Retrait d'un document de la bibliothèque ou modification de sa description

Guillaume Lafon et Nicolas Gast

Présentation du problème

On considère une bibliothèque électronique, dans laquelle à chaque document est associé un certain nombre de mots clefs. Ces mots clefs sont organisés en une taxonomie et on note $<$ la relation d'ordre sur cette taxonomie.

Un document est consitué d'une adresse représentant le corps du document et d'une liste de documents que l'on appelera fils du document. Un document sans fils est appelé document atomique.

À chaque document est associée une liste de mots de la taxonomie qui décrivent le document. Cette liste est obtenue à partir des informations du documents fournies par l'auteur. Elle dépend aussi des fils du documents. Pour des raisons de concision de la description, celle-ci est *minimale*, c'est à dire que si elle contient deux mots t et t' et que $t < t'$, alors seul t est gardé, t' n'étant pas assez pertinent.

Pour constuire les liens entre document et taxonomie, l'algorithme de calcul de la description réduite est utilisé :

- il prend le produit des description des documents fils
- il réduit les éléments du produit en utilisant le plus petit ancètre commun des mots de la taxonomie.
- il prend l'union des élements du produit réduit et de la description propre au document et la réduit, c'est à dire enlève les t' tels que $t < t'$

Ce qui se traduit par :

si $d = (D_1, D_2, \ldots, D_n)$
$RedDesc(d) := reduce[DescrPropre(d) \cup ReduProduit(D_1, D_2, \ldots, D_n)]$

Cela nous mène directement au problème du retrait d'un document, qui est similaire à celui de la modification de sa description.

En effet, la suppression d'un document, tout comme la modification de sa description entraine des suppressions de mots dans la description d'autres documents.

Par exemple, si $d = (D_1, D_2, \ldots, D_n)$, la suppression de D_1 entraine une modification de $ReduProduit(D_1, D_2, \ldots, D_n)$ et par récurence de la même modification sur tous les ancètres de D_1.

Dans la phase de suppression d'un document, il y a une étape préalable qui est de supprimer les liens vers ce document qui apparaissent entre la taxonomie et le document ou entre un autre document et celui-ci mais cette étape est peu couteuse et ne peut être améliorée. Nous nous intéresserons donc à l'étape

suivante, c'est à dire la modification de la description d'un fichier.

1 L'algorithme naïf

Cet algorithme est le premier auquel on pense en réfléchissant au problème. Les premières étapes semblent à peu près inévitables et automatiques. Soit D le document à supprimer. On commence par supprimer du catalogue toutes les paires faisant intervenir D, ce qui se fait par une simple requête sur la base de données constituant le catalogue.

Il faut ensuite modifier les liens faisant intervenir des documents dont D est une composante. Pour chaque tel document, on supprime donc toutes les paires du catalogue le faisant intervenir, puis on reconstruit sa description réduite pour mettre à jour la base. Plus précisément, on remonte dans le graphe des documents de la manière suivante :

- on sélectionne les documents dont D est un fils (c'est-à-dire ses pères dans le graphe des documents de la bibliothèque)
- on recalcule la description réduite de chacun de ces documents
- on applique à nouveau l'algorithme à chacun des pères

Cet algorithme correspond à un parcours récursif dans le graphe des ancêtres de D, le point important étant évidemment qu'on a besoin que la description réduite des fils d'un document soit bonne pour calculer celle du document lui-même. On peut vérifier au passage qu'aucun cycle n'intervient dans dans le graphe des documents, ce qui ferait boucler l'algorithme, mais ne devrait normalement pas arriver (et devrait d'ailleurs plutôt être vérifié au moment de la création du document).

Dans le cas d'une modification, on peut appliquer essentiellement le même algorithme, seule la première étape est légèrement modifiée.

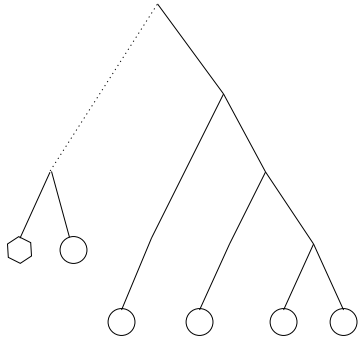
2 Amélioration de l'algorithme

Cet algorithme présente bien entendu le gros inconvénient de devoir appliquer plusieurs fois l'algorithme de calcul de la description réduite, qui est assez lourd, pour chaque modification. En effet, chaque réduction est exponentiel en fonction du nombre de mot dans la description des fils :

si $d = (D_1, d_2, \ldots, D_n)$ où $card(Description(D_i)) = 2$, le nombre de couples où il faut calculer le plus petit ancètre (que nous appelerons PPA par la suite) est 2^n.

En fait, cet algorithme recalcule entièrement toute une partie de la base de données. Nous avons pensé améliorer l'algorithme en essayant de ne pas recalculer toutes les réductions à chaque fois, en remarquant que $PPA(d) = PPA(D_1, PPA(D_2, \ldots, D_n))$ mais le problème est de calculer $PPA(D_2, \ldots, D_n)$ de manière efficace (un calcule de PPA coute la somme des distances entre le PPA et les fils)

Le problème est qu'il y a une dépendance forte entre les différents arguments :



En effet, comment distinguer si il y a deux fils parmis l'arbre de gauche, dans ce cas le PPA ne va pas changer où s'il n'y en a qu'un, dans ce cas le PPA va fortement changer ?

Conclusion

Notre conclusion est qu'il parait difficile d'améliorer cet algorithme. Ainsi nous pensons que le mieux est :

- soit d'espérer que les documents ont un nombre de fils limité, ce qui semble assez cohérent
- soit de ne faire tourner notre l'algorithme que toutes les n modifications (avec n grand)

On peut penser que la base de la bibliothèque n'a pas besoin d'être mise à jour en temps réel. C'est pourquoi on peut se contenter de faire la première étape de notre algorithme en temps réel et de ne faire la deuxième qu'une fois par jour.