Blitz 2013 coveo

Vous avez 8 heures pour concevoir un moteur de recherche de musique, comportant :

- Un module d'indexation
- L'API de recherche
- Une interface de recherche permettant des recherches complexes et présentant les résultats de façon conviviale

Vous serez évalués sur votre talent à produire des solutions performantes et fiables et pour votre créativité pour la présentation des résultats.

Les aspects esthétiques et fonctionnels des interfaces utilisateurs seront notés par un jugement collaboratif.

Vous aurez à votre disposition

- Cette documentation
- Une Machine Virtuelle (Linux ou Windows, à votre choix!) hébergée sur Amazon EC2
    - Vous pourrez vous y brancher en SSH (Linux) ou en Remote Desktop Connection (Windows)

Vous devrez développer ces trois composantes

- **Un « connecteur »** qui vous permettra de vous connecter au service qui fournit les données et d'indexer l'information nécessaire pour votre application.
- **Un indexeur** qui structurera les données récupérées par le module précédent afin de pouvoir répondre rapidement aux requêtes (vos requêtes doivent fournir une réponse en moins d'une seconde!)
- **Une interface utilisateur** qui permettra de faire de recherches dans le contenu que vous aurez indexé et afficher les résultats ainsi que des facettes. Celle-ci devra offrir la possibilité de :
    a. Filtrer les résultats avec des facettes (par exemple, année, genre musical, etc.). Ces dernières devront indiquer le compte de résultats

pour chaque valeur (exemple : nombre de résultats dont le Genre est « Rock Music »).

b. Faire des recherches avec les opérateurs « AND » (implicite, exécuté par défaut) et « OR »

c. Faire des recherches sur des champs en particulier (par exemple, l'artiste est « Rush »)

Rondes d'évaluation

1. Une première ronde avec environ 1500 documents
2. Une 2e ronde avec environ 50 000 documents
3. Une 3e ronde avec environ 210 000 documents

Pour gagner vous devrez

Cumuler un maximum de points. Vous pouvez réaliser ceci de différentes façons, soit :

1. Lors de chaque ronde d'évaluation
    a. Avoir indexé le nombre maximum d'items disponibles, le plus rapidement possible.
    b. Offrir des résultats de recherches pertinents pour un ensemble de requêtes de test que le service de Coveo soumettra à votre API, et ce, le plus rapidement possible.
2. En soumettant votre interface de recherche pour évaluation collaborative.

L'évaluation de l'indexation et de la recherche sera faite lors des 3 rondes d'évaluation. Les points reçus sont cumulatifs. Les détails de l'évaluation sont fournis à la fin du document (voir Évaluation page 22). Les différents résultats et réalisations seront affichés en temps réel sur les différents écrans placés dans nos bureaux, ainsi que sur le tableau de bord auquel vous aurez accès, question de nourrir votre esprit de compétition.

Quelques règles pour une saine compétition

- L'utilisation de librairies externes doit être autorisée par Coveo.
- Tout le code doit être écrit sur place la journée même.

- Toute action portant à nuire à une autre équipe est interdite.
- Coveo se garde le droit de regard sur le code source des solutions pour s’assurer que les règles ont été respectées.
- Coveo se garde un droit de regard sur le contenu de l’index utilisé par l’application et pourrait réviser le pointage si le contenu comporte des irrégularités.
- Le non-respect des règles peut mener à l’élimination de l’équipe.
- Un test antidopage sera réalisé à la fin du défi pour s'assurer que le code réalisé est conforme aux règles. :P

Amusez-vous, soyez compétitifs, mais pas antisportifs !

Déroulement du défi

6h30	Arrivée des participants et rencontre avec Coveo
8h00	Présentation du défi
8h30	Début de la compétition
10h30	Première ronde d’évaluation
12h00	Dîner
14h00	Deuxième ronde d’évaluation
16h00	Dernière ronde d’évaluation
16h30	Fin du concours
17h00	Présentation des applications, notation UI, 5 à 7
20h00	Remise des prix

Adresses de vos VMs, clés, et détails de dernière minute

Un wiki: http://bit.ly/blitzwiki
Dashboard: http://bit.ly/blitzdashboard
Cette documentation : http://bit.ly/blitzdoc

Vous y trouverez différentes informations utiles, tels que les adresses du tableau de bord, les adresses de vos serveurs de données respectifs, etc. Les clefs et mots de passe requis pour accéder à vos VMs Amazon vous seront remis par email.

Description des rondes

Note : vous ne pouvez soumettre qu'une seule fois pour chaque ronde. Assurez-vous donc que votre index fonctionne avec les données de test !

Ronde « Les Peewees du Carnaval » - dès 8h30

Cette micro-ronde sera ouverte dès 8h30 et sera disponible toute la journée. Vous pourrez vous évaluer sur cette ronde et voir vos résultats sur le tableau de bord. Cette ronde comporte un peu plus de cent documents et une seule requête contenant un seul mot. Nous vous suggérons d'utiliser cette ronde pour effectuer l'équivalent d'un test unitaire.

Rondes de test

Dès la fin d'une ronde officielle, ces dernières resteront disponibles pour des fins de tests et d'évaluation.

Ronde 1 –11h (15% des points)

Environ 1500 documents.
Nous ouvrons les données à 11h. La soumission de votre « Run » doit débuter avant 11h05 et vous avez 15 minutes pour récupérer les données & les indexer.

5 requêtes avec q=un seul mot.

Répartition des points : **90% récupération des données**, 10% requêtes.

Ronde 2 –14h30 (25% des points)

Environ 50 000 documents.
Nous ouvrons les données à 14h30. La soumission de votre « Run » doit débuter avant 14h35 et vous avez 20 minutes pour récupérer les données & les indexer.

2 requêtes avec un seul mot
2 requêtes avec q=plusieurs mots
1 requête incluant métadonnée (une seule).

Répartition des points : 25% récupération des données**, 75% requêtes**.

Ronde 3 –16h30 (35% des points)

Environ 210 000 documents.
Nous ouvrons les données à 16h30. La soumission de votre Ronde doit débuter avant 16h35 et vous avez 30 minutes pour récupérer les données & les indexer.
Requêtes avec OR, plusieurs contraintes de type métadonnée.

1 requête avec un seul mot

1 requête avec plusieurs mots
2 requêtes avec plusieurs mots en OR
1 requête avec plusieurs contraintes de type métadonnée

Répartition des points : **50% récupération des données, 50% requêtes**.

Présentations des Interfaces – début vers 17h00 (25% des points)

Vous présenterez votre UI à l'ensemble des équipes à partir de 17h. Un bulletin de vote vous sera remis. Vous devrez choisir les 3 équipes qui selon vous ont fait le meilleur UI. Vous ne pouvez pas voter pour vous et svp, ne négociez pas vos votes!

Détermination des gagnants

Deux prix seront remis durant la soirée.

1er prix : Un MacBook Air à chaque participant de l'équipe ayant amassé le plus de points au total.

2e prix : 250$ en carte-cadeau chez FutureShop à chaque participant de l'équipe ayant amassé le plus de points pour la partir UI (si même équipe que 1er prix, alors le prix sera remis à la 2e équipe ayant globalement amassé le plus de points).

Faire évaluer votre indexeur lors des rondes d'évaluation

Dashboard

Vous aurez accès à un tableau de bord qui vous permettra de consulter diverses informations, dont votre pointage et votre position parmi les équipes. De plus, au signal de Coveo aux heures indiquées à l'horaire, les rondes deviendront disponibles. Vous pourrez obtenir l'identifiant des rondes dont vous aurez besoin afin de démarrer votre indexeur.

De plus, avant de procéder, vous devez nous remettre l'adresse complète sous format URL de votre API de recherche, dont nous nous servirons afin d'effectuer l'évaluation.

Start/Stop

Pour démarrer l'acquisition des données, vous devez exécuter la commande suivante (http get):

serveur_données*/BlitzDataWebService/evaluationRun/start?runId=*id

où id est l'identifiant que vous aurez récupéré dans le tableau de bord.

À la fin de votre indexation, vous devrez soumettre un stop (http get) à :

serveur_données /BlitzDataWebService/evaluationRun/stop avant la fin

Nous vous recommandons chaudement de tester l'ensemble d'une ronde avec la ronde « Peewees du Carnaval ».

Pour connaître nos méthodes de calcul de pointage, consultez la section Évaluation vers la fin du document.

Notes importantes :

- Vous n'avez droit qu'à **une seule soumission**, qui est déclenchée par le Start.
- Vous êtes récompensés pour une récupération de données rapide, via un boni qui peut atteindre 60%.
- Le temps que nous utilisons pour calculer votre éventuel bonus est celui entre l'appel à Start et Stop (vous ne pourrez évidemment plus rien ajouter après le Stop!).
- Si vous dépassez le temps alloué, nous déclenchons automatiquement le Stop.
- Chaque ronde possède une fenêtre de temps pendant laquelle vous pouvez débuter le Start. Par exemple, pour la Ronde 1, vous devez déclencher le Start entre 11h et 11h05. En théorie, si vous lancez alors la Ronde, cette dernière pourra s'exécuter jusqu'à 11h20.
- Notre système d'évaluation **lancera automatiquement les requêtes d'évaluation** dès que la commande Stop aura été reçue.

Partie I: Extraction des données

La connexion à une source de données est la première étape dans l'implantation d'un moteur de recherche. L'ensemble des données doit être transféré vers votre index qui sera réutilisé pour les évaluations des résultats de recherche et par votre interface utilisateur.

Informations sur les données

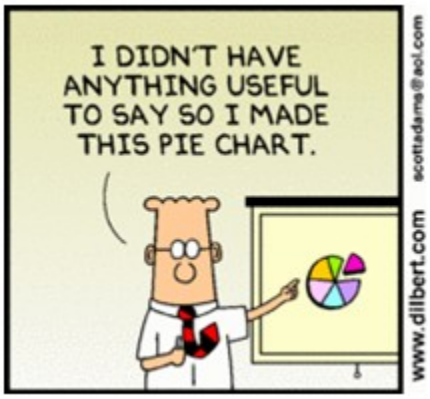

Figure 1. Graphique inutile.

- Il y a deux types de données, des « artists » et des « albums »
- Toutes les métadonnées sont de type « String »

Serveur de données

Coveo vous fournira une adresse où vous connecter pour aller chercher les données au début du concours.

Paramètres de pagination et nombre de résultats par appel

size

Permet de déterminer le nombre de résultats à recevoir par appel. Maximum de 100.

page

Permet d'effectuer une pagination pour aller chercher tous les documents disponibles.

Exemple

size=5&page=2, va chercher les résultats 11 à 15 pour une page de 10 résultats

Artistes

/artists

Exemple : /artists?size=5&page=2


```
{
   "content": [
      {
         "id": "011_vz"
      },
      {
         "id": "011_x9"
      },
      {
         "id": "011h2j"
      },
      {
         "id": "011hdn"
      },
      {
         "id": "011hxp"
      }
   ],
   "firstPage": false,
   "totalPages": 24770,
   "numberOfElements": 5,
   "totalElements": 123846,
   "lastPage": false,
   "size": 5,
   "number": 1
}
```


Notez les paramètres qui vous permettent de paginer votre acquisition des données.

Pour aller chercher un artiste en particulier, vous devez utiliser le id dans le URL selon le format

```
/artists/id
```

Exemple : /artists/011_x9


```
{
   "id": "011_x9",
   "name": [
      "Joe Hahn"
   ],
   "origin": [
      "Los Angeles"
   ],
   "genres": [
      "Electronica",
      "Rapcore",
```


```
        "Electronic dance music",
        "Rap metal",
        "Hip hop music",
        "Industrial rock",
        "Nu metal",
        "Alternative rock"
    ],
    "labels": [
        "Warner Bros. Records"
    ],
    "group_names": [
        "Linkin Park",
        "Xero"
    ],
    "instruments_played": [
        "Keyboard",
        "Sampler"
    ],
    "text": "'''Joseph \"Joe\" Hahn''' (born March 15, 1977), also known by his stage name, '''Mr. Hahn''', is an American turntablist and director best known as the DJ and sampler for the American rock band Linkin Park."
}
```


Les métadonnées à produire sous forme de facettes sont : *name*, *origin*, *genres*, *labels*, *group_names* et *instruments_played*.

Albums

/albums

Pour aller chercher un album en particulier

/albums/id

Exemple :

/albums/01hcg8p


```
{
    "id": "01hcg8p",
    "name": [
        "Big Pimpin'"
    ],
    "artists": [
        "01vw20h"
    ],
    "release_date": [
        "2000-09-12"
    ],
    "track_names": [
        "Big Pimpin' (instrumental)",
        "Girls Best Friend",
        "Big Pimpin' (radio edit)"
    ]
}
```


Dans cet exemple d'album, les metadata que vous devez produire en facette sont : *name*, *artists*, *release_date* et *track_names*.

Note 1: les champs « id » et « text » ne doivent pas être utilisés en facette.

Note 2: vous remarquerez parfois que des espaces précèdent les valeurs dans les champs, par exemple : " Post-modern factory rock". Vous devez enlever les espaces avant et après le premier caractère des valeurs de champ.

Partie II: Indexation et Recherche

Tokenization

Contenu du texte

L'une des premières choses que vous devrez mettre en place est votre routine de tokenization, qui consiste à diviser le texte en « mots » qui seront indexés. Vous devez absolument suivre les règles suivantes :

- la recherche ne doit pas tenir compte de la casse (i.e : case-insensitive)
- les caractères permis dans un mot sont les caractères alphanumériques, incluant les caractères accentués en français (é, è, etc.), ainsi que le tiret – et le souligné _ (donc, *Del-Lords* doit être indexé comme un seul mot). Bonus si vous supportez l'arabe et le Tagalog.
- Les mots formés uniquement de tirets sont ignorés. Ceci dit, nous ne chercherons pas ----.

Exemple :

```
This is, if I'm not mistaken, a half-good example of what a less naïve tokenizer (if such a contraption exists) should be able to index! -- For good measure, it should also parse numbers like 1000... We can index python magic methods like __init__, but we will ignore things like -----! Aren't we good?
```

Tokens:

```
['this', 'is', 'if', 'i', 'm', 'not', 'mistaken', 'a', 'half-good', 'example', 'of', 'what', 'a', 'less', 'naïve', 'tokenizer', 'if', 'such', 'a', 'contraption', 'exists', 'should', 'be', 'able', 'to', 'index', 'for', 'good', 'measure', 'it', 'should', 'also', 'parse', 'numbers', 'like', '1000', 'we', 'can', 'index', 'python', 'magic', 'methods', 'like', '__init__', 'but', 'we', 'will', 'ignore', 'things', 'like', 'aren', 't', 'we', 'good']
```

Métadonnées

En ce qui a trait aux métadonnées, vous devez les indexer telles quelles (donc, les recherches par métadonnées (avec un paramètre tel que *genres=Rock music* devront contenir exactement la même chaîne et les facettes devront présenter exactement ces valeurs).

Vous devez ajouter le contenu des métadonnées au document indexé en suivant les règles de tokenization précédentes. Ainsi, si la métadonnée contient « Rock Progressive », la requête *progressive* va retourner ce document (mais pas la requête *genres=progressive* parce que la chaîne complète doit être cherchée dans une recherche par métadonnée).

API de Recherche

Mon API, il est full REST

L'API de recherche est un API de type REST

http://en.wikipedia.org/wiki/Representational_state_transfer.

Il a une seule méthode qui sait répondre à des requêtes HTTP de la forme suivante:

```
GET /your/search/endpoint?q=Salut HTTP/1.1
Accept: application/json
Content-Type: application/json; charset=utf-8
Host: 192.168.12.34
User-Agent: Coveo
```

Découpons la requête afin de voir les différentes composantes.

HTTP method

```
GET /your/search/endpoint?q=Salut HTTP/1.1
```

La requête HTTP est de type GET et se fait sur le path de votre choix. Elle utilise HTTP 1.1.

Voir http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods

Accept

```
Accept: application/json
```

La requête accepte les réponses en format JSON uniquement.

Voir http://en.wikipedia.org/wiki/JSON

Content-Type

```
Content-Type: application/json; charset=utf-8
```

La requête contient du JSON en UTF-8.

Voir http://en.wikipedia.org/wiki/JSON

Voir http://en.wikipedia.org/wiki/UTF-8

Host

```
Host: 192.168.12.34
```

La requête se fait sur votre serveur (cette adresse IP sera remplacée par le hostname de votre serveur).

User-Agent

```
User-Agent: Coveo
```

La requête sera effectuée par Coveo (le User-Agent pourrait changer, vous pouvez l'ignorer).

Format général des requêtes

Voyons maintenant comment la requête elle-même est envoyée à votre service.

- Les paramètres de la requête sont passés via query string.

  Voir http://en.wikipedia.org/wiki/Query_string

- Tous les paramètres ont des valeurs « percent encoded » et il est de votre responsabilité de bien lire la requête.

  Voir http://en.wikipedia.org/wiki/Percent-encoding

Voici les paramètres que votre service doit supporter, en ordre de complexité:

- **q=**

  Le paramètre 'q' contient le texte recherché.

  "Bonjour", "Bonjour%20terre", "Bonjour%20OR%20Allo", "Bonjour%20AND%20Allo"

  Le fonctionnement du paramètre 'q' est couvert plus en détail dans la section *Syntaxe détaillée du paramètre q*.

- **metadata=**

  Les paramètres *metadata* sont en fait le nom de métadonnées, par exemple "origin" ou "year". **Les métadonnées ne supportent que l'opérateur OR entre les valeurs.**

  Le fonctionnement des métadonnées est couvert plus en détail dans la section *Syntaxe des paramètres de type metadata***.**

Syntaxe détaillée du paramètre *q*

Le paramètre 'q' représente la requête entrée par l'utilisateur. Ce paramètre permet de rechercher dans le contenu de vos items.

Pensez à Google:

```
http://www.google.com/search?q=Recursion
http://www.google.com/search?q=Zerg%20rush
http://www.google.com/search?q=Do%20a%20barrel%20roll
```

Comme mentionné dans la liste des fonctionnalités que votre API doit supporter, la requête peut contenir les mots clés "OR". C'est également via le paramètre 'q' que ces mots clés seront passés.

Note : **Par défaut, l'opérateur AND est appliqué entre les mots**. Nous n'enverrons pas de requêtes qui mélange le AND implicite et le OR. Exemple, nous pourrons envoyer *tom sawyer* (AND implicite), *tom OR sawyer OR rush*, mais pas *tom sawyer OR rush* (combinaison non supportée de *AND* et de *OR*).

Exemples :

```
q=Rush%20Tom%20Sawyer
```

(Trouve les documents qui contiennent tous les mots *Rush*, *Tom* et *Sawyer*).

```
q=Rush%20OR%20Tom%20OR%20Sawyer
```

(Trouve les documents qui contiennent au moins un des mots)

Syntaxe des paramètres de type *metadata*

Les autres paramètres passés via la requête représentent des métadonnées. Les items que vous ajouterez à vos index contiennent de nombreuses métadonnées.

Par exemple, un artiste a un pays d'origine. Cette information est contenue dans la métadonnée "origin" dans le corpus.

Votre index doit donc permettre de rechercher via un champ métadonnée "origin".

Pour continuer avec l'exemple de Google, il est possible de filtrer par la langue de la page recherchée.

Ainsi, si on veut trouver les pages qui parlent de Coveo en allemand sur Google, il est possible d'utiliser la metadonnée "lr" (language result) avec comme valeur "lang_de" (Deutsch)

```
https://www.google.com/search?q=Coveo&lr=lang_de
```

Dans le cas de votre API, les métadonnées ne supportent que l'opérateur OR entre les valeurs.

Voici des exemples de paramètres de recherche de métadonnées :

```
genres=Rock%20music%20OR%20Thrash%20Metal
```

```
genres=Hard%20rock%20OR%20Thrash%20Metal
```

À noter qu'une valeur entière doit être trouvée et non seulement une partie des mots de la métadonnée. Par exemple, la recherche *genres=rock* ne doit pas retourner les documents qui ont pour genre *rock progressive*.

Opérateur entre les catégories de métadonnées et la requête

Un opérateur logique « AND » est appliqué entre la requête q et chacune des catégories de métadonnées spécifiées.

Par exemple, pour la requête suivante :

```
http://server/search?q=Olympic%20games&origin=United%20States&genres=
Rock%20music%20OR%20Thrash%20Metal
```

Les résultats doivent contenir les mots *olympics* et *games*, le pays (origin) doit être *United States* et le genre doit être *Rock* music ou *Thrash Metal*.

Requêtes sans paramètre « q »

Certaines requêtes peuvent ne pas avoir de section *q*. Par exemple, pour obtenir tous les albums *Rock* provenant des États-Unis :

```
http://server/search?origin=United%20States&genres=Rock%20Music
```

Exemples d'URL complets de recherche

- Trouver tous les items qui parlent de "Rush":

```
http://server/search?q=Rush
```

- Trouver tous les items qui parlent de "Canadian Rock Music":

```
http://server/search?q=Canadian%20Rock%20Music
```

- Trouver tous les items qui parlent de "Olympic games" et qui ont comme metadonnée "origin" la valeur "United States" et comme metadonnée "genres" la valeur "Rock music":

```
http://server/search?q=Olympic%20games&origin=United%20States&genres=Rock
%20music
```

- Trouver tous les items qui parlent de « Olympic » ou de « Olympics » et qui ont comme metadonnée origin" la valeur "United States"

```
http://server/search?q=Olympic%20OR%20Olympics&origin=United%20States
```

- Trouver tous les items qui ont comme metadonnée "genres" la valeur "Rock music" OU la valeur "Techno":

```
http://server/search?genres=Rock%20music%20OR%20Techno
```

La réponse

La réponse de votre API ReST doit être de la forme suivante :

```
HTTP/1.1 200 Ok
Content-Type: application/json
Server: YourAwesomeServer
Content-Length: 163

{
  "facets": {
    "origin": {
      "Georgia": 1,
      "White Plains": 1
    },
    “type”: {
       "albums": 32,
       "artists": 19
    },
    "genres": {
      "Alternative rock": 2,
      "Pop rock": 1,
      "Indie rock": 1
    }
   },
   "results": [{
                "id": "the original id"
            }, {
                ....
            }]
}
```

Découpons la réponse pour voir les différentes composantes.

HTTP Status Code

```
HTTP/1.1 200 Ok
```

La réponse utilise HTTP 1.1 et indique que tout s'est bien déroulé avec un status de type "200 - OK".

Voir http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Content-Type

```
Content-Type: application/json
```

La réponse contient du JSON.

Server

```
Server: YourAwesomeServer
```

Le type du serveur web qui a répondu à la requête.

Content-Length

```
Content-Length: 163
```

Si la longueur de la réponse est connue par le serveur au moment de la requête et qu'il envoie tout le contenu en une seule réponse, Content-Length peut être utilisé afin de spécifier la longueur de la réponse.

Voir http://en.wikipedia.org/wiki/List_of_HTTP_header_fields#Content-Length

Format du corps de la réponse

Le corps de la réponse contient, sous la forme de JSON, les résultats de la requête.

Voici les différents objets présents en JSON :

Results

results [Collection de Result]. Vous devez retourner TOUS les résultats qui répondent à la requête.

Result

id [string]. Vous devez retourner l'id original tel qu'il a été fourni par notre API de données. Cet ID sera utilisé pour évaluer les documents que vous retournez.

text[string]. Optionnel, mais vous en aurez probablement besoin pour afficher un extrait dans vos résultats de recherche.

metadata. Optionnel. Bien que notre service ne valide pas cette information, vous pouvez ajouter l'information dont vous aurez besoin dans votre UI pour afficher des métadonnées.

Facets

Vous devez retourner dans la section « facets » (voir l'exemple) les métadonnées agrégées de tous les résultats qui satisfont la requête.

- Les facettes sont un dictionnaire de métadonnées, où pour chaque métadonnée, la liste des différentes valeurs trouvées dans les résultats, ainsi que le nombre d'occurrences est retourné.
- **N'oubliez pas la facette *type***, qui ne fait pas partie des métadonnées fournies par le serveur de données, mais que vous devez ajouter vous-mêmes, soit *album* pour les albums et *artist* pour les artistes!
- Certaines facettes vont contenir des ids (d'albums ou d'artistes). Vous devez laisser les ids tels quels dans votre réponse car c'est ainsi que nous allons évaluer le résultat. Ceci dit, côté UI, vous devriez peut-être envisager d'afficher le nom de l'album ou de l'artiste correspondant...
- Si vous êtes en .NET, pour sérialiser votre dictionnaire, nous vous recommandons newton Json.NET, car si vous utilisez la sérialisation par exemple de *DataContractJsonSerializer*, vous pourriez avoir des difficultés à sérialiser le dictionnaire de métadonnées à moins de spécifier l'existence de toutes les métadonnées (*origin*, *genres*, etc.) explicitement dans le code.
-

Partie III : Interface utilisateur

La finalité de ce concours est évidemment de faciliter la vie des visiteurs du site Web de musique. Une interface utilisateur devra donc leur permettre de trouver la musique qu'ils cherchent. Votre application devra mettre à profit l'information que vous avez extraite et stockée dans l'index. N'hésitez pas à poser des questions à l'équipe Coveo qui représente votre « client » dans ce projet. Ils seront également vos juges pour l'évaluation finale. Vraiment, rien ne vaut une bonne discussion pour bien comprendre ce que les gens recherchent.

Boîte de recherche

Évidemment, une boîte de recherche devra être présentée. Au-delà de cette boîte de recherche et des résultats affichés, vous devez mettre votre créativité en œuvre afin de faciliter l'exploration des données. Inspirez-vous de sites connus tels qu'Amazon pour créer des facettes, des liens qui créent des recherches automatiques, des façons de présenter l'information relative aux albums dans les résultats qui retournent des artistes, etc.

Facettes

Les facettes permettent de naviguer dans les résultats de recherche en restreignant les valeurs retournées à celles qui correspondent à des critères sur des métadonnées. Voir, par exemple, le site http://www.platt.com/search.aspx?q=bulbs qui présente, dans la partie gauche, diverses facettes permettant de préciser sa recherche. Votre interface de recherche doit donc inclure des facettes fonctionnelles permettant aux utilisateurs de spécifier des critères tels que le pays d'origine, le genre de musique, etc. À vous de déterminer quelles facettes sont requises et pertinentes! Lorsqu'une facette est sélectionnée, elle doit produire une requête avec des métadonnées correspondant à la sélection.

Vous devez avoir une facette « type » qui contient « artist » ou « album ».

Les images

Afin de fournir une belle présentation, il est suggéré de présenter des vignettes (*thumbnails*) des albums et artistes retournés par les requêtes. Les images sont disponibles sur votre serveur de données, via l'adresse suivante :

/BlitzDataWebService/images/id

Évaluation

Chaque ronde a une pondération, la ronde 1 vaut 15%, la ronde 2 vaut 25% et la ronde 3 vaut 35% dans le pointage final, tandis que le UI comptera pour 25%.

Connecteur

La capacité de votre système à se connecter à l'API de données de Coveo et à récupérer les données rapidement sera évaluée de la façon suivante :

Lors de chacune des rondes d'évaluation, un sous-ensemble des données sera « visible ». Votre connecteur doit récupérer le maximum de données à chacune des itérations.

Proportion des documents de chaque ronde récupérée

Lors de chaque ronde, nous mesurons le nombre de documents qui étaient disponibles que vous avez récupérés avec succès. Votre score pour chaque ronde est alors calculé de la façon suivante :

$$score = 100 * proportion_{récupérée}^{2}$$

Par exemple, si vous récupéré 90% des documents, votre score sera de 81. Ainsi, chaque pourcentage additionnel grappillé vous est quadratiquement récompensé.

Boni pour acquisition des données rapide

Pour récompenser les équipes qui réussissent à aller chercher les données plus rapidement que le temps alloué, un boni pouvant atteindre 60% des points calculés dans la section précédente sera accordé. Ce boni correspond à 3% par minute utilisée (entre le Start et le Stop) de moins que le temps alloué.

Exemple : le temps alloué est de 15 minutes. Il s'écoule 9 minutes entre la réception de votre Start et de votre Stop, pendant lesquelles vous avez récupéré 90% des données. Vous avez donc 81 points + un boni de 18% sur ces points, donc un total de 95.6 points.

Indexation et recherche

Afin de mesurer la bonne indexation des documents, nous allons mesurer également les résultats que vous retournerez pour un ensemble de requêtes de tests.

Coveo a établi exactement, pour chaque requête, les deux ensembles suivants :

1) l'ensemble des documents (mesuré à l'aide de leur ID) qui doivent être retournés (l'ordre n'est pas important) (70% des points)

2) l'ensemble des facettes et les comptes de chacune des valeurs respectives de ces facettes (30% des points)

Évaluation de l'ensemble de résultats

En ce qui concerne le premier point, la mesure utilisée est la mesure F1 de «précision/rappel » (voir http://en.wikipedia.org/wiki/Precision_and_recall). La précision s'assure que vous retourniez uniquement des documents qui contiennent les valeurs recherchées, et le rappel s'assure que vous retourniez tous les documents qui devraient l'être. La mesure F1 combine ces deux paramètres en un seul score.

Évaluation des résultats de facette

Les valeurs de facette et leurs comptes respectifs, l'approche est similaire. Pour chaque valeur de facette attendue, nous mesurerons votre score selon l'équation suivante :

$$\frac{\sum_{valeurs} max\left(100 - \left|valeur_{attendue} - valeur_{retournée}\right|^2, 0\right)}{nombre_{¿}}$$

Aussi, à partir d'une différence de 10 entre la valeur attendue et la valeur retournée, vous n'obtenez plus aucun score pour une valeur de facette évaluée. Évidemment, pour une requête donnée, il peut y avoir des dizaines de valeurs évaluées (par exemple, toutes les valeurs de *country*, de *genre*, etc.) par requête.

Exemple :

Nous attendons les valeurs suivantes :

Country
Canada, 14
United States, 108
United Kingdom, 36
Spain, 2
Ireland, 1

Genres
Rock music, 89
Pop music, 76
Death metal, 1

Mais vous retournez des erreurs

Country
Canada, 16 **<== au lieu de 14**
United States, 108

United Kingdom, 36
Spain, 2
Ireland, 1

Genres
Rock music, 81 **<== au lieu de 89**
(il manque Pop Music)
Death metal, 1

Il y a 8 valeurs de facette évaluées, vous obtenez donc

= (96 + 100 + 100+ 100+ 100 + 36 + 0 + 100) / 8

= **79% des points** pour l'évaluation des facettes de cette requête.

Malus pour requêtes lentes

La vitesse compte! Chez Coveo, on considère qu'une requête qui prend plus d'une seconde est un échec (au moins partiellement). Aussi, pour chaque requête à laquelle vous répondrez en plus d'une seconde, vous perdrez 1% des points associés à cette requête par tranche de 0.1s au-delà d'une seconde.

Par exemple, si une de vos requêtes devait prendre 1.8 seconde, vous subiriez une pénalité de 8%. À partir de 10 secondes, vous perdez donc tous vos points.

Si votre moteur est lent, mais précis, à vous de déterminer si vous avez avantage à retourner un ensemble partiel de résultats, mais rapidement!

Étapes d'évaluations

Voici le sétapes à faire pour chaque ronde d'évaluation :

1- Vous appelez la méthode **Start**

2- Vous démarrez la **récupération et l'indexation** des données

3- Vous appelez la méthode **Stop** lorsque votre **récupération** ainsi que **l'indexation** sont complétées

4- Les services de Coveo lanceront automatiquement l'évaluation par requête dès la méthode **Stop** est appelée

Interface de recherche

En ce qui a trait à l'interface de recherche, les critères évalués sont :

- La qualité, la créativité, l'ergonomie du design et l'expérience utilisateur
- Le bon fonctionnement des "facettes"

- L’insertion de liens entre albums et artistes (exemple : lister les albums d’un artiste)

Au cours de la journée, si vous le souhaitez, vous pourrez présenter des maquettes aux « clients » (juges Coveo) afin d’obtenir leurs impressions.

À la fin du défi, vous serez évalués par trois juges. Vous aurez 10 minutes pour leur démontrer la force de votre application. Soyez prêts dès leur arrivée.

Il est fortement conseillé de commencer la réalisation de l’interface utilisateur dès le début du défi.

Autres informations utiles

Index inversé

Selon wikipédia :

> En informatique, un **index inversé** est une correspondance entre du contenu, comme des mots ou des nombres, et sa position dans un ensemble de données comme un enregistrement en base de données, un document ou un ensemble de documents ; sur le même principe qu'un index terminologique. Le but de l'index inversé est de permettre une recherche plein texte plus rapide, contre un temps d'insertion de nouvelles données augmenté.

Par exemple, si doc1 = « a b c » et doc2 = « b c d », la structure conserve :

a -> doc1
b -> doc1, doc2
c -> doc1, doc2
d -> doc2

Extraction des données

Le multi-threading, c'est efficace!

Requêtes qui retournent 1 000 000 000 résultats (ou qui prennent autant de secondes!)

Lors des tests, vous devez retourner TOUS les résultats qui répondent aux requêtes. Servez-vous de cette info pour éviter des requêtes qui accidentellement prendraient le reste de vos jours.

Aussi, étant donné le malus pour les requêtes qui prennent plus de 1 seconde, il est impossible d'obtenir des points pour une requête si cette dernière prend 10 secondes ou plus. Vous pouvez donc mettre un délai d'expiration dans votre algorithme pour essayer de répondre au moins partiellement dans des délais plus courts!

Checklists

Voici une liste pour vous aider à valider que vous avez rempli les exigences du concours:

Extraction des données

- ☐ Support du Start avec ID de la Ronde
- ☐ Support du Stop
- ☐ Acquisition des albums et des artistes
- ☐ Ajout de la métadonnée *type* qui détermine si l'objet est un *artist* ou un *album*
- ☐ « Trimmage » des espaces pour les métadonnées, avant et après

Index et recherche

- ☐ Support des appels API : HTTP GET, utf8, application/json, etc.
- ☐ Support des requêtes percent-encoded
- ☐ Support des règles de tokenization (mots = séquences alphanumériques, incluant - et _)
- ☐ Retour des résultats et des facettes en format Json selon les requis
- ☐ Requêtes retournent en moins d'une seconde (pénalités progressives jusqu'à 10 secondes)

Ronde 1:

- ☐ Assurez-vous de nous avoir fourni votre nom d'équipe et qu'il est visible sur le tableau de bord!
- ☐ Assurez-vous de nous avoir communiqué l'adresse de votre API pour les recherches!
- ☐ Contenu textuel des métadonnées peut être recherché via le paramètre *q*
- ☐ Facettes incluent toutes les métadonnées, sauf *text* et *id*

Ronde 2:

- ☐ Paramètre *q* : Support de l'opérateur par défaut (AND implicite)

- ☐ Paramètre *q* : Support de l'opérateur explicite *OR*
- ☐ Support du paramètre de type métadonnées (dois « matcher » les valeurs exactes complètes seulement)
- ☐ Opération de type « AND » entre l'opérateur *q* et les métadonnées (en somme : entre les items séparés par des & dans l'URL)
- ☐ Support de recherches sans paramètre *q* (i.e : uniquement métadonnées)

Ronde 3:

- ☐ Paramètre métadonnées avec *OR* entre les valeurs d'une même métadonnée

Interface-utilisateur

- ☐ Afficher les facettes contenant liste des valeurs ayant au moins 1 occurrence
- ☐ Afficher les facettes avec des comptes de valeurs
- ☐ Sélections dans les facettes génèrent les bons paramètres de requête
- ☐ Facettes « multi-sélection » avec opérateur « OR » entre les valeurs
- ☐ Images / Thumbnails