Université de Neuchâtel

Institut Interfacultaire d'Informatique

Apprentissage en recherche d'informations

Thèse

présentée à la Faculté des Sciences

pour obtenir le grade de Docteur ès Sciences

par

Dana VRAJITORU

Directeur de thèse :

Prof. Dr Pierre-Jean ERARD, Université de Neuchâtel, Suisse

Codirecteur de thèse :

Prof. Dr Jacques SAVOY, Université de Neuchâtel, Suisse

Experts:

Prof. Dr Dan CRISTEA, Université de Iasi, Roumanie

Prof. Dr Jacques PASQUIER, Université de Fribourg, Suisse

Mai 1997

IMPRIMATUR POUR LA THÈSE

Apprentissage en recherche d'informations

de Mme Dana Vrajitoru

UNIVERSITÉ DE NEUCHÂTEL

FACULTÉ DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel sur le rapport des membres du jury,

MM. P.-J. Erard (directeur de thèse),
J. Savoy (co-directeur de thèse)
J. Pasquier (Fribourg) et D. Cristea (Iasi, Roumanie)

autorise l'impression de la présente thèse.

Neuchâtel, le 4 juin 1997

Le doyen:

R. Dändliker

R. Dändliker

Table de matières

Introduction

En cette fin de siècle, le développement des connaissances s'accroît de plus en plus, le volume d'information dont on dispose ne cesse d'augmenter rendant une gestion manuelle trop onéreuse, voir impossible. Ainsi, il devient important d'élaborer, de concevoir et de mettre au point des systèmes informatisés et automatisés capables de manipuler les connaissances sur support électronique. Parmi les formes de représentation de la connaissance humaine, les textes présentent plusieurs avantages, parmi lesquels la compacité et la simplicité.

Dans le cadre de cette nécessité générale, le but de la recherche d'informations est de dépister, dans une collection d'objets, ceux qui répondent à une requête. Le plus souvent, les systèmes de recherche d'informations permettent d'organiser et d'exploiter une grande quantité de documents, afin qu'un utilisateur retrouve l'information souhaitée dans un temps convenable et avec un minimum d'efforts. Plus particulièrement, on cherche à représenter et à manipuler de grandes collections de documents (plus de 1'000 documents) pour lesquelles des modèles robustes doivent être mis au point.

Un modèle de recherche d'informations doit spécifier dans le détail comment les documents et les requêtes sont représentés, comment s'effectue l'appariement entre les documents et la requête. Afin d'évaluer la performance du système, il est nécessaire que l'usager spécifie quels documents, parmi tous ceux qu'il a reçu comme réponse, correspondent effectivement à son besoin, ce qui nous donne des jugements de pertinence (voir figure I).

Le développement actuel de l'informatique et, en particulier de l'intelligence artificielle, s'oriente vers la notion d'apprentissage automatique. Un tel algorithme utilise les succès et les échecs rencontrés lors de son utilisation afin de s'améliorer. Il doit être capable d'inférer de nouvelles connaissances à partir des données dont il dispose et de s'adapter plus facilement à des besoins changeants.

La recherche d'informations est un domaine particulièrement propice à l'apprentissage, car cet aspect devient non seulement utile, mais nécessaire avec l'accroissement de la taille des collections utilisées. Ainsi, il n'est plus possible d'envisager une solution statique à ce problème, car le sens attaché aux termes peut varier ou s'enrichir avec le temps. L'indexation initiale peut paraître obsolète quelques années plus tard. La notion d'apprentissage qui apparaît ainsi naturellement, constitue le sujet de cette thèse.

Dans ce sens, l'objectif concret de cette thèse est d'étudier les possibilités d'utiliser les jugements de pertinence pour l'apprentissage afin d'améliorer ainsi la performance d'un système de recherche d'informations, comme le démontre aussi la figure 1.

Plusieurs chercheurs se sont déjà intéressés à une partie de ce sujet et ont proposé certaines techniques basées sur la *rétroaction* ou *relevance feedback*.

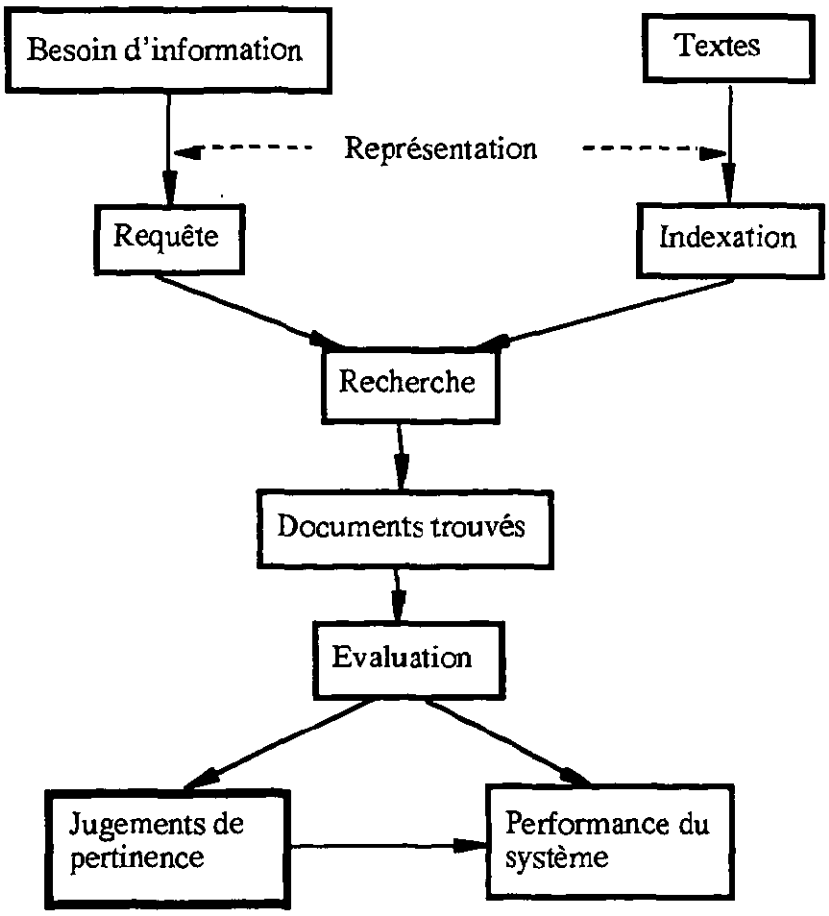

Fig. 1 Objectif de la thèse

Cependant, les derniers modèles des informaticiens impliqués dans l'apprentissage les rapprochent des chercheurs en neurosciences ou en systèmes naturels. Dans cette convergence, on peut citer le développement des réseaux de neurones, du concept d'acteur, du concept de système autonome et de la réalité virtuelle. La recherche d'informations vient tirer profit, comme d'autres domaines, de la découverte de ces algorithmes.

Inspiré des mécanismes de reproduction des organismes vivants, l'algorithme génétique ou d'adaptation génétique s'avère une méthode intéressante qui semble pouvoir répondre à beaucoup de problèmes différents. Il fait partie de la classe des algorithmes probabilistes qui utilisent l'intervention du hasard à certains moments et se comportent différemment lorsqu'on l'applique deux fois sur les mêmes données. Grâce à son concept original et attractif, et à sa grande flexibilité, cet algorithme a été utilisé également en recherche d'informations. Dans ce sens, nous avons exploré plusieurs applications de l'algorithme génétique à la recherche d'informations et étudié certains aspects de son fonctionnement liés à ce domaine.

Dans différents domaines, on dispose de démonstrations mathématiques rigoureuses pour confirmer une théorie. En ce qui concerne la recherche d'informations, les techniques utilisées sont souvent fondées sur des observations empiriques, ce qui rend la comparaison

des méthodes et la validation des hypothèses plus difficiles. L'incertitude a plusieurs sources : l'indexation, qui est une représentation approximative du document, la formulation de la requête, qui reflète souvent le besoin d'information d'une manière ambiguë ou trop générale, et le fait que deux utilisateurs peuvent juger différemment le contenu d'un texte par rapport à la même requête [Saracevic 75].

Pour faire face à ces difficultés, plusieurs méthodes d'évaluation ont été conçues. Dans les recherches présentées dans cette thèse, nous avons prêté une attention particulière à une évaluation juste et formelle. Ainsi, pour chaque méthode, nous avons estimé le taux d'erreur de la manière la plus juste possible, afin de pourvoir juger de l'efficacité réelle des algorithmes.

Plan de la thèse

La thèse est structurée sur 6 chapitres:

Le premier chapitre introduit les modèles de recherche d'informations utilisés dans cette thèse, ainsi qu'une classification des algorithmes d'apprentissage étudiés.

Le deuxième chapitre traite un sujet très important qui concerne l'évaluation des stratégies d'apprentissage. L'adaptation à la recherche d'informations de la méthode d'évaluation appelée *leaving one out* , décrite dans ce chapitre, constitue une approche originale.

Le troisième chapitre présente une réévaluation de plusieurs stratégies récurrentes avec la méthode du *leaving one out,* et ce, sur des collections de plus grande taille que lors de recherches antérieures. A la fin du chapitre, nous introduisons une stratégie originale d'apprentissage récurrent.

Le quatrième chapitre introduit brièvement la méthode du *relevance feedback* que nous avons évaluée dans cette thèse à des fins de comparaison avec les autres algorithmes d'apprentissage.

Le cinquième chapitre est consacré à l'étude de la pertinence de l'application de l'algorithme génétique en recherche d'informations, sujet pour lequel nous nous sommes inspirés des recherches de Gordon (1988, 1991). Notre application de ce paradigme au domaine de recherche se distingue des recherches antérieures par la taille des collections utilisées, par la construction de la population initiale, par l'utilisation d'une indexation pondérée et par une évaluation plus réaliste.

Le dernier chapitre présente un complément au chapitre cinq, qui consiste dans l'analyse et l'amélioration de l'algorithme génétique en recherche d'informations. Nous présentons une analyse mathématique et pratique des mécanismes de fonctionnement de cet algorithme et, pour finir, nous introduisons une nouvelle opération de croisement, plus performante que l'opération classique.

1. Modèles de recherche d'informations et apprentissage

Le présent chapitre introduit deux modèles de recherche d'informations que nous avons utilisés pour nos recherches : le modèle booléen et le modèle vectoriel. Les sections de ce chapitre correspondent à une brève introduction à l'indexation automatique, à l'appariement, à quelques méthodes traditionnelles d'évaluation et à la notion d'apprentissage.

Le modèle booléen, basé sur la logique binaire, est encore assez répandu dans les systèmes commerciaux. Parmi ses avantages, on peut citer la simplicité, la facilité de représentation et la rapidité de la recherche. Plus performant que ce modèle, le modèle vectoriel présente une grande importance du point de vue historique et est très répandu dans les laboratoires de recherche. Son importance est également associée au fait que l'usager peut décrire son besoin d'information en langue naturelle et non pas uniquement à l'aide des opérateurs logiques (ET, OU, ET NON).

1.1. Indexation automatique

L'indexation est un procédé qui a pour but de construire une représentation des documents et des requêtes de telle manière que leurs traits sémantiques importants soient mis en évidence. Généralement, l'indexation peut se faire par des attributs extemes (objectifs) (nom de l'auteur, nombre de pages, éditeur, lieu de publication, date de publication), ou par des attributs internes (subjectifs) qui s'intéressent au contenu sémantique.

L'indexation peut être manuelle ou automatique, avec un vocabulaire contrôlé ou libre. L'indexation manuelle, éventuellement avec un vocabulaire contrôlé, est encore très courante car les services commerciaux publient également des journaux et répertoires [Cleverdon 84] [Blair 90].

Les techniques simples d'indexation automatique sont aussi performantes que l'indexation manuelle [Salton 69], [Salton 72]. Certains auteurs préfèrent une indexation manuelle complétée par une indexation automatique [Svenonius 86].

L'indexation automatique comprend généralement les opérations suivantes :

1. Repérer les mots (suite de lettres et/ou de chiffres).
2. Éliminer les mots très courants, peu ou pas porteurs d'information [Fox 90], [Rijsbergen 79].
3. Supprimer les séquences terminales [Lovins 68], [Paice 90], [Porter 80]. Dans nos évaluations, nous avons choisi l'algorithme de Porter.
4. Représenter par un poids w_{ij} chaque "concept" ou terme simple t_j, $j = 1,2,\ldots,t$, dans chaque document d_i, $i = 1,2,\ldots,n$, en utilisant les formules suivantes :

$w_{ij} = 0$ si le terme t_j n'apparaît pas dans le document d_i, pour les deux modèles,

$$w_{ij} = 1 \quad \text{si le terme } t_j \text{ apparaît dans le document } d_i \tag{1}$$

pour l'indexation binaire, et :

$$w_{ij} = ntf_{ij} \cdot nidf_j \quad \text{avec} \quad ntf_{ij} = \frac{tf_{ij}}{\max_k tf_{ik}} \quad \text{et} \quad nidf_j = \frac{\log(n) - \log(df_j)}{\log(n)} \tag{2}$$

pour le modèle vectoriel.

Dans l'équation (2), suggérée par Salton [Salton 83] et Turtle [Turtle 90], tf_{ij} indique le nombre d'occurrences du terme t_j dans le document d_i, et df_j indique le nombre de documents dans lesquels le terme t_j apparaît.

La pondération des termes tient compte de la fréquence (normalisée) des termes dans les documents mais aussi de la répartition des termes dans la collection. Plus précisément, un terme fréquent dans un document aura un poids relativement plus important dans la description de ce document. Ce poids sera diminué si le terme est fréquent dans la collection car, dans ce cas, il est peu utile pour distinguer les documents les uns des autres.

1.2. Représentation et appariement

Le **modèle booléen** que nous avons utilisé représente les documents comme des ensembles de termes et les requêtes sous forme d'arbres binaires selon la définition contenue dans la figure 2.

arbre = terme
arbre = arbre et arbre
arbre = arbre **ou** arbre
arbre = **et non** arbre

Fig. 2 Arbre logique binaire

Le document d_j est extrait du corpus si son représentant satisfait la requête.

La classification des documents dépistés par le système pour une requête donnée n'est pas basée sur une mesure de distance entre les documents et la requête, mais sur des attributs externes (par exemple, par la date du plus récent au plus ancien, pratique courante dans les systèmes commerciaux).

Le **modèle vectoriel** [Salton 71] représente les requêtes et les documents comme des points (ou des vecteurs) dans un espace à t dimensions (figure 3).

Il existe plusieurs mesures qui permettent de calculer la similarité entre le vecteur du document et celui de la requête. Dans nos recherches, nous avons utilisé le cosinus de l'angle entre le vecteur document et le vecteur requête ([Salton 71], p. 163). Si d_i et q représentent

les vecteurs du document i, respectivement de la requête, alors la similarité entre le document et la requête est donnée par :

$$sim(d_i, q) = \frac{\sum_{k=1}^{t} w_{ik} \cdot w_{qk}}{\sqrt{\sum_{k=1}^{t} w_{ik}^2 \cdot \sum_{k=1}^{t} w_{qk}^2}}$$

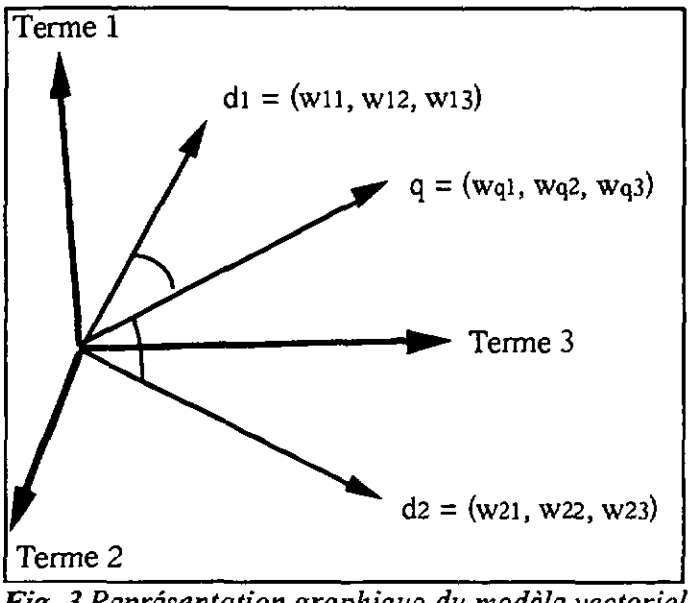

Fig. 3 Représentation graphique du modèle vectoriel

où w_{ij} est le poids du terme t_j dans la requête Q et il est calculé selon l'équation (2). Par exemple, soient le document et la requête présentés dans le tableau 1 :

***Tab. 1** Petit exemple d'appariement*

Document \ Terme	t_1	t_2	t_3
d_1	0.2	0.3	0.5
q	0.1	0	0.2

Alors $sim(d_1, q) = \frac{0.2 \cdot 0.1 + 0.5 \cdot 0.2}{\sqrt{\left(0.2^2 + 0.3^2 + 0.5^2\right) \cdot \left(0.1^2 + 0.2^2\right)}} = 0.217.$

Les documents sont classés selon leur degré de similarité avec la requête, de la similarité la plus forte à la plus faible et en cas d'égalité, par date de publication.

I.3 Collections utilisées

L'évaluation des différents modèles d'apprentissage présentés dans cette thèse a été effectuée sur deux collections : **CACM** (articles du journal Communications of the Association for Computing Machinery) et **CISI** (Collection of the Institute for Scientific Information).

Pour les deux collections, nous disposons de requêtes munies de leurs jugements de pertinence qui nous permettent d'évaluer la performance des systèmes. Le tableau 2 illustre la composition des deux collections :

Tab. 2 Statistiques des collections

Collection	CACM	CISI
Nombre de documents	3'204	1'460
Nombre de requêtes	50	35
Nombre de termes uniques	5'935	5'823
Nombre d'occurrences	187'670	174'913
Nombre moyen de termes par requête	11.24	7.43
Nombre moyen de termes communs à deux requêtes	0.52 (4.63%)	1.10 (14.75%)
Nombre moyen de documents pertinents par requête	15.84	49.77
Nombre moyen de documents pertinents communs à deux requêtes	0.31 (1.94%)	2.38 (4.78%)

I.4 Méthodes traditionnelles d'évaluation

Cette section présente les principales mesures de performance utilisées dans notre recherche pour l'évaluation des algorithmes :

- les notions de précision et de rappel,
- la notion de précision à 11 points de rappel.

La précision et le rappel sont les mesures les plus utilisées dans la recherche d'informations pour évaluer à quel point une liste de documents répond au besoin d'information exprimé dans une requête.

La **précision** est la proportion de documents pertinents extraits par rapport à l'ensemble des documents retournés :

$$précision = \frac{pertinent \cap retourné}{retourné}$$

Le **rappel** est le rapport entre le nombre de documents pertinents extraits et le nombre de documents pertinents pour la requête donnée :

$$rappel = \frac{pertinent \cap retourné}{pertinent}$$

Par exemple, si la réponse à une requête est constituée de 10 documents, parmi lesquels 3 sont pertinents, et si nous savons que dans toute la collection il y a 7 documents pertinents pour cette requête, la précision est de 3/10 = 0.33 et le rappel est égal à 3/7 = 0.43.

Dans la pratique, et spécifiquement au cours de cette thèse, nous sommes confrontés à la comparaison entre deux méthodes d'apprentissage ou d'indexation différentes. Pour que l'algorithme A soit meilleur que l'algorithme B, il faut que la précision et le rappel de A soient supérieurs à ceux de B ($A > B$ si $p_A > p_B$ et $r_A > r_B$). Cette contrainte est difficile à satisfaire, et une mesure de comparaison unique serait plus utile.

Dans le sens des ces arguments, nous avons choisi la **précision à 11 points de rappel**, comme base de comparaison de tous les algorithmes testés. En fait, cette méthode largement répandue consiste à fixer le rappel à 11 valeurs (0.0, 0.1, 0.2, ..., 0.9 et 1.0) et à calculer la moyenne des précisions correspondantes à ces 11 valeurs.

En réalité, étant donné une liste de documents retournés par le système pour une requête, nous pouvons calculer la précision et le rappel par rapport à toute coupure de la liste après un nombre quelconque de documents. Imaginons, par exemple, que pour une requête il y ait 10 documents pertinents et que le système retourne la liste de 5 éléments du tableau 3. Alors, en coupant la liste après chaque document retrouvé, du premier jusqu'au dernier, nous obtenons les valeurs de la précision et du rappel illustrées dans le tableau 3. Ainsi, pour une valeur de rappel de 0.1, on retrouve les valeurs de précision {1/3, 1/4, 1/5}. De manière générale, à une valeur fixe du rappel correspondent plusieurs valeurs de la précision. Pour calculer la précision à 11 point fixes de rappel, nous avons besoin d'une interpolation de ces valeurs.

***Tab. 3** Exemple de liste retournée*

Document	Pertinence	Précision	Rappel
1	non	0	0
2	non	0	0
3	oui	1/3	0.1
4	non	1/4	0.1
5	non	1/5	0.1

Sur la base d'études empiriques [Salton 71] ou suivant un modèle théorique [Gordon 89], il a été établi qu'il existe une relation inverse entre la précision et le rappel. Ce principe général a été utilisé pour construire des interpolations entre les points (rappel, précision) afin de pouvoir obtenir une seule valeur de la précision pour chaque valeur de rappel. Nous avons choisi l'interpolation néo-Cleverdon, qui établit la courbe monotone décroissante unissant par le "haut" les points existants.

Finalement, pour décider si un algorithme est meilleur qu'un autre, nous utilisons la règle d'usage suivante : une différence de 5% de la précision moyenne à 11 points fixes de rappel est considérée comme significative [Sparck Jones 77]; si la différence atteint 10%, elle est considérée comme très significative.

1.5 Classification des algorithmes d'apprentissage

Généralement, un algorithme d'apprentissage est capable d'améliorer sa réponse au fil du temps. Par exemple, un programme capable de jouer aux échecs va améliorer sa stratégie en retenant les bons mouvements, ses erreurs ou celles de son adversaire.

En recherche d'informations, le but d'un algorithme d'apprentissage est d'augmenter la qualité de la réponse exprimée, par exemple, par la précision à 11 points de rappel.

Un premier critère de classification des différents algorithmes d'apprentissage est la durée de l'apprentissage. Plus précisément, l'apprentissage réalisé pour une requête sera-t-il utilisé pour des requêtes ultérieures ou non ? Selon ce critère, un algorithme d'apprentissage peut être

- **récurrent** (ou à long terme) si l'apprentissage d'une requête a des effets ultérieurs,
- **transiant** (ou à court terme) si l'apprentissage améliore uniquement la réponse à la requête courante sans tenir compte des résultats obtenus par les requêtes passées.

Un deuxième critère de classification considère la nature des changements effectués par l'algorithme. Ainsi, l'apprentissage peut :

- modifier la représentation des documents afin de rendre :
  - les documents pertinents plus "proches" de la requête,
  - les documents non pertinents moins similaires avec la requête,
- modifier la requête selon la représentation des documents pertinents et non pertinents afin de mieux exprimer le besoin d'information.

La figure 4 illustre la classification des algorithmes d'apprentissage que nous allons étudier dans les chapitres suivants selon les critères introduits dans cette section.

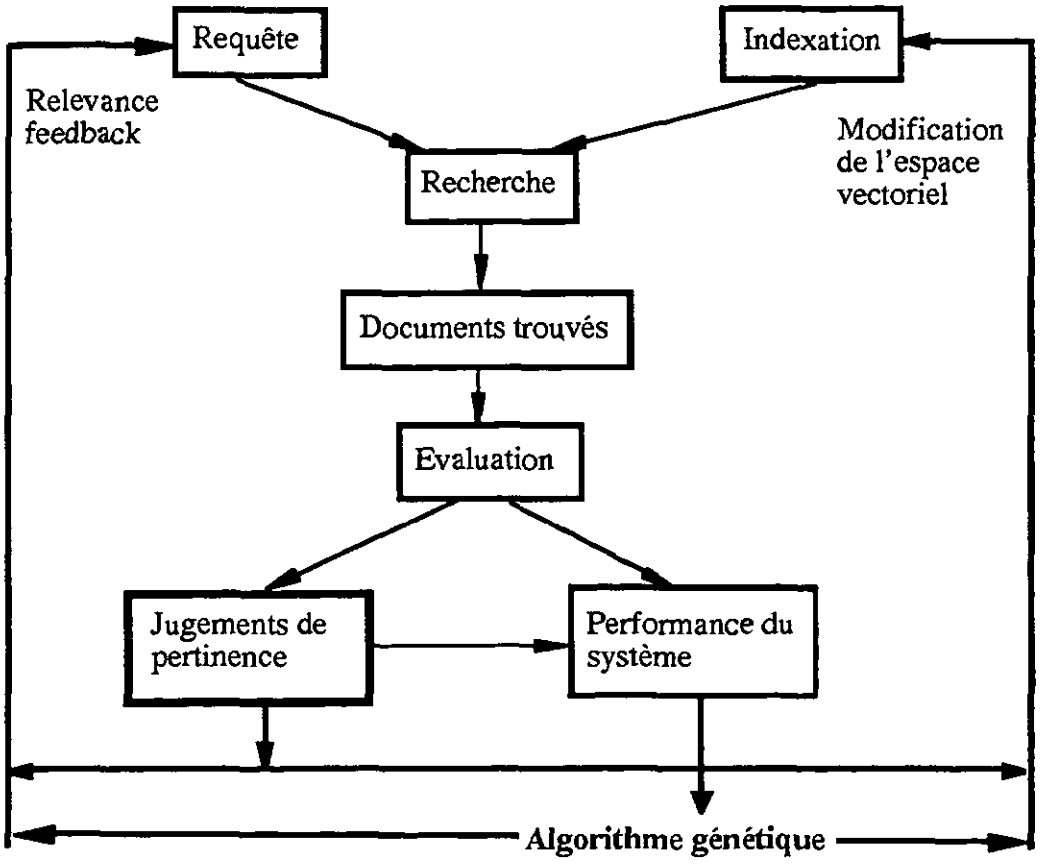

Fig. 4 Algorithmes d'apprentissage

2. Méthodologie d'évaluation

Au cours de ce chapitre, nous allons introduire la notion de taux d'erreur et présenter les différentes méthodes d'évaluation de ce taux.

2.1 Taux d'erreur et taux d'erreur apparent

L'exemple classique d'algorithme d'apprentissage est la classification automatique. Sa tâche consiste à attribuer une classe prédéfinie à tout objet qui lui sera présenté (par exemple, l'OCR). La performance de ces algorithmes peut s'estimer selon le taux d'erreur commis par le système lors de la classification. Ce taux est calculé comme le pourcentage d'objets mal classés :

$$taux\ d'erreur = \frac{nombre\ d'erreurs}{nombre\ de\ cas}$$

Le taux d'erreur réel est défini statistiquement comme la valeur de convergence asymptotique du taux d'erreur du classificateur quand le nombre de cas est de plus en plus grand, à condition que ces cas suivent la loi de répartition de la population actuelle [Efron 86].

Lors de l'apprentissage automatique, l'algorithme ajuste ses paramètres à l'aide d'un échantillon pour lequel on connaît la solution au problème. Malheureusement, on ne dispose pas d'un autre ensemble pour évaluer l'efficacité de l'algorithme d'apprentissage. Ceci pose un problème difficile à résoudre, c'est-à-dire comment estimer de manière fiable le taux d'erreur à partir d'un seul ensemble d'échantillons ?

Cet ensemble doit donc fournir le matériel à deux opérations distinctes : l'apprentissage proprement-dit et le test de l'algorithme. Ainsi, on le sépare en deux ensembles :

- l'ensemble d'apprentissage,
- l'ensemble de test.

Selon la manière dont ces ensembles sont définis, on peut calculer le taux d'erreur apparent (méthode rétrospective), ou estimer le taux d'erreur par des méthodes moins biaisées.

2.2 Évaluation rétrospective

Le taux d'erreur apparent implique l'utilisation du même ensemble pour l'apprentissage et pour l'évaluation. Appelé également taux d'erreur de resubstitution ou de reclassification, c'est une première estimation de la performance d'une méthode, un point de départ pour toute évaluation, car le taux réel est généralement plus élevé. Une évaluation d'algorithme d'apprentissage basée sur cette méthodologie est appelée **rétrospective**. Dans notre cas, l'information disponible pour l'algorithme d'apprentissage est constituée par les

jugements de pertinence. Une évaluation rétrospective utilise les mêmes requêtes pour l'apprentissage et pour le test.

Il semble à priori normal qu'un algorithme d'apprentissage soit capable de reconnaître les objets qu'il a rencontré. Mais, dans ce cas, il suffirait de créer une table de correspondance entre chaque objet vu et sa classe d'appartenance pour avoir un algorithme possédant un taux d'erreur proche de 0%. Cette observation nous indique que le taux d'erreur apparent est trop optimiste, et peut induire une erreur dans la sélection de la meilleure méthode d'apprentissage. Il arrive même qu'un algorithme donne de très bons résultats rétrospectifs mais qu'il soit incapable de répondre d'une manière cohérente à des requêtes ultérieures. Ce phénomène est appelé la sur-spécialisation aux cas présentés [Weiss 91].

Malgré les résultats trop optimistes obtenus avec cette méthode, on l'utilise pour avoir une idée de la limite supérieure de l'augmentation de la performance avec certains algorithmes d'apprentissage [Efron 86]. Dans le cadre de cette thèse, nous n'insisterons pas trop sur une méthodologie d'évaluation sujette à de multiples réserves.

2.3 Autres méthodes d'évaluation

Le but recherché dans l'évaluation d'un algorithme d'apprentissage est la performance réelle de l'algorithme dans de nouvelles conditions. Dans la section 2.1 nous avons vu que le taux d'erreur réel doit être estimé, par exemple, avec le taux d'erreur apparent. Mais d'autres estimateurs sont disponibles.

Pour obtenir une évaluation moins biaisée que l'évaluation rétrospective, nous devons séparer l'ensemble d'apprentissage de l'ensemble de test, c'est-à-dire, utiliser des cas distincts lors de l'apprentissage et lors de l'évaluation ("train and test"). Il n'est pas possible de vérifier autrement le comportement réel de l'algorithme.

Cette séparation est relativement simple pour les méthodes récurrentes. Dans ce cas, nous pouvons utiliser des méthodes d'évaluation développées pour les classificateurs. En général, à partir d'un nombre fixe de cas, la proportion suggérée entre les ensembles d'apprentissage et de test est de 2/3 - 1/3. Le nombre d'échantillons nécessaires pour que le taux d'erreur résultant du test soit proche du taux réel, par exemple dans un intervalle de confiance de 5%, n'est pas très grand. Pourtant, les 50 requêtes (CACM) et les 35 pour la CISI avec leurs jugements de pertinence ne sont pas suffisantes pour une estimation selon cette démarche.

Une autre solution est d'effectuer plusieurs expérimentations semblables et indépendantes en séparant l'ensemble des échantillons selon plusieurs manières. Le plus couramment, on divise les n cas en n/j parties de j cas. L'expérimentation s'effectue n/j fois avec l'ensemble d'apprentissage de taille $n-j$ et l'ensemble de test de taille j (voir tableau 4). Il suffit finalement de prendre la moyenne de ces n/j valeurs pour définir le taux d'erreur. Cette

démarche se nomme validation croisée. La méthode du *leaving one out* en est un cas particulier dans lequel on apprend sur *n-1* cas et on teste sur le $n^{ième}$ cas. Le taux d'erreur estimé par le *leaving one out* est considéré comme très proche du taux réel [Weiss 91].

Soit R l'ensemble de requêtes. La méthode *leaving one out* a la forme suivante :

```
for i = 1 to n do
    { trainSet := {Q1, .., Qn} -{Qi};
      testSet :={Qi};
      train (trainSet);
      ri := evalOn (testSet); }
result := mean (r)
```

Le tableau 4 montre le partage des échantillons en ensembles d'apprentissage et de test ainsi que le nombre d'itérations effectuées dans chaque cas.

Tab. 4 Ensemble d'apprentissage et de test

	Validation croisée	Leaving One Out
Apprentissage	n-j	n-1
Test	j	1
Itérations	n/j	n

Selon le *leaving one out*, l'apprentissage se fait sur l'ensemble des requêtes, sauf une, qui sert à l'évaluation. L'avantage de cette méthode est qu'elle permet d'évaluer l'algorithme d'apprentissage avec le maximum de requêtes autant pour l'apprentissage que pour l'évaluation, tout en donnant des résultats réalistes de la performance. Le temps de calcul, qui est supérieur à celui d'autres méthodes d'évaluation, représente le seul défaut majeur.

Pour évaluer les algorithmes d'apprentissage transiants, il n'est pas simple de séparer l'ensemble d'apprentissage de l'ensemble de test. Nous avons recouru à une solution connue qui consiste à empêcher l'algorithme d'apprentissage de connaître tous les jugements de pertinence lors de la phase d'entraînement. En effet, les 5 à 30 documents les mieux classées seront soumis à l'utilisateur qui les jugera comme pertinents ou non. L'algorithme effectue l'apprentissage à partir de cette information uniquement et l'évaluation estime sa performance par rapport à l'ensemble entier des jugements de pertinence. Selon la méthode d'apprentissage présentée, les chapitres respectifs contiendront plus de détails sur ces évaluations.

3. Apprentissage récurrent dans le modèle vectoriel

Ce troisième chapitre présente quelques méthodes récurrentes d'apprentissage dans le cadre du modèle vectoriel. Tous les algorithmes exposés dans les sections suivantes opèrent des modifications de l'indexation des documents. L'hypothèse sous-jacente à ces modèles est que les jugements de pertinence des requêtes passées peuvent améliorer la performance du système pour l'évaluation des requêtes futures.

3.1 Modèles 1 et 2 de Ide et Salton

E. Ide et G. Salton [Ide 71] proposent une méthode d'apprentissage, appelée standard, modifiant l'espace des documents en fonction des jugements de pertinence obtenus sur un ensemble de requêtes. Dans cette section, nous allons présenter deux variantes de cette stratégie.

Modification standard

Le but de la modification standard de l'espace vectoriel est de rapprocher les vecteurs des documents pertinents de celui de la requête. Ce rapprochement se réalise par :

1 la modification de la représentation des documents pertinents, et non de la requête, dans le but d'un apprentissage récurrent;

2 l'augmentation ou diminution du poids des termes d'indexation dans les documents pertinents en fonction du poids de ces termes dans la requête;

3 l'enrichissement de la représentation des documents pertinents avec les termes de la requête qui n'apparaissent pas dans ces documents.

Désignons par d_k le vecteur correspondant à un document d à l'itération k, et par q le vecteur correspondant à la requête; si le document d est pertinent, alors à l'itération k+1, on construit un vecteur d_{k+1} de la manière suivante :

$$d_{k+1} = (1-\alpha)\cdot d_k + \alpha \cdot q, \qquad 0 \le \alpha \le 1 \tag{3}$$

Dans le cas contraire, le document, n'étant pas pertinent, on ne le modifie pas :

$$d_{k+1} = d_k \tag{4}$$

Ainsi, cette méthode admet une deuxième hypothèse selon laquelle on ne peut rien apprendre de documents non pertinents pour une requête (le système n'apprend rien de ses erreurs). En effet, si le document n'est pas pertinent et qu'il est extrait, alors ce document possède au moins un terme commun avec la requête.

Par exemple, si la requête était "photographies de lions", ses termes seraient "photographie" et "lion". Un document traitant de photographies de paysages serait non

pertinent pour la requête. Néanmoins, le terme "photographie" ne devrait pas être modifié dans la description, car le document pourrait être pertinent pour d'autres requêtes contenant le terme "photographie".

Le tableau 5 présente les résultats de l'évaluation **rétrospective** de la modification standard. Les résultats de l'évaluation *leaving one out* du même modèle sont présentés dans le tableau 6.

Tab. 5 Résultats du modèle 1 par évaluation rétrospective

CACM		CISI	
Modèle 1, Eq. 3-4	**Précision (%ch.)**	**Modèle 1, Eq. 3-4**	**Précision (%ch.)**
Départ	32.56		20.12
$\alpha = 0.3$	90.09 (+176.69)	$\alpha = 0.1$	61.76 (+206.94)
$\alpha = 0.4$	93.06 (+185.82)	$\alpha = 0.3$	87.91 (+336.94)
$\alpha = 0.6$	95.43 (+193.03)	$\alpha = 0.4$	90.92 (+352.0)

Tab. 6 Résultats du modèle 1 par leaving one out

CACM		CISI	
Modèle 1, Eq. 3-4	**Précision (%ch.)**	**Modèle 1, Eq. 3-4**	**Précision (%ch.)**
Départ	32.56		20.12
$\alpha = 0.2$	35.02 (+7.54)	$\alpha = 0.06$	20.64 (+2.57)
$\alpha = 0.3$	35.40 (+8.72)	$\alpha = 0.08$	20.83 (+3.55)
$\alpha = 0.4$	35.29 (+8.38)	$\alpha = 0.1$	20.92 (+3.99)
$\alpha = 0.5$	34.88 (+7.13)	$\alpha = 0.12$	20.85 (+3.64)
$\alpha = 0.6$	34.01 (+4.45)	$\alpha = 0.2$	20.78 (+3.29)

Une variante du modèle 1

Cette deuxième méthode de Ide et Salton est inspirée des stratégies utilisées pour la reconnaissance adaptative des "patterns". L'idée de base est d'apparier chaque document pertinent retrouvé à un document non pertinent pas encore modifié. Si le document non pertinent présente une similarité avec la requête plus grande qu'un document pertinent, on effectue un échange pour rapprocher le document pertinent de la requête et pour en éloigner le document non pertinent. Formellement :

a) Si pour tout d_i pertinent pour q_0 et pour tout d_j non pertinent

$$sim(d_i, q_0) > sim(d_j, q_0) + \theta$$

alors aucune modification n'est faite.

b) Autrement, on traite tout vecteur d_i dénotant un document pertinent i dans l'ordre de classement avec q_0 : s'il existe un document k non pertinent pour q_0 et pas encore ajusté avec q_0 tel que

$$sim(d_k, q_0) + \theta > sim(d_i, q_0)$$

$$\text{alors : } d'_i = (1-\alpha)\cdot d_i + \alpha \cdot q_o \text{ et} \quad (5)$$

$$d'_k = (1-\alpha)\cdot d_k - \alpha \cdot q_o \quad (6)$$

où d_k est le document correspondant aux critères annoncés ayant la plus grande similarité avec q_0, et d'_i et d'_k sont les nouveaux documents modifiés.

c) S'il n'existe aucun document non pertinent pas encore modifié, la modification de d_i est exécutée quand même.

Les résultats de la variante du modèle 1 de Ide et Salton (que nous appelons *modèle 2*) pour l'évaluation rétrospective et *leaving one out* sont présentés respectivement dans les tableaux 7 et 8.

Tab. 7 Évaluation rétrospective du modèle 2 sur les deux collections

CACM			CISI		
α	θ	Précision (%changement)	α	θ	Précision (%changement)
Départ		32.56			20.12
0.1	0.46	74.44 (+128.59)	0.06	0.0	57.67 (+186.67)
0.3	0.23	86.87 (+166.76)	0.1	0.28	72.96 (+262.67)
0.4	0.46	87.52 (+168.75)	0.3	0.0	85.12 (+323.15)

Tab. 8 Évaluation leaving one out du modèle 2 sur les deux collections

CACM			CISI		
α	θ	Précision (%changement)	α	θ	Précision (%changement)
Départ		32.56			20.12
0.1	0.46	34.95 (+7.33)	0.06	0.0	22.08 (+9.74)
0.3	0.0	35.56 (+9.19)	0.1	0.0	22.33 (+11.0)
	0.23	36.09 (+10.84)		0.14	22.29 (+10.82)
	0.46	36.15 (+11.01)		0.28	22.27 (+10.71)
0.4	0.46	35.75 (+9.77)	0.3	0.0	19.19 (-4.59)

Sur la base des tableaux 7 et 8, les deux formulations améliorent la performance pour les deux collections. Pour l'évaluation *leaving one out*, la différence est significative pour la collection CACM avec les deux méthodes (+8.72 modification standard, +11.01 la variante).

Pour la collection CISI, seul le deuxième modèle réalise une augmentation significative (+3.99 modification standard, +11.0 la variante).

Notons que la variante présente de meilleurs résultats sur les deux collections, mais la différence entre les meilleures performances respectives n'est significative que pour la collection CISI (+2.12 - CACM, +6.73 - CISI).

3.2 La méthode de Brauen

Dans cette section nous allons présenter un modèle proposé par Brauen dans [Brauen 71]. La modification de l'espace des documents se fait de manière à accroître la similarité entre le document et la requête :

a) si le terme d'indexation existe dans la requête, mais pas dans le document pertinent, on le rajoute au document avec un poids $\beta \geq 0$ (constante arbitraire);

b) si le terme apparaît dans un document pertinent et dans la requête, son poids est augmenté de la manière suivante :

$$w'_{ij} = w_{ij} + \gamma\left(1 - w_{ij}\right) \quad \text{avec} \quad 0 \leq \gamma \leq 1 \tag{7}$$

c) si le terme apparaît dans le document pertinent, mais pas dans la requête, on diminue son poids de la manière suivante :

$$w'_{ij} = \delta \cdot w_{ij} \quad \text{avec} \quad 0 \leq \delta \leq 1 \tag{8}$$

La méthode originale a été conçue avec une indexation pondérée, ce qui influence les paramètres des équations 7 et 8. Ainsi, le nombre 1 dans l'équation (7) représente une valeur maximale du poids des termes, et il peut être différent pour une autre indexation.

On remarque qu'en remplaçant la valeur 1 dans l'équation (7) par w_{qj}, on revient à la modification standard. Cet algorithme est donc essentiellement équivalent à la méthode de Ide et Salton, si l'on considère que les termes dans la requête ont des poids égaux à la valeur maximale permise par l'indexation.

Évaluation

Nous avons testé l'algorithme avec $\beta = \gamma$, et les résultats sont illustrés dans les tableaux 9 (évaluation rétrospective) et 10 (leaving one out). Nous avons donné une importance particulière aux évaluations avec $\delta = 1.0$, car ceci signifie que les termes apparaissant dans les documents pertinents, mais pas dans la requête, ne sont pas modifiés.

Les résultats du tableau 10 semblent confirmer le fait que l'algorithme est plus performant sans l'équation 8 pour la collection CACM. Pour la collection CISI, ce point de l'algorithme améliore la performance pour de grandes valeurs de δ, mais les accroissements ne sont pas significatifs.

Tab. 9 Évaluation rétrospective de l'algorithme de Brauen

Eq. 7-8		Précision (% changement)	
δ	γ	CACM	CISI
Départ		32.56	20.12
0.9	0.1	89.17 (+173.82)	83.09 (+313.06)
	0.4	97.90 (+200.67)	89.00 (+332.44)
	0.6	97.87 (+200.53)	90.06 (+347.68)
1.0	0.05	74.71 (+129.42)	65.84 (+227.23)
	0.1	89.23 (+174.04)	79.27 (+294.08)
	0.4	97.88 (+200.56)	88.20 (+338.44)

Tab. 10 Résultats de l'algorithme de Brauen sur les deux collections

CACM			CISI		
Eq. 7-8		Précision (%ch.)	Eq. 7- 8		Précision (%ch.)
Départ		32.56			20.12
δ	γ		δ	γ	
0.5	0.1	25.57 (-21.47)	0.5	0.05	18.46 (-8.24)
0.9	0.1	35.31 (+8.44)	0.9	**0.05**	**20.36 (+1.2)**
	0.15	35.46 (+8.89)		0.07	19.30 (-4.06)
1.0	0.05	35.47 (+8.92)	1.0	**0.05**	**19.51 (-3.02)**
	0.1	**36.65 (+12.57)**		0.1	18.08 (-10.14)
	0.15	**36.45 (+11.94)**		0.15	16.47 (-18.14)
	0.2	35.68 (+9.59)			

3.3 Approche de Friedman, Maceyak et Weiss

Friedman, Maceyak et Weiss [Friedman 71] proposent une stratégie qui prend en compte non seulement les termes se trouvant dans la requête, mais aussi leurs "synonymes". Ainsi, pour trouver les synonymes d'un terme donné, cet algorithme cherche tous les termes qui semblent caractériser la requête. Soit un terme t_i et les mesures suivantes :

$$R_i = \frac{\sum_{\forall dj,\ \text{document pertinent}} w_{ij}}{\text{Nombre de documents pertinents}}$$

$$N_i = \frac{\sum_{\forall dj,\ \text{document non pertinent}} w_{ij}}{\text{Nombre de documents non pertinents extraits}}$$

$$D_i = R_i - N_i$$

dans lesquelles D_i indique la valeur discriminante pour le terme t_i, qui est la différence entre l'importance moyenne du terme dans les documents pertinents (R_i) et dans les documents non pertinents (N_i). Les termes pour lesquels $D_i > \delta$ sont considérés comme caractérisant le besoin d'information auquel répond l'ensemble des documents pertinents. Ensuite, on calcule le poids relatif du terme t_i dans la requête:

$$F_i^1 = \frac{w_{qi}}{\sum_j w_{qj}}$$

où w_{qi} est le poids du terme t_i dans la requête,

$$F_i^2 = \frac{\sum_{\forall dj,\ document\ pertinent} w_{ij}}{\sum_{k=1}^{t} \sum_{\forall dj,\ document\ pertinent} w_{kj}} \qquad F_i^3 = \frac{\sum_{\forall dj,\ document\ non\ pertinent} w_{ij}}{\sum_{k=1}^{t} \sum_{\forall dj,\ document\ non\ pertinent} w_{kj}}$$

Pour les termes se trouvant dans la requête, dans les documents retournés ou pertinents, on calcule :

si $D_i > \delta$ ou si le terme apparaît dans la requête :

$$T_i = \alpha_1 F_i^1 + \alpha_2 F_i^2 \tag{9}$$

si $D_i < \delta$ et si le terme n'apparaît pas dans la requête :

$$T_i = -\alpha_2 F_i^3$$

Le vecteur du document d_j est modifié pour ces termes de la manière suivante :

- si d_j est pertinent,

$$w_{ij} = w_{ij}(1 + T_i) \tag{10}$$

- si d_j est non pertinent, son poids est mis à 0 : $w_{ij} = 0$.

Les équations (9) et (10) nous disent que l'augmentation des poids est plus accentuée pour les termes déjà importants dans les documents.

En fait, les termes "discriminants" pour une requête apparaissent avec des poids forts dans les documents pertinents et avec des poids faibles ou nuls dans les documents non pertinents. C'est la raison pour laquelle ils sont considérés comme synonymes des termes de la requête.

Évaluation

Pour évaluer cet algorithme, les changements suivants ont été apportés :

- T_i est calculé seulement pour les termes discriminants ou apparaissant dans les documents pertinents;
- seuls ces termes sont modifiés dans les documents pertinents;
- les documents non pertinents ne sont pas modifiés.

L'évaluation rétrospective de l'algorithme est présentée dans le tableau 11 :

Tab. 11 Évaluation rétrospective de l'algorithme de Friedman, Maceyak et Weiss

CACM				CISI			
Eq. 10				Eq. 10			
δ	α_1	α_2	Précision (%ch.)	δ	α_1	α_2	Précision (%ch.)
0.05	1.2	1.2	93.31 (+186.58)	0.01	0.2	0.2	46.74 (+132.32)
0.1	1.2	1.2	94.02 (+188.76)	0.05	0.8	0.8	86.71 (+330.95)
0.3	1.2	1.2	94.22 (+189.37)	0.1	1.2	1.2	89.19 (+343.31)

Les résultats de l'évaluation de cet algorithme selon les modifications décrites avec la méthode *leaving one out* se trouvent dans le tableau 12 :

Tab. 12 L'algorithme de Friedman, Maceyak et Weiss selon la méthode leaving one out

CACM				CISI			
Eq. 10				Eq. 10			
δ	α_1	α_2	Précision (%ch.)	δ	α_1	α_2	Précision (%ch.)
0.02	0.5	0.5	35.69 (+9.61)	0.01	0.1	0.1	20.67 (+2.73)
0.05	0.5	0.5	35.94 (+10.39)		0.2	0.2	20.86 (+3.7)
	0.7	0.7	37.00 (+13.63)		0.2	0.5	20.69 (+2.84)
	1	1	37.83 (+16.19)		0.5	0.2	20.36 (+1.18)
	1.2	1.2	38.45 (+18.09)	0.03	0.5	0.5	19.71 (-2.03)
0.1	0.5	0.5	35.59 (+9.31)	0.05	0.5	0.5	19.81 (-1.54)

Le choix du paramètre δ s'avère différent dans les deux collections. En effet, les valeurs discriminantes D_i sont différentes, et l'une des raisons est le nombre moyen de termes discriminants. Le tableau 13 démontre que le nombre moyen de termes ajoutés à la requête varie selon la collection et selon la valeur du paramètre δ. Pour la collection technique CACM, une valeur de $\delta = 0.05$ permet d'ajouter, en moyenne, 14 termes mais, la valeur δ "idéale" pour la collection CISI s'élève à 0.01, ce qui apporte, en moyenne, 96 termes. Une expansion plus forte semble utile à la collection CISI.

Les résultats de cet algorithme sont comparables à la modification standard et à l'algorithme de Brauen. Malgré ses bonnes performances, cet algorithme est défavorisé par le temps de calcul, plus grand que celui des autres algorithmes.

Tab. 13 Nombre moyen de termes discriminants selon δ

CACM		CISI	
δ	nombre moyen de termes	δ	nombre moyen de termes
0.1	3.62	0.05	6.65714
0.05	14.12	0.03	16.0857
0.02	55.28	0.01	96.3714

Nous avons jugé intéressant de voir si cet algorithme trouve effectivement des synonymes aux termes de la requête. Pour cela, nous avons considéré la première requête correspondant à chaque collection et la valeur de δ pour laquelle nous avons obtenu la meilleure performance. Le tableau 14 montre la répartition de synonymes trouvés par l'algorithme. Une présentation plus détaillée de cette analyse se trouve dans [Vrajitoru 95].

Tab. 14 Statistiques sur la première requête des deux collections

Nombre de	CACM	CISI
termes ∈ requête	10	12
synonymes transmis par la machine	13	112
termes ∈ requête ∩ synonymes	0	3
synonymes effectifs	0	7
termes ∈ domaine	6	26
synonymes qui n'ont rien à voir avec la requête	7	76

Ce tableau semble suggérer que la bonne performance de cet algorithme vient surtout des termes de la requête, et non de leur synonymes.

3.4 Notre modèle

Dans cette section, nous proposons une autre stratégie d'apprentissage. Cette dernière va modifier uniquement l'indexation des documents pertinents et va tenir compte de leur classement dans la liste retournée par le système.

Comme d'autres auteurs, nous avons suivi le principe d'augmenter, dans la description des documents pertinents, le poids des termes inclus dans la requête. La nouveauté de notre méthode consiste à traiter chaque document pertinent d'une manière quelque peu différente, car nous pensons que le poids associé à ces termes doit augmenter plus fortement si le document est mal placé dans la liste retournée. Ainsi, il semble plus

important d'augmenter le poids des termes de la requête dans les documents pertinents ayant un faible degré de similarité que dans ceux ayant un fort degré de similarité.

Notre modification va donc se baser sur la similarité du document avec la requête. L'application opérationnelle de cette idée est illustrée dans l'équation (11) :

$$w'_{ij} = w_{ij} + \alpha \cdot q_j + \beta \cdot \left(sim\left(d_{premier}, q\right) - sim\left(d_i, q\right) \right) \qquad (11)$$

Cette équation signifie que l'augmentation du poids d'un terme dans un document pertinent dépend directement de la différence entre la similarité de ce document et la similarité du premier document retourné. Ainsi pour un document pertinent ayant un faible degré de similarité avec la requête, la troisième partie de l'équation (11) sera nettement plus forte.

Évaluation

L'évaluation rétrospective de l'algorithme a donné les résultats présentés dans le tableau 15, et l'évaluation selon la méthode du *leaving one out* dans le tableau 16.

Tab. 15 Résultats rétrospectifs du modèle proposé

Collection		CACM	CISI
Eq. 11		Précision (%changement)	
Départ		32.56	20.12
α	β		
0.06	0.03	64.54 (+98.88)	57.0 (+183.31)
0.3	0.2	96.48 (+196.31)	89.31 (+343.97)
0.4	0.5	98.90 (+203.7)	88.91 (+341.96)
0.6	0.5	98.85 (+203.55)	89.68 (+345.79)

Tab. 16 Résultats du modèle proposé sur les deux collections

CACM			CISI		
Eq. 11		Précision (%ch.)	Eq. 11		Précision (%ch.)
Départ		32.56			20.12
α	β		α	β	
0.2	0.3	39.1 (+20.09)	0.04	0.05	20.83 (+3.52)
	0.4	39.26 (+20.57)		0.1	20.48 (+1.78)
0.3	0.1	39.0 (+19.78)	0.06	0.03	20.91 (+3.91)
	0.2	39.38 (+20.95)		0.05	20.84 (+3.56)
	0.3	39.07 (+19.99)		0.1	20.51 (+1.93)
0.4	0.1	38.74 (+18.99)	0.08	0.02	20.79 (+3.31)
	0.2	38.74 (+18.99)			

3.5 Commentaires des résultats

Pour terminer ce chapitre, nous allons étudier le classement des meilleures performances obtenues par chacune des quatre méthodes décrites précédemment. Ces résultats sont regroupés dans le tableau 17.

Tab. 17 Classement des méthodes

CACM		CISI	
Méthode	**Précision (%ch.)**	**Méthode**	**Précision (%ch.)**
Départ	32.56		20.12
Vrajitoru, éq. 11 $\alpha = 0.3, \beta = 0.2$	39.38 (+20.95)	Ide et Salton 2, éq. 5-6 $\alpha = 0.1, \theta = 0.0$	22.33 (+11.0)
Friedman, Maceyak et Weiss, éq. 10, $\delta = 0.05, \alpha_1 = \alpha_2 = 1.2$	38.45 (+18.09)	Ide et Salton 1, éq 3-4, $\alpha = 0.1$	20.92 (+3.99)
Brauen, éq. 7-8 $\gamma = 0.1, \delta = 1.0$	36.65 (+12.57)	Vrajitoru, éq. 11 $\alpha = 0.06, \beta = 0.03$	20.91 (+3.91)
Ide et Salton 2, éq. 5-6, $\alpha = 0.3, \theta = 0.46$	36.15 (+11.01)	Friedman, Maceyak et Weiss, éq. 10, $\delta = 0.01, \alpha_1 = \alpha_2 = 0.2$	20.86 (+3.7)
Ide et Salton 1, éq 3-4, $\alpha = 0.3$	35.4 (+8.72)	Brauen, éq. 7-8 $\gamma = 0.05, \delta = 0.9$	20.36 (+1.2)

Pour la collection CACM, la différence entre la méthode que nous avons proposée et l'algorithme de Friedman, Maceyak et Weiss n'est pas significative (39.38 vs. 38.45 (+2.42)). Par contre, notre méthode est significativement meilleure que toutes les autres (+7.45 par rapport à Brauen). La méthode de Friedman, Maceyak et Weiss est significativement meilleure que la méthode 2 de Ide et Salton (+6.36), mais équivalente à l'algorithme de Brauen (+4.91). Dans cette collection, tous les algorithmes d'apprentissage étudiés possèdent des performances significativement meilleures que la solution de base.

Pour la collection CISI, la deuxième variante de l'algorithme de Ide et Salton s'avère significativement meilleure que toutes les autres méthodes. Ainsi, la différence entre celle-ci et la modification standard (Ide et Salton 1) s'élève à +6.74 (22.33 vs. 20.92). La différence entre les autres méthodes, par contre, n'est pas significative. Ainsi, par exemple, la méthode 1 de Ide et Salton et la méthode de Brauen donnent des performances similaires. Il n'y a que la variante de la méthode 1 de Ide et Salton qui améliore la performance de départ de manière significative. Ceci semble suggérer que pour la collection CISI, on gagne à modifier la représentation des documents non pertinents.

La figure 5 présente une synthèse de ces commentaires. Les algorithmes sont regroupés par classes d'équivalence. Ainsi, nous avons deux classes d'équivalence pour chaque collection, dont une contient une seule stratégie (Ide et Salton 2, pour la collection

CISI). L'ellipse marque le fait que la différence entre les deux classes d'équivalence pour la collection CACM, est quasiment significative (4.9%).

Comme le modèle de Ide et Salton 2 (éq. 2) améliore de manière significative la performance dans les deux collections, nous pouvons soutenir la thèse selon laquelle l'apprentissage améliore la performance indépendamment de la collection.

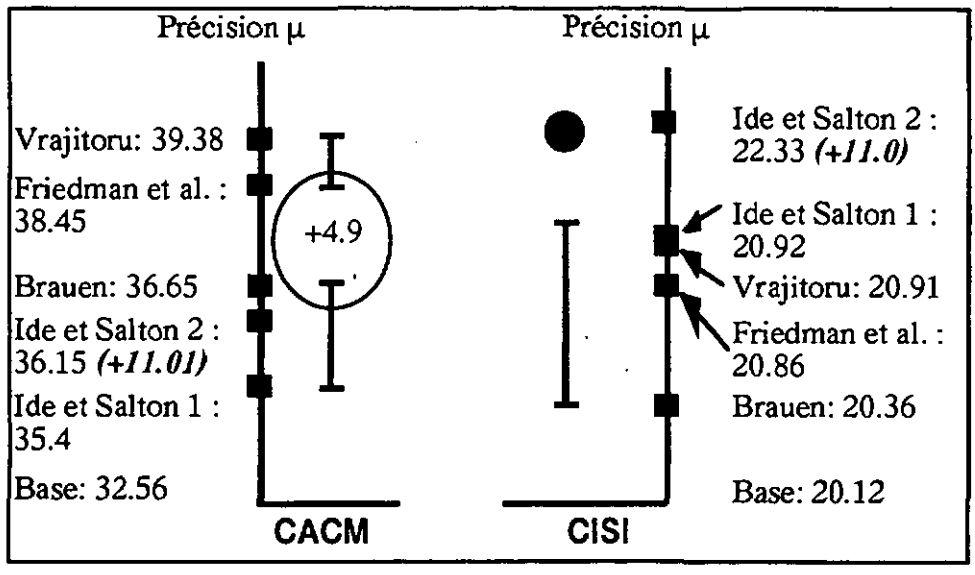

Fig. 5 Classification des méthodes récurrentes

La différence de performance des algorithmes entre les collections a plusieurs sources, telles que :

- Pour la collection CISI, le nombre moyen de documents pertinents par requête est plus élevé (49.77 vs 15.84 pour la CACM) et ceci entraîne le fait que le repositionnement d'un document (pertinent ou non) a moins d'impact sur la performance.
- Pour la collection CISI, les requêtes possèdent plus de termes en commun que de documents pertinents, comme le démontre le tableau 2, section 1.3. Par exemple, un terme très fréquent, "system", apparaît dans 14 (28%) requêtes pour la CACM et dans 18 (51%) requêtes pour la CISI. Parmi ces requêtes, 4 ont au moins un document commun pour les deux collections, ce qui représente respectivement 28.57% pour la CACM et 22.22% pour la CISI.

Les valeurs optimales pour les paramètres nécessaires aux algorithmes sont, selon le tableau 17, très différentes d'une collection à l'autre. Ceci entraîne le fait que, pour une nouvelle collection, on ne pourrait pas assigner une valeur sans essais préalables.

Nous avons trouvé ces valeurs par des recherches suivant deux principes : la recherche dichotomique et la méthode du *hill-climbing*. Les deux méthodes sont des adaptations naturelles des algorithmes connus dans la littérature sous le même nom.

Par exemple, si la valeur du paramètre α suggérée par les auteurs d'une stratégie est de 0.1, on peut d'abord essayer le *hill-climbing* par pas de 0.1. Si, pour la valeur 0.2, on obtient une plus petite performance que pour 0.1, on peut appliquer la recherche dichotomique dans l'intervalle [0.05, 0.15]. Dans le cas contraire, on peut poursuivre notre recherche par *hill-climbing* avec le même pas jusqu'à obtenir une ou deux valeurs de α pour lesquelles la performance n'augmente plus.

4. Apprentissage transiant dans le modèle vectoriel

Au cours de la section 1.5, nous avons précisé que les algorithmes d'apprentissage transiants s'appliquent uniquement à la requête courante. La méthode du *relevance feedback*, présentée ci-après, constitue une méthode classique de ce type d'apprentissage.

Dans ce cadre, on fait l'hypothèse que les documents pertinents à une requête donnée possèdent des termes d'indexation similaires et que, si ces termes sont judicieusement sélectionnés et pondérés, leur addition à la requête permet d'améliorer significativement la performance.

4.1 Présentation du *relevance feedback*

Beaucoup d'auteurs ont utilisé cette méthode efficace et simple, et il en résulte de nombreuses variantes [Salton 90]. En principe, cet algorithme consiste à modifier la requête en tenant compte de certains jugements de pertinence donnés par l'usager, afin de mieux cerner et représenter son besoin d'information.

Une application de cette méthode pour des indexations binaires est possible [Dillon 80]. L'indexation pondérée représente le cas le plus souvent rencontré et c'est le type d'indexation que nous avons utilisé. Parmi les variantes de l'algorithme, nous avons choisi celle qui présente les meilleurs résultats dans [Salton 90], c'est-à-dire le dec-hi [Ide 71].

La méthode consiste à modifier les poids des termes de la requête en

- ajoutant les poids des termes apparaissant dans les documents pertinents,
- soustrayant les poids des termes apparaissant dans le premier document non pertinent retrouvé.

Plus précisément, la nouvelle requête est obtenue selon l'équation (12) :

$$q_{k+1} = q_k + \sum_{pertinents} d_i - d_{1er\ non\ pert} \qquad (12)$$

4.2 Évaluation

Pour évaluer le *relevance feedback* de manière rétrospective, les jugements de pertinence sont entièrement utilisés, autant pour l'apprentissage que pour le test. Le tableau 18 montre les résultats de cette évaluation.

Afin d'estimer le taux d'erreur d'une manière plus réaliste, nous avons adopté une évaluation résiduelle. Ainsi, les 5 à 30 documents les mieux classés par le système sont considérés comme "vus par l'utilisateur" et sont utilisés pour modifier la requête. Par la suite, ils sont supprimés, qu'ils soient pertinents ou non, de la seconde liste retournée. Cette dernière est évaluée selon les jugements de pertinence résiduels. Les résultats de cette

évaluation sont présentés dans le tableau 19. Dans ce tableau, la colonne sous l'étiquette "Avant" indique la précision moyenne sans le *relevance feedback* et la colonne "Après", désigne cette précision avec le *relevance feedback.*

Tab. 18 Relevance feedback selon l'évaluation rétrospective

Docs. jugés	CACM	CISI
0	32.56	20.12
5	43.50 (+33.57)	26.12 (+29.87)
10	47.95 (+47.25)	28.60 (+42.16)
15	51.23 (+57.3)	30.88 (+53.53)
20	54.01 (+65.84)	33.06 (+64.37)
30	58.77 (+80.47)	36.35 (+80.71)

Tab. 19 Relevance feedback en évaluation résiduelle

	CACM		CISI	
Docs. jugés	Avant	Après	Avant	Après
5	19.19	25.50 (+32.88)	17.69	22.31 (+26.16)
10	16.67	23.85 (+43.05)	16.70	22.71 (+36.02)
15	15.57	23.83 (+53.08)	16.34	23.24 (+42.28)
20	13.48	22.10 (+64.01)	14.62	22.83 (+56.14)
30	11.70	21.87 (+86.95)	13.07	23.73 (+81.62)

Parfois tous les documents pertinents se retrouvent parmi les documents utilisés pour l'apprentissage. Ce phénomène apparaît lorsqu'il y a un petit nombre de documents pertinents pour la requête et que ceux-ci sont initialement très bien placés. En conséquence, les jugements de pertinence résiduels correspondent à un ensemble vide. Ces requêtes sont éliminées de l'évaluation résiduelle. Le tableau 20 présente les requêtes dans cette situation selon le nombre de documents utilisés pour l'apprentissage :

Tab. 20 Requêtes retirées selon le nombre de documents jugés par l'utilisateur

Docs. jugés	CACM
5, 10	31, 33, 64
15, 20, 30	6, 31, 33, 64

Afin d'uniformiser les résultats, nous avons réalisé une dernière évaluation en retirant les requêtes 6, 31, 33 et 64, présentée par le tableau 21. Ces résultats sont synthétisés dans la figure 6.

***Tab. 21** Relevance feedback avec les jugements de pertinence résiduels*

	CACM	
Docs. jugés	Avant	Après
5	19.34	25.95 (+34.19)
10	16.31	24.24 (+48.62)

Cette méthode ne doit pas être comparée avec les algorithmes d'apprentissage modifiant la représentation des documents car :

- le type d'apprentissage n'est pas le même (récurrent - transiant)
- la performance de départ n'est pas la même.

Malgré ces contraintes, on peut remarquer que les pourcentages d'augmentation de la performance s'avèrent nettement plus significatifs pour le *relevance feedback* que pour les méthodes récurrentes vues au chapitre précédent.

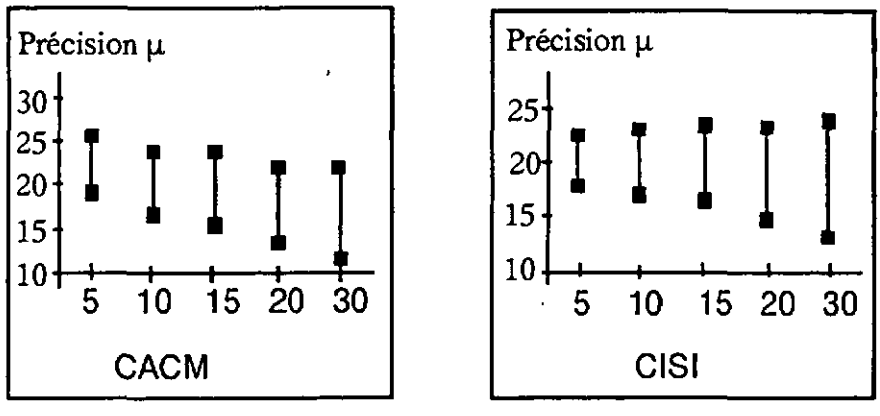

Fig. 6 Résultats selon l'évaluation résiduelle

5. L'algorithme génétique en recherche d'informations

Des algorithmes complexes d'apprentissage développés en intelligence artificielle peuvent être appliqués en recherche d'informations et l'algorithme génétique (A.G.) en est un exemple. Les applications que nous avons développées de l'A.G. s'inscrivent dans plusieurs des catégories présentées auparavant. Ce chapitre présente les traits principaux des A.G. ainsi que leurs applications en recherche d'informations.

5.1 Présentation de l'algorithme génétique

L'algorithme génétique est généralement utilisé pour résoudre des problèmes d'optimisation. Soit P un ensemble dont les éléments seront nommés individus et une fonction $f(i)$ $(f: P \rightarrow R)$ associant une performance à chaque individu i. Par itération, l'A.G. cherchera le(s) individu(s) i_o offrant la meilleure performance, performance qui ne représente pas forcément la solution optimale.

L'application de l'A.G. requiert une codification des individus sous forme d'un vecteur de longueur L dans lequel chaque élément i_k représente un gène, et k le locus.

$$i = (i_1, i_2, \cdots, i_L)$$

La codification binaire (la valeur d'un gène pouvant être 0 ou 1), est la forme la plus répandue voir recommandée par certains auteurs [Goldberg 89]. L'A.G. le plus simple, consiste à itérer les trois opérations suivantes :

1. reproduction
2. croisement avec une probabilité p_c
3. mutation avec la probabilité p_m

La population générée à chaque itération s'appelle génération, notée P_k,, où k indique le temps (ou le numéro de l'itération). Reprenons ces trois opérations.

Étant donnée une population initiale P_o, la **reproduction** consiste, dans le schéma le plus simple, à choisir d'une manière aléatoire un nombre d'individus $\in P_o$, égal au $cardinal(P_o)$, chaque individu ayant une chance d'être sélectionné proportionnelle à sa performance $f(i)$.

Le croisement constitue la deuxième opération. Soient i_1 et i_2 , deux individus choisis aléatoirement, et k , une position aléatoire entre 1 et L -1, nommée le site de croisement. On construit par **croisement** deux autres individus i_3 et i_4 de la manière suivante :

$$i_3(j) = \begin{cases} i_1(j) & \text{si} \quad j \le k \\ i_2(j) & \text{si} \quad j > k \end{cases} \qquad i_4(j) = \begin{cases} i_2(j) & \text{si} \quad j \le k \\ i_1(j) & \text{si} \quad j > k \end{cases}$$

Soit i_1 l'individu choisi lors de la phase de reproduction, et k, un nombre aléatoire entre 1 et L indiquant le site de mutation. On construit i_2 par **mutation** en remplaçant le gène correspondant à cette position par sa négation :

$$i_2(j) = i_1(j) \quad \text{si} \quad j \neq k$$
$$i_2(k) = \overline{i_2(k)}$$

L'itération s'arrête après un nombre de générations fixé d'avance ou lorsque la population converge selon un critère donné.

La population initiale est construite de différentes façons qui varient selon la nature du problème à résoudre. Le plus souvent, les individus formant la population initiale sont choisis de manière aléatoire et couvrent uniformément l'ensemble des solutions possibles.

Illustration

Nous allons illustrer l'A.G. à l'aide d'un petit exemple. Supposons que nous recherchons le maximum de la fonction $f(x) = 1 + 3 \cdot x - 0.2 \cdot x^2$ sur l'intervalle [0 .. 15]. La figure 7 montre la partie du graphe de cette fonction qui nous intéresse.

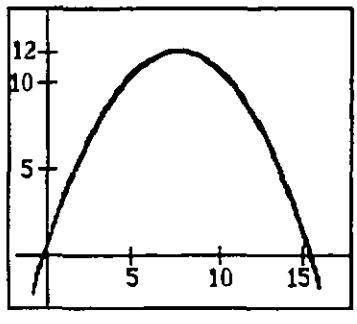

Fig. 7 Graphe de la fonction

Pour codifier les individus, nous avons représenté les nombres entiers de l'intervalle sur 4 bits. La fonction atteint son maximum (12.25) pour $x = 7.5$, qui ne se trouve pas parmi les 16 individus. En tenant compte de la codification, le maximum (12.2) est atteint aux points 0111 = 7 et 1000 = 8, symétriques par rapport au point 7.5.

Prenons une population initiale formée de 4 individus tirés au hasard, que nous avons triés par ordre décroissant de performance :

***Tab.** 22 Population initiale*

individu	nb. entier	f(i)	p(i)
0110	6	11.8	32.78%
0100	4	9.8	27.22%
1100	12	8.2	22.78%
1011	13	6.2	17.22%

La construction de la nouvelle génération commence avec 4 tirages aléatoires pour choisir 4 individus, la probabilité de chacun d'eux étant proportionnelle à sa fonction de performance. Le tirage aléatoire a choisi les 4 individus suivants : 1011, 0110, 0110 et 1100.

Pour la phase de croisement ou celle de mutation, ces 4 individus sont groupés en deux paires, soient (1011; 0110) et (0110; 1100). Supposons que le croisement s'effectue avec une probabilité de 0.8 (p_c).

Pour la première paire, le hasard décide que ces deux individus seront croisés. On choisit aléatoirement une position de croisement entre 0 et 4 (soit 3) et les deux nouveaux individus seront :

$$\begin{matrix} 101 \| 1 \\ 011 \| 0 \end{matrix} \Rightarrow \begin{cases} 1010 \\ 0111 \end{cases}$$

Pour la deuxième paire, le hasard n'indique pas de croisement.

Avec une probabilité fixée de mutation ($p_c = 0.01$), chaque individu passe tour à tour pour déterminer si un de ses gènes est à modifier.

Pour l'individu 0110, une mutation s'opère sur le deuxième gène : 0110 -> 0010

et pour l'individu 1100, une mutation s'exerce sur le dernier gène : 1100 -> 1101

La nouvelle population est donc composée par :

***Tab. 23** Nouvelle génération*

individu	nb. entier	f(i)
1010	10	11
0111	7	12.2
0010	2	6.2
1101	13	6.2

On peut remarquer que l'un des deux individus de performance maximale (0111 = 7) se trouve dans cette nouvelle génération. Cela illustre le fonctionnement de l'A.G.

5.2 Application en recherche d'informations

L'espace de recherche d'un A.G. appliqué en recherche d'informations est constitué par l'ensemble de tous les descripteurs possibles des documents. Pour un document donné d_j, avec $j = 1..m$ et un ensemble de termes t_k avec $k=1..n$, le descripteur associé à d_j possède la forme suivante :

$$d_j = \langle t_{1j}, t_{2j}, \ldots, t_{nj} \rangle$$

Le gène t_{ij} correspond au terme t_i dans le document d_j. Les valeurs prises par ce gène dépendent du type d'indexation utilisé, comme l'explique la section 2.1.

Un individu s'obtient par concaténation des descripteurs de tous les documents d_j, pour $j = 1..m$:

$$i = \langle d_1, d_2, \ldots, d_m \rangle = \langle t_{11}, \ldots t_{n1}, t_{12}, \ldots t_{n2}, \ldots, t_{1m}, \ldots t_{nm} \rangle$$

Cette représentation est une généralisation par rapport à l'application décrite dans [Gordon 88] et [Blair 90]. Dans ces recherches, un individu est un descripteur particulier d'un seul document. En passant à une collection de plus grande taille, nous ne pouvons plus nous permettre de traiter chaque document séparément, phénomène qui nous a conduit à cette approche différente.

Pour réduire la taille des individus au strict nécessaire dans nos évaluations, nous avons pris en compte uniquement les termes qui apparaissent dans les requêtes. Ainsi, nous avons gardé 313 termes pour la collection CACM et 136 termes pour la collection CISI.

La performance d'un individu est représentée par la précision moyenne à 11 points fixes de rappel (section 1.4). Les sections suivantes présentent les différentes manières d'appliquer l'A.G.

Diverses exécutions de l'A.G. nous ont fait remarquer que la meilleure performance peut décroître d'une génération à l'autre. Comme le phénomène est courant, nous avons développé une forme monotone d'A.G. qui ressemble à la sélection élitiste [De Jong 75].

Ainsi, si la meilleure performance des individus de la nouvelle génération s'avère plus faible que la meilleure performance de ceux contenus dans l'ancienne génération, nous remplaçons le pire individu de la nouvelle génération par le meilleur individu de l'ancienne. En d'autres termes, si $P = (i_1, i_2, \ldots, i_k)$ est la population lors d'une génération telle que

$$f(i_1) \leq f(i_2) \leq \cdots \leq f(i_k)$$

et $P' = (i'_1, i'_2, \ldots, i'_k)$ est la population lors de la génération suivante telle que

$$f(i'_1) \leq f(i'_2) \leq \cdots \leq f(i'_k)$$

et si $f(i'_k) < f(i_k)$, alors on remplace i'_1 par i_k. Au cours des chapitres suivants, tous les résultats présentés sont obtenus en suivant ce principe de monotonie.

La différence entre notre méthode et celle développée par De Jong réside dans le fait que le meilleur individu est préservé dans tous les cas lors de la sélection élitiste.

5.3 Indexation binaire ou pondérée

Nous avons appliqué l'A.G. avec une indexation binaire ou pondérée, selon les indexations présentées au chapitre 1. Ainsi, pour les deux indexations, t_{ij} est égal à w_{ij}, défini selon l'équation (1) (binaire) ou (2) (pondérée).

Dans le cas binaire, t_{ij} est un gène. Mais, pour l'indexation pondérée, les gènes recouvrent plusieurs bits, et aucune coupure ne peut intervenir à l'intérieur d'un gène.

Pour réduire cette représentation, nous avons discrétisé les poids et ces valeurs réelles sont remplacées par des entiers. Pour les deux collections, nous avons choisi 10 classes, représentant le nombre minimal de classes qui altère le moins possible la performance de l'individu.

Ces valeurs entières entre 0 et 10 sont représentées de manière binaire à l'aide de 4 bits. Aucune précaution n'est prise pour éviter la coupure d'un gène parmi ces 4 bits.

Le croisement des valeurs entre 0 et 10 codifiées sur 4 bits peut donc engendrer des valeurs supérieures à 10. Cela n'entraîne aucun problème, ni pour notre modèle théorique, ni pour l'évaluation. Par contre, si les gènes représentaient 3 couleurs, codifiées sur 2 bits, le croisement pourrait engendrer une 4^{e} couleur, sans signification concrète.

5.4 Approche récurrente ou transiante

Conformément à la classification des algorithmes d'apprentissage, l'application de l'A.G. peut se réaliser selon une approche récurrente ou une stratégie transiante (section 1.5). A cette distinction, nous ajoutons la constitution de la population de départ (section 5.5).

Dans le cas **récurrent,** un individu est composé par les termes apparaissant dans l'ensemble des requêtes. La fonction de performance est définie comme la précision moyenne obtenue sur l'ensemble des requêtes.

Dans le cadre de l'approche **transiante,** on travaille uniquement sur **la requête courante.** Dans ce cas, la population de départ considère seulement les termes de la requête courante et la fonction de performance est définie comme la précision moyenne obtenue par l'individu sur la requête courante.

Nous avons vu au chapitre 2 que si la performance d'un algorithme d'apprentissage est faible selon une méthode rétrospective d'évaluation, il n'est pas nécessaire d'estimer sa performance par d'autres méthodes, car ces dernières produiront un taux d'erreur plus fort et exigeront un temps de calcul nettement plus important. Ainsi, les résultats de l'A.G. récurrent sont si peu concluants rétrospectivement que nous n'avons pas évalué cette approche au moyen d'autres méthodes.

Par contre, dans le cas transiant, les résultats rétrospectifs nous ont poussé à estimer le taux d'erreur de manière moins biaisée. Pour cela, nous avons utilisé une méthode similaire à celle décrite au chapitre 4. Elle consiste à "montrer à l'utilisateur" les 30 premiers documents retournés par le système et à appliquer l'algorithme d'apprentissage à partir des documents ainsi jugés pertinents. En fait, la liste des documents pertinents est modifiée à chaque évaluation à partir des 30 documents les mieux classés. L'individu le plus performant selon

les jugements de pertinence partiels est évalué finalement avec l'ensemble des jugements de pertinence, ce qui nous donne la performance de l'apprentissage (**transiant - utilisateur**).

5.5 Constitution de la population de départ

Le contenu de la population de départ constitue le troisième critère de classification des apprentissages par A.G.. Les individus peuvent être construits de plusieurs manières :

- par indexation automatique,
- aléatoirement,
- par indexation automatique partielle,
- à partir des jugements de pertinence.

L'indexation automatique nous donne un individu dont la performance correspond à la référence, c'est-à-dire que tous les résultats seront comparés à cette performance correspondant au modèle booléen ou vectoriel.

Généralement, lors de l'application de l'A.G., la population de base est souvent constituée par des individus choisis aléatoirement. Pour suivre ce principe, nous avons utilisé l'opération de mutation que nous avons appliquée 80'000 fois pour la collection CACM et 40'000 fois pour la collection CISI. Le nombre de mutations à été choisi en fonction du nombre de documents contenus dans chaque collection.

Une première population de départ est constituée de 7 individus construits aléatoirement, notée "**aléatoire**" (voir le tableau 24). En ajoutant l'individu indexé automatiquement à cette population, on en obtient une deuxième (voir les tableaux 25-27 sous l'étiquette "**indexé**").

Pour l'"**apprentissage des autres requêtes**" (**A.A.R.**), la population de base contient l'individu indexé et 7 autres individus construits à partir des jugements de pertinence répartis de la manière suivante :

```
identIndividu := 1;
pour chaque (requête q)
    pour chaque (terme t ∈ q)
        pour chaque (d, document pertinent pour q)
        {       ajouter (t, d) à l'individu;
                identIndividu : = (identIndividu mod 7) + 1;
        }
```

Fig. 8 Construction de la population de départ A.A.R.

Les indexations partielles sont construites uniquement à partir de la section logique d'un document soit

- le titre,
- les mots-clés donnés par les auteurs (seulement pour CACM),
- CR (Computer Review Abstract, seulement pour CACM).

Pour ces évaluations, notées par "**titre**", l'individu indexé par rapport au document entier est aussi utilisé.

Une dernière manière de construire la population de base utilise l'individu indexé et des individus "**vides**", dont tous les gènes ont la valeur 0. L'intérêt de cette méthode consiste dans le fait que le nombre de positions "1" attendues dans un individu est nettement inférieur à celui de "0", et bien que les "bonnes positions de 1" font qu'un individu est performant, les "bonnes positions de 0" aident à éliminer le bruit.

L'évaluation récurrente de l'A.G. ayant amélioré très peu la performance, elle est uniquement rétrospective. Pour une évaluation transiante, lors de la construction de la population de base pour chaque requête, les jugements de pertinence de la requête courante ne sont pas pris en compte. Ainsi, nous avons utilisé de nouveau la méthode du *leaving one out* (section 2.4).

5.6 Évaluation

Les résultats des différentes expériences décrites dans ce chapitre sont présentés dans les tableaux 24 - 27. La figure 9 présente une visualisation des résultats du tableau 28.

Tab. 24 Résultats de l'A.G. avec population de départ aléatoire selon différentes méthodes d'application en 10 générations

	CACM		CISI	
	Binaire	**Pondéré**	**Binaire**	**Pondéré**
départ	1.78	1.12	5.08	4.02
récurrent	1.80 (+1.17)	1.26 (+12.6)	5.49 (+8.02)	5.05 (+32.36)
transiant	2.37 (+33.24)	2.85 (+154.81)	6.54 (+28.68)	9.02 (+124.75)
transiant - utilisateur	2.35 (+32.16)	2.59 (+31.17)	6.17 (+21.42)	8.41 (+109.34)

L'application récurrente de l'A.G. n'a montré de changement significatif de la performance que dans un cas : l'apprentissage des autres requêtes (A.A.R.). Ceci n'est pas du à l'A.G., mais à la construction de la population de départ, qui suit le principe rétrospectif. La performance obtenue apparaît avant la première génération. L'application de la méthode du *leaving one out* à cette population correspond au cas transiant.

*Tab. 25 Résultats de l'A.G. **récurrent rétrospectif** selon différentes méthodes d'application en 10 générations*

	CACM		CISI	
	Binaire	Pondéré	Binaire	Pondéré
départ	21.00	32.70	13.66	19.83
indexé	21.00 (0.0)	32.70 (0.0)	13.66 (0.0)	19.91 (+0.43)
A.A.R.	86.4 (+311.36)	97.86 (+199.29)	63.90 (+367.91)	92.50 (+366.52)
titre	21.08 (+0.39)	32.78 (+0.25)	14.0 (+2.54)	19.83 (0.0)
vide	21.00 (0.0)	32.70 (0.0)	14.31 (+4.75)	19.83 (0.0)

*Tab. 26 Résultats de l'A.G. **transiant rétrospectif** selon différentes méthodes d'application en 10 générations*

	CACM		CISI	
	Binaire	Pondéré	Binaire	Pondéré
départ	21.00	32.70	13.66	19.83
indexé	21.84 (+4.01)	34.41 (+5.24)	15.40 (+12.75)	21.72 (+9.54)
A.A.R.	22.09 (+5.18)	38.16 (+16.7)	16.73 (+22.43)	24.90 (+25.59)
titre	24.00 (+14.28)	37.93 (+15.99)	17.20 (+25.95)	21.38 (+7.8)
vide	21.72 (+3.44)	36.00 (+10.11)	16.45 (+20.43)	22.88 (+15.38)

*Tab. 27 Résultats de l'A.G. **transiant utilisateur** selon différentes méthodes d'application en 10 générations*

	CACM		CISI	
	Binaire	Pondéré	Binaire	Pondéré
départ	21.00	32.70	13.66	19.83
indexé	21.24 (+1.14)	33.78 (+3.32)	13.61 (-0.37)	21.33 (+7.56)
A.A.R.	21.76 (+3.62)	37.85 (+15.77)	15.36 (+12.46)	22.01 (+11.02)
titre	23.47 (+11.73)	37.92 (+15.97)	14.42 (+5.58)	20.00 (+0.87)
vide	21.05 (+0.22)	34.00 (+4.0)	13.95 (+2.15)	21.11 (+6.46)

Les résultats montrent que dans le cas transiant, on peut obtenir des augmentations significatives de la performance, même avec l'intervention de l'utilisateur (tableau 28). Cependant, la méthode du *relevance feedback*, présentée au chapitre 4, montre des différences plus spectaculaires de la performance par rapport à la base. On peut supposer que l'A.G. serait avantagé par un plus grand nombre de générations, mais le temps de calcul nécessaire ne serait plus comparable aux autres algorithmes d'apprentissage.

Une conclusion qui s'impose à ce chapitre est que l'utilisation de la connaissance du domaine est utile, voir indispensable pour le bon fonctionnement de l'algorithme génétique pour deux raisons:

- la population de base "aléatoire" présente une très faible performance dans tous les cas, toutes les autres populations initiales étant donc, avantagées par l'inclusion de l'individu obtenu par indexation automatique;
- la population présentant, dans chaque cas, la meilleure performance est soit celle nommée "A.A.R.", soit celle nommée "titre" (voir figure 7); ces deux populations utilisent une connaissance supplémentaire du domaine, représentée par les jugements de pertinence (A.A.R.), ou par les sections logiques du document (titre).

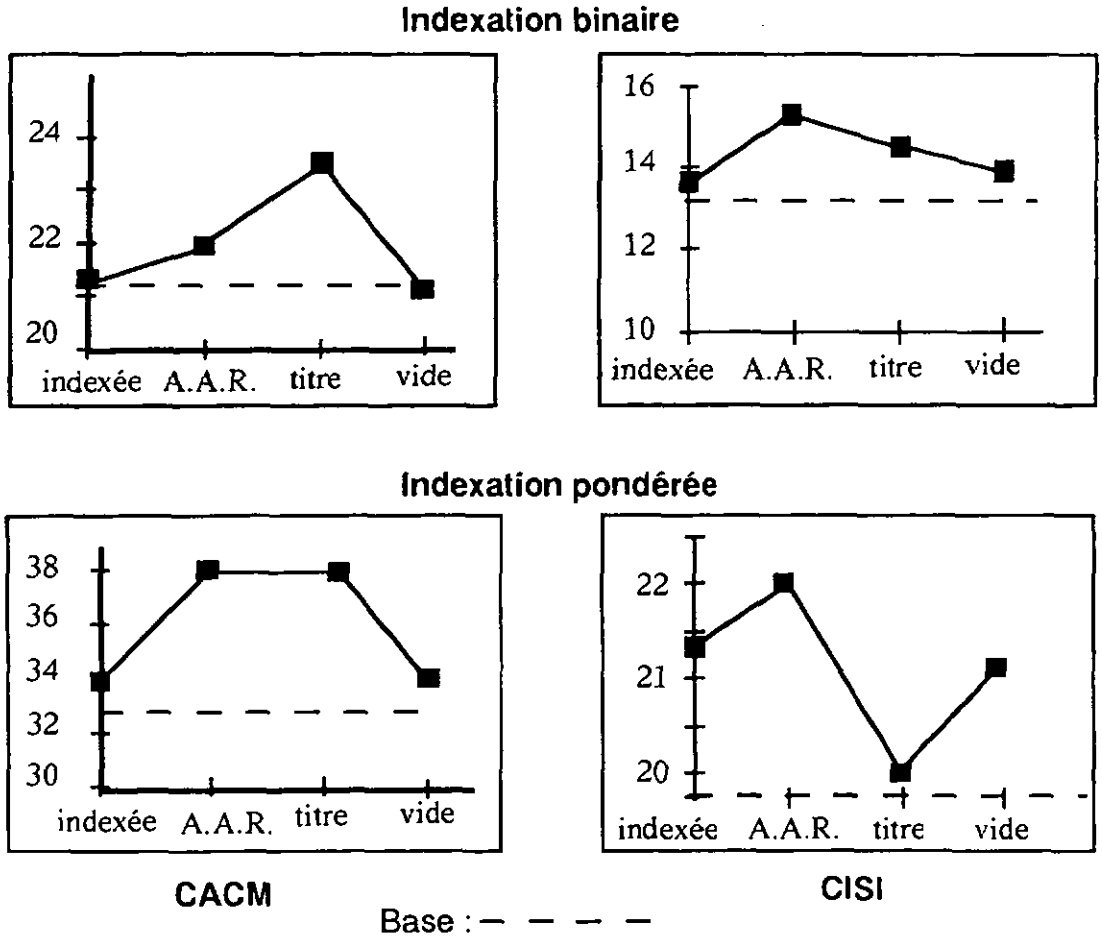

Fig. 9 Résultats transiants - utilisateur de l'A.G.

6. Analyse et modification possible de l'algorithme génétique

Lors du chapitre précédent nous avons vu que l'A.G. n'est pas très performant en recherche d'informations. Ce chapitre propose une analyse des raisons de ces échecs et des améliorations possibles de l'A.G.

Définition. Un *pattern* est un individu partiellement défini. Nous allons employer cette notion pour désigner une partie d'un individu.

Définition. Soient I_1 et I_2 deux individus. Les *patterns utiles réciproques* H_1 et H_2 représentent l'intersection de I_1 et I_2 avec l'individu de performance maximale qui peut être obtenu par n'importe quelle séquence de croisements de I_1 et I_2.

$$H_k = I_k \otimes I_m, \quad k = 1,2 \quad \text{où}$$
$$f(I_m) = \max\left\{ f(I_j) \,\middle|\, \forall I_j \text{ tel que } I_j - (I_1 \oplus I_2) = (0,0,...,0) = \bar{0} \right\}$$

dans laquelle $\oplus$ et $\otimes$ sont les opérateur booléens "ou" et "et" appliqués sur chaque gène.

Notations :

$C_m(algorithme)$ = complexité en moyenne de l'algorithme;

$C(algorithme)$ = complexité de l'algorithme dans le cas où les fonctions de complexité du pire cas, du cas moyen et du meilleur cas sont identiques;

$g(n) \in \Theta(h(n))$ si $\exists c_1, c_2 \in R$ tels que $\forall n \in N, \quad c_1 \cdot g(n) < h(n) < c_2 \cdot g(n)$

$\forall$ pattern H,

$o(H)$ est le nombre de positions (locus) fixées dans H (0 ou 1);

$\delta(H)$ est la longueur du pattern (dernière position définie - première position définie);

Si I est un intervalle (un nombre fixé de locus), et H un pattern, on note $I(H)$ la longueur de l'intersection (nombre de locus communs) de H avec I :

$$I(H) = o(H \cap I)$$

6.1 Relation entre la taille de l'individu et la performance de l'A.G.

Quelques chercheurs proposent de recourir à l'A.G. en recherche d'informations et une partie de leurs résultats tendent à démontrer qu'il est efficace. Malgré l'optimisme qui peut en découler, on peut remarquer que le paramètre *nombre de gènes* est très restreint dans la plupart des études. En voici quelques exemples :

- Goldberg [89] montre que l'A.G. fonctionne bien sur des individus de 30 gènes;
- Gordon [91] essaye son modèle sur 18 documents et environ 200 termes (~ 3600 gènes);

- Yang [92] utilise l'A.G. pour modifier les requêtes de la conférence TREC'1, et dans son article fait référence à 21 termes (gènes) au plus;
- Chen [95] essaye l'A.G. sur des collections de documents contenant au plus 10 documents et 33 mots-clés (330 gènes).

La question qui en résulte est de savoir si ces bons résultats obtenus avec de petits exemples nous garantissent son efficacité dans des cas réels, c'est-à-dire sur des collections de taille moyenne. Cette section essaie de répondre à cette question.

Recherches antérieures

Pour expliquer le bon fonctionnement de l'A.G., Goldberg [89] s'est orienté vers la structure des patterns qui survivent et a démontré que les chances de survie d'un pattern dépendent de deux paramètres, à savoir :

- sa performance : une meilleure performance entraîne une meilleure chance de survie;
- sa dispersion : des patterns dont les gènes sont bien groupés ont plus de chances de survie.

Parallèlement, d'autres auteurs [Mitchell 91], [Forrest 93] ont étudié l'influence de certaines caractéristiques de la fonction de performance sur la qualité de la solution trouvée par l'A.G. comme, par exemple, les régions isolées de haute performance, solutions multiples conflictuelles. En particulier, ces auteurs ont démontré que certaines catégories de fonctions mènent l'A.G. vers de mauvais résultats comme, par exemple, $f(x) = (x - 0.5)^2$.

Pour évaluer la difficulté d'un problème soumis à l'A.G., Koza [92] propose d'estimer le nombre d'individus nécessaires pour trouver la solution optimale avec une probabilité de 99%. Sa méthode combine le calcul probabiliste et l'expérimentation pratique.

Les résultats de nos recherches suggèrent une direction complémentaire à ces recherches précédentes. Nous avons étudié le comportement de l'approche génétique par rapport à la longueur de l'individu (nombre de gènes) et aux opérations appliquées, notamment le croisement.

Étude de complexité

Pour évaluer l'efficacité de l'A.G., nous devons connaître sa complexité étant donné la taille de l'individu (L), la complexité du calcul de la fonction de la performance ($\Theta(f)$), le nombre de générations ($n_{générations}$) et le nombre d'individus par génération ($n_{individus}$). Le calcul complet de cette complexité moyenne se trouve dans [Vrajitoru 96]. Il nous a permis d'affirmer que

$$C_m(L) = n_{générations} \cdot n_{individus} \cdot Ind(L), \quad \text{ou} \quad Ind(L) \in \Theta(f + n_{individus} + L) \qquad (13)$$

Selon cette formule, la complexité moyenne de l'A.G. est de l'ordre du nombre total d'individus générés ($n_{générations} \cdot n_{individus}$) multiplié par la fonction de complexité de la construction d'un nouvel individu, notée *Ind(L)*. Cette dernière est de l'ordre du plus grand des trois termes suivants : le calcul de la fonction de performance ($\Theta(f)$), l'opération de sélection ($\in \Theta(n_{individus})$) et l'opération de croisement (linéaire sur L). Dans notre étude, le nombre d'individus contenus dans une génération est plus petit que la taille d'un individu. On peut également remarquer que la taille de l'individu peut être réduite (représentation sous forme de matrice rare, comme dans notre cas) et alors L est remplacé par la longueur maximale effective d'une telle représentation.

Le premier terme, contenant le calcul de la fonction de performance, n'est pas toujours négligeable. Par exemple, dans notre cas, ce calcul s'avère coûteux :

$$C_m(f) \in \Theta\left((\mu_{term/req.} \cdot \mu_{doc/terme})^2\right) \in \Theta(L^2)$$

Ce que nous aimerions connaître dans la formule (13) est le nombre d'individus dans la population initiale et le nombre de générations nécessaires pour obtenir un individu très performant en fonction de L, la taille d'un individu. C'est un problème complexe en raison des nombreux facteurs qui influencent le déroulement de l'A.G. Les expériences (voir chapitre 5, tableaux 24-27) suggèrent que ce nombre augmente fortement avec L. Les paragraphes suivants tentent de démontrer qu'une grande valeur de L diminue les chances de trouver un individu performant. Nous allons commencer avec une analyse théorique suivie d'expérimentations.

Relation entre la taille de l'individu et le nombre de générations

Dans ce paragraphe, nous allons analyser le nombre de croisements nécessaires pour obtenir un individu performant à partir de deux individus donnés. De plus, notre analyse va mettre en évidence le paramètre déterminant ce nombre.

Prenons le cas de deux individus I_1 et I_2 et de leurs patterns utiles réciproques H_1 et H_2. On voudrait garder ces patterns dans la population résultant des générations suivantes et, si possible, les regrouper dans un seul individu. Pour cela, on essaie de former, par une séquence de croisements, un individu I_3 contenant à la fois H_1 et H_2.

Pour simplifier les choses et sans nuire à la généralité, considérons que $H_1 \otimes H_2 = \tilde{0}$. Décomposons H_1 et H_2 dans n, respectivement m groupes compacts intercalés les uns aux autres. Graphiquement, les deux individus et leurs patterns se présentent de la manière suivante :

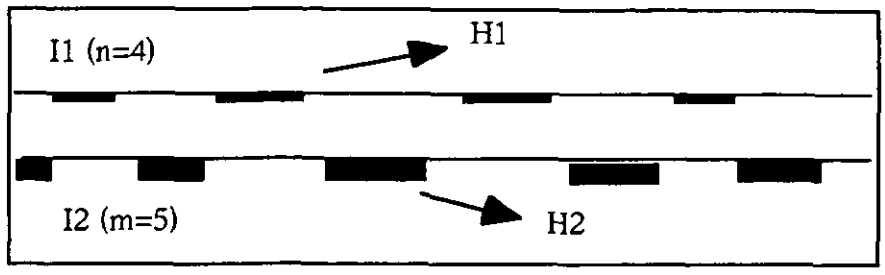

***Fig. 10** Emplacement des groupes compacts des deux individus*

Considérons le cas d'une opération de croisement. Dans le meilleur des cas, le site de croisement tiré au hasard apparaît entre deux groupes compacts consécutifs, l'un appartenant à H_1 et l'autre à H_2. Suite à l'opération, les deux groupes compacts sont réunis et l'on obtient deux nouveaux individus avec la somme totale des groupes compacts $= n + m - 1$. Par exemple, sur la base des deux individus de la figure 10, on peut obtenir

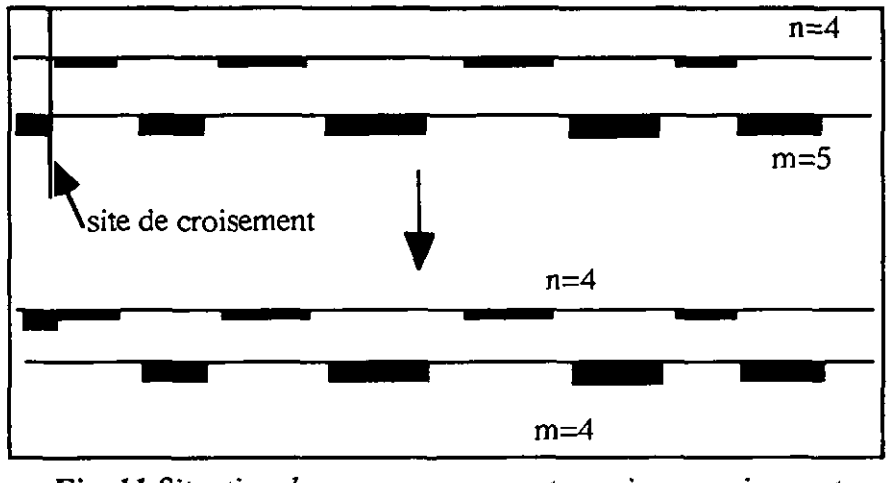

***Fig. 11** Situation des groupes compacts après un croisement*

Dans le cas simple ou $n = m = 1$ un seul croisement suffit, dans le meilleur des cas, pour obtenir l'individu "idéal" recherché. Par induction, pour n et m quelconques, on a besoin de $n + m - 1$ croisements. Pour effectuer un tel nombre de croisements entre deux individus, on a besoin d'un nombre de générations appartenant au moins à l'ordre du logarithme de $n + m - 1$ (voir section 6.2, paragraphe *emboîtement des croisements*).

On peut conclure que le nombre de générations nécessaires à une bonne performance est lié à la compacité de l'individu le plus performant et à sa dispersion dans les individus de la population de base, au sens de la figure 10. Ceci relie implicitement la performance de l'A.G. à la taille de l'individu car plus l'individu est grand, plus il y a de chances que l'information recherchée soit dispersée.

Une analyse de notre cas, présentée dans le tableau 28, nous montre que la dispersion de H_{max} est très grande pour les deux collections, ce qui rend difficile l'amélioration de la performance.

Tab. 28 Analyse des deux collections

	CACM	CISI
L	1092564	207320
#individus	$2^{1002852}=10^{328894.5}$	$2^{198560}=10^{62409.5}$
$o(H_{max})$	8292	9961
o(bruit)	1084272	197359
$\delta(H_{max})$	1091285	206573
compacité	faible	faible
$o(H_{max})/L$	0.007589	0.048
$o(H_{max})/\delta(H_{max})$	0.007598	0.0482
$o(H_{max})/o(bruit)$	0.007647	0.0505
taille de la population	8	8

Petit exemple

Pour vérifier pratiquement que la taille de l'ensemble de recherche détermine le bon fonctionnement de l'A.G., nous avons construit un exemple de petite taille à partir des deux collections. Dans ce but, nous n'avons utilisé qu'une ou deux requêtes par collection et un nombre réduit de documents choisis tels que tous les documents pertinents soient sélectionnés. L'indexation utilisée pour cet exemple est uniquement binaire.

Le tableau 29 indique le nombre de requêtes, l'identificateur des requêtes, le nombre de termes, le nombre de documents et le nombre de documents pertinents utilisés dans notre exemple.

Les résultats présentés dans le tableau 30 démontrent une évolution spectaculaire de la performance dans le cas d'un individu de petite taille.

Tab. 29 Structure de l'exemple réduit

	CACM	CISI
Nombre de requêtes	2	1
Identificateurs des requêtes	1 + 3	4
Nombre de termes	10 + 7	10
Nombre de documents	50	40
Nombre de documents	5 + 6	8

Pour mieux illustrer l'effet de l'augmentation de la taille de l'individu sur l'efficacité de l'A.G., le tableau 31 présente une comparaison des résultats des trois catégories d'évaluation selon la taille : récurrent, transiant et avec un petit exemple. La population de base est indexée pour les apprentissages récurrent set transiants, et est aléatoire pour le petit exemple. Ceci souligne encore l'efficacité de l'A.G. sur un exemple restreint.

Tab. 30 Évaluation des exemples réduits

CACM		CISI	
#générations	Précision (%change)	#générations	Précision (%change)
0	22.96	0	21.72
2	22.97 (+0.03)	1	27.71 (+27.62)
3	32.04 (+39.52)	2	30.14 (+38.79)
11	41.10 (+78.97)	7	42.74 (+96.82)
		10	47.59 (+119.14)
		17	50.90 (+134.4)
		20	56.31 (+159.32)

Tab. 31 Comparaison des différents résultats selon la taille de l'individu

	Départ	10-11 gen.	#docs	#termes	L
CACM					
récurrent	21.00	21.00 (0.0)	3204	341	1092564
transiant	21.00	21.84 (+4.01)	3204	11.24	36013
petit ex.	22.96	41.10 (+78.97)	50	17	850
CISI					
récurrent	13.66	13.66 (0.0)	1460	142	207320
transiant	13.66	15.40 (+12.75)	1460	7.43	10846
petit ex.	21.72	42.74 (+96.82)	40	10	400

6.2 Grande population de départ ou beaucoup de générations ?

Le nombre d'individus contenus dans la population de base représente un paramètre important pour tout A.G. Le choix de ce paramètre s'avère limité parfois par la mémoire disponible, parfois par d'autres conditions. Dans cette section, nous analyserons l'influence du nombre d'individus par génération sur la performance de l'algorithme. En d'autres termes, il s'agit de savoir si, pour un temps de calcul donné, une population de départ importante mais avec un nombre limité de générations apporte de meilleurs résultats qu'une population plus restreinte mais avec un nombre de générations plus important.

Expériences

Pour essayer de répondre objectivement à cette question, deux contraintes doivent être respectées :

1. Le nombre d'individus générés doit être le même dans chaque expérience ($n_{générations} \cdot n_{individus}$). Avec un nombre de générations constant, il est évident

que les expériences conduites avec une plus grande population de départ doivent mener à une plus grande performance.

2. L'information contenue dans la population de base doit être la même. Il serait injuste de comparer des résultats obtenus à partir de données de départ différentes. Pour respecter cette contrainte, nous pouvons créer la première génération, soit :
   - avec la population de départ nommée "vide" (voir section 5.5), en variant le nombre d'individus composés uniquement de "0",
   - par l'apprentissage à l'aide des autres requêtes (A.A.R.), en variant le nombre d'individus construits sur la base des jugements de pertinence (originellement 7 dans la figure 8, section 5.5).

Le tableau 32 montre les résultats des deux méthodes pour les deux collections selon le nombre d'individus par génération et le nombre de générations.

***Tab. 32** Résultats de la variation de la taille de la population de base*

	CACM		CISI	
Individus/ générations	A.A.R.	Vide	A.A.R.	Vide
1/0	36.26	32.70	20.29	19.83
2/40	37.24 (+2.71)	33.05 (+1.09)	22.15 (+9.17)	21.06 (+6.21)
4/20	35.37 -> 37.37 (+5.65)	33.59 (+2.74)	20.01 -> 21.71 (+8.53)	21.86 (+10.24)
6/13	35.10 -> 37.96 (+8.15)	35.18 (+7.6)	20.80 -> 23.21 (+11.61)	21.75 (+9.7)
8/10	34.42 -> 38.16 (+10.84)	36.00 (+10.11)	20.52 -> 24.9 (+21.34)	22.88 (+15.38)
10/8	35.48 -> 39.43 (+11.15)	36.17 (+10.63)	20.48 -> 24.14 (+17.87)	22.85 (+15.26)
14/6	34.54 -> 40.62 (+17.61)	36.71 (+12.28)	20.55 -> 24.77 (+20.5)	23.73 (+19.68)
16/5	34.72 -> 37.96 (+9.34)	37.65 (+15.16)	20.54 -> 24.24 (+18.02)	22.81 (+15.04)
20/4	34.36 -> 41.61 (+21.11)	38.30 (+17.13)	21.48 -> 24.96 (+16.23)	22.88 (+15.4)

Les résultats montrent un maximum de performance pour la collection CACM avec 20 individus / 4 générations. Pour la collection CISI, le maximum est atteint avec 8 individus / 10 générations pour l'apprentissage des autres requêtes (A.A.R.), et 14 individus / 6 générations pour l'apprentissage débutant avec des individus vides.

Le tableau 33 présente une comparaison entre les méthodes ayant la performance maximale et minimale. Les deux premières lignes indiquent la taille de la population de base et le nombre de générations (taille/générations) pour la performance minimale et maximale. La troisième ligne contient le pourcentage de différence en faveur de l'application de performance maximale. Les quatre lignes qui suivent montrent le nombre de requêtes pour lesquelles le modèle de performance maximale est respectivement, meilleur, significativement meilleur, moins bon, significativement moins bon que le modèle ayant la performance minimale. Les quatre dernières lignes montrent le nombre de requêtes pour lesquelles, lors de l'application de performance minimale et maximale, il y a eu une amélioration et une amélioration significative de la performance.

Cette analyse montre qu'un bon choix des paramètres peut influencer la performance de l'A.G. et suggère qu'une population de départ (nombre d'individus par génération) plus importante est avantageuse par rapport à un grand nombre de générations.

***Tab. 33** Comparaison par requêtes entre les applications de performance maximale et minimale*

	CACM		CISI	
	A.A.R.	Vide	A.A.R.	Vide
min (ind/gén)	2/40	2/40	4/20	2/40
max (ind/gén)	20/4	20/4	8/10	14/6
max/min - 100%	+11.73%	+15.87%	+14.67%	+12.69%
max - min > 0	38	44	24	30
max - min > 5%	24	28	14	22
max - min < 0	10	1	9	4
max - min < -5%	7	1	5	2
min > base	10	9	23	18
min - base > 5%	7	5	16	8
max > base	48	45	33	33
max - base > 5%	32	30	24	27

Plusieurs facteurs agissent sur la performance de l'A.G. Dans les paragraphes suivants, nous allons en analyser quelques uns.

Convergence

Le premier facteur est la convergence de l'algorithme vers l'individu de performance maximale I_{max}. La croissance exponentielle du nombre d'occurrences des bons individus dans les générations est bien connue dans la littérature [Koza 92]. Nous allons étudier la rapidité de cette croissance en fonction du nombre d'individus par génération.

Prenons le cas simple d'une population contenant un individu de performance f_{max} et *n-1* individus de performance f_{min}, avec $f_{max} > f_{min}$. Soit e_k le nombre attendu d'individus de performance f_{max} pour la génération *k*. Soit *P(I)* la probabilité que l'individu *I* apparaisse dans la génération suivante (probabilité proportionnelle à *f(I)*). Nous avons :

$$e_0 = 1$$

$$e_{k+1} = n \cdot P(I_{max}) = n \cdot \frac{e_k \cdot f_{max}}{e_k \cdot f_{max} + (n - e_k \cdot f_{min})}$$

Si la suite e_k converge, on peut alors calculer la limite avec l'équation

$$e_{k+1} = e_k \Rightarrow$$

$$\frac{n \cdot e_k \cdot f_{max}}{e_k \cdot f_{max} + (n - e_k) \cdot f_{min}} = e_k \quad \Rightarrow \quad e_{lim1} = 0, \quad \text{sinon} \quad (e_k \neq 0)$$

$$\frac{n \cdot f_{max}}{e_k \cdot f_{max} + (n - e_k) \cdot f_{min}} = 1 \Rightarrow n \cdot f_{max} = e_k \cdot f_{max} + (n - e_k) \cdot f_{min} \Rightarrow$$

$$n \cdot (f_{max} - f_{min}) = e_k \cdot (f_{max} - f_{min}) \Rightarrow \quad \text{si} \quad f_{max} \neq f_{min} \quad \text{alors} \quad e_{lim2} = n$$

e_{lim1} et e_{lim2} sont les deux limites possibles de la suite. Nous allons étudier les conditions qui mènent à chacun des deux cas. En fait, si la suite est monotone ascendante, la limite est *n*, et si elle est monotone décroissante, la limite est 0. La condition deviendrait :

$$e_{k+1} \geq e_k \Leftrightarrow \frac{n \cdot e_k \cdot f_{max}}{e_k \cdot f_{max} + (n - e_k) \cdot f_{min}} \geq e_k \quad \Leftrightarrow$$

$$\frac{n \cdot f_{max}}{e_k \cdot f_{max} + (n - e_k) \cdot f_{min}} \geq 1 \Leftrightarrow n \cdot f_{max} \geq e_k \cdot f_{max} + (n - e_k) \cdot f_{min} \Leftrightarrow$$

$(n - e_k) \cdot f_{max} \geq (n - e_k) \cdot f_{min}$ qui est vrai si $f_{max} \geq f_{min}$

Pour résoudre l'inégalité, nous avons tenu compte du fait que $e_k > 0$ et que $n - e_k > 0$.

Cette dernière inégalité veut dire que l'individu de performance minimale a tendance a disparaître et que la population converge au fil des générations vers l'individu le plus performant. Cette tendance s'accentue plus fortement si *n* est petit. Par exemple, pour la population "vide", nous avons

Tab. 34 CACM : $f_{max} = 32.70$, $f_{min} = 1.43$

n	e_0	e_1	e_2
2	1	1.92 (95.8%)	2.00 (99.8%)
4	1	3.64 (90.9%)	3.96 (99.1%)
20	1	10.88 (54.4%)	19.26 (96.3%)

Tab. 35 CISI : $f_{max} = 19.83$, $f_{min} = 4.65$

n	e_0	e_1	e_2	e_3
2	1	1.60 (80.2%)	1.87 (93.6%)	1.96 (98.0%)
4	1	2.34 (58.5%)	3.43 (85.7%)	3.84 (96.0%)
20	1	3.67 (18.3%)	9.26 (46.3%)	15.71 (78.5%)

En reprenant le tableau 33, on constate qu'un nombre restreint de requêtes a une performance qui s'améliore dans le modèle à performance minimale ("min", n = 2/4). Ce nombre s'avère plus faible pour la collection CACM que pour le corpus CISI. En effet, le rapport entre f_{max} et f_{min} est plus important dans le cadre de la CACM, ce qui entraîne une convergence plus rapide de la population.

Position de croisement

Nous avons vu, au cours de la section 6.1, que la position de croisement est importante pour la performance de l'algorithme. Dans les conditions décrites lors de l'analyse de la convergence, la probabilité d'avoir une position de croisement à une bonne place augmente avec la taille de la population n. Pour deux individus quelconques, soit p_1 la probabilité de tirer au sort une "bonne" position de croisement pour une opération. Alors, pour deux croisements ayant comme parents les mêmes individus, nous avons :

$$p_2 = p_1 + p_1 - p_1^2 = 1 - (1 - p_1)^2.$$

De manière générale, pour $n > 3$

$$p_{n+1} = p_1 + p_{n-1} - p_1 \cdot p_{n-1} = 1 - (1 - p_1)(1 - p_{n-1}) \Rightarrow$$

$$p_n = 1 - (1 - p_1)^n$$

Pour $n \to \infty$ la suite $p_n \to 1$ car $1 - p_1 < 1 \Rightarrow (1 - p_1)^n \to 0$.

Ce deuxième facteur contribue à la bonne performance des populations nombreuses.

Emboîtement des croisements

Dans cette sous-section, nous allons montrer que, quelle que soit la taille de la population de départ, le nombre minimal de générations nécessaires pour arriver au résultat optimal est le même.

Nous allons montrer que pour effectuer trois croisements, deux générations sont nécessaires et suffisantes. La figure 12 montre le résultat des trois croisements appliqués successivement, c'est-à-dire dans au moins trois générations. Chaque site (1, 2, 3) de croisement correspond à un croisement différent, et l'ordre d'application des croisements n'a pas d'importance. Ces croisement nécessitent la construction de trois générations.

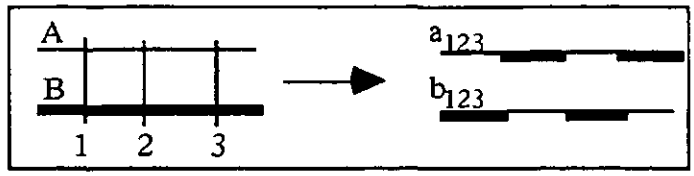

Fig. 12 Trois croisements à 1 site

Est-ce que les deux individus ainsi créés (a_{123} et b_{123}) peuvent aussi être obtenus en deux générations ? La réponse est affirmative et nous allons l'expliquer à l'aide de la figure 13. On applique d'abord les croisement 1 et 3 aux deux parents (sélectionnés ainsi chacun deux fois dans la phase de reproduction) et l'on obtient les fils intermédiaires a_1, b_1, a_3, et b_3. Ensuite, on applique le croisement noté "2" aux individus a_1 et a_3, ce qui nous donne l'individu a_{123} (et un deuxième sans intérêt). Finalement on applique de nouveau le croisement "2" , cette fois aux individus b_1 et b_3, dont en résulte b_{123}. Comme les croisements "1" et "3" peuvent s'appliquer indépendamment, ainsi que les deux croisements de site "2", toutes ces opérations peuvent se faire par la construction de deux nouvelles générations.

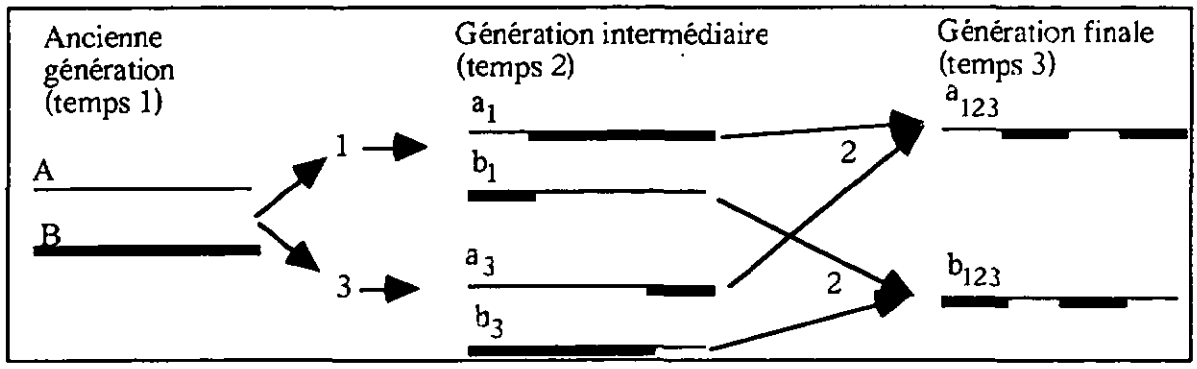

Fig. 13 Emboîtement des croisements

Supposons que nous ayons deux individus et que pour obtenir un individu maximal nous ayons besoin d'effectuer m croisements successifs. En suivant le raisonnement précédent, les m croisements peuvent être organisés dans une structure d'arbre binaire ayant m noeuds (figure 14). Chaque niveau de l'arbre représente une génération. Il est bien connu qu'un tel arbre possède $\lceil m/2 \rceil$ feuilles et $\lceil \log_2(\lceil m/2 \rceil) \rceil + 1 = \lceil \log_2 m \rceil$ niveaux. Cette valeur représente le nombre minimal de générations nécessaires pour obtenir l'individu optimal, quel que soit le nombre d'individus par génération.

Si l'on garde constant le nombre de générations multiplié par la taille de la population de départ, les considérations antérieures nous conduisent à conclure que la performance de l'A.G. devrait diminuer face à un nombre trop restreint de générations.

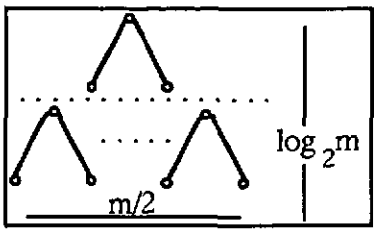

Fig. 14 Arbre binaire à m noeuds

Les trois paramètres analysés nous indiquent que, dans les conditions énoncées pour les expériences, la performance de l'A.G. devrait être faible pour une population de départ de petite taille, devrait augmenter avec ce paramètre, atteindre un maximum et diminuer ensuite. Cette courbe semble apparaître pour l'évaluation avec la population de départ "vide" pour la collection CISI (voir figure 15), même si nous ne pouvons pas tirer une vraie conclusion dans ce sens sur un si petit nombre d'individus par génération.

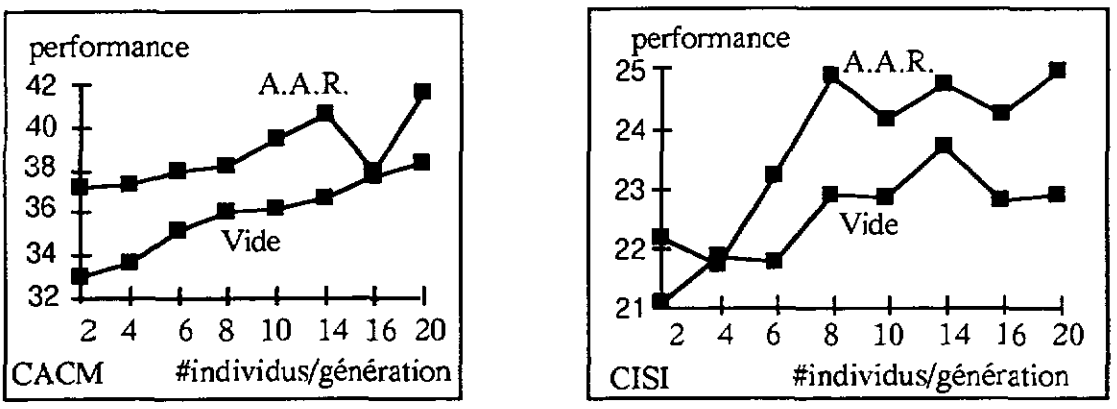

Fig. 15 Visualisation des courbes de performance pour les deux collections

6.3 Nouvelle méthode de croisement

Cette section est consacrée à l'analyse d'un certain phénomène qui survient au cours du croisement et à la description d'une nouvelle méthode de croisement.

L'opération de croisement, telle que nous l'avons présentée et utilisée, constitue la manière la plus simple de considérer cette opération. Il existe de nombreuses autres formes, et dans les paragraphes suivants nous en présentons quelques-unes.

La première généralisation est le **croisement généralisé** [De Jong 75]. Pour cette opération, on choisit au hasard un nombre de positions de croisement et on échange les parties d'ordre pair ainsi obtenues, comme le montre la figure 16.

Cette opération est équivalente à l'application répétée du croisement à un site.

L'opération de **croisement restreint** est identique au croisement à un site, avec une différence dans le choix du point de croisement. Ainsi, ce point peut seulement être choisi entre la première et la dernière des positions où les parents ont des gènes différents. Si k est la position de croisement, alors

$$\min[i|parent_1(i) \neq parent_2(i)] \leq k \leq \max[i|parent_1(i) \neq parent_2(i)]$$

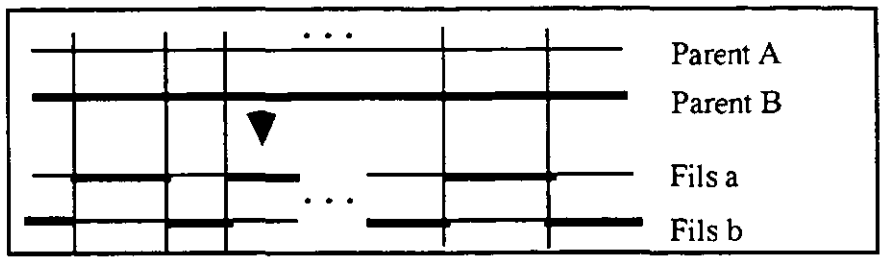

Fig. 16 Croisement à plusieurs positions

L'opérateur de **croisement uniforme** consiste à choisir chaque gène du fils à partir du gène correspondant de l'un ou l'autre des parents pris au hasard [Syswerda 89]. Pour toute position i, telle que $1 \leq i \leq L$, soit *rand* un nombre aléatoire de valeur 0 ou 1. Alors

$$fils(i) = \begin{cases} parent_1(i) \text{ si } rand = 0 \\ parent_2(i) \text{ si } rand = 1 \end{cases}$$

L'opérateur de **fusion** produit un seul fils à partir de deux parents [Beasley 95]. Pour chaque gène du fils, on choisit la valeur du gène de l'un ou l'autre des parents, avec la probabilité de chaque parent proportionnelle à sa performance :

$$\forall i, 1 \leq i \leq l, \quad fils(i) = \begin{cases} parent_1(i) \text{ de probabilité } \dfrac{f(parent_1(i))}{f(parent_1(i)) + f(parent_2(i))} \\ parent_2(i) \text{ de probabilité } \dfrac{f(parent_2(i))}{f(parent_1(i)) + f(parent_2(i))} \end{cases}$$

Analyse du croisement

Lors du croisement, on espère qu'au moins l'un des fils présentera une meilleure performance que celle de ses parents. En pratique, le contraire arrive assez souvent et ce phénomène nous a poussé vers cette analyse.

Pour un croisement des parents I_1 et I_2, considérons les patterns utiles réciproques de ces individus et nommons les $H_1 \subset I_1$ et $H_2 \subset I_2$. Dans notre cas, ces patterns représentent la présence des termes de la requête dans la description des documents pertinents ou l'absence de ces termes dans la description des documents non pertinents. On essaie de former, par une suite de croisements, un individu I_3 contenant à la fois H_1 et H_2.

Considérons que $o(H_1) > o(H_2)$. Si la performance de chaque individu dépend de la longueur de son pattern utile, (donc de $o(H_i), i = 1,2$), nous avons intérêt à obtenir par croisement de H_1 et H_2, un pattern H_3 avec $o(H_3) > o(H_1)$.

Le point de croisement divise H_1 et H_2 en deux parties de longueur

$$\alpha \cdot o(H_1) \quad \text{et} \quad (1-\alpha) \cdot o(H_1)$$

$$\beta \cdot o(H_2) \quad \text{et} \quad (1-\beta) \cdot o(H_2)$$

avec α et β deux constantes telles $0 \leq \alpha, \beta \leq 1$. Alors, on peut calculer que :

$$o(H_3) = \alpha \cdot o(H_1) + (1-\beta) \cdot o(H_2)$$

et

$$o(H_4) = (1-\alpha) \cdot o(H_1) + \beta \cdot o(H_2)$$

En essayant de remplir la condition $o(H_3) > o(H_1)$, on obtient :

$$o(H_3) > o(H_1) \Leftrightarrow$$
$$\alpha \cdot o(H_1) + (1-\beta) \cdot o(H_2) > o(H_1) \Leftrightarrow$$
$$(1-\beta) \cdot o(H_2) > (1-\alpha) \cdot o(H_1) \Leftrightarrow$$
$$\frac{(1-\beta)}{(1-\alpha)} > \frac{o(H_1)}{o(H_2)}$$

Cette équation nous indique que la chance d'obtenir un meilleur individu que ses parents augmente avec la différence entre α et β. Nous avons considéré le nombre de termes communs aux requêtes et aux documents pertinents dans l'individu indexé (H_1) en rapport avec le même nombre dans les individus aléatoires (H_2) pour les deux collections.

CACM

$$\frac{1588}{458} \leq \frac{o(H_1)}{o(H_2)} \leq \frac{1588}{384} \Leftrightarrow$$
$$3.46 \leq \frac{o(H_1)}{o(H_2)} \leq 4.13$$

CISI

$$\frac{2699}{1151} \leq \frac{o(H_1)}{o(H_2)} \leq \frac{2699}{1083} \Leftrightarrow$$
$$2.34 \leq \frac{o(H_1)}{o(H_2)} \leq 2.49$$

Généralement, la population de base pour un A.G. est créée d'une manière aléatoire (par exemple, par un nombre de mutations). C'est l'uniformité de la population qui rend parfois l'A.G. inefficace, comme nous allons l'expliquer.

Imaginons $H_{\max}$ le pattern ayant la performance maximale. Nous savons que H_1 et H_2 sont inclus dans $H_{\max}$. L'uniformité de la population fait que l'on s'attend à avoir dans un intervalle I approximativement

$$I(H_1) \approx I(H_{\max}) \cdot \frac{o(H_1)}{o(H_{\max})} \quad \text{et} \quad I(H_2) \approx I(H_{\max}) \cdot \frac{o(H_2)}{o(H_{\max})} \tag{14}$$

ce qui veut dire que la proportion entre deux individus est approximativement égale à la proportion des intersections avec l'intervalle :

$$\frac{o(H)}{o(H_{\max})} \approx \frac{o(H \cap I)}{o(H_{\max} \cap I)}$$

Cette observation nous permet de calculer

$$\alpha \approx \frac{I(H_1)}{o(H_1)} = \frac{I(H_{\max})}{o(H_{\max})} = \frac{I(H_2)}{o(H_2)} \approx \beta \Leftrightarrow$$

$$\alpha \approx \beta$$

Comme les coefficients α et β sont presque égaux, le rapport (1-α)/(1-β) est proche de 1 et, selon la loi des grands nombres, la probabilité d'avoir un tel comportement augmente avec L . Dans ce cas, il n'est plus possible de surpasser le rapport entre $o(H_1)$ et $o(H_2)$, indépendamment de la position de croisement choisie.

L'hypothèse sous-jacente à l'équation (14) s'avère difficile à vérifier. Pour s'assurer que cette hypothèse demeure plausible, nous avons analysé la fréquence des paires (terme appartenant à la requête, document pertinent pour la requête) classées dans des intervalles définis selon le nombre du document et qui forment une partition de chaque collection. Pour cela, nous avons effectué des histogrammes des ces paires (terme, document), en considérant l'indexation automatique et une indexation aléatoire pour les deux collections. Afin de travailler avec un nombre presque identique d'intervalles (classes) pour les deux collections, nous avons groupé les documents en classes de 300 documents à numéros consécutifs pour la CACM et de 150 documents à numéros consécutifs pour la CISI. Quelques résultats sont présentés dans le tableau 36 et dans la figure 17.

La colonne "#paires" représente le nombre total de paires (terme occurant dans une requête, document pertinent pour cette requête) trouvés dans chaque classe, pour toute requête R. La moyenne, notée μ représente l'identificateur moyen de classe et σ l'écart type :

$$\mu = \frac{\sum_{\forall classe_i} (nombre\ de\ paires \in classe_i) * i}{\sum_{\forall classe_i} nombre\ de\ paires \in classe_i}$$

$$\sigma = \sqrt{\frac{\sum_{\forall classe_i} \left((nombre\ de\ paires \in classe_i) * (i - \mu)\right)^2}{\sum_{\forall classe_i} nombre\ de\ paires \in classe_i}}$$

On remarque que la classe moyenne est pratiquement la même pour les trois individus, que l'écart type est stable, et que les graphiques des histogrammes se ressemblent beaucoup. Par conséquent, les coefficients α et β ont des valeurs similaires. De plus, le nombre de documents dans chaque classe correspond en général à la proportion respective de $H_{\max}$.

Ceci confirme l'hypothèse émise (équation 14) et explique le fait que, dans une grande majorité des croisements observés, la performance des fils est généralement plus grande que la performance du pire parent mais souvent, plus petite que la performance du meilleur parent. Il est alors évident qu'un accroissement de la performance s'avère difficile.

Tab. 36 Moyenne et écart-type

	CACM			CISI		
individu	#paires	μ	σ	#paires	μ	σ
H_{max}	8292	8.14	2.45	9961	4.84	3.53
Aléatoire	422	7.91	3.43	1112	4.81	3.48
Indexé	1588	8.13	1.93	2669	4.87	3.54

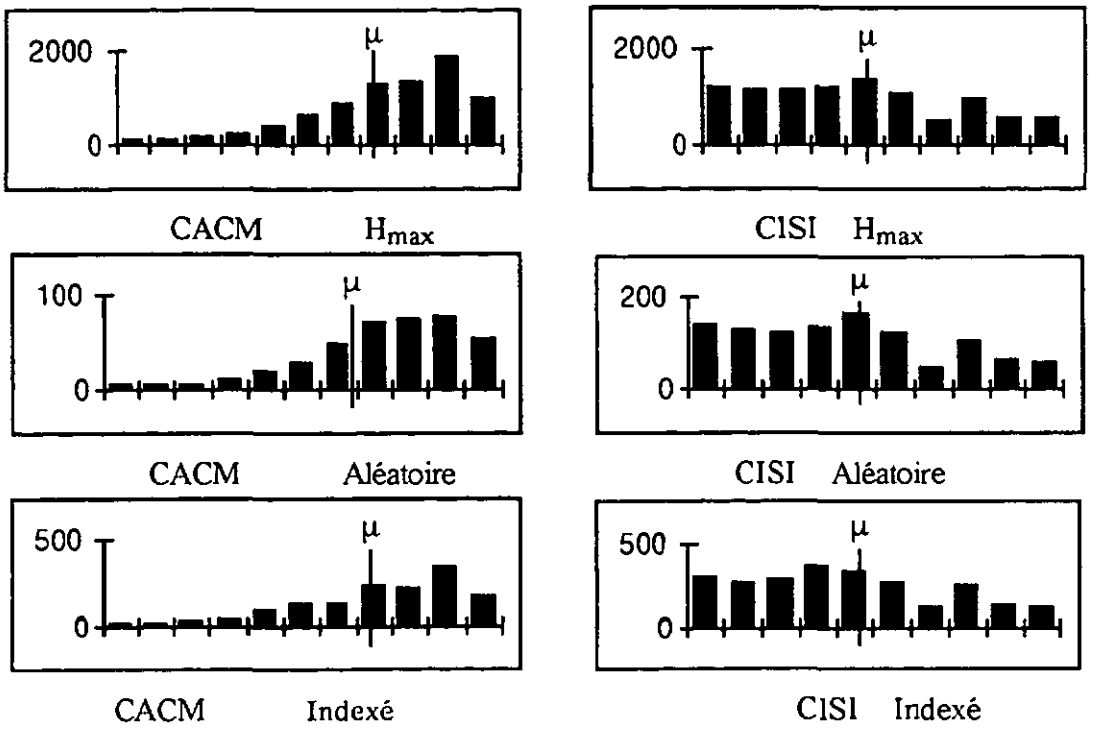

Fig. 17 Histogrammes

Description de notre méthode

L'analyse décrite à la section précédente nous a conduit vers une nouvelle méthode de croisement, que nous avons appelée **croisement dissocié**. Elle consiste à dissocier les paramètres α et β indépendamment de la nature des individus. Pour cela on choisit non pas une position de croisement, mais deux.

Soient I_1 et I_2 deux individus, et $1 \leq i \leq L$ et $1 > j \leq L$ deux positions de croisement. Les nouveaux individus I_3 et I_4 sont créés de la manière suivante :

$$I_3(k) = \begin{cases} I_1(k) & \text{si } k \leq i \\ I_1(k) \text{ ou } I_2(k) & \text{si } i < k \leq j \\ I_2(k) & \text{si } k > j \end{cases} \qquad I_4(k) = \begin{cases} I_2(k) & \text{si } k \leq i \\ 0 & \text{si } i < k \leq j \\ I_1(k) & \text{si } k > j \end{cases}$$

La performance des individus engendrés par l'A.G. modifié au cours des générations montrent une plus grande diversité. Les résultats de la nouvelle méthode de croisement

montrent une amélioration presque systématique par rapport à la performance de l'A.G. classique. Les tableaux 37 - 39 présentent une comparaison entre les deux méthodes.

Tab. 37 Comparaison entre les deux méthodes pour la population de départ aléatoire en 10 générations

	CACM		CISI	
	Classique	Dissocié	Classique	Dissocié
rétrospectif	2.85	6.14 (+115.44)	9.02	10.85 (+20.29)
utilisateur	2.59	4.28 (+165.25)	8.41	10.84 (+28.89)

Tab. 38 Résultats de l'A.G. rétrospectif transiant en 10 générations

	CACM		CISI	
	Classique	Dissocié	Classique	Dissocié
indexé	34.41	41.17 (+19.65)	21.72	24.15 (+21.77)
A.A.R.	38.16	43.95 (+15.17)	24.90	25.74 (+29.82)
titre	37.93	42.01 (+10.76)	21.38	23.18 (+16.88)
vide	36.00	42.46 (+17.94)	22.88	23.75 (+19.75)

Tab. 39 Résultats de l'A.G. transiant utilisateur en 10 générations

	CACM		CISI	
	Classique	Dissocié	Classique	Dissocié
indexé	33.78	40.12 (+18.76)	21.33	23.14 (+8.49)
A.A.R.	37.85	42.90 (+13.37)	22.01	24.57 (+11.63)
titre	37.92	41.49 (+9.41)	20.00	21.82 (+9.1)
vide	34.00	41.38 (+21.71)	21.11	22.00 (+4.22)

Comparaison avec le *relevance feedback*

Afin d'évaluer la performance de l'A.G. en apprentissage transiant, nous l'avons comparé avec le *relevance feedback*. Pour chaque collection, nous avons choisi les deux meilleures évaluations de l'A.G. Ensuite, nous avons fait varier le nombre de documents "jugés par l'utilisateur" (voir la section 5.4, suivant le même nombre de documents que pour le *relevance feedback*). Les tableaux 40 et 41 présentent la comparaison entre l'A.G. traditionnel, le croisement *dissocié* en 10 générations, et le *relevance feedback* en évaluation résiduelle.

A cause de l'évaluation résiduelle du *relevance feedback*, les méthodes comparées n'ont pas la même performance de départ. On ne peut donc pas les comparer directement. Par contre, en regardant les pourcentages d'augmentation par rapport à la performance de départ de chaque méthode, on peut dire que, malgré l'amélioration de performance apportée à l'A.G. par la nouvelle opération de croisement, la méthode du *relevance feedback* reste meilleure.

Tab. 40 Comparaison entre les méthodes pour la collection CACM

	5	10	15	20	30
titre	36.01 (+10.59)	36.58 (+12.35)	37.03 (+13.74)	36.89 (+13.31)	37.92 (+15.97)
A.A.R.	38.14 (+16.65)	37.15 (+14.08)	38.67 (+18.26)	38.13 (+17.09)	37.85 (+15.77)
titre diss.	38.95 (+19.62)	42.57 (+30.74)	41.14 (+26.34)	42.0 (+29.0)	41.49 (+26.89)
A.A.R. diss.	41.57 (+27.67)	44.78 (+37.53)	44.07 (+35.36)	44.64 (+37.1)	42.9 (+31.2)
Rel FB - utilisateur	25.50 (+32.88)	23.85 (+43.05)	23.83 (+53.08)	22.1 (+64.01)	21.87 (+86.95)

Tab. 41 Comparaison entre les méthodes pour la collection CISI

	5	10	15	20	30
indexé	19.27 (-4.23)	19.82 (-1.5)	20.67 (+2.71)	20.89 (+3.83)	21.33 (+7.56)
A.A.R.	21.19 (+6.88)	22.03 (+9.48)	22.45 (+11.56)	22.08 (+9.73)	22.01 (+11.02)
indexé diss.	20.86 (+3.69)	21.85 (+8.58)	23.79 (+18.22)	23.16 (+15.09)	23.14 (+16.71)
A.A.R. diss.	22.55 (+12.08)	24.81 (+23.3)	25.44 (+26.46)	25.24 (+25.44)	24.57 (+23.9)
Rel FB - utilisateur	22.31 (+26.16)	22.71 (+36.02)	23.24 (+42.28)	22.83 (+56.14)	23.73 (+81.62)

En conclusion, le nombre de documents utilisés dans l'apprentissage influence moins l'A.G. que le *relevance feedback*. Ainsi, on peut imaginer une application de l'A.G. même si l'usager juge peu de documents ("loi du moindre effort").

Conclusion

Au cours de cette thèse, nous nous sommes proposé d'analyser différents algorithmes d'apprentissage automatique appliqués en recherche d'informations. Plus précisément, comment peut-on tirer profit des jugements de pertinence pour améliorer la réponse aux besoins de l'utilisateur ?

Les chapitres de la thèse présentent, tour à tour, des algorithmes qui opèrent sur la représentation des documents ou sur celle de la requête, utilisant les jugements de pertinence passés ou uniquement ceux de la requête courante. Dans les chapitres 5 et 6, nous avons présenté l'A.G. nous permettant d'effectuer différents types d'apprentissage.

Dans la catégorie des algorithmes récurrents, utilisant les jugements de pertinence des requêtes passées pour améliorer la réponse à des requêtes futures, nous avons

- évalué une série de méthodes existantes dans la littérature, pour lesquelles nous avons proposé :
  - une évaluation plus réaliste, en utilisant la méthode du *leaving one out*,
  - des collections de test de plus grande taille (81 et 425 documents pour les recherches antérieures vs. 1460 et 3204 documents dans cette thèse),
- développé une nouvelle stratégie d'apprentissage récurrent bien située du point de vue de la performance par rapport aux autres (meilleure performance pour la CACM et de performance équivalente à la majorité pour la CISI),
- une application de l'algorithme génétique
  - sur des collections de plus grande taille, contrairement aux collections expérimentales de petite taille pour les recherches antérieures,
  - avec l'indexation pondérée en plus de l'indexation binaire.

Dans la catégorie des algorithmes transiants, améliorant uniquement la réponse à la requête courante, nous avons proposé

- une application transiante de l'algorithme génétique, dans les conditions déjà décrites pour son application récurrente,
- une évaluation de la méthode du *relevance feedback* de manière cohérente avec les autres stratégies du point de vue des collections utilisées et de la mesure de performance, à des fins de comparaison.

L'évaluation de toutes ces stratégies a donné les classifications illustrées dans la figure 18 pour les évaluations rétrospectives et dans la figure 19 pour les autres évaluations (*leaving one out*, collection résiduelle, utilisateur).

Pour synthétiser ces résultats, on peut conclure que :

- les algorithmes transiants sont plus performants que les algorithmes récurrents, avec la réserve que cette forme d'apprentissage n'a pas d'impacte sur le long terme;
- l'algorithme génétique est moins performant que les autres algorithmes d'apprentissage, conclusion qui suggère qu'il n'est pas pertinent de l'utiliser pour modifier l'indexation des documents pour des collections de taille moyenne ou grande.

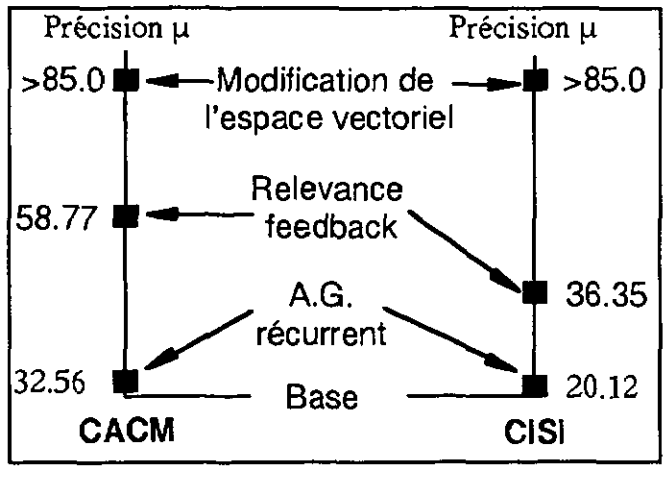

Fig. 18 Classification des algorithmes en évaluation rétrospective

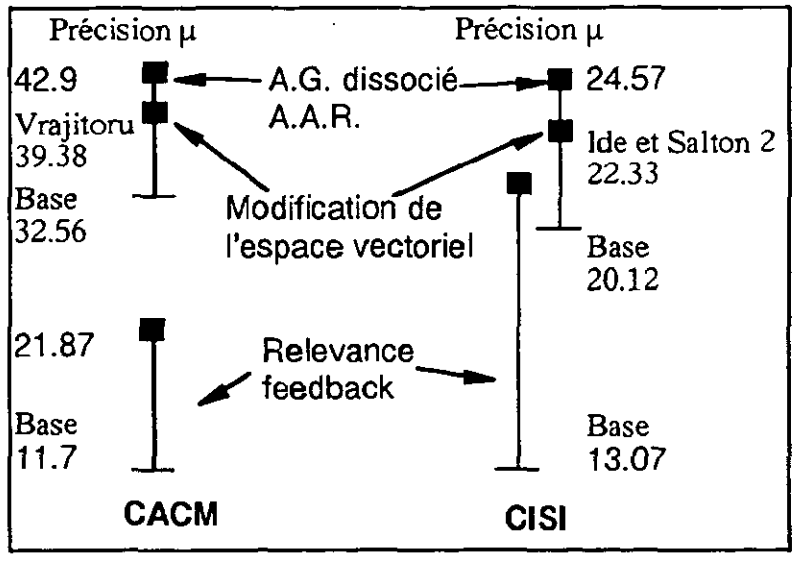

Fig. 19 Classification des algorithmes selon d'autres méthodes d'évaluation

En plus de ces apports en recherche d'informations, cette thèse propose une contribution à la théorie de l'algorithme génétique, à savoir :

- une analyse des mécanismes de fonctionnement de l'algorithme génétique, comme l'influence des paramètres taille de l'individu, nombre d'individus par génération et nombre de générations sur sa performance,

- une nouvelle opération de croisement, appelée *croisement dissocié*, qui s'est avérée plus performante que l'opération classique autant pour des raisons théoriques qu'expérimentales.

La conclusion la plus importante que l'on peut tirer de nos recherches est que l'apprentissage en recherche d'informations s'avère une approche possible et performante au point de vue précision/rappel.

Remerciements

Cette recherche a été soutenue en partie par le Fond National Suisse de la Recherche Scientifique avec la subvention 21-37'345.95.

Les algorithmes présentés dans cette thèse ont été implémentés en Smalltalk à partir des travaux de J. Savoy de l'Université de Neuchâtel, Suisse, ainsi que de D. Derbais, S. Simard, M. Choquette et F. Mercil de l'Université de Montréal, Quebec. L'auteur les remercie d'avoir mis à sa disposition leurs programmes.

La correction du texte français a été réalisée grâce aux contributions de O. Maumary et A. Le Calvé. L'auteur leur en est reconnaissant.

BIBLIOGRAPHIE

[Blair 90] D. C. Blair (1990) : *Language and Representation in Information Retrieval.* Elsevier, Amsterdam (NL).

[Brassard 94] G. Brassard, P. Bratley (1994) : *Fundamentals of Algorithmics.* Prentice-Hall.

[Brauen 71] T. L. Brauen (1971) : Document Vector Modification. Dans G. Salton (Editor) *The SMART Retrieval System - Experiments in Automatic Document Processing.* Prentice-Hall, Inc., Englewood Cliffs, (NJ), 456-484.

[Bush 45] V. Bush (1945) : As We May Think. *Atlantic Monthly.* 176, 101-108.

[Chen 95] H. Chen (1995) : Machine Learning for Information Retrieval : Neural Networks, Symbolic Learning, and Genetic Algorithms. *Journal of the American Society for Information Science.* 46(3), 194-216.

[Cleverdon 84] C. W. Cleverdon (1984) : Optimizing Convenient On-Line Access to Bibliographic Databases. *Information Service & Use.* 4, 37-47.

[Cooper 92] W. Cooper, F.C. Grey & D.P. Gabney (1992) : Probabilistic Retrieval Based on Staged Logistic Regression. *Proceedings of SIGIR'92.* Copenhague (Dk), 198-210.

[De Jong 75] K. A. De Jong (1975) : An Analysis of the Behavior of a Class of Genetic Adaptive Systems. (Doctoral disertation, University of Michigan). *Dissertation Abstracts International,* 36(10), 5140B.

[Dillon 80] M. Dillon & J. Desper (1980) : Automatic Relevance Feedback in Boolean Retrieval Systems. *Journal of Documentation.* 36, 197-208.

[Efron 86] B. Efron (1986) : How Biased Is the Apparent Error Rate of a Prediction Rule. *Journal of the American Statistical Association,* 81 (394), 461-470.

[Forrest 93] S. Forrest, M. Mitchell (1993) : What Makes a Problem Hard for a Genetic Algorithm ? Some Anomalous Results and Their Explanation. *Machine Learning.* 13, 285-319.

[Fox 83] E.A. Fox (1983) : *Caracterization of Two Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts.* Technical Report TR 83-561, Department of Computer Science, Cornell University.

[Fox 90] C. Fox (1990) : A Stop List for General Text. *ACM-SIGIR Forum.* 24(1-2), 19-35.

[Friedman 71] S. Friedman, J. A. Maceyak, S. F. Weiss (1971) : A Relevance Feedback System Based on Document Transformations. Dans G. Salton (Editor) *The SMART Retrieval System - Experiments in Automatic Document Processing.* Prentice-Hall, Inc., Englewood Cliffs, (NJ), 447-455.

[Fuhr 91] N. Fuhr & C. Buckley (1991) : A Probabilistic Learning Approach for Document Indexing. *ACM Transactions on Information Systems.* 9(3), 223-248.

[Fuhr 94] N. Fuhr & U. Pfeifer (1994) : Probabilistic Information Retrieval As a Combination of Abstraction, Inductive Learning and Probabilistic Assumptions. *ACM Transactions on Information Systems.* 12(1), 92-115.

[Furnas 87] G. Furnas, T.K. Landauer, L.M. Gomez & S.T. Dumais (1987) : The Vocabulary Problem in Human-System Communication. *Communications of the ACM.* 30(11), 964-971.

[Goldberg 89] D.E. Goldberg (1989) : *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading (MA).

[Gordon 88] M. Gordon (1988) : Probabilistic and Genetic Algorithms for Document Retrieval. *Communications of the ACM.* 31(10), 1208-1218.

[Gordon 89] M. Gordon, M. Kochen (1989) : Recall-Precision Trade-off : A Derivation. *Journal of the American Society for Information Science.* 40(3), 145-151.

[Gordon 91] M. Gordon (1991) : User-Based Document Clustering by Redescribing Subject Descriptions with a Genetic Algorithm. *Journal of the American Society For Information Science.* 42(5), 311-322.

[Grey 94] F.C. Grey (1994) : Inferring Probability of Relevance Using the Method of Logistic Regression. *Proceedings of the ACM-SIGIR'94.* Dublin (Ir), 222-231.

[Ide 71] E. Ide, G. Salton (1971) : Inteactive Search Strategies and Dynamic File Organization in Information Retrieval. Dans G. Salton (Editor) *The SMART Retrieval System*. Prentice-Hall, Inc., Englewood Cliffs, (NJ), 373-393.

[Koza 92] J. Koza (1992) : *Genetic Programming : on the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge (MA).

[Krovetz 92] R. Krovetz & W.B. Croft (1992) : Lexical Ambiguity and Information Retrieval. *ACM Transactions on Information Systems.* 10(2), 115-141.

[Kwok 88] K.L. Kwok (1988) : On the Use of Bibliographically Related Titles for the Enhancement of Document Representations. *Information Processing & Management.* 24(2), 123-131.

[Kwok 90] K.L. Kwok (1990) : Experiments with a Component Theory of Probabilistic Information Retrieval Based on Single Terms as Document Components. *ACM Transactions on Information Systems.* 8(4), 363-386.

[Lovins 68] J. B. Lovins (1968) : Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics.* 11(1), 22-31.

[Masand 92] B. Masand, L. G. Linoff & D. Waltz (1992) : Classifying News Stories Using Memory Based Reasoning. *Proceedings of SIGIR '92.* Copenhague (Dk), 6/92, 59-65.

[Mitchell 91] M. Mitchell, S. Forrest, J.H. Holland (1991) : The Royal Road for Genetic Algorithms : Fitness Landscapes and G.A. Performance. *Toward a Practice of Autonomous Systems : Proceeding of the First European Conference on Artificial Life*. The MIT Press, Cambridge (MA).

[Paice 90] C. D. Paice (1990) : Another Stemmer. *ACM-SIGIR Forum*. 24(3), 56-61.

[Porter 80] M. F. Porter (1980) : An Algorithm for Suffix Stripping. *Program*. 14(3), 130-137.

[Rijsbergen 79] C. J. van Rijsbergen (1979) : *Information Retrieval*. Butterworths, 2nd ed., London (UK).

[Salton 69] G. Salton (1969) : A Comparison Between Manual and Automatic Indexing Methods. *American Documentation*. 20(1), 61-71.

[Salton 71] G. Salton (Ed.) (1971) : *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice-Hall Inc., Englewood Cliffs (NJ).

[Salton 72] G. Salton (1972) : A New Comparison Between Conventional Indexing (MEDLARS) and Automatic Text Processing (SMART). *Journal of the American Society for Information Science*. 23(2), 75-84.

[Salton 83] G. Salton, E. Fox, U. Wu (1983) : Extended Boolean Information Retrieval. *Communications of the ACM*. 26(12), 1022-1036.

[Salton 88] G. Salton, C. Buckley (1988) : Term Weighting Approaches in Automatic Text Retrieval. Dans *Information Processing & Management*. 24(5), 513-523.

[Salton 90] G. Salton, C. Buckley (1990) : Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science*. 41(4), 288-297.

[Saracevic 75] T. Saracevic (1975) : Relevance : A Review of and a Framework for Thinking on the Notion in Information Science. *Journal of the American Society for Information Science*. 26, 321-343.

[Savoy 94a] J. Savoy, M. Ndarugendamwo, D. Vrajitoru (1994) : Report on the Trec-3 Experiment : A Learning Scheme in a Vector Space Model. *Proceedings of TREC-3*. NIST, Gaitherburgs (MD), 361-372.

[Savoy 94b] J. Savoy (1994) : A Learning Scheme for Information Retrieval in Hypertext. *Information Processing & Management*. 30(4) 515-533.

[Sparck Jones 77] K. Sparck Jones, R. G. Bates (1977) : *Research on Automatic Indexing 1974-1976*. Technical Report, Computer Laboratory, University of Cambridge.

[Stone 74] M. Stone (1974) : Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society, Series B*. 36(2), 111-147.

[Svenonius 86] E. Svenonius (1986) : Unanswered Questions in the Design of Controlled Vocabularies. *Journal of the American Society for Information Science*. 37(5), 331-340.

[Syswerda 89] G. Syswerda (1989) : Uniform Crossover in Genetic Algorithms. Dans J. D. Schaffer (Ed.), *Proceedings of the Third Internationall Conference on Genetic Algorithms*. Morgan Kaufmann Publishers, San Mateo (CA).

[Turtle 90] H. Turtle (1990) : *Inference Networks for Document Retrieval*. University of Massachusetts, Computer and Information Science Department, Doctoral Dissertation, Technical Report COINS Report 90-92, October 1990.

[Turtle 91] H. Turtle & W.B. Croft (1991) : Evaluation of an Inference Network-Based Retrieval Model. *ACM Transactions on Information Systems*. 9(3), 187-222.

[Vrajitoru 95] D. Vrajitoru (1995) : *Modification de l'indexation par apprentissage dans le modèle vectoriel.* Cahier de recherche en informatique CR-I-95-02, Université de Neuchâtel, Faculté de droit et des Sciences Économiques, Division économique et sociale.

[Vrajitoru 96] D. Vrajitoru (1996) : *Apprentissage par algorithme génétique dans la recherche d'informations.* Cahier de recherche en informatique CR-I-96-02, Université de Neuchâtel, Faculté de droit et des Sciences Économiques, Division économique et sociale.

[Weiss 91] A. C. Kulikowski, M. S. Weiss (1991) : *Computer Systems That Learn*. Morgan Kaufmann, San Mateo (CA).

[Yang 92] J.-J. Yang, R.R. Korfhage & E. Rasmussen (1992) : Query Improvement i n Information Retrieval Using Genetic Algorithms. A Report on the Experiments of the TREC Project. *Proceedings of TREC'1*. NIST, Gaitherburgs (MD), 31-58 .

Présentation de l'auteur

Dana VRAJITORU
Institut Interfacultaire d'Informatique
Pierre-à-Mazel 7
2000 NEUCHÂTEL
Tel. (+4132)718.13.73
e-mail: dana.vrajitoru@seco.unine.ch

Née le 1 février 1969
Nationalité: Roumaine
Célibataire

Diplômes :

Diplôme d'informaticienne de l'Université de Neuchâtel, mention très bien, 1994.
Diplôme en informatique de l'Université de Iasi, Roumanie, 1993.

Publications :

Savoy, J., & Vrajitoru, D. (1997) : Evaluation of Learning Schemes Used in Information Retrieval. *Journal of the American Society for Information Science*, janvier 1997 (accepté avec révisions).

Vrajitoru, D. (1997) : Genetic Algorithms in Information Retrieval. Présentation acceptée pour *AIDRI'97 (La VIIe Conférence de l'Association Internationale pour le Development de la Recherche Interdisciplinaire)*, Université de Genève, juin 1997.

Savoy J., Le Calvé A., Vrajitoru D. (1997) : Report on the TREC-5 Experiment: Data Fusion and Collection Fusion. Proceedings *TREC'5 (The Fifth Text REtrieval Conference)*, NIST, Gaithersburg (MD), (à paraître).

Publications internes :

Vrajitoru, D., Erard, P.-J. (1996) : *Facial Animation by Digitalized Image Deformation.* Cahier de recherche en informatique, CR-I-96-09.

Vrajitoru, D. (1996) : *Amélioration du croisement pour l'algorithme génétique dans la recherche d'informations.* Cahier de recherche en informatique, CR-I-96-06.

Savoy, J., Vrajitoru, D. (1996) : *Algorithme génétique et recherche d'informations.* Cahier de recherche en informatique, CR-I-96-05.

Vrajitoru, D. (1996) : *Apprentissage par algorithme génétique dans la recherche d'informations.* Cahier de recherche en informatique, CR-I-96-02.

Vrajitoru, D. (1995) : *Modification de l'indéxation par apprentissage dans le modèle vectoriel.* Cahier de recherche en informatique, CR-I-95-02.

Savoy, J., Vrajitoru, D. (1996) : Algorithme génétique et recherche d'informations. Dans *Université de Neuchâtel Informations*, 126, 19-28.