Vers un nombre raisonnable de résultats

Nicolas Béchet

1 Introduction

1.1 Contexte général et existant

Les travaux présentés dans ce rapport visent à répondre aux besoins des systèmes de recherche d'informations (SRI) dont les problématiques suivantes, liées aux résultats qu'ils fournissent, ont été constatées. En fonction d'une requête émise par un utilisateur, les résultats fournis par le système sont souvent trop denses ou bien au contraire trop faibles ou nuls. Il est en effet assez fréquent qu'un utilisateur de moteur de recherche Web se voit proposer un nombre important de résultats, suite à une requête, comme l'illustre la figure 1. Nous montrons par le biais de cet exemple qu'une requête sur un moteur de

FIGURE 1 – Exemple de requête avec un moteur de recherche générant un nombre trop important de résultats

recherche Web peu générer un nombre conséquent de résultats. L'utilisateur se voit alors contraint, s'il n'a pas obtenu le résultat escompté sur la première page de résultat, soit de consulter les pages suivantes, soit de reformuler sa requête de manière plus précise afin de converger vers un résultat pertinent.
A l'inverse, il se peut qu'un utilisateur, suite à une requête sur un SRI, n'obtienne aucun résultat comme l'illustre la figure 2. Cet exemple montre qu'en fonction des critères choisis par l'utilisateur, le système de recherche de location de logements ne lui fait aucune proposition. Dans ce cas, l'utilisateur n'a pas d'autre choix que de reformuler sa requête afin d'obtenir des résultats. Le problème rencontré par l'utilisateur est alors double . Il n'y a d'une part aucun résultat retourné suite à sa requête et d'autre part aucune indication relative au critère responsable de ce résultat. En d'autres termes, n'y a t-il aucun résultat car le prix maximal souhaité par l'utilisateur est trop faible, ou alors est-ce car il n'y a plus d'appartements disponibles sur Versailles ?

Les travaux présentés dans ce rapport peuvent s'apparenter à une tâche d'expansion de requêtes. Des nombreux travaux de la littérature traitent de cette problématique. L'objectif initiale d'une telle tâche est de permettre de remédier au problème du faible nombre de résultats obtenus en réponse à une requête. Parmi les premières approches d'expansion de requêtes, nous trouvons un certain nombre d'approches citées dans [GGM92] appliquant une expansion de requête collaborative dans un système de questions/réponses. Les auteurs présentent un survol des différentes approches proposées dans les années 80 afin de pouvoir reformuler des requêtes en fonction de remarques d'utilisateurs.
En se fondant sur une représentation vectorielle des documents contenus dans un SRI, la tâche d'expansion de requête revient à effectuer une pondération de composantes du vecteur représentant la requête utilisateur en fonction d'informations supplémentaires. Citons par exemple les travaux de [QF93] ou

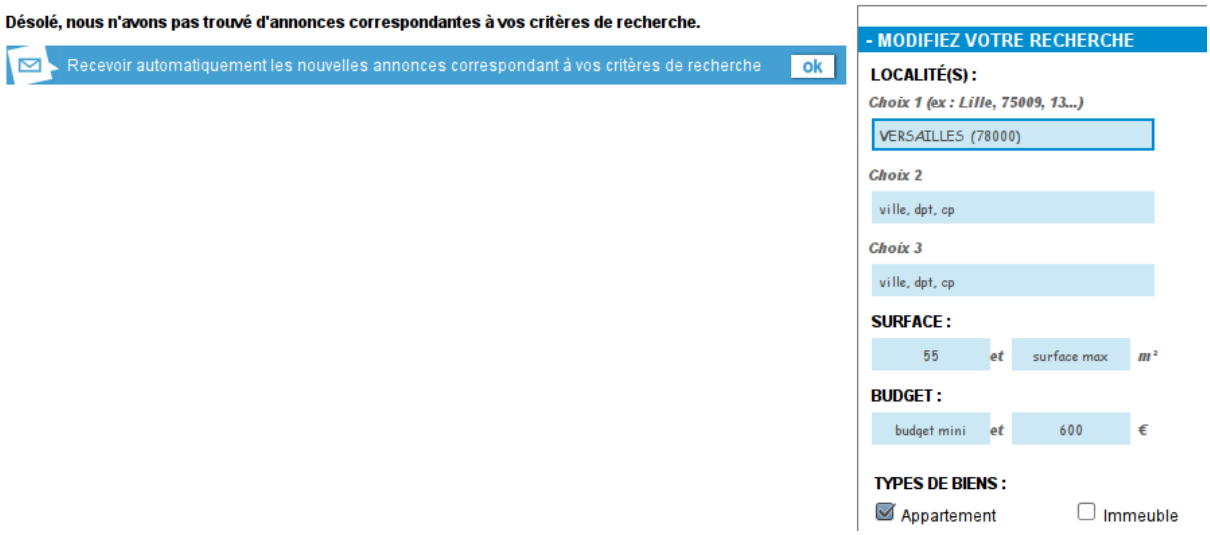

FIGURE 2 – Exemple de requête ne retournant aucun résultat

encore de [Voo94] qui réalises ces tâches en se basant sur les méthodes introduites dans [Sal86]. Ces travaux proposent d'étendre les composantes des vecteurs, qui ne sont autres que des concepts sémantiques, à d'autres concepts sémantiques proches. Il est également évoqué la possibilité de rajouter de nouveaux concepts aux vecteurs mais ces approches peuvent dégrader les résultats s'il elle n'est pas contrôlée humainement tels que le montrent [MM00].
En outre, l'utilisation de ressources sémantiques, et plus particulièrement Wordnet [Mil85], est fréquemment employée dans la littérature afin de réaliser des expansions de requêtes. Citons [GMV99] qui ont proposés le système OntoSeek, dans lequel ils ont introduits une méthodologie d'expansion de requêtes sur différents catalogues de produits et sur les pages jaunes en utilisant Wordnet. Citons également [MM00] ou encore [Voo94] qui exploitent également Wordnet. Des travaux plus récents exploitent d'autres des ressources sémantiques tel que [NHT+07]. Les auteurs appliques leurs travaux à l'expansion de requêtes sémantiques et de reclassement de résultats dans un contexte de sélection de documents multimédias. Le principe est, à partir de mots clés, d'identifier les concepts auxquels ils appartiennent. Dès lors, ils reformulent une nouvelle requête ou simplement reclassent les résultats initiaux en fonction de ces concepts. Citons également [VCLV08] qui proposent d'exploiter la notion de concepts "non partagés". Ces travaux reposent sur la notion de classes sémantiques, construites à partir de ressources multiples, et les concepts "non partagés" sont des concepts provenant d'une ressource particulière, n'ayant pas de corrélations avec d'autres concepts provenant d'autres ressources. Leurs travaux, exploitant ces concepts, montrent une amélioration significatives par rapport à l'existant.
Outre l'exploitation de ressources sémantiques, citons [CWNM02] qui proposent une méthode afin de sélectionner des termes à ajouter à une requête existante. Leur méthode s'appuie sur des analyses statistiques de corpus de documents en identifiant les termes co-occurrents. D'autres approches comme celle proposée dans les travaux de [LWC06] propose d'effectuer un apprentissage en fonction des documents consultés par des utilisateurs, personnalisant également les réponses fournies par retour de pertinence.
D'autres travaux d'expansion de requêtes traitent également de la suggestion ou de la personnalisation de requêtes sur le Web. Citons tout d'abord [PKM07] qui proposent de construire un profil d'utilisateur dont la finalité est d'effectuer des recommandations collaboratives aux utilisateurs, reformulant leurs requêtes sur le Web, en les adaptant à leurs comportements mais également à celui d'utilisateurs ayant un comportement similaire.
Par ailleurs, [JZH09] effectuent des suggestions de requêtes en fonction d'une requête initiale sur le

Web. Le principe est de chercher à classer en tête dans de nouvelles requêtes des documents provenant de réponses de la requête initiale. Dès lors, l'utilisateur peut choisir une nouvelle requête ciblant ainsi son intérêt pour un document particulier de la requête initiale.

Nos travaux présentés dans ce document peuvent s'inscrire dans une démarche similaire visant à proposer à l'utilisateur d'obtenir des résultats plus pertinent en fonction de la requête qu'il a effectuée. Notons cependant que notre méthodologie ne s'appuie pas sur la notion d'expansion de requête, mais plus sur le fait d'obtenir toujours un nombre constant de résultats. Ainsi, nous ne nous focalisons pas sur les requêtes elles-même mais sur les résultats que produit la requête utilisateur.
Les travaux proposés dans ce contexte visent ainsi à proposer une méthodologie permettant de proposer à un utilisateur de système de SRI d'obtenir, quelque soit la requête qu'il émet, un nombre dit "raisonnable" de résultat. Ainsi, dans le cas ou le SRI ne fourni pas un nombre raisonnable de résultats, nous distinguons deux cas :

1. Le SRI ne retourne aucun résultat
2. Le SRI retourne trop de résultats

Le nombre raisonnable de résultats peut-être défini comme suit. L'objectif est de proposer à un utilisateur une seule page de résultats accessible rapidement et consultable à moindre coût. Par exemple, pour un SRI contenant principalement des données textuelles, un nombre raisonnable de résultats pourrait être 10. Citons par exemple les archives photographiques de la collection Tucci (présentée dans la section suivante), pour lesquels 10 résultats sont pertinents.
Les travaux présentés dans ce rapport ont été réalisés dans le cadre du projet européen IDEAS, et expérimentés avec les données fournies par l'ISIAO, relativement aux ressources photographiques provenant des différentes expéditions menées par Giuseppe Tucci. Nous présentons dans la section suivante l'illustration de la problématique du nombre raisonnable de résultats appliquée aux systèmes de recherche d'informations du site Web de l'ISIAO.

1.2 La base de donnée Tucci

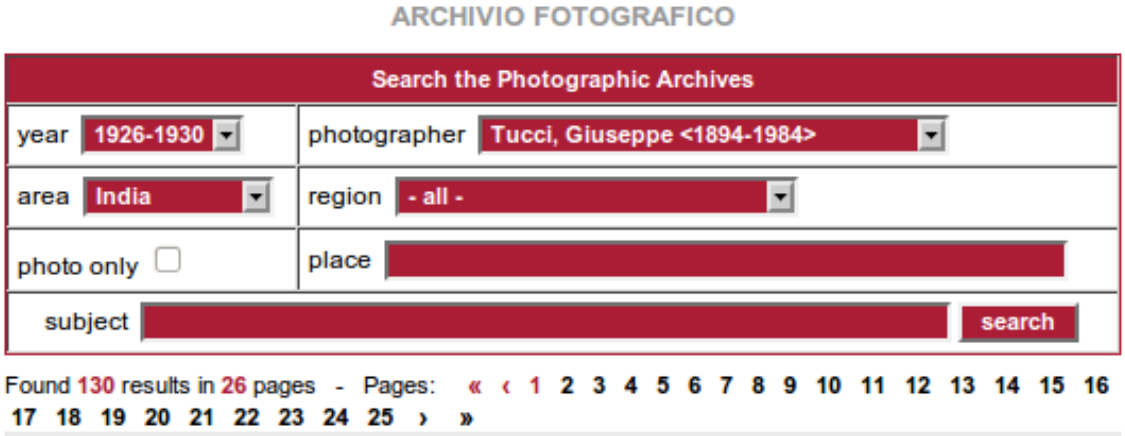

FIGURE 3 – Exemple de requête retournant trop de résultats

Les travaux que nous avons réalisé afin d'aboutir à une méthodologie sont fondés sur les données fournies par l'ISIAO (Istituto Italiano per l'Africa e l'Oriente). Ce dernier maintient en effet le site Web relatif aux archives photographiques provenant des expéditions menées par Giuseppe Tucci. Ce site Web

contient en autres, un système de recherche d'informations permettant à un utilisateur d'effectuer une requête permettant de cibler plus précisément des photographies provenant des archives. Parmi les critères disponibles, les utilisateurs peuvent saisir un intervalle de temps, une zone géographique avec différents niveaux de granularité, un photographe précis, le sujet de la photographie et finalement s'il souhaite que le résultat soit illustré.

Le SRI Tucci, illustré sur la figure 3, est également confronté à la problématique du nombre de résultats trop important ou nul. En effet, comme le montre la requête effectuée dans l'exemple de la figure

FIGURE 4 – Exemple de requête ne retournant aucun résultat

3, un utilisateur peut obtenir un nombre trop important de résultats ou bien à l'inverse pas de résultat (figure 4), nécessitant une reformulation de sa requête ou bien de parcourir toutes les pages de résultats.

1.3 Méthodologie

La méthodologie présentée dans ce document se fonde sur la mise en place de deux approches :

1. Aucun résultat suite à une requête : approche d'expansion
2. Trop de résultats suite à une requête : approche de réduction

Les approches de réduction et d'expansion vont permettre d'adapter la requête initiale de l'utilisateur afin de converger vers un nombre raisonnable de résultats. L'expansion peut par exemple, dans le cas de l'exemple de la figure 2, consister à modifier la requête initiale en augmentant le budget maximal initialement saisi par l'utilisateur. Notons que dans certains cas, effectuer une expansion fourni un nombre supplémentaire de résultats pouvant être trop . En effet, en reprenant l'exemple de la figure 2, l'utilisateur peut se voir proposer plus de 100 résultats avec la nouvelle requête, ce qui ne correspond pas à un nombre raisonnable de résultats. Ainsi, une étape de réduction pourra également intervenir dans le cadre des méthodes d'expansion.

Pour les SRI, les notions de réduction et d'expansion sont assez dépendantes des données manipulées. Faire une expansion sur un prix ou sur une zone géographique nécessite par exemple deux méthodologies différentes. Nous pouvons alors distinguer trois types de données couramment manipulés avec ce type de systèmes :

1. Les données spatiales.
2. Les données temporelles.
3. Les données propres aux systèmes.

Nous discuterons dans un premier temps des pré-traitement à apporter aux données manipulées. Nous présenterons alors les méthodes d'expansion et de réduction avant de conclure en présentant des perspectives à ces travaux.

2 Pré-traitement des données

2.1 Pourquoi un pré-traitement ?

Le pré-traitement de données est une étape primordiale lors de la mise en place d'un SRI. En effet, cette étape consiste à formater les données brutes, de telle sorte qu'elles soient manipulables par le système. En d'autres termes, cette étape revient à se questionner sur l'aspect de la représentation des données manipulées.

2.2 Analyse de la répartition des données

Le première étape à réaliser lors de la mise en place d'une méthode visant à proposer un nombre raisonnable de résultats, consiste à analyser la répartition des données constituant le SRI étudié. En effet, la méthodologie proposée repose en partie sur la distribution des différents types de données évoqués en section 1.3. Par exemple, il sera inutile de faire une expansion de résultats sur un type de données ne contenant qu'une seule instance. Si l'ensemble de nos photographies, dans le cas de la base de données Tucci, ont toutes été prises par le même photographe, faire une expansion à d'autres photographes semble superflu.
Il a donc été réalisé une étude de la répartition des données fournies par l'ISIAO relatives aux photographies de Tucci. Nous distinguons, tel qu'évoqués dans la section 1.3, trois types de données que sont les données spatiales, temporelles et celles propres aux systèmes.

Les données spaciales
La répartition des photographies d'un point de vue géographique est assez hétérogène telle que montrée

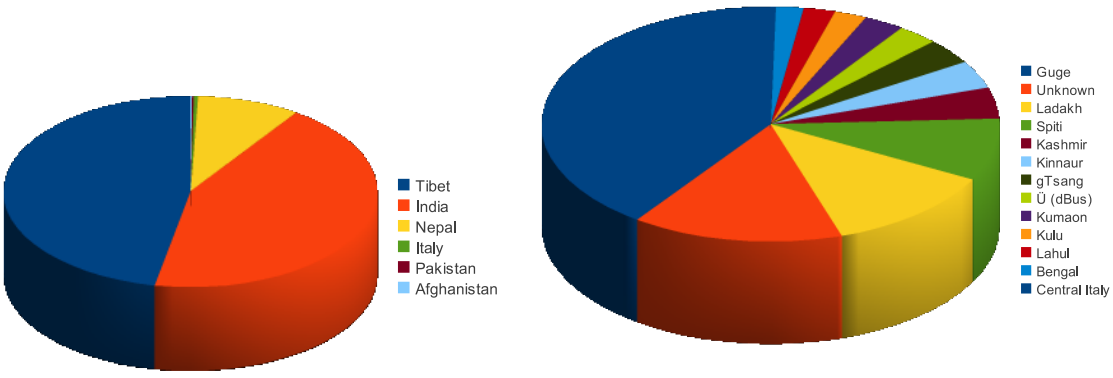

FIGURE 5 – Data repartition : Area

sur la figure 5. Les principaux pays où ont été prises les photographies sont le Tibet et l'Inde.
Cette même figure 5 montre également la répartition des photographies en fonction des différentes régions du Tibet. Cette répartition est plus homogène avec néanmoins un tiers des photographies situées

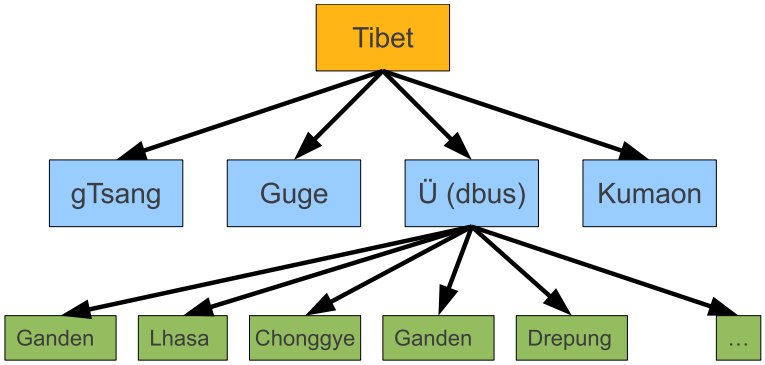

FIGURE 6 – Data repartition : Area Hierarchy.

dans la région du Gange.
Outre les régions et les zones géographiques (area), les données géographiques de la collection Tucci sont organisées sous forme hiérarchique. Ainsi, le Tibet possède un certain nombre de régions, elles mêmes décomposables en un certain nombre de lieux (place).
La figure 6 illustre cette hiérarchie pour le Tibet en montrant qu'il se décompose en un certain nombre de régions, elles mêmes possédant des 'places'.

Les données temporelles
Les données temporelles de la base de données Tucci ont la particularité de posséder des dates incertaines. Ainsi, la répartition des photographies pour le critère temporel a été redéfini de la manière suivante.
Considérons dans un premier temps le tableau suivant représentant la distribution précédente des photographies en fonction des années où elles ont été prises tel qu'illustré dans la figure 7. Cette figure

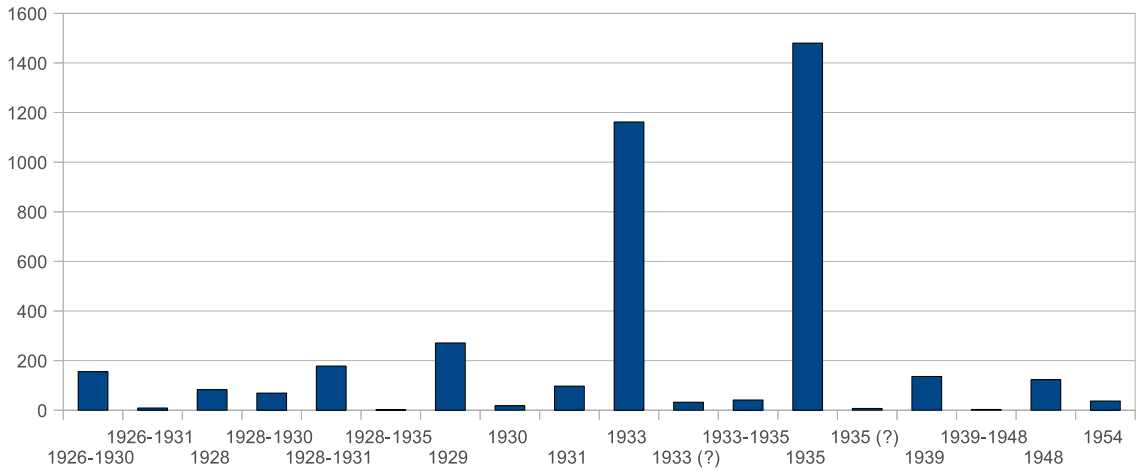

FIGURE 7 – Nombre de photographies en fonction de l'année.

met en évidence les années incertaines représentées sous forme d'intervalle, comme *1926-1930*, mais également sous la forme 'Date (?)', comme par exemple avec *1933 (?)*. Le pré-traitement réalisé afin de pouvoir traiter ce type d'informations a consister à reconsidérer la répartition des dates pour ces données incertaines. Ainsi, les deux types ont été traités.

– Les intervalles
Les photographies ayant été affectées à un intervalle de temps ont été redistribuées sur l'ensemble des années couvertes par cet intervalle. Par exemple, dans l'intervalle *1926-1930*, 155 photographies ont été réalisées. Ainsi, nous avons réparti ces photographies de manière homogène dans la période *1926-1930* en affectant 31 photographies pour chaque année soit :
1926 : 31 photographies,
1927 : 31 photographies,
etc.
Notons que notre tâche consiste ici à répartir les photographies par années et non pas sur un intervalle d'années. En outre, nous n'affectons pas de photographie particulière à une année mais il s'agit juste d'équilibrer la répartition des photographies.
– Les ' ?'
Afin de traiter les années avec un point d'interrogation, nous avons fait le choix de considérer ces années comme un intervalle de type 'année-1 – année+1'. Par exemple, pour l'année *1933 (?)*, nous avons interprété cette date comme l'intervalle *1932-1934*. Finalement, si 12 photographies sont concernées, la répartition devient la suivante.
1932 : 4 photographies,
1933 : 4 photographies,
1934 : 4 photographies. Finalement, après avoir appliqué ces règles à l'ensemble des photographies de la collection Tucci, nous obtenons la nouvelle répartition des dates présentée dans la figure 8.

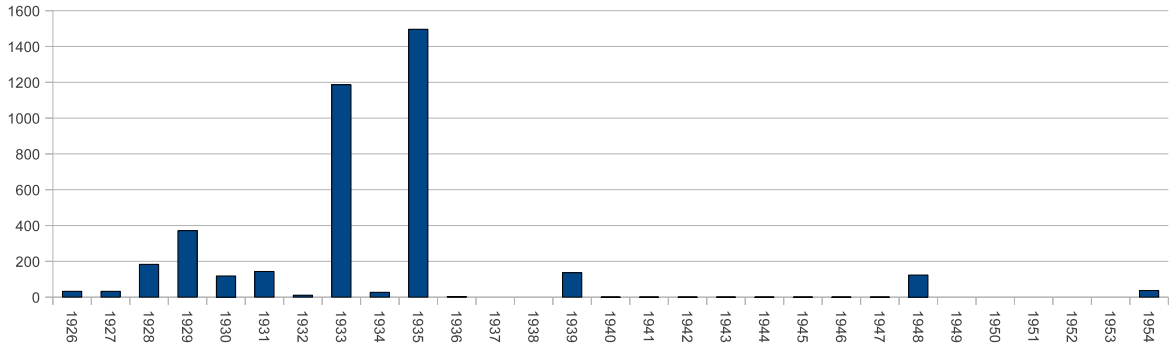

FIGURE 8 – Nombre de photographies en fonction de la nouvelle répartition temporelle.

Cette nouvelle répartition a fait émerger des périodes significatives durant lesquelles des photographies ont été prises. Nous avons ainsi défini cinq périodes :

– Entre 1926 et 1932
– 1933
– 1935
– Période seconde guerre mondiale
– Période post-guerre

Notons pour finir que ces périodes ne reflètes pas les dates des expéditions menées par Tucci mais correspondent aux résultats des analyses que nous avons réalisées avec les données temporelles.

Les données propres au système

Parmi les données propres à la base de données Tucci, nous pouvons citer les photographes et les sujets des photographies. La figure 9 présente la répartition des photographes ayant contribué à la collection

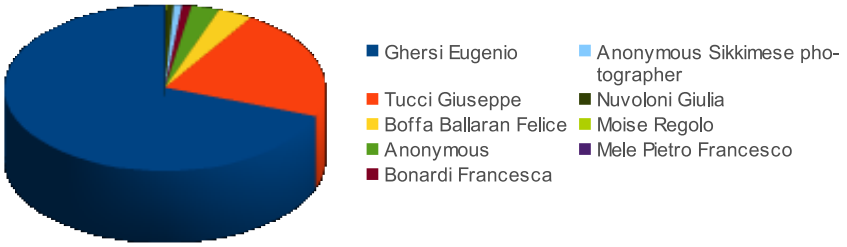

FIGURE 9 – Data repartition : Photograph

Tucci.
Cette figure montre que 90% des photographies ont été réalisée par Eugenio Ghersi ou par Guiseppe Tucci. La répartition des photographes est donc assez hétérogène et montre qu'un utilisateur choisissant par exemple d'obtenir des photographies de Regolo Moise obtiendra peu de résultats.
Par ailleurs, les sujets des photographes pourraient constituer un type de données spécifique puisque qu'il s'agit de données textuelles, décrivant les thématiques des photographies. Parmi les sujets, nous avons relevé avec notre échantillon un total de 1768 sujets distincts, dont 513 possèdent plus d'une instance. Environs 38 % des lemmes utilisés dans les sujets sont des noms, et les cinq termes les plus employés sont 'Mural painting', 'Temple', 'Landscape', 'Inscription' et 'Expedition'.

3 Expansion

Rappelons que l'objectif de ces travaux est de proposer à un utilisateur de système de recherche d'informations un nombre raisonnable de résultats. Deux cas de figures peuvent alors être rencontrés :

- Soit les résultats proposés à l'utilisateur sont nuls ou très peu nombreux
- Soit les résultats proposés à l'utilisateur sont trop nombreux

Ainsi nous traiterons dans une première partie les différentes approches mise en œuvre afin d'effectuer une expansion des résultats obtenus, puis présenterons dans un second temps une méthodologie visant à réduire le nombre de résultats (en section 4).
Les méthodes d'expansions peuvent être décomposées en fonction du type de données à traiter. En effet, en prenant par exemple la base de données des archives photographies de Tucci, nous avons précédemment évoqué les différents types de données manipulés. Dès lors, faire une expansion avec des données hiérarchiques n'est pas la même tâche que celle consistant à faire une expansion avec des données temporelles.
Nous distinguons alors dans les sections suivantes les différents types d'expansion proposés en fonction des types de données manipulés. Notons que la dernière sous section relative à l'expansion est consacrée au choix du type de données à étendre (en section 3.4).

3.1 Données hiérarchiques

Afin d'effectuer une expansion de données hiérarchiques, et plus particulièrement de données spatiales avec la collection Tucci, plusieurs approches peuvent être envisagées. D'une part exploiter la hiérarchie et considérer par exemple qu'une 'place' peut-être étendue à une 'région'. Une autre possibilité serait de considérer les zones géographiquement proches, en tenant compte d'un certain nombre de paramètres, comme le relief, l'histoire commune de deux pays, etc. Ce dernier point peut être défini par un expert qui indiquera des zones considérées comme proches.

La prise en compte de la hiérarchie

Afin justifier le choix de l'expansion basée sur la hiérarchie, appuyons nous sur l'exemple suivant.

FIGURE 10 – Requête utilisateur ne retournant aucun résultat.

Considérons une requête utilisateur sur la collection Tucci telle qu'illustré sur la figure 10. Cette requête ne retourne aucun résultat en considérant les photographies prises par Tucci au Tibet dans la région gTsang. Ainsi, le principe de l'expansion est dans ce cas de ce focaliser sur la hiérarchie en reformulant la requête de l'utilisateur sans prendre en compte le critère relatif à la région. Ainsi, la requête consiste à demander des photographies prises par Tucci au Tibet.

Les résultats obtenus pour la seconde requête sont présentés dans la figure 11. On constate alors que six

FIGURE 11 – Même requête en occultant le critère 'région'.

photographies répondent à ce critère et l'utilisateur obtient donc un nombre raisonnable de résultats à sa requête qui initialement ne retournait aucune photographie.

Finalement, en automatisant le processus d'expansion de données hiérarchique avec la collection Tucci, la soumission de la requête de la figure 10 retourne directement des résultats étendus comme montré dans la figure 12. Nous constatons bien sur cette figure que l'utilisateur se voit désormais retourner six résultats '**étendus**'.

Notons qu'avec cette méthode, il n'est pas nécessaire de stocker la hiérarchie complète des données mais uniquement la structure de l'arbre afin de pouvoir reformuler la requête. En d'autres termes, il faut dans notre cas stocker la structure "*Area->Region->Place*". Reste finalement à définir une fonction qui retournera le père d'un critère données comme "*père(Region)*" qui retournera "*Area*".

Proximité géographique définie par un expert

Dans ce cas, l'expert fournit une liste pouvant être modélisée à l'aide un graphe (avec GraphML par

FIGURE 12 – Expansion réalisée en s'appuyant sur la hiérarchie géographique.

exemple) ou encore dans un simple fichier texte. L'idée d'un telle méthode revient alors à ne plus s'appuyer sur la hiérarchie des données afin de faire une expansion mais sur une expertise humaine, permettant de définir réellement la proximité de zones géographiques. Plus concrètement, il est par exemple plus logique d'un point de vue distance géographique, de faire une expansion de la région du Kashmir en Inde aux Nepal, Tibet et Pakistan plutôt qu'à toutes les régions de l'Inde dont notamment la région du Bengal qui est assez éloignée du Kashmir. Nous présentons ci-dessous un exemple de fichier texte pouvant définir la proximité de zone géographique.

```
'Afghanistan' = 'Pakistan', 'India', 'Tibet', 'Nepal', 'All'
'India' = 'Nepal', 'Pakistan', 'Tibet', 'Afghanistan', 'All'
                        'Italy' = 'All'
'Nepal' = 'India', 'Tibet', 'Pakistan', 'Afghanistan', 'All'
'Pakistan' = 'Afghanistan', 'India', 'Tibet', 'Nepal', 'All'
'Tibet' = 'Nepal', 'India', 'Pakistan', 'Afghanistan', 'All'
```

Ce fichier utilisé pour le critère 'Area' permet de définir la proximité des Area entres-eux. Ainsi, une requête de donnant aucun résultat en Inde sera dans un premier temps étendu au Népal, puis si nous n'avons toujours pas obtenu un nombre raisonnable de résultats, la requête sera alors étendu au Pakistan, etc. Notons que 'All' signifie dans ce fichier que l'ensemble des Area sont pris en compte.

Algorithme

L'algorithme proposé ci-dessus permet d'effectuer une expansion de requêtes via des données hiérarchiques.

```
#####################################################################
ExpandGeo(Results, Nb, Nb_raisonnable, Node, Tree, Query)

 ENTREE : Results : liste des résultats d'une requête,
          Nb      : nombre de résultats,
```

```
          Nb_raisonnable : le nombre de résultats à atteindre,
          Node   : position dans la hiérarchie,
          Tree   : la hiérarchie,
          Query  : la requête utilisateur
 SORTIE : Results : les résultats étendus

 While Nb < Nb_raisonnable
     Node <- Father(Node, Tree)
     Results <- UpdateQuery(Query, Node, Tree)
     Nb <- Size(Results)
 End While
 Return Results

#####################################################################
Father(Node, Tree)

 ENTREE : Node, Tree
 SORTIE : Node : le noeud correspondant au père dans Tree

#####################################################################
UpdateQuery(Query, Node, Tree)

 ENTREE : Query, Node, Tree
 SORTIE : Results : La liste des résultats de la nouvelle requête

 Cette fonction reformule la requête initiale en supprimant toutes
 les condition de rang inférieur à Node dans la hiérarchie

#####################################################################
Size(Results)

 ENTREE : Results
 SORTIE : Nb : Le nombre d'élément de Results
```

Notons que l'algorithme proposé est uniquement adapté aux données hiérarchiques, mais il pourrait être adapté aux données fournies par un expert.

3.2 Données temporelles

Les méthodes d'expansion

Les données temporelles ont la particularité d'être linéaires et continues. En s'appuyant sur les prétraitements réalisés sur les données temporelles en section 2.2, nous pouvons considérer deux types de requêtes possibles pour un utilisateur avec ces données. Soit l'utilisateur choisi un intervalle de dates (de 1941 à 1943), soit il choisi une classe comme par exemple '*Periode post-guerre*'. Dès lors, deux types d'expansion sont possibles, soit une expansion aux dates voisines, soit une expansion aux classes voisines. L'exemple 13 illustre le cas ou un utilisateur choisi un intervalle de dates et se voit proposer des résultats étendus à des dates voisines. Dans ce cas, la requête émise ne proposait aucun résultat entre 1926 et 1930, mais six résultats étendus ont été proposés, pour l'année 1931. De manière similaire, l'expansion temporelle aux classes voisines illustrée sur la figure 14, montre qu'aucune photographie n'a été réalisée par Tucci dans la région du Gange durant la période de la seconde guerre mondiale.

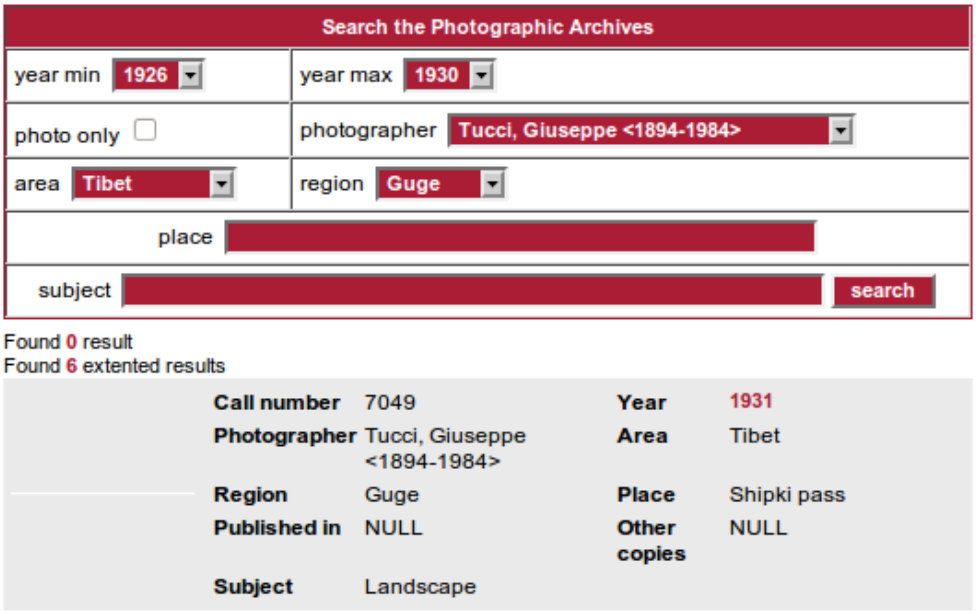

FIGURE 13 – Exemple d'expansion temporelle.

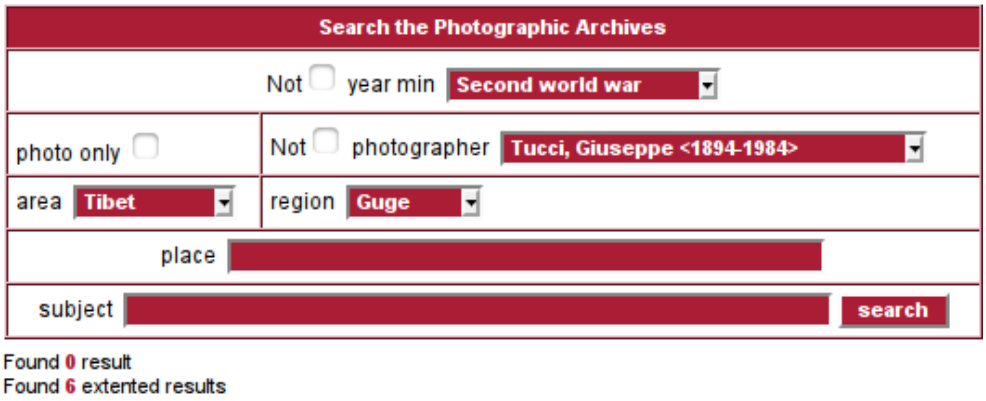

FIGURE 14 – Exemple d'expansion temporelle à partir des classes.

Discussions

Bien que les approches d'expansions aux dates voisines sembles plus précises, évitant de faire une requête sur des dates trop éloignées, cette dernière parait cependant bien trop coûteuse. En effet, elle nécessite actuellement des requêtes pour chaque test d'expansion. Par exemple, en reprenant la figure 13, l'algorithme suppose de réinterroger la base de données tant que l'on a pas obtenu un nombre raisonnable de résultats. Nous avons dans exemple obtenu un résultat voisin à une année près mais supposons qu'il n'y a pas de résultats avant 1950. Il aurait alors fallu exécuter 20 requêtes supplémentaires. De plus, la collection Tucci de possède pas une amplitude temporelle très importante (de 1926 à 1954 actuellement). Mais il est tout à fait possible de rencontrer dans un autre système une amplitude temporelle supérieur à 1000, le nombre de requêtes serait alors beaucoup trop important afin de faire une expansion de ce type. Ainsi, la méthode se fondant sur les classes semble mieux adaptées en terme de temps de traitement ou complexité.

Notons pour finir que les classes de dates définies suite à une analyse de la répartition des données peut être redéfini par un expert, en s'appuyant par exemple sur les dates réelles des expéditions menés par Tucci. Ces informations peuvent être fournies, à la manière des informations géographiques évoquées dans la section précédente, sous forme de graphes, ou encore de simples fichiers textes.

Algorithme
L'algorithme proposé ci-dessus permet d'effectuer une expansion de requêtes via des données temporelles.

```
#########################################################################
ExpandTime(Results, Nb, Nb_raisonnable, Query, Interval)

 ENTREE : Results : liste des résultats d'une requête,
          Nb      : nombre de résultats,
          Nb_raisonnable : le nombre de résultats à atteindre,
          Query  : la requête utilisateur,
          Interval : un couple de dates minimum et maximum ou
                     une classe de dates
 SORTIE : Results : les résultats étendus

 While Nb < Nb_raisonnable and not Border(Interval)

     Results = UpdateQuery(Query, Interval)
     Nb <- Size(Results)

 End While

 Return Results

#########################################################################
Border(Interval)

 ENTREE : Interval : un tableau de dates minimum et maximum ou
                     une classe de dates
 SORTIE : Out : Un booléen retournant vrai si l'on a atteint les
                bornes de l'intervalle

 Cette fonction doit distinguer le cas ou l'intervalle est un couple
 et celui ou il est une classe (il peut être dans ce cas utilisé
 avec la fonction 'Convert')

#########################################################################
UpdateQuery(Query, Interval, ClassList, Position)

 ENTREE : Query  : la requête utilisateur,
          Interval : un tableau de date minimum et maximum ou
                     une classe de date
          ClassList : la table des classes ordonnées
                      (optionnel)
          Position : position de la classe choisie par
```

```
                    l'utilisateur dans ClassList
 SORTIE : Results : la liste obtenue avec la nouvelle requête

 Switch Interval[1]

     Case Null
         Interval[0] <- Interval[0]-1
         Interval[1] <- Interval[1]+1

     Other Case
         Class  <-  Interval[0]
         (Date_min, Date_max) <- Convert(Class)
         If (Exists List[Position-1])
             [Date_min, Temp] <- Convert(List[Position-1])
         Fin If
         If (Exists List[Position+1])
             [Temp, Date_max] <- Convert(List[Position-1])
         Fin If
         Results <- Querying(Query, Interval)

 End Switch

 Return Results

#####################################################################
Convert(Class)

 ENTREE : Class    : Une classe temporelle,
 SORTIE : Date_min : La date minimum de l'intervalle,
          Date_max : La date maximum de l'intervalle

 Cette fonction convertie une classe temporelle en un intervalle
 composé d'une date minimum et d'une date maximum

#####################################################################
Exists(Tab)

 ENTREE : Tab    : Une composante supposé d'une tableau,
 SORTIE : Out : Un booléen retournant vrai si la composante existe
```

3.3 Autre type de données

Les autres types de données qui sont propres aux différents systèmes étudiés peuvent d'une manière générale être manipulées en utilisant la notion de classes. L'objectif est d'obtenir des relations de proximité entre les instances de ce type de données. Par exemple avec la collection Tucci, il s'agirait de construire une table de proximité des photographes entres eux. En d'autres termes, quels sont les photographes proches d'un autre par exemple.
Ainsi, notre proposition afin de traiter ce type de données est de construire des vecteurs de photographes avec Tucci par exemple, dont les descripteurs seront un des autres critères du système (ou plusieurs).

Nous pouvons par exemple décrire un photographe en fonction des lieux où il a photographié, ou encore en fonction des périodes. Afin de concrétiser cette méthode, nous nous appuierons sur la collection Tucci avec le type de données 'photographe'. Nous avoir choisi de décrire les photographes de Tucci en fonction des sujets qu'ils ont photographiés.

Construction de la matrice

La première étape consiste à construire une matrice de co-occurrences des photographes en fonction des différents mots contenus dans les sujets des photographies qu'il a pris. Nous caractérisons alors un photographe par la présence ou l'absence d'un mot dans les sujets de ses photographies. Un extrait de la matrice obtenue est donné dans la table 1.

Photograph	*meadow*	*arch*	*meal*	*baby*
Anonymous	0	0	0	0
A. Sikkimese photographer	0	0	0	0
Boffa Ballaran, Felice	0	0	0	0
Bonardi, Francesca	0	0	0	0
Ghersi, Eugenio	1	1	1	1
Mele, Pietro Francesco	0	0	0	0

TABLE 1 – Extrait de la matrice de co-occurrences.

Calcul des similarités

Une fois la matrice obtenue, nous possédons pour chaque photographe un vecteur qui le défini. L'étape suivante consiste à calculer la proximité de tous les photographes deux à deux. Nous avons opter ici pour un calcul de cosinus entre les vecteurs. Cette mesure fut notamment la première à être utilisée dans le système de recherche d'information SMART [Sal71]. Le cosinus entre deux vecteurs est obtenu en calculant le *produit scalaire* entre ces deux vecteurs, que nous divisons par le produit de la norme des deux vecteurs. Le cosinus entre deux vecteurs $\overrightarrow{u}$ et $\overrightarrow{v}$ de telle sorte que θ soit l'angle formé par ces deux vecteurs est défini par l'équation suivante :

$$\theta = arccos \frac{\overrightarrow{u}.\overrightarrow{v}}{\|\overrightarrow{u}\|.\|\overrightarrow{v}\|}$$

Les scores obtenus sont présentés dans le tableau 2.

L'expansion

Le dernière étape est l'expansion elle même. Avec la matrice du tableau 2, nous sommes maintenant en mesure d'établir un classement de photographes. Par exemple, le photographe '*Eugenio Ghersi*' est proche de '*Giuseppe Tucci*' (score de 0.45), puis des photographes '*anonymes*' (score de 0.26), puis de '*Giulia Nuvoloni*' (score de 0.19), etc.

Nous pouvons finalement faire une expansion sur le critère photographe. Si un utilisateur souhaite consulter des photographies de '*Eugenio Ghersi*', nous ferons si nécessaire une expansion avec les photographies de '*Giuseppe Tucci*', des photographes '*anonymes*', etc.

La liste complète des proximités de photographes est donnée dans le tableau 3. Ainsi, le photographe numéro 9 '*Giuseppe Tucci*' est proche des photographes numéro 5, 1, 8, 3, etc.

	Anonymous	*Anonymous Sikkimese Photographer*	*Boffa Ballaran, Felice*	*Bonardi, Francesca*	*Ghersi, Eugenio*	*Mele, Pietro Francesco*	*Moise, Regolo*	*Nuvoloni, Giulia*	*Tucci, Giuseppe*
Anonymous	1,00	0,18	0,19	0,17	0,26	0,12	0,11	0,10	0,31
A. Sikkimese photographer	0,18	1,00	0,13	0,27	0,10	0,38	0,17	0,19	0,14
Boffa Ballaran, Felice	0,19	0,13	1,00	0,00	0,17	0,00	0,08	0,17	0,21
Bonardi, Francesca	0,17	0,27	0,00	1,00	0,07	0,00	0,00	0,00	0,09
Ghersi, Eugenio	0,26	0,10	0,17	0,07	1,00	0,05	0,10	0,19	0,45
Mele, Pietro Francesco	0,12	0,38	0,00	0,00	0,05	1,00	0,00	0,00	0,06
Moise, Regolo	0,11	0,17	0,08	0,00	0,10	0,00	1,00	0,15	0,14
Nuvoloni, Giulia	0,10	0,19	0,17	0,00	0,19	0,00	0,15	1,00	0,24
Tucci, Giuseppe	0,31	0,14	0,21	0,09	0,45	0,06	0,14	0,24	1,00

TABLE 2 – Proximité des photographes en fonction des sujets de leurs photographies.

1	*Anonymous*	9	5	3	2	4	6	7	8
2	*A. Sikkimese photographer*	6	4	8	1	7	9	3	5
3	*Boffa Ballaran, Felice*	9	1	5	8	2	7	4	6
4	*Bonardi, Francesca*	2	1	9	5	3	6	7	8
5	*Ghersi, Eugenio*	9	1	8	3	2	7	4	6
6	*Mele, Pietro Francesco*	2	1	9	5	3	4	7	8
7	*Moise, Regolo*	2	8	9	1	5	3	4	6
8	*Nuvoloni, Giulia*	9	5	2	3	7	1	4	6
9	*Tucci, Giuseppe*	5	1	8	3	2	7	4	6

TABLE 3 – Proximité des photographes en fonction des sujets de leurs photographies (classement).

3.4 Choix du critère à étendre

Nous avons évoqué précédemment des approches visant à réaliser une expansion des résultats obtenus suite à une requête utilisateur. Ces approches proposes à un utilisateur de converger vers un nombre raisonnable de résultats, en fonction d'un critère donné. En effet, nous faisons expansion sur un critère temporelle, spatiale, ou autre. Dès lors, nous devons nous interroger sur le choix du critère à étendre afin de réaliser une expansion. En d'autres termes, pourquoi faire une expansion de dates plutôt que géographique ?
Ainsi, le choix du critère à étendre peut être énuméré de la manière suivante.

1. D'une part, ne sélectionner que les critères retournant de nouveaux résultats.
2. Puis, ordonner les critères restant en fonction d'une liste de préférences de critères.
3. Finalement, ajouter les nouvelles réponses à la requête d'un utilisateur en effectuant une expansion pour chaque critère jusqu'à atteindre le nombre idéal de résultats.

Sélection de critères pertinents
La première étape peut être réalisée simplement en réalisant une requête, comptant le nombre de résultats possibles en faisant une expansion sur un critère particulier. Ainsi, nous sommes en mesure de connaître les critères intéressants pour l'expansion. Cependant, cette approche à l'inconvénient d'être

coûteuse car une requête est nécessaire pour chaque critère, même ceux qui ne pourront pas être utilisé pour l'expansion.
Une seconde approche peut consister à charger en mémoire un hypercube de données comprenant les effectifs des résultats possibles par critère afin de prendre une décision sur le critère à étendre. Le principe de l'hypercube est de reposer sur un module extérieur réalisant des requêtes en amont, indiquant les critères intéressant à conserver. Cette approche à l'avantage d'être moins coûteuse que la précédente car une fois le cube chargé, il ne reste plus qu'a le consulter afin de connaître les critères pertinents. Il n'y donc plus d'interrogations de la base de données. Notons cependant que la construction de l'hypercube doit se limiter à quatre ou cinq dimensions pour de ne pas nécessiter trop d'espace en mémoire. Une dimension de l'hypercube est alors un critère.
Dans nos expérimentations, nous avons sélectionné quatre critères afin de réaliser notre hypercube, produisant ainsi un hypercube à quatre dimensions. Nous avons ainsi conservé les dimensions '*Classes de temps*', '*Photographes*', '*Area*' et '*Region*'. Un extrait de l'hypercube généré est donné ci-dessous.

```
$service{'Class2'}{'Ghersi, Eugenio'}{'India'}{'Kinnaur'} = 23;
$service{'Class2'}{'Ghersi, Eugenio'}{'India'}{'Kulu'} = 19;
$service{'Class2'}{'Ghersi, Eugenio'}{'India'}{'Lahul'} = 13;
$service{'Class2'}{'Ghersi, Eugenio'}{'India'}{'Spiti'} = 35;
$service{'Class2'}{'Ghersi, Eugenio'}{'India'}{'Unknown'} = 4;
```

Cette exemple nous montre que pour la seconde classe de temps, signifiant l'année '*1933*', le photographe '*Eugenio Ghersi*' a réalisé dans la région du '*Kinnaur*' en '*Inde*' 23 photographies. Le principe est ici de couvrir l'ensemble des requêtes possibles avec ces quatre dimensions afin de pouvoir rapidement effectuer une sélection de critères. Néanmoins, cette méthode est moins précise que celle consistant à faire une requête pour chaque critère.

La liste de priorités
Une fois la sélection de critères pertinents réalisés, il est alors nécessaire de concevoir une liste de priorités indiquant l'ordre de préférence des critères à sélectionner. Lors de nos expérimentations réalisées avec la collection Tucci, nous avons proposé la liste suivante. Notons que cette dernière peut-être modifiée par un expert.

```
1 : Temporelle
2 : Photographe
3 : Spatiale
4 : Sujet
```

Algorithme
L'algorithme ci-dessous propose de synthétiser le processus globale d'expansion de réponses à fournir à un utilisateur afin d'obtenir un nombre raisonnable.

```
#####################################################################
Expansion(Results, Query, Nb, Nb_raisonnable,
          Hypercube, PriorityList, Node, Tree, Interval)

 ENTREE : Results : liste des résultats d'une requête,
          Nb      : nombre de résultats,
          Nb_raisonnable : le nombre de résultats à atteindre,
          Query  : la requête utilisateur,
          PriorityList : liste des priorités de critères
```

```
          Tree   : la hiérarchie géographique,
          Query  : la requête utilisateur,
          Interval : un tableau de dates minimum et maximum ou
                     une classe de dates
 SORTIE : Results : les résultats étendus

 i <- 0

 While Nb < Nb_raisonnable and not EndOfList(PriorityList[i])

     If (Hypercube[PriorityList[i]] > 0)

         Results <- Push(Results,
                         Expand(PriorityList[i], Results, Nb,
                                Nb_raisonnable, Query,
                                Node, Tree, Interval))
         Nb <- Size(Results)

     End If

     i <- i + 1

 End While

 Return Results
#######################################################################
Expand(Criteria, Results, Nb, Nb_raisonnable, Query, Node,
       Tree, Interval)

 ENTREE : Criteria : le nom du critère
          Results : liste des résultats d'une requête,
          Nb      : nombre de résultats,
          Nb_raisonnable : le nombre de résultats à atteindre,
          Query  : la requête utilisateur,
          Tree   : la hiérarchie géographique,
          Query  : la requête utilisateur,
          Interval : un tableau de dates minimum et maximum ou
                     une classe de dates
 SORTIE : Results : les résultats étendus

 Switch Criteria

     Case 'Spatiale'
         Return ExpandGeo(Results, Nb, Nb_raisonnable,
                          Node, Tree, Query)

     Case 'Time'
         Return ExpandTime(Results, Nb, Nb_raisonnable,
                           Query, Interval)
```

```
 End Switch

 Return Results

#########################################################################
Push(List1, List2)

 ENTREE : List1, List2 : deux listes
 SORTIE : List : une liste contenant les éléments de
                 list1 et ceux de list2
```

Exemple d'expansion multicritères
Afin d'illustrer l'algorithme de choix des critères à étendre, nous nous appuyons sur l'exemple 15. La

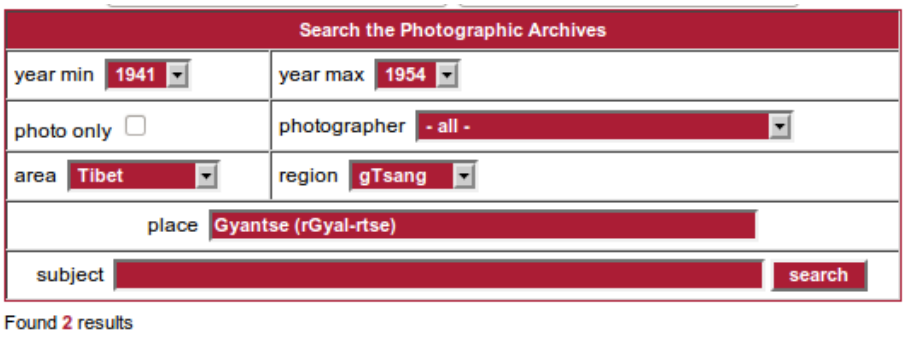

FIGURE 15 – Exemple d'expansion avec choix de critères

requête émise par l'utilisateur n'a dans ce cas retournée que deux résultats. Le nombre raisonnable de résultats fixé par l'utilisateur est ici de 10. Ainsi, une expansion va être réalisée en fonction des différents critères afin d'atteindre ce nombre. Le choix des critères a été fixé par l'utilisateur. Dans un premier temps, l'expansion est réalisée sur l'année, puis sur les aspects géographiques.
Dans un premier temps au niveau 'place' puis au niveau région. Cette dernière expansion, illustrée sur la figure 16, montre le nombre raisonnable de résultats a ici été atteint. Notons cependant que ce résultats est même dépassé car nous obtenons avec cette dernière expansion 123 résultats supplémentaires, auxquels s'ajoute les précédents (6 autres réponses).
Nous devons ainsi introduire à ce niveau une méthode de réduction de résultats que nous détaillons dans la section 4.

4 Réduction

Après avoir présenté une méthodologie dans les sections précédentes visant à augmenter le nombre de résultats provenant d'une requête utilisateur, nous proposons dans cette section de définir une méthode permettant de faire une réduction de résultats. Comme évoquée à la fin de la section 3.4, l'étape de réduction peut également intervenir lorsqu'une expansion a produit un nombre trop important de résultats. Nous distinguons deux approches possibles afin de mettre en place une telle réduction. D'une part une réduction par tirages aléatoires parmi les résultats de la requête, ou bien un tirage aléatoire stratifié.

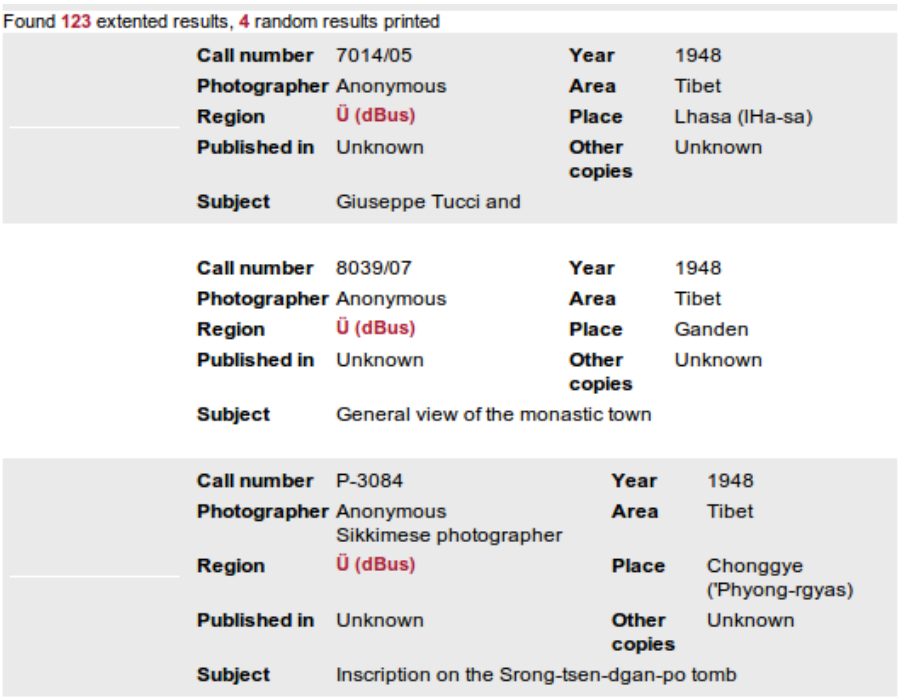

FIGURE 16 – Exemple d'expansion avec choix de critères

4.1 Réduction par tirages aléatoires

La réduction par tirage aléatoire consiste simplement en la sélection, parmi les résultats obtenus, du nombre nécessaire de résultats afin d'obtenir un nombre raisonnable. Les avantages de cette approche sont bien entendu la simplicité de mise en place, ainsi que sa rapidité d'exécution. La figure 16 montre un exemple de résultats qui ont été sélectionnés aléatoirement afin de converger vers un nombre raisonnable de résultats. Sur cet exemple, 4 résultats ont été tirés aléatoirement parmi 123.
Notons que cette méthode est applicable à la fois lors d'une réduction de résultats fournis par une requête utilisateur, mais peut également être employée afin de réduire un expansion.
Néanmoins, cette méthode fournie des résultats peu pertinents en ce sens qu'elle peut par exemple sélectionner des résultats du même photographe en s'appuyant sur la collection Tucci, alors qu'il y avait parmi les résultats des photographies de tous les photographes. Ainsi, nous proposons dans la section suivante une méthode originale permettant de tirer des résultats aléatoirement parmi un ensemble de réponses, mais en stratifiant au préalable nos résultats.

4.2 Réduction par tirages aléatoires stratifiés

L'idée de la stratification des résultats est la suivante. Plutôt que de proposer à un utilisateur des résultats aléatoires, le principe est de proposer des résultats caractérisant au mieux les différents thèmes ou domaines présents dans les résultats de la requête utilisateur (ou de l'expansion à réduire). Notons que la notion de thème ou domaine correspond aux critères évoqués dans les sections précédentes.
La mise en place d'une telle méthode revient alors à répondre à la question : comment sélectionner ces différents domaines et leurs instances ?

La sélection des domaines pertinents

Cette première étape consiste à identifier parmi les différents critères spécifiés dans la requête, quels seront les plus significatifs. Par exemple, considérons la collection Tucci avec la requête : photographies d'Eugenio Ghersi en Inde. Cette requête retourne trop de résultats (1126). Nous devons donc réduire le nombre de réponses. Les domaines candidats afin d'être jugés pertinents sont ici les critères pour lesquels l'utilisateur n'a pas spécifié de contrainte (l'année, la région, le lieu et le sujet). Notons que dans le cas de données hiérarchiques, nous privilégions le niveau le plus élevé comme dans ce cas la région.

Dès lors, les domaines sont dit pertinents si la répartition de leurs instances sont homogènes. Nous considérons comme homogène un domaine où l'instance la plus présente n'excède pas un certain seuil α * *le nombre total de réponses fournies par la requête initiale*. Avec notre précédent exemple, l'instance la plus représentative des années est la classe de dates '*entre 1926 et 1932*' totalisant 675 réponses. En fixant $\alpha = 2/3$, les 675 réponses représentent moins que les 2/3 de 1126. Ainsi, ce critère date peut être considéré comme domaine pertinent car sa répartition parmi les réponses est homogène.

La seconde étape de la méthode de réduction proposée est de ne conserver parmi les domaines pertinents les deux plus significatifs, donc les deux plus homogènes en termes de répartition d'instances. Dans notre exemple, il s'agit des critères '*date*' et '*région*'.

La sélection des instances

Après avoir sélectionnés deux domaines significatifs, il reste à sélectionner des instances de ces domaines afin de présenter à l'utilisateur un nombre raisonnable de résultats pertinents en fonction de sa requête initiale. Cette étape peut se décomposer de la manière suivante :

1. Construire une matrice de co-occurrences avec les deux domaines significatifs et leurs nombres d'instances.
2. Ne conserver que les n plus fréquentes instances de cette matrice, par domaine.
3. Sélectionner aléatoirement des réponses représentatives parmi les instances restantes.

Afin d'illustrer cette étape, nous nous appuyons sur l'exemple précédent à savoir : photographies d'Eugenio Ghersi en Inde. Nous avons dans la précédente étape sélectionné deux domaines pertinents : les dates et les région. La matrice de co-occurrences est dès lors proposée dans la figure 17. En fixant pour

	1926-1932	*1933*	*1935*	*Second World War*	*After War*	sum
Bengal	24	8	28	20	34	*114*
Kashmir	102	46	27	4	34	***213***
Kinnaur	79	7	25	13	12	*136*
Kulu	0	11	5	18	41	*75*
Kumaon	48	8	4	19	30	*109*
Ladakh	89	28	20	29	11	***177***
Rubshu	67	27	37	6	2	*139*
Sikkim	12	38	39	11	46	*146*
Spiti	254	17	31	30	13	***345***
sum	***675***	*190*	***216***	*150*	***223***	

FIGURE 17 – Matrice de co-occurrences entre les domaines Date et Région de l'Inde

cet exemple la valeur de n à 3, les instances sélectionnées sont pour les dates '1926 à 1932', '1935', 'After War' et pour les régions de l'Inde 'Kashmir', 'Ladakh', 'Spiti'. Nous pouvons alors extraire ces trois instances pour chaque domaine et obtenir une nouvelle matrice présentée dans la figure 18. La dernière étape consiste finalement à sélectionner parmi les 9 instances possibles des occurrences aléatoirement.

	1926-1932	1935	After War
Kashmir	102	27	34
Ladakh	89	20	11
Spiti	254	31	13

FIGURE 18 – Matrice de co-occurrences entre les domaines Date et Région de l'Inde

Algorithme

Ce paragraphe présente l'algorithme de la réduction par tirages aléatoires stratifiés.

```
#####################################################################
ReductionStrat(Results, Query, Nb_raisonnable, alpha, n)

 ENTREE : Results : liste des résultats d'une requête,
          Query  : la requête utilisateur,
          Nb_raisonnable : le nombre de résultats à atteindre,
          alpha : Taux qui une fois multiplié par le nombre total
                  de résultats doit être inférieur au nombre
                  d'occurrences de l'instance la plus représentative
                  d'un domaine donné
          n : Le nombre d'instances à conserver dans la matrice
              finale de co-occurrences
 SORTIE : Results : les résultats réduits

 Fields <- SelectFieds(Query, Results, alpha)

 If (Size(Fields) > 1)
     Matrix <- BuildMatrix(Fields, Results)
     Matrix <- XtractMatrix(Matrix, n)
     Results <- Rand_Select(Matrix, Results, Nb_raisonnable)
 Else If
     Results <- Reduction(Results, Query, Nb_raisonnable)
 End If

 Return Results

#####################################################################
SelectFields(Query, Results, alpha)

 ENTREE : Query
          Results
          alpha
 SORTIE : Fields : les deux domaines les plus pertinents

 AllFields <- XtractFields(Query)

 Min[0] <- size(Results)
 Min[1] <- size(Results)
 # initialisés par le nombre total de résultats
```

```
 Foreach Field of AllFields
     max <- 0
     Instances <- XtractInstances(Field, Result)
     Foreach Field of AllFields
         max <- maximum(max, size(Field))
         # compte le nombre d'occurrence d'une instance d'un domaine
         # comme 1933 pour date, et retourne le maximum entre cette
         # valeur et la précédente
     End Foreach
     If (max < alpha * size(Results))
         # Le domaine devient candidat
         If (max < Min[0])
             Min[0] <- max
             Fields[0] <- Field
         Else If
             If (max < Min[1])
                 Min[1] <- max
                 Fields[1] <- Field
             End If
         End If
         # test si le candidat est parmi les deux meilleurs
     End if
 End Foreach

 Return Fields
```

```
######################################################################
BuildMatrix(Fields, Results)

 ENTREE : Fields
          Results
 SORTIE : Matrix : une matrice contenant le nombre d'occurrences
                   existant dans Result avec les critères imposés
                   par les domaines (par exemple 1933 et Inde, 1935
                   et Inde, 1933 et Tibet, etc. si les deux domaines
                   sont Area et Date)

######################################################################
XtractMatrix(Matrix, n)

 ENTREE : Matrix : la matrice résultante de BuildMatrix
          n : Le nombre d'instances à conserver dans la matrice
              finale de co-occurrences
 SORTIE : Matrix : matrice de co-occurrences contenant les n instances
                   les plus fréquentes pour chacun des deux domaines
```

Le principe de cette fonction est simplement de compter le nombre d'occurrences de chaque instances des deux domaines et de ne conserver que les n plus fréquentes.

```
#####################################################################
Rand_Select(Matrix, Results, Nb_raisonnable)

 ENTREE : Matrix : la matrice résultante de XtractMatrix
          Results : liste des résultats d'une requête,
          Nb_raisonnable : le nombre de résultats à atteindre
 SORTIE : Results : les résultats finaux

 Nb <- 0

 Nb_Row <- RowSize(Matrix) # compte le nombre de lignes

 Nb_Column <- ColumnSize(Matrix) # compte le nombre de colonnes

 While Nb < Nb_raisonnable
     result <- Matrix(random(1, Nb_Row) , random(1, Nb_Column))
     If not exists Results(result)
         Push(Results, result)
         Nb <- Nb + 1
     End If
 End While

Le principe de cette fonction est de sélectionner dans la matrice
issue de XtractMatrix un nombre raisonnable de résultats,
en tirant de manière homogène parmi les couples de la matrice.

#####################################################################
Reduction(Results, Query, Nb_raisonnable)

 ENTREE : Results : liste des résultats d'une requête,
          Query  : la requête utilisateur,
          Nb_raisonnable : le nombre de résultats à atteindre
 SORTIE : Results : les résultats finaux tirés aléatoirement

 Cette fonction effectue un simple tirage de Nb_raisonnable résultats
 aléatoire parmi les résultats de la requête initiale. Elle
 correspond à la réduction par tirages aléatoires.
```

5 Conclusion et perspectives

Synthèse

Nous avons présenté dans ce rapport une méthodologie visant à proposer un nombre raisonnable de résultats à un utilisateur de système de recherche d'informations (SRI). Deux problématiques ont alors été résolues.

1. La requête utilisateur ne produit pas assez de résultats. Nous avons alors proposé une méthode d'expansion s'adaptant aux différents types de données manipulées dans le SRI.
2. La requête utilisateur produit trop de résultats. Nous avons alors introduit un processus de réduction aléatoire stratifié, qui vise à proposer à l'utilisateur un échantillon représentatif des résultats obtenus à partir de sa requête.

Cette méthodologie a été expérimentée avec la collection Tucci. Nous avons pu vérifier l' efficacité des méthodes proposées et leur intérêt. Ce travail peut encore être étendu par l'ajout de perspectives comme celles présentées dans les paragraphes suivants.

L'expansion à base de listes
Une première perspective à ces travaux serait d'améliorer l'algorithme d'expansion en se fondant sur la notion de liste. L'idée est de se passer des différentes approches spécifiques aux données textuelles, temporelles, spatiales, etc. Dès lors, l'objectif serait d'avoir des modules externes qui calculs une liste de priorité de critères à étendre. Finalement, l'étape d'expansion consisterait à suivre les recommandations fournies dans ces différentes listes, propres à chaque critère.

Gestion du multicritère
Pour aller plus loin dans le processus d'expansion, il serait intéressant de pouvoir réaliser une expansion sur plusieurs critères simultanément. Par exemple, imaginons une requête ou l'ensemble des expansions pour chacun des critères n'apporte aucun résultat supplémentaire. Il serait alors intéressant, plutôt que de ne pas faire d'expansion ou que de faire une expansion sur l'ensemble des critères, de faire une expansion à deux ou trois critères. Appliqué à la collection Tucci, nous pourrions avoir une expansion sur l'année et la région simultanément.

La personnalisation des requêtes
Une autre perspective intéressante serait, lors d'une étape de réduction dans un premier temps, de permettre à un utilisateur de personnaliser les résultats. En effet, une étape de réduction intervient lorsque la requête initiale a retournée un nombre trop important de résultats. Une fois la réduction appliquée, un nombre raisonnable de résultats homogènes est proposé à l'utilisateur. C'est ici qu'interviendrait le processus de personnalisation en permettant à l'utilisateur de sélectionner une ou plusieurs réponses et de reformuler la requête de l'utilisateur en prenant en compte ces choix. La requête générée fournie alors des résultats personnalisés à l'utilisateur et également des résultats plus précis en limitant le nombre de critères. Nous pourrions également à terme proposer un système d'authentification sur le site web contenant le SRI afin de personnaliser d'avantage les requêtes utilisateurs en fonction de ces précédentes visites d'une part (recommandation basée sur le contenu) et également en fonction des préférences d'utilisateurs ayant un profil similaire (recommandation collaborative).

Références

[CWNM02] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Probabilistic query expansion using query logs. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 325–332, New York, NY, USA, 2002. ACM.

[GGM92] Terry Gaasterland, Parke Godfrey, and Jack Minker. An overview of cooperative answering, 1992.

[GMV99] N. Guarino, C. Masolo, and G. Vetere. OntoSeek : content-based access to the web. *Intelligent Systems and Their Applications, IEEE*, 14(3) :70–80, 1999.

[JZH09] Shen Jiang, Sandra Zilles, and Robert Holte. Query suggestion by query search : A new approach to user support in web search. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, pages 679–684, Washington, DC, USA, 2009. IEEE Computer Society.

[LWC06] Hsi-Ching Lin, Li-Hui Wang, and Shyi-Ming Chen. Query expansion for document retrieval based on fuzzy rules and user relevance feedback techniques. *Expert Syst. Appl.*, 31(2) :397–405, 2006.

[Mil85] George A. Miller. Wordnet : a dictionary browser. In *the First International Conference on Information in Data*, 1985.

[MM00] Rada Mihalcea and Dan Moldovan. An iterative approach to word sense disambiguation. In *In Proceedings of FLAIRS-2000*, pages 219–223, 2000.

[NHT+07] Apostol (Paul) Natsev, Alexander Haubold, Jelena Tešić, Lexing Xie, and Rong Yan. Semantic concept-based query expansion and re-ranking for multimedia retrieval. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 991–1000, New York, NY, USA, 2007. ACM.

[PKM07] Pallavi Palleti, Harish Karnick, and Pabitra Mitra. Personalized web search using probabilistic query expansion. In *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, WI-IATW '07, pages 83–86, Washington, DC, USA, 2007. IEEE Computer Society.

[QF93] Yonggang Qiu and Hans-Peter Frei. Concept based query expansion. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, pages 160–169, New York, NY, USA, 1993. ACM.

[Sal71] Gerard Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[Sal86] Gerard Salton. On the use of term associations in automatic information retrieval. In *Proceedings of the 11th coference on Computational linguistics*, COLING '86, pages 380–386, Stroudsburg, PA, USA, 1986. Association for Computational Linguistics.

[VCLV08] Anthony Ventresque, Sylvie Cazalens, Philippe Lamarre, and Patrick Valduriez. Improving interoperability using query interpretation in semantic vector spaces. In *Proceedings of the 5th European semantic web conference on The semantic web : research and applications*, ESWC'08, pages 539–553, Berlin, Heidelberg, 2008. Springer-Verlag.

[Voo94] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 61–69, New York, NY, USA, 1994. Springer-Verlag New York, Inc.