Documentation projet RI

Contents

Module `indexer` **1**
Classes . . . 1
Class `BiInverIndex` . . . 1
Attributes . . . 1
Args . . . 1
Methods . . . 2

Module `requester` **3**
Classes . . . 3
Class `Requester` . . . 3
Attributes . . . 4
Args . . . 4
Methods . . . 4

Module `indexer`

Classes

Class `BiInverIndex`

```
class BiInverIndex(index_path='/INDEX')
```

Indexe inversé bilingue français-anglais.

Attributes

`index : dict` L'index de la forme

`{'term1':{'id_doc1':freq, 'id_doc2':freq, 'id_doc3':freq, ... }, ... }`

`index_document : dict` L'index des documents de la forme

`{'id_doc1':{'nom':'nom_doc1', 'titre':'titre_doc1', 'taille':500}, ... }.`

`plain_word_fr : Pattern` Le regex-pattern pour la détection des lemmes français.
`plain_word_en : Pattern` Le regex-pattern pour la détection des lemmes anglais.
`fr_tagger : TreeTagger` Le tagger pour le français.
`en_tagger : TreeTagger` Le tagger pour l'anglais.
`save_folder : str` Emplacement où l'index est sauvegardé.
`keep_path : str` Le chemin du répertoire où seront stocké les fichiers indexés.
`index_name : str` Le nom du fichier json pour sauvegarder l'index.
`index_document_name : str` Le nom du fichier json pour sauvegarder l'index des documents.

Args

`index_path : str` le chemin de l'index.

Methods

Method `add_doc`

```
def add_doc(self, file, id)
```

Ajoute un document à l'index.

Args

`file : str` Le chemin du document à ajouter.
`id : int` l'identifiant du document.

Method `build_index`

```
def build_index(self, corpus_path, update_index=False)
```

Construit l'index inversé à partir d'un réperoire contenant des fichiers xml à indexer. Les fichiers doivent être sous la forme :

```
<article>
    <titre> </titre>
    <texte> </texte>
</article>
```

Args

`corpus_path : str` Le chemin du répertoire contenant les fichiers à indexer.
`update : bool` Indique si c'est une mise à jour de l'index. Dans ce cas l'état de l'index actuel sera récupéré. Sinon un nouvel index est crée.
Si un index existe déjà il sera supprimé avec accord de l'utilisateur False par défaut.

Returns

`index : dict` L'index nouvellement crée.

Method `check_state`

```
def check_state(self)
```

Vérifie si un dossier "INDEX" existe déjà et s'il contient bien les élements requis.

Returns: True si l'index est en bon état, False si l'index n'a pas été trouvé ou s'il est détérioré.

Method `clean_state`

```
def clean_state(self)
```

Tente de nettoyer l'environnement d'index. Vérifie si le dossier "INDEX" existe déjà. Une demande de confirmation est demandée avant de les supprimer.

Returns

True si il n'y avait pas d'état ou que l'état a bien été réinitialisé. False si l'utilisateur a refusé le nettoyage.

Method `dump`

```
def dump(self)
```

Sauvegarde l'index et l'index de document dans des fichiers json nommés "index.json" et "index_document.json".

Method `get_stats`

```
def get_stats(self, text)
```

Récupère la fréquence des termes contenus dans un texte.

Args

`text : str` Le textes à utiliser.

Returns : freq_term (dict): Les fréquences des termes trouvés dans le texte. size (int): Le nombre total de token dans le texte.

Method `import_index`

```
def import_index(self)
```

Récupère l'état actuel de l'index. Utilisé dans le cas d'un mise à jour de l'index.

Returns

`currents_docs : list` La liste des documents actuellement indexés.
`id : int` Le nouvel id, où va commencer l'indexation.

Method `keep_doc`

```
def keep_doc(self, file)
```

Copie un document dans le dossier de sauvegarde : documentsIndex.

Args

`file : str` Le chemin du fichier à copier.

Method `parse_doc`

```
def parse_doc(self, xml_file)
```

Parse un document xml de la forme :

```
<article>
    <titre> </titre>
    <texte> </texte>
</article>
```

Args

`xml_file : str` Le chemin du fichier xml.

Returns

`text : str` Le contenu de la balise texte du fichier.
`title : str` Le contenu de la balise title du fichier.

Module `requester`

Classes

Class `Requester`

```
class Requester(index, sim='euc')
```

Le requêteur qui utilise l'index créé par l'indexeur.

Attributes

`index_folder : str` Le chemin de l'index.
`index : dict` L'index recupéré du fichier index.json.
`index_document : dict` L'index des documents récupérés du fichier index_document.json.
`document_size : int` Le nombre de document.
`sim : function` La fonction utilisée pour le calcule de similarité.
`tf : function` Fonction pour calcul du tf.
`idf : function` Fonction pour calcul de l'idf.
`tfidf : function` Fonction pour calcul du tfidf.

Args

`index : str` Le chemin de l'index sur lequel effectuer les requêtes.
`sim : str` La méthode utilisée pour le calcul de similarité.

Methods

Method `filter_documents`

```
def filter_documents(self, docs, keywords)
```

Les documents sont filtrés en fonction des opérateurs.

Args

`docs : dict` le dictionnaire de documents ids - fréquences
`keywords : dict` les mots-clés trés par opérateurs

Returns

`match_docs : dict` un nouveau dictionnaire contenant seulements les documents respectants les règles des opérateurs

Method `filter_keywords`

```
def filter_keywords(self, keywords)
```

Répartit les mots-clés de la requête en sous-listes en fonction des opérateurs

Args

`keywords : list` la liste des mots-clés

Returns

`keywords_clean : list` la liste des mots-clés sans les opérateurs
`keywords_gr : dict` un dictionnaire avec les mots-clés triés par opérateur

Method `get_documents`

```
def get_documents(self, keywords_gr)
```

Extrait les documents qui contiennent les mots-clés de l'index.

Args

`keywords : list` le dictionnaire des mots-clés trié par opérateur

Returns

`extrac_docs : dict` un dictionnaire contenant les documents extraits, les clés sont les ids des documents, les valeurs sont les fréquences associées

Method `request`

```
def request(self, keywords)
```

Lance la requête.

Args

`keywords : list` La liste de mots clés

Trois opérateurs peuvent être utilisés avec les mots-clés : - + (plus) : le terme doit être présent - - (moins) : le terme doit être absent - ø (aucun opérateur) : le terme peut être absent ou présent

Returns

`result : list` Le résultat de la requête.

Method `sorted_documents`

```
def sorted_documents(self, docs, keywords)
```

Calcule le score de pertinence des documents, puis les ordonne.

Args

`docs : dict` le dictionnaire des documents ids - fréquences
`keywords : dict` les mots-clés trés par opérateurs

Returns

`result : list` la listes des documents ordonnées en fonction du score de pertinence

---

Generated by *pdoc* 0.7.5 (https://pdoc3.github.io).