants (en-tête *__variety_std_deviation__*) et la dernière colonne le nombre de sous-échantillons (en-tête *__variety_count__*).

Reprenons l'exemple des segmentations *mots* et *lettres* de la chaîne *un exemple simple* (voir section 5.5.1 ci-dessus). La mesure la plus élémentaire effectuée par le widget est celle du nombre de types ou *variété*. Par exemple. pour la segmentation *lettres*, en définissant les unités sur la base du contenu des segments:

| *__context__* | *__variety__* |
|---|---|
| *lettres* | 9 |

Naturellement, il est possible de définir les types plutôt sur la base des valeurs associées à une clé d'annotation, par exemple *type de lettre*:

| *__context__* | *__variety__* |
|---|---|
| *type de lettre* | 2 |

Une première variation sur ce thème consiste à *pondérer* la variété en fonction de la fréquence des types. A cet effet, on peut calculer la *perplexité* de la distribution des segments, c'est-à-dire l'exponentielle de l'entropie sur cette distribution. Cette mesure est égale à la variété si et seulement si les types de segments ont une fréquence uniforme; elle décroît et tend vers 0 à mesure que la distribution des segments s'écarte de l'uniformité et s'approche du déterminisme. A titre d'exemple, voici la perplexité obtenue pour *type de lettres*:

| *__context__* | *__variety__* |
|---|---|
| *type de lettre* | 1.96013170421 |

L'écart observé entre variété pondérée (1.96) ou non (2) traduit l'écart à l'uniformité de la distribution des types de lettres dans cet exemple.

Plutôt que de s'intéresser à la variété (pondérée ou non) des types de segments *en général*, on peut se pencher sur leur variété moyenne au sein d'une *catégorie*. Par exemple, on peut se demander quelle est la variété moyenne des lettres *en fonction du type de lettre*:

| __context__ | __variety__ |
|---|---|
| lettres | 4.5 |

En moyenne, dans notre exemple, un type de lettre (*consonne* ou *voyelle*) est donc représenté par 4.5 lettres distinctes – pour autant qu'on attribue le même poids à chaque catégorie. L'alternative consiste à pondérer les catégories en fonction de leur fréquence, ce qui aboutirait dans notre cas à donner plus de poids dans le calcul de moyenne à la variété des consonnes (dont la fréquence est 9) qu'à celle des voyelles (dont la fréquence est 6):

| __context__ | __variety__ |
|---|---|
| lettres | 4.8 |

De l'augmentation observée par rapport au cas où les catégories ne sont pas pondérées, on peut déduire que le nombre de consonnes distinctes est plus élevé que celui de voyelles.

En résumé, la pondération (ou non) par la fréquence des unités fonde la distinction entre variété et perplexité; par ailleurs, dans le cas où l'on calcule la variété/perplexité moyenne par catégorie, il est possible d'effectuer (ou non) une pondération par la fréquence des catégories.

Les diverses mesures de variété présentées ci-dessus peuvent ensuite être combinées avec les mêmes modes de spécification des *contextes* – donc des lignes de la table – que dans le widget **Length** (voir section 5.5.2 ci-dessus): le premier mode consiste à définir les contextes sur la base des contenus ou annotations d'une segmentation donnée; le second repose sur le concept d'une "fenêtre" de *n* segments que l'on fait "coulisser" progressivement du début à la fin de la segmentation.

Toute les mesures de variété (pondérées ou non, simples ou par catégorie) sont sensibles à la taille de l'échantillon, c'est-à-dire dans notre cas la longueur de la segmentation. A ce titre, elles ne sont en principe pas directement comparables entre des segmentations de longueurs différentes. Considérons par exemple la variété (non pondérée) des *lettres* (unités) dans les *mots* (contextes):

| __context__ | __variety__ |
|---|---|
| un | 2 |
| exemple | 5 |
| simple | 6 |

Etant donné que le mot *un* ne contient que 2 lettres, la variété de 2 observée pour ce mot est relativement élevée en comparaison de la variété de 5 observée pour le mot de 7 lettres *exemple*.

Pour réduire l'effet de cette dépendance à la longueur de la segmentation, il est possible d'adopter la stratégie suivante: tirer un nombre fixé de sous-échantillons dans chaque segmentation à comparer et rapporter la variété moyenne par sous-échantillon. Par exemple, en fixant la taille des sous-échantillons à 2 segments, et en tirant 100 sous-échantillons pour chaque mot, on peut obtenir les valeurs suivantes (qui dépendent du hasard de l'échantillonnage):[24]

[24] L'exemple a une vocation pédagogique; en pratique, on utilisera typiquement une taille de sous-échantillon nettement plus élevée, par exemple 50 segments ou plus.

| __context__ | __variety_average__ | __variety_std_deviation__ | __variety_count__ |
|---|---|---|---|
| *un* | 2.0 | 0.0 | 100 |
| *exemple* | 1.88 | 0.324961536185 | 100 |
| *simple* | 2.0 | 0.0 | 100 |

Cette nouvelle mesure montre bien que la variété des lettres dans le mot *un* est proportionnellement plus élevée que celle de *exemple* (et comparable avec celle de *simple*).

Nous passons maintenant à la présentation de l'interface du widget (voir figure 34 ci-dessous). Celle-ci contient quatre sections séparées pour la spécification des unités (**Units**), des catégories (**Categories**), des contextes (**Contexts**), et des paramètres de rééchantillonnage (**Resampling**).

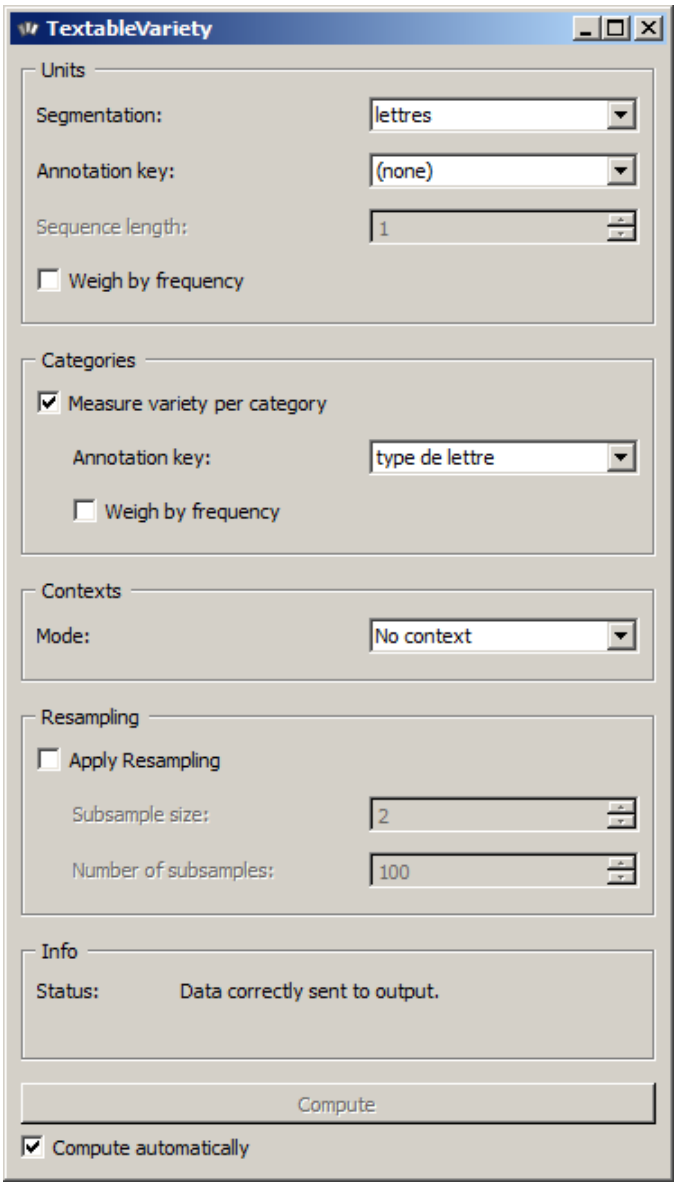



**Figure 34: Widget *Variety*.**

Dans la section **Units**, le menu déroulant **Segmentation** permet de sélectionner parmi les segmentations entrantes celle dont les segments serviront de base au calcul de la variété. Le menu **Annotation key** affiche les éventuelles clés d'annotation associées à la segmentation choisie; si l'une de ces clés est sélectionnée, ce sont les valeurs d'annotation correspondantes qui seront utilisées; si

en revanche la valeur **(none)** est sélectionnée, ce sera le *contenu* des segments. Le menu déroulant **Sequence length** permet d'indiquer s'il faut considérer les segments isolés ou les *n–grammes* de segments. Enfin, la case à cocher **Weigh by frequency** permet d'activer la pondération des unités par leur fréquence (donc la mesure de la perplexité plutôt que de la variété).

Dans la section **Categories**, la case à cocher **Measure diversity per category** déclenche le calcul de la diversité moyenne par catégorie. Le cas échéant, le menu déroulant **Annotation key** permet de sélectionner la clé d'annotation dont les valeurs seront utilisées pour la définition des catégories. La case à cocher **Weigh by frequency** permet d'activer la pondération par la fréquence des catégories.

Comme pour le widget **Count** (voir section 5.5.1 ci-dessus), la section **Contexts** se décline en plusieurs variantes selon la valeur sélectionnée dans le menu déroulant **Mode**. Celui-ci permet de choisir entre les modes de spécification des contextes décrits plus haut. Le mode **No context** correspond au cas où la mesure de variété est appliquée globalement à l'ensemble de la segmentation définissant les unités de mesure (spécifiée dans la section **Units**).

Le mode **Sliding window** (identique à celui du widget **Count**, voir figure 30 ci-dessus) implémente la notion de "fenêtre coulissante" introduite plus haut. Il permet d'observer l'évolution de la variété au fil de la segmentation. Le seul paramètre est la taille de la fenêtre (en nombre de segments), fixée au moyen du curseur **Window size**.

Enfin, le mode **Containing segmentation** (identique à celui du widget **Count**, voir figure 32 ci-dessus) correspond au cas où les contextes sont définis par les types de segments apparaissant dans une segmentation (qui sera le plus souvent distincte des segmentations fournissant les unités de mesure et les unités de moyennisation). Cette segmentation est sélectionnée parmi les segmentations entrantes au moyen du menu déroulant **Segmentation**. Le menu **Annotation key** affiche les éventuelles clés d'annotation associées à la segmentation sélectionnée; si l'une de ces clés est sélectionnée, ce sont les types de valeurs d'annotation correspondantes qui constitueront les en-têtes des lignes; si en revanche la valeur **(none)** est sélectionnée, c'est le *contenu* des segments qui sera utilisé. La case à cocher **Merge contexts** permet de mesurer la variété de façon globale dans l'ensemble de la segmentation définissant les contextes.

Dans la section **Resampling**, la case **Apply resampling** permet d'activer le calcul de la diversité moyenne par sous-échantillon de taille donnée. Le nombre de segments par sous-échantillon est déterminé par le curseur **Subsample size**, et le nombre de sous-échantillons avec **Number of subsamples**.

La section **Info** indique si une table a pu être correctement produite en sortie le cas échéant, ou la raison pour laquelle aucune table n'est émise (pas de données en entrée, typiquement).

Le bouton **Compute** et la case à cocher **Compute automatically** fonctionnent de la même façon que dans le widget **Count** (voir section 5.5.1 ci-dessus).

### 5.5.4 Widget *Annotation*

Ce widget prend une ou plusieurs segmentations en entrée et produit une représentation tabulée des valeurs d'annotation associées aux segments de l'une d'entre elles (pour une clé d'annotation donnée).

Les tables produites par ce widget ne contiennent que deux colonnes. La première (en-tête *__context__*) contient les en-têtes correspondant aux contextes – qui sont définis essentiellement de la même façon qu'avec le widget **Count**, mode **Containing segmentation** (voir section 5.5.1 ci-dessus): par les types de segments apparaissant dans une segmentation. La seconde colonne (en-tête *__annotation__*) contient la ou les annotations associées à chaque type de segment.

De telles tables sont typiquement destinées à être combinées (au moyen du widget **Merge Data**, onglet **Data**) avec des tables produites par les widgets **Count**, **Length** ou **Variety**, dans le but de créer un classifieur automatique. Danc ce contexte, la chaîne figurant dans la colonne *__annotation__* sera généralement interprétée comme la *classe* associée à chaque contexte – qu'il s'agira pour le classifieur d'apprendre à prédire sur la base des autres variables.

Reprenons l'exemple des segmentations *mots* et *lettres* de la chaîne *un exemple simple* (voir section 5.5.1 ci-dessus). On peut produire la table suivante, donnant la valeur d'annotation associée à la clé *type de lettre* pour chaque lettre distincte:

| *__context__* | *__annotation__* |
|---|---|
| *u* | *voyelle* |
| *n* | *consonne* |
| *e* | *voyelle* |
| *x* | *consonne* |
| *m* | *consonne* |
| *p* | *consonne* |
| *l* | *consonne* |
| *s* | *consonne* |
| *i* | *voyelle* |

Dans cette illustration, chaque lettre n'est associée qu'à une seule valeur d'annotation. Dans le cas le plus général, les contextes peuvent être associés à plusieurs valeurs; par exemple, si les contextes sont définis comme les *types de mots* de la segmentation *mots* et les annotations comme les contenus des segments de la segmentation *lettres*: [25]

| *__context__* | *__annotation__* |
|---|---|
| *grammatical* | *u,n* |
| *lexical* | *e,m,l,p,i,s,x* |

Dans ce cas, il s'agira pour l'utilisateur de choisir (a) l'ordre (fréquentiel ou ASCII-bétique) dans lequel seront triées les valeurs multiples et (b) si elles doivent être toutes affichées ou uniquement la première (dans l'ordre sélectionné).

L'interface du widget (voir figure 35 ci-dessous) contient trois sections séparées pour la spécification des unités (**Units**), du traitement des valeurs multiples (**Multiple Values**) et des contextes (**Contexts**). Dans la section **Units**, le menu déroulant **Segmentation** permet de sélectionner parmi les segmentations entrantes celle dont les segments seront examinés pour déterminer les valeurs d'annotation. Le menu **Annotation key** affiche les éventuelles clés d'annotation associées à la segmentation choisie; si l'une de ces clés est sélectionnée, ce sont les valeurs d'annotation correspondantes qui seront utilisées; si en revanche la valeur **(none)** est sélectionnée, ce sera le *contenu* des segments. Le menu déroulant **Sequence length** permet d'indiquer s'il faut considérer les segments isolés ou les *n–grammes* de segments. dans ce dernier cas, la chaîne (optionnelle) spécifiée

[25] Comme le montre cet exemple, les contenus des segments peuvent également être utilisés comme "valeurs d'annotation".

dans le champ de texte **Intra-sequence delimiter** sera utilisée pour séparer le contenu ou la valeur d'annotation correspondant à chaque segment individuel.

Dans la section **Multiple Values**, le menu déroulant **Sort by** permet de sélectionner le critère de tri des valeurs multiples (le cas échéant), soit par fréquence (**Frequency**) ou par ordre ASCII (**ASCII**). La case à cocher **Sort in reverse order** inverse l'ordre de tri, et la case **Keep only first value** permet de ne conserver que la première valeur (dans l'ordre sélectionné). Le champ **Value delimiter** sert à indiquer la chaîne de caractère à insérer entre les valeurs multiples.

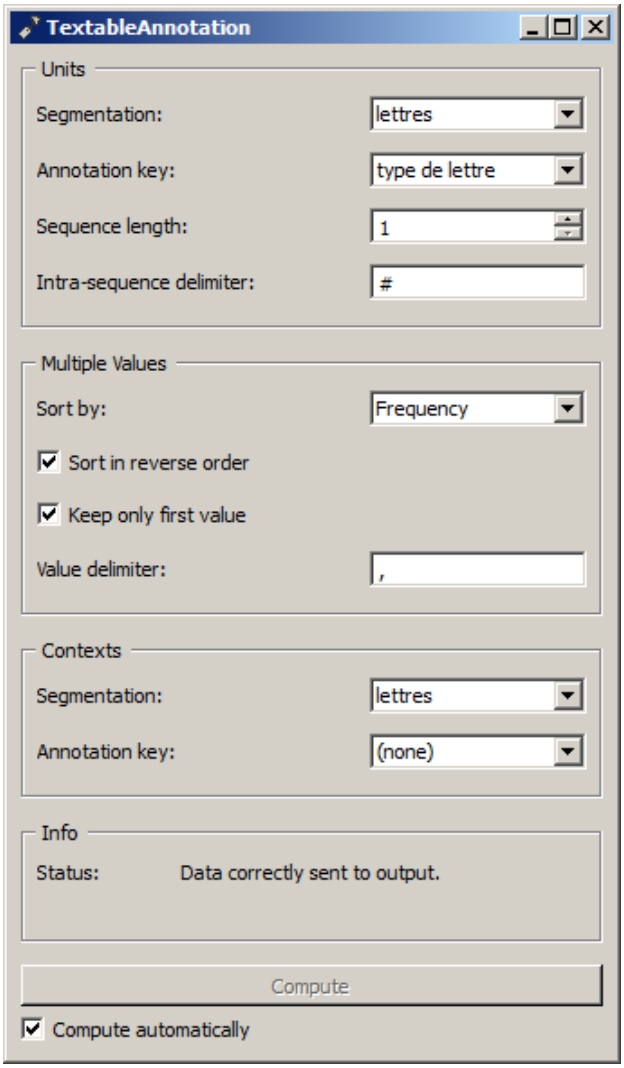



**Figure 35: Widget *Annotation*.**

Contrairement aux autres widgets de cette catégorie, la spécification des contextes ne peut se faire ici que par rapport à une segmentation contenant celle des unités (soit l'équivalent du mode **Containing segmentation** des widgets **Count**, **Length** et **Variety**). Cette segmentation est sélectionnée parmi les segmentations entrantes au moyen du menu déroulant **Segmentation**. Le menu **Annotation key** affiche les éventuelles clés d'annotation associées à la segmentation sélectionnée; si l'une de ces clés est sélectionnée, ce sont les types de valeurs d'annotation correspondantes qui constitueront les en-têtes des lignes; si en revanche la valeur **(none)** est sélectionnée, c'est le *contenu* des segments qui sera utilisé.

La section **Info** indique si une table a pu être correctement produite en sortie le cas échéant, ou la raison pour laquelle aucune table n'est émise (pas de données en entrée, typiquement).

Le bouton **Compute** et la case à cocher **Compute automatically** fonctionnent de la même façon que dans le widget **Count** (voir section 5.5.1 ci-dessus).

### 5.5.5 Widget *Context*

Ce widget prend une ou plusieurs segmentations en entrée et produit, sous forme de table, des concordances ou des listes de collocations permettant d'examiner les contextes dans lesquels apparaissent des segments donnés.

Le fonctionnement de ce widget (apparu dans la version 1.2 de TEXTABLE) repose sur les notions d'unités et de contextes, comme tous les widgets de sa catégorie. Le rôle de la segmentation des unités est central: c'est elle qui définit les segments jouant le rôle de *pivots* dans la construction des concordances (ou des listes des collocations), c'est-à-dire les segments dont le contexte est analysé.

Le cas le plus simple est celui où une seule segmentation est considérée; la seule façon de définir les contextes est alors en termes d'un nombre donné de segments voisins. Par exemple, étant donné la seule segmentation en *lettres* de la chaîne *un exemple simple* (voir section 5.5.1 ci-dessus), on peut construire des concordances comme la suivante:

| __id__ | __pos__ | 1L | __pivot__ | 1R |
|---|---|---|---|---|
| 1 | 1 | | u | n |
| 2 | 2 | u | n | e |
| 3 | 3 | n | e | x |
| 4 | 4 | e | x | e |
| 5 | 5 | x | e | m |
| 6 | 6 | e | m | p |
| 7 | 7 | m | p | l |
| 8 | 8 | p | l | e |
| 9 | 9 | l | e | s |
| 10 | 10 | e | s | i |
| 11 | 11 | s | i | m |
| 12 | 12 | i | m | p |
| 13 | 13 | m | p | l |
| 14 | 14 | p | l | e |
| 15 | 15 | l | e | |

Dans cette table, la colonne *__id__* donne le numéro de chaque segment "pivot" (sa position dans la table). La colonne *__pos__* indique la position de chaque segment pivot dans la segmentation des unités, et dans le cas d'espèce cette information duplique la précédente (nous verrons ci-dessous que ce n'est pas toujours le cas). Le segment pivot lui-même apparaît dans la colonne *__pivot__*, et ses voisins immédiats à gauche et à droite apparaissent respectivement dans les colonnes *1L* et *1R*.

Le nombre de voisins affichés à gauche et à droite peut naturellement être plus élevé, de même qu'on peut afficher des valeurs d'annotation plutôt que le contenu des segments (qu'il s'agisse des segments pivots ou de leurs voisins). Par exemple, la table suivante donne les 3 voisins immédiats de chaque lettre en affichant leur valeur d'annotation pour la clé *type de lettre*:

| __id__ | __pos__ | 3L | 2L | 1L | __pivot__ | 1R | 2R | 3R |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | | u | consonne | voyelle | consonne |
| 2 | 2 | | | voyelle | n | voyelle | consonne | voyelle |
| 3 | 3 | | voyelle | consonne | e | consonne | voyelle | consonne |
| 4 | 4 | voyelle | consonne | voyelle | x | voyelle | consonne | consonne |
| 5 | 5 | consonne | voyelle | consonne | e | consonne | consonne | consonne |
| 6 | 6 | voyelle | consonne | voyelle | m | consonne | consonne | voyelle |
| 7 | 7 | consonne | voyelle | consonne | p | consonne | voyelle | consonne |
| 8 | 8 | voyelle | consonne | consonne | l | voyelle | consonne | voyelle |
| 9 | 9 | consonne | consonne | consonne | e | consonne | voyelle | consonne |
| 10 | 10 | consonne | consonne | voyelle | s | voyelle | consonne | consonne |
| 11 | 11 | consonne | voyelle | consonne | i | consonne | consonne | consonne |
| 12 | 12 | voyelle | consonne | voyelle | m | consonne | consonne | voyelle |
| 13 | 13 | consonne | voyelle | consonne | p | consonne | voyelle | |
| 14 | 14 | voyelle | consonne | consonne | l | voyelle | | |
| 15 | 15 | consonne | consonne | consonne | e | | | |

La particularité de telles tables est qu'elles donnent le contexte de *tous* les segments de l'unique segmentation considérée. En général, on s'intéresse plutôt à certains segments spécifiques, ce qu'il est possible de préciser au moyen d'une segmentation distincte. Supposons ainsi qu'on dispose, en plus de la segmentation *lettres*, d'une segmentation dont le label serait *pivots* et ne contenant que les occurrences de la lettre *e* (toujours dans la chaîne *un exemple simple*)[26]:

```
(
   ( 4,   4,      { (type de lettre: voyelle)} ),
   ( 6,   6,      { (type de lettre: voyelle)} ),
   ( 10,  10,     { (type de lettre: voyelle)} ),
   ( 17,  17,     { (type de lettre: voyelle)} )
)
```

En spécifiant les unités au moyens de la segmentation *pivots* et les contextes (soit en l'occurrence les segments voisins) au moyen de la segmentation *lettres*, on peut alors produire une table comme la suivante:

| __id__ | __pos__ | 3L | 2L | 1L | __pivot__ | 1R | 2R | 3R |
|---|---|---|---|---|---|---|---|---|
| 1 | 3 | | voyelle | consonne | e | consonne | voyelle | consonne |
| 2 | 5 | consonne | voyelle | consonne | e | consonne | consonne | consonne |
| 3 | 9 | consonne | consonne | consonne | e | consonne | voyelle | consonne |
| 4 | 15 | consonne | consonne | consonne | e | | | |

Cet exemple de concordance plus typique montre bien, par ailleurs, que la position du segment pivot dans la table (colonne __*id*__) n'est pas nécessairement égale à sa position dans la segmentation définissant les contextes (colonne __*pos*__).

Dans les exemples précédents, le contexte de chaque segment pivot est défini en termes des segments *voisins* dans une segmentation. Une autre possibilité est de définir le contexte sur la base d'une autre segmentation dont les segments *contiennent* les segments pivots. Pour illustrer ce second mode de caractérisation des contextes, considérons le cas où les unités sont spécifiées par la segmentation *pivots*, comme précédemment, et les contextes par la segmentation *mots* (voir section 5.5.1):

[26] C'est typiquement au moyen du widget **Select** (voir section 5.4.5) qu'on pourrait produire une telle segmentation.

| __id__ | __pos__ | __left__ | __pivot__ | __right__ |
|---|---|---|---|---|
| 1 | 2 | | e | xemple |
| 2 | 2 | ex | e | mple |
| 3 | 2 | exempl | e | |
| 4 | 3 | simpl | e | |

Cet exemple montre les implications de ce changement de mode de spécification du contexte. En premier lieu, la table résultante a maintenant une largeur fixe[27] de 5 colonnes: *__id__* et *__pivot__* ont la même fonction que précédemment; *__pos__* indique la position du segment contexte correspondant à chaque segment pivot (ce qui permet de retrouver facilement le contexte en question au moyen du widget **Display**, cf. section 5.4.8); enfin les colonnes *__left__* et *__right__* donnent respectivement la partie gauche et droite de chaque segment contexte contenant un segment pivot.

Notons par ailleurs que dans ce cas, remplacer le contenu du segment contexte par une de ses valeurs d'annotation n'aurait guère de sens. En revanche, il peut être utile d'indiquer une telle valeur dans une colonne séparée, comme *partie du discours* dans l'exemple suivant, qui illustre également la possibilité de remplacer le contenu du pivot par une valeur d'annotation (en l'occurrence *type de lettre*):

| __id__ | __pos__ | __left__ | __pivot__ | __right__ | partie du discours |
|---|---|---|---|---|---|
| 1 | 2 | | voyelle | xemple | nom |
| 2 | 2 | ex | voyelle | mple | nom |
| 3 | 2 | exempl | voyelle | | nom |
| 4 | 3 | simpl | voyelle | | adjectif |

Ces exemples mettent en évidence la versatilité du widget **Context**, dont les possibilités sont plus diversifiées que celles que peut offrir un concordancier de base – au prix d'une mise en œuvre plus complexe puisqu'elle implique généralement de pouvoir construire et mettre en relation deux segmentations distinctes du texte analysé.

Nous concluons ce tour d'horizon des capacités du widget par la construction de listes de collocations. Notons d'abord que cette fonctionnalité est conçue ici comme une option de visualisation applicable à une concordance du premier type évoqué ci-dessus, soit où le contexte est défini en termes de segments *voisins* plutôt que "contenant". Plutôt que de représenter les segments voisins de chaque occurrence du pivot, on peut en effet donner la liste de ces (types de) segments assortis d'une indication de l'attraction ou, au contraire, la répulsion entre chacun d'eux et le pivot.

Reprenons par exemple la concordance présentée plus haut où les unités sont données par la segmentation *pivots* et les contextes par les valeurs d'annotation *type de lettre* de la segmentation *lettres*:

| __id__ | __pos__ | 3L | 2L | 1L | __pivot__ | 1R | 2R | 3R |
|---|---|---|---|---|---|---|---|---|
| 1 | 3 | | voyelle | consonne | e | consonne | voyelle | consonne |
| 2 | 5 | consonne | voyelle | consonne | e | consonne | consonne | consonne |
| 3 | 9 | consonne | consonne | consonne | e | consonne | voyelle | consonne |
| 4 | 15 | consonne | consonne | consonne | e | | | |

Sous forme de liste de collocations, les mêmes données permettent de produire la représentation suivante:

[27] Sauf dans le cas "pathologique" ou aucun segment pivot n'est contenu dans un segment contexte.

| __unit__ | __mutual_info__ | __local_freq__ | __local_prob__ | __global_freq__ | __global_prob__ |
|---|---|---|---|---|---|
| consonne | 0.206450877467 | 9 | 0.692307692308 | 9 | 0.6 |
| voyelle | -0.378511623254 | 4 | 0.307692307692 | 6 | 0.4 |

La colonne *__mutual_info__* donne l'information mutuelle (en bits) entre les pivots (soit, pour mémoire, les occurrences de la lettre *e*) et chaque valeur d'annotation *type de lettre* apparue dans le voisinage (soit ici à une distance maximale de 3 segments) des pivots. Cette quantité est le logarithme binaire du rapport entre la probabilité du *type de lettre* en question dans le voisinage des pivots et sa probabilité dans la segmentation des contextes en général.

Ainsi, le type *consonne* est apparu 9 fois dans le voisinage de *e* (*__local_freq__*), sur un total de 13 segments apparus dans ce voisinage, d'où une probabilité "locale" de 9/13 = 0.69 (*__local_prob__*); par ailleurs le même type est apparu 9 fois dans l'ensemble de la segmentation *lettres* (*__global_freq__*), sur un total de 15 segments, d'où une probabilité "globale" de 9/15 = 0.6 (*__global_prob__*). Enfin, le logarithme binaire de 0.69/0.6 = 1.15 vaut 0.21 bits (*__mutual_info__*), et cette valeur (légèrement) positive traduit la (faible) attraction entre *e* et le type *consonne* à une distance maximale de 3 segments. A l'inverse, l'information mutuelle négative entre *e* et *voyelle* montre que ces catégories sont plutôt en relation de répulsion dans le voisinage considéré.

L'interface du widget (voir figure 36 ci-dessous) contient deux sections séparées pour la spécification des unités (**Units**) et des contextes (**Contexts**). Dans la section **Units**, le menu déroulant **Segmentation** permet de sélectionner parmi les segmentations entrantes celle dont les segments joueront le rôle de pivots. Le menu **Annotation key** affiche les éventuelles clés d'annotation associées à la segmentation choisie; si l'une de ces clés est sélectionnée, ce sont les valeurs d'annotation correspondantes qui seront utilisées; si en revanche la valeur **(none)** est sélectionnée, ce sera le *contenu* des segments. La case **Separate annotation**, activée seulement lorsqu'une clé d'annotation est sélectionnée, permet d'indiquer que les valeurs associées à cette clé doivent apparaître dans une colonne séparée (dont l'en-tête est la clé correspondante) plutôt que de remplacer le contenu des segments dans la colonne *__pivot__*. Notons que les deux contrôles (**Annotation key** et **Separate annotation**) sont désactivés lorsque la case **Use collocation format** est cochée, voir ci-dessous.

Dans la section **Context**, le menu **Mode** permet de choisir entre les deux modes de caractérisation des contextes mentionnés plus haut: en termes de segments *voisins* des segments pivots (**Neighboring segments**) ou alors de segments les *contenant* (**Containing segmentation**). Dans les deux cas, la segmentation en question est sélectionnée parmi les segmentations entrantes au moyen du menu déroulant **Segmentation** et le menu **Annotation key** affiche les éventuelles clés d'annotation associées à cette segmentation. Si l'une de ces clés est sélectionnée, l'affichage des valeurs correspondantes diffère en fonction du **Mode** utilisé: s'il s'agit de **Neighboring segments**, les valeurs d'annotation remplacent le contenu des segments dans les colonnes *1R*, *1L*, ... ; dans le cas contraire, elles apparaissent dans une colonne séparée dont l'en-tête est la clé d'annotation correspondante.

En mode **Neighboring segments**, la section **Contexts** permet encore d'indiquer si une limite doit être posée au nombre de segments voisins affichés pour chaque segment pivot et, le cas échéant, laquelle (**Max. distance**). La case **Use collocation format** sert à formater le résultat comme une liste de collocations (plutôt qu'une concordance); lorsqu'elle est cochée, le menu déroulant **Min. frequency** permet de spécifier la fréquence (globale) minimale que doit atteindre un type de segment pour apparaître dans la liste.

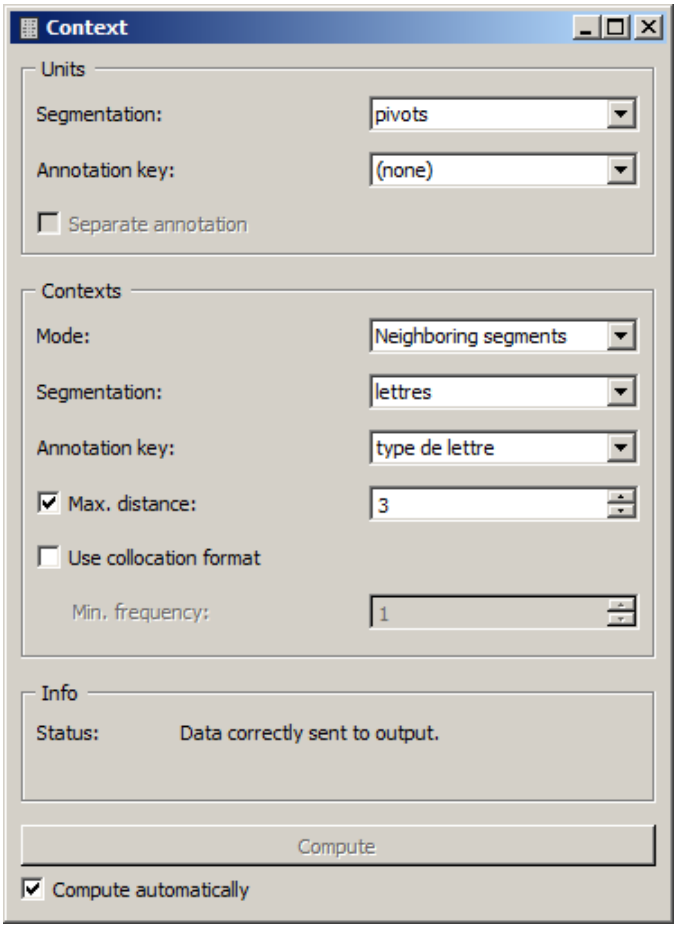



**Figure 36: Widget *Context*.**

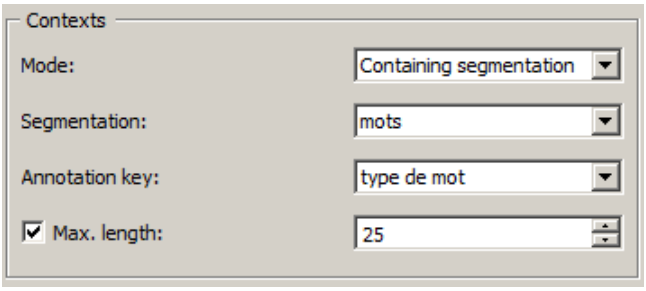



**Figure 37: Widget *Context* (mode *Containing segmentation*).**

En mode **Containing segmentation** (voir figure 37 ci-dessus), la section **Contexts** permet de spécifier le nombre de caractères maximal devant apparaître dans le contexte gauche et droit du pivot.

La section **Info** indique si une table a pu être correctement produite en sortie le cas échéant, ou la raison pour laquelle aucune table n'est émise (parce qu'elle est vide, typiquement).

Le bouton **Compute** et la case à cocher **Compute automatically** fonctionnent de la même façon que dans le widget **Count** (voir section 5.5.1 ci-dessus).

## *5.6 Widget de conversion/exportation de tables*

L'unique widget de cette catégorie, **Convert**, prend en entrée une table dans le format interne de TEXTABLE v1.4 (type *Table*) et permet de la modifier (tri, normalisation, etc.), la convertir vers le type *ExampleTable* propre à Orange Canvas, ou l'exporter vers un fichier.

Comme indiqué en section 5.2 ci-dessus, le format de représentation de table de Orange Canvas (*ExampleTable*) présente des problèmes de compatibilité avec les données encodées en Unicode. Pour permettre l'usage de cet encodage courant dans le contexte spécifique de l'analyse de données textuelles, TEXTABLE v1.4 utilise son propre format de représentation de table, nommé (fort à propos) *Table*.

Les widgets présentés dans la section précédente (**Count**, **Length**, **Variety** et **Annotation**) produisent donc des données de type *Table*. Pour pouvoir être manipulées au moyen des nombreux widgets de traitement de données tabulées offerts par Orange Canvas, ces données doivent être converties du format *Table* vers le format *ExampleTable* propre à Orange Canvas et, le cas échéant, vers un encodage supporté par ce dernier format.

Notons encore que le type *Table* auquel appartiennent toutes les tables produites par les widgets de TEXTABLE v1.4 se subdivise en plusieurs sous-types. En particulier, les tables de contingence (voir section 5.5.1 ci-dessus) appartiennent au sous-type *Crosstab*, qui lui-même se subdivise en *PivotCrosstab*, *FlatCrosstab* et *WeightedFlatCrosstab*. Ces trois formats sont équivalents du point de vue de l'information qu'ils permettent de stocker, et la façon la plus simple de comprendre ce qui les distingue est d'en voir un exemple.

Considérons la table de contingence suivante, du type *PivotCrosstab* (telle que produite par le widget **Count**, voir section 5.5.1 ci-dessus):

| *__context__* | *unité1* | *unité2* |
|---|---|---|
| *contexte1* | 1 | 4 |
| *contexte2* | 2 | 1 |

Voici la même information convertie en format *FlatCrosstab*:

| *__id__* | *__unit__* | *__context__* |
|---|---|---|
| *1* | *unité1* | *contexte1* |
| *2* | *unité2* | *contexte1* |
| *3* | *unité2* | *contexte1* |
| *4* | *unité2* | *contexte1* |
| *5* | *unité2* | *contexte1* |
| *6* | *unité1* | *contexte2* |
| *7* | *unité1* | *contexte2* |
| *8* | *unité2* | *contexte2* |

Cette représentation contient toujours trois colonnes portant les en-têtes *__id__*, *__unit__* et *__context__*, et un nombre de lignes égal à l'effectif total de la table de contingence. C'est la façon standard d'encoder une table de contingence dans Orange Canvas, et elle est requise par des widgets tels que *Correspondence Analysis* notamment (après conversion en *ExampleTable*, toutefois).

Le format *WeightedFlatCrosstab* produit une représentation plus compacte en ne conservant qu'une copie de chaque paire unité–contexte distincte et en ajoutant une colonne *__count__* pour stocker l'information du nombre de répétitions de chaque paire:

| *__id__* | *__unit__* | *__context__* | *__count__* |
|---|---|---|---|
| *1* | *unité1* | *contexte1* | *1* |
| *2* | *unité2* | *contexte1* | *4* |
| *3* | *unité1* | *contexte2* | *3* |
| *4* | *unité2* | *contexte2* | *2* |

Ce format est parfois utilisé pour représenter des tables de contingence dans des programmes de traitement de données tiers (par exemple SPSS).

Après ces considérations générales sur les types de tables propres à TEXTABLE v1.4, nous passons à la présentation de l'interface du widget **Convert** (voir figure 38 ci-dessous).

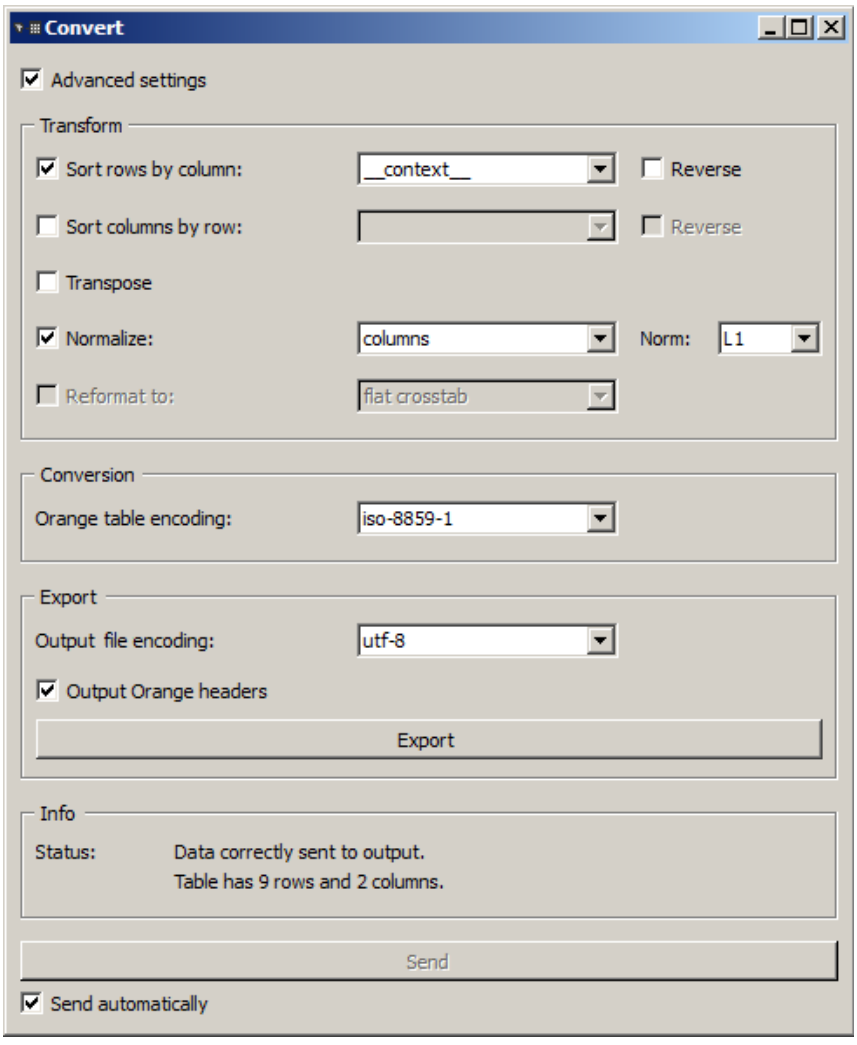



**Figure 38: Widget *Convert*.**

La section **Transform** de l'interface (accessible uniquement lorsque la case **Advanced settings** est sélectionnée) permet d'appliquer des opérations de tri, normalisation et reformatage aux données entrantes.

La case à cocher **Sort rows by column** déclenche le tri des lignes. Si elle est sélectionnée, les en-têtes de colonnes de la table apparaissent dans le menu déroulant immédiatement à droite et l'utilisateur peut ainsi sélectionner la colonne sur la base de laquelle les lignes seront triées. Si la case **Reverse** à droite du menu déroulant est cochée, le tri s'effectuera dans l'ordre inverse (soit décroissant).

La ligne suivante (**Sort columns by row**) contrôle de façon identique le tri des colonnes. Il est à noter que dans ce cas, la première colonne (celle des en-têtes de lignes) restera toujours à la même position, le tri ne s'appliquant qu'aux colonnes suivantes. Pour trier les colonnes sur la base de la ligne d'en-tête, il faut généralement choisir la première option dans le menu déroulant à droite de **Sort columns by row**. Celle-ci contiendra typiquement un nom prédéfini par TEXTABLE v1.4 mais qui ne s'affiche en principe pas sur la table (*__unit__* s'il s'agit d'une table de contingence de type *PivotCrosstab* telle que produite par le widget **Count**, voir section 5.5.1 c-dessus, et l'en-tête générique *__col__* dans tous les autres cas).

La case à cocher **Transpose** permet de transposer la table, c'est-à-dire d'inverser ses lignes et ses colonnes. Cette option n'est disponible que pour les tables de contingence de type *PivotCrosstab.*

La case à cocher **Normalize** déclenche la normalisation de la table; elle n'est applicable que pour les tables de contingence du type *PivotCrosstab*. Si elle est sélectionnée, l'utilisateur peut choisir dans le menu déroulant immédiatement à droite si la normalisation doit s'effectuer par ligne (**lines**), par colonne (**columns**) ou encore sur l'ensemble de la table (**table**); le menu déroulant **Norm** permet de sélectionner le type de normalisation, soit **L1** (division par la somme de la ligne/colonne/table) ou **L2** (division par la racine de la somme des carrés de la ligne/colonne/table).

Deux autres modes de normalisation peuvent être sélectionné dans le menu déroulant: **quotients** et **presence/absence** (dont la sélection désactive l'option **L1/L2**). En mode **quotients**, l'effectif présent dans chaque case d'une table de contingence (de type *PivotCrosstab*) est divisé par l'effectif "théorique" correspondant sous l'hypothèse d'indépendance entre lignes et colonnes de la table. Ce quotient est supérieur à 1 si la ligne et la colonne en question sont en relation d'attraction mutuelle, inférieur à 1 en cas de répulsion de la ligne et la colonne, enfin égal à 1 si ligne et colonne ne se repoussent ni ne s'attirent particulièrement. En mode **presence/absence**, les effectifs supérieurs à 1 sont remplacés par la valeur 1, si bien que la table résultante ne peut contenir que les valeurs 0 et 1.

Enfin, la case à cocher **Reformat to** permet d'effectuer des conversions entre les trois formats de tables de contingence présentés dans la section 5.6 ci-dessus. Elle n'est évidemment applicable que pour les tables de contingence (type *Crosstab*). Il est à noter que cette option et l'option **Normalize** sont mutuellement exclusives, en ce sens que la sélection de l'une désactive l'autre.

Le cas échéant, les divers traitements définis dans cette section sont appliqués aux données entrantes dans l'ordre où ils apparaissent dans l'interface, de haut en bas. Les données modifiées peuvent ensuite être émises sur les connexions sortantes ou exportées dans un fichier.

Dans la section **Conversion** de l'interface, le menu déroulant **Orange table encoding** permet de spécifier vers quel encodage les données émises sur les connexions sortantes doivent être converties (rappelons que les widgets de traitement de tables d'Orange Canvas présentent des difficultés pour le traitement de l'encodage Unicode). Si certains caractères ne peuvent être convertis vers le format spécifié (par exemple des caractères accentués dans le format ASCII), ils sont automatiquement remplacés par les entités HTML correspondantes (par exemple *é* pour *é*).

La dernière section de l'interface, **Export**, permet de sauver le contenu de la table dans un fichier texte. A cet effet, le menu déroulant **Output file encoding** sert à indiquer vers quel encodage les données doivent être converties avant d'être sauvegardées. Cet encodage est entièrement distinct de celui sélectionné sous **Orange table encoding** (voir ci-dessus) et n'affecte pas les données émises sur les connexions sortantes mais uniquement celles sauvées dans un fichier. Typiquement, à moins

d'une limitation imposée par le traitement ultérieur prévu pour les données sauvegardées (par exemple dans le cadre d'un autre logiciel d'analyse de données), on cherchera à conserver ici le maximum d'information en spécifiant soit l'encodage original des données, soit un encodage plus général (par exemple Unicode). La case à cocher **Output Orange headers** permet d'indiquer si la sortie doit inclure toutes les lignes d'en-tête du format *.tab* propre à Orange Canvas (**Output Orange headers**).[28] Enfin le bouton **Export** ouvre un dialogue permettant de choisir le fichier dans lequel les données seront exportées, et d'effectuer l'exportation le cas échéant.

La section **Info** indique le nombre de lignes et colonnes dans la table produite en sortie le cas échéant, ou les raisons pour lesquelles aucune table n'est émise (pas de données en entrée).

Le bouton **Send** et la case à cocher **Send automatically** fonctionnent de la même façon que dans le widget **Preprocess** (voir section 5.4.1 ci-dessus).

[28] Voir à ce sujet http://orange.biolab.si/doc//reference/tabdelimited.htm.

# Annexe A – formats d'im-/exportation JSON

## *Généralités*

L'interface avancée des widgets **Text Files**, **URLs**, **Recode** et **Segment** permet d'importer ou d'exporter en format JSON[29] des listes de sources (voir sections 5.3.2 et 5.3.3), de substitutions (voir section 5.4.2) et d'expressions régulières (voir section 5.4.4). En effet, au-delà d'un nombre restreint d'entrées, il devient plus laborieux de créer ces listes au moyen de l'interface des widgets qu'en éditant directement un fichier texte dans un format spécifié.

Le format général de ces fichiers est le suivant:

```
[
    {
        "cle_1": valeur_1,
        "cle_2": valeur_2,
        ...
        "cle_N": valeur_N,
    },
    ...
    {
        "cle_1": valeur_1,
        "cle_2": valeur_2,
        ...
        "cle_N": valeur_N,
    }
]
```

NB:

- le fichier doit être encodé en utf-8
- l'entier du fichier est compris entre crochets carrés `[ ]`
- chaque entrée de la liste est comprise entre accolades `{ }` et séparée de la suivante par une virgule
- chaque entrée contient une liste de paires clé–valeur séparées par des virgules, dans un ordre indifférent
- clé et valeur sont séparées par deux-points
- la clé est toujours une chaîne entre guillemets doubles
- la valeur peut être une chaîne entre guillemets doubles, ou l'un des mots-clés booléens `true` et `false`
- à l'intérieur des chaînes, la barre oblique /, la barre oblique inverse \ et les guillemets doubles " doivent être précédées ("échappées") par une barre oblique inverse; retour à la ligne et tabulation s'obtiennent avec *n* et *t* respectivement; la notation *uCCCC* (ou chaque *C* représente un chiffre) pour les caractères Unicode est admise.
- certaines clés possèdent une valeur par défaut et sont donc optionnelles; les autres sont obligatoires.

[29] Voir http://www.json.org/.

### *Liste de fichiers (widget* Text Files*)*

Les clés (et valeurs associées) pour les listes de fichiers sont les suivantes:

| *Clé* | *Type* | *Défaut* | *Valeur* | *Remarque* |
|---|---|---|---|---|
| ***path*** | *chaîne* | *–* | *chemin (absolu ou relatif) du fichier* | *attention à échapper les barres obliques normales et inverses* |
| ***encoding*** | *chaîne* | *–* | *encodage du fichier* | *cf. http://docs.python.org/2/library/codecs.html#standard-encodings* |
| ***annotation_key*** | *chaîne* | *""* | *clé d'annotation* | |
| ***annotation_value*** | *chaîne* | *""* | *valeur d'annotation* | |

Exemple:

```
[
    {
        "path":             "data\\Balzac\\Eugenie_Grandet.txt"
        "encoding":         "iso-8859-1",
        "annotation_key":   "auteur",
        "annotation_value": "Balzac",
    },
    {
        "path":             "data\\Balzac\\Le_Pere_Goriot.txt"
        "encoding":         "iso-8859-1",
        "annotation_key":   "auteur",
        "annotation_value": "Balzac",
    },
    {
        "path":             "data\\Daudet\\Lettres_de_mon_moulin.txt"
        "encoding":         "iso-8859-15",
        "annotation_key":   "auteur",
        "annotation_value": "Daudet",
    },
    {
        "path":             "data\\Daudet\\Tartarin_de_Tarascon.txt"
        "encoding":         "iso-8859-15",
        "annotation_key":   "auteur",
        "annotation_value": "Daudet",
    }
]
```

### *Liste d'URLs (widget* URLs*)*

Les clés (et valeurs associées) pour les listes d'URLs sont les suivantes:

| *Clé* | *Type* | *Défaut* | *Valeur* | *Remarque* |
|---|---|---|---|---|
| ***url*** | *chaîne* | *–* | *url (absolue) du document* | *veiller à inclure l'indication `http://` et à échapper les barres obliques* |
| ***encoding*** | *chaîne* | *–* | *encodage du document* | *cf. http://docs.python.org/2/library/codecs.html#standard-encodings* |
| ***annotation_key*** | *chaîne* | *""* | *clé d'annotation* | |
| ***annotation_value*** | *chaîne* | *""* | *valeur d'annotation* | |

Exemple:

```
[
    {
        "url":              "http://www.imsdb.com/scripts/Alien.html"
        "encoding":         "iso-8859-1",
        "annotation_key":   "genre",
        "annotation_value": "sci-fi",
    },
    {
        "url":              "http://www.imsdb.com/scripts/Pulp-Fiction.html"
        "encoding":         "iso-8859-1",
        "annotation_key":   "genre",
        "annotation_value": "crime",
    }
]
```

### *Liste de substitutions (widget* Recode*)*

Les clés (et valeurs associées) pour les listes de fichiers sont les suivantes:

| *Clé* | *Type* | *Défaut* | *Valeur* | *Remarque* |
|---|---|---|---|---|
| ***regex*** | *chaîne* | *–* | *expression régulière* | *attention à échapper les barres obliques normales et inverses* |
| ***unicode_dependent*** | *booléen* | *false* | *option -u* | *cf. http://docs.python.org/library/re.html* |
| ***ignore_case*** | *booléen* | *false* | *option -i* | *cf. http://docs.python.org/library/re.html* |
| ***multiline*** | *booléen* | *false* | *option -m* | *cf. http://docs.python.org/library/re.html* |
| ***dot_all*** | *booléen* | *false* | *option -s* | *cf. http://docs.python.org/library/re.html* |
| ***replacement_string*** | *chaîne* | *–* | *chaîne de remplace-ment* | |

Exemple:

```
[
    {
        "regex":              "<.+?>",
        "replacement_string": ""
    },
    {
        "ignore_case":        true,
        "regex":              "\u0153",
        "replacement_string": "oe"
    },
    {
        "regex":              "(\\w+) (peut|veut)-il",
        "replacement_string": "est-ce que &1 &2",
        "unicode_dependent":  true
    }
]
```

### *Liste d'expressions régulières (widget* Segment*)*

Les clés (et valeurs associées) pour les listes de fichiers sont les suivantes:

| *Clé* | *Type* | *Défaut* | *Valeur* | *Remarque* |
|---|---|---|---|---|
| ***regex*** | *chaîne* | *–* | *expression régulière* | *attention à échapper les barres obliques normales et inverses* |
| ***mode*** | *chaîne* | *–* | *"split" ou "tokenize"* | |
| ***unicode_dependent*** | *booléen* | *false* | *option -u* | *cf. http://docs.python.org/library/re.html* |
| ***ignore_case*** | *booléen* | *false* | *option -i* | *cf. http://docs.python.org/library/re.html* |
| ***multiline*** | *booléen* | *false* | *option -m* | *cf. http://docs.python.org/library/re.html* |
| ***dot_all*** | *booléen* | *false* | *option -s* | *cf. http://docs.python.org/library/re.html* |
| ***annotation_key*** | *chaîne* | *""* | *clé d'annotation* | |
| ***annotation_value*** | *chaîne* | *""* | *valeur d'annotation* | |

Exemple:

```
[
    {
        "annotation_key":    "type",
        "annotation_value":  "autre",
        "dot_all":           true,
        "mode":              "Tokenize",
        "regex":             "."
    },
    {
        "annotation_key":    "type",
        "annotation_value":  "voyelle",
        "mode":              "Tokenize",
        "regex":             "\\w",
        "unicode_dependent": true
    },
    {
        "annotation_key":    "type",
        "annotation_value":  "consonne",
        "ignore_case":       true,
        "mode":              "Tokenize",
        "regex":             "[bc\u00e7dfghjklmnpqrstvwxz]"
    },
    {
        "annotation_key":    "type",
        "annotation_value":  "chiffre",
        "mode":              "Tokenize",
        "regex":             "[0-9]"
    }
]
```