Algorithmes génétiques en recherche d'informations

Vrajitoru Dana[1]

Abstract : Information retrieval (IR) proposes solutions for searching, in a given set of objects, for those replying to a given description. Often, these objects are documents, however other forms like images, videos, sounds can be considered. Besides statistical approaches, artificial intelligence models present an attractive paradigm to improve performance in IR systems by learning, and the genetic algorithm (GA) represents one of them. How can the GA be used in IR and under which conditions? Does this learning method present an attractive performance ? What are the reasons of its failures? What can be done to improve its performance ? In this paper we try to answer these questions.

Keywords: *information retrieval, genetic algorithm, feedback.*

Introduction

Un des développements de l'informatique durant ces dernières années tend à rapprocher les informaticiens des chercheurs en neurosciences. De cette convergence, on peut citer les réseaux de neurones, le concept d'acteur, le concept de système autonome, et les algorithmes génétiques.

---

[1]Université de Neuchâtel, IIIUN, Pierre-à-Mazel 7, 2000 Neuchâtel, Suisse, e-mail: dana.vrajitoru@seco.unine.ch

L'algorithme génétique, comme l'indique son nom, simule les mécanismes de reproduction des organismes vivants en vue de trouver une solution à un problème donné. La recherche de solutions s'effectue par des opérations comme la reproduction, le croisement et la mutation. Parmi ses avantages, la robustesse et la possibilité d'application sans connaître à priori les propriétés du domaine sont, à nos yeux, les plus importantes.

Comme pour beaucoup d'algorithmes, les paramètres de l'algorithme génétique doivent être soigneusement choisis pour obtenir de bons résultats. Souvent, ces paramètres dépendent du système et une conclusion générale quand à leur valeur est difficile à tirer. Ce type de question constitue le but principal de cet article qui apporte une méthodologie justifiée pour résoudre ce problème.

Cet article se divise en trois parties. La première présente les grandes lignes de l'algorithme génétique. La deuxième explique les problèmes de la recherche d'informations et évalue l'algorithme génétique en recherche d'informations. La troisième décrit et évalue notre nouvel opérateur de croisement pour l'algorithme génétique.

1. L'algorithme génétique

Mis au point par Holland (Holland 1975, De Jong 1975) , l'algorithme génétique (AG) est généralement utilisé pour résoudre des problèmes d'optimisation. Soit P un ensemble dont les éléments sont appelés individus. A chaque individu i on associe une performance, notée $f(i)$, dont l'AG devra chercher la valeur optimale, ainsi que l'un des points de l'espace P où cette valeur est atteinte.

Pour appliquer l'AG, les individus sont codés sous forme de vecteur de longueur L (le chromosome) dans lequel chaque élément i_k représente un gène : $i = (i_1, i_2, ..., i_L)$. Le plus fréquemment ces vecteurs sont binaires (Goldberg 1989).

Nous avons utilisé un AG simple, consistant à itérer les trois opérations suivantes: reproduction, croisement et mutation. La population ainsi créée lors de chaque itération s'appelle une génération et se note P_t.

Étant donné P_O, une population initiale, la **reproduction** que nous avons choisie pour nos recherches tire au sort avec répétition un nombre d'individus de P_O égal à son cardinal. Les individus présentant une haute performance peuvent apparaître plusieurs fois, tandis que ceux de faible performance ont tendance à disparaître.

Nous avons employé le **croisement à 1 point** avec une probabilité de 0.8 et la mutation simple de très faible probabilité (Goldberg 1989).

La construction des générations suit le schéma classique selon lequel les enfants remplacent les parents. Cependant, nous y avons apporté une modification pour empêcher la performance maximale d'une génération de décroître. Ainsi, si le cas se présente, le meilleur parent peut prendre la place de l'enfant de plus faible performance. Cette méthode se rapproche de la sélection élitiste.

2. L'algorithme génétique en recherche d'informations

La section suivante introduit les notions de base concernant la recherche d'informations et présente l'application de l'algorithme génétique à ce domaine.

2.1 La recherche d'informations

La recherche d'informations tente de résoudre le problème suivant: "Étant donné une très grande collection d'objets (le plus souvent des documents), trouver ceux qui répondent à un besoin d'information exprimé par un usager (requête)".

Pour évaluer nos travaux, nous avons recourt à la collection CACM (3204 documents et 50 requêtes avec leurs jugements de pertinence) et la collection CISI (1460 documents, 35 requêtes).

Généralement, une première étape consiste indexer les documents (Salton 1989). Durant cette opération, on élimine les termes très courants qui sont peu ou pas porteurs d'information, ainsi que les suffixes qui ne changent souvent pas le sens du mot. Pour finir, on établit une représentation de chaque document sous forme d'une liste de termes, éventuellement pondérés.

Dans notre recherche, le poids (t_{hj}) d'un terme (j) dans un document (h) est calculé avec la formule de l'équation (1), formule introduite par Salton et al. (1983). Intuitivement, un terme s'avère plus important s'il apparaît souvent dans le document (composante *ntf*). Ce poids tend à diminuer si le terme en question apparaît dans presque tous les documents de la collection.

$$t_{hj} = ntf_{hj} \cdot nidf_j \quad \text{avec}$$
$$ntf_{hj} = \frac{tf_{hj}}{\max_g tf_{hg}} \quad \text{et} \quad nidf_j = \frac{\log(n) - \log(df_j)}{\log(n)} \tag{1}$$

Dans cette équation, *tf* représente la fréquence du terme dans le document, *df* la fréquence du terme dans la collection (indicé selon le terme *j*) et *n* le nombre de documents dans la collection.

Finalement, pour retrouver et classer les documents correspondant à la requête, on calcule un indice de similarité entre la requête et les documents. Cet indice est calculé selon l'équation (2), signifiant le cosinus de l'angle entre les vecteurs imaginaires représentant la requête (q) et le document (d_h) (Salton 1989). Dans cette équation, w_{qj} représentent les poids des termes dans la requête et sont calculé selon une formule similaire à l'équation (1).

$$sim(d_h, q) = \frac{\sum w_{qj} t_{hj}}{\sqrt{\sum w_{qj}^2 \sum t_{hj}^2}} \tag{2}$$

Par exemple, prenons le document #1613 appartenant à la collection CACM, intitulé "One-Pass Compilation of Arithmetic Expressions for a Parallel Processor" ainsi que la requête #17 exprimée par le texte "Optimization of intermediate and machine code".

Suite à l'indexation, la requête sera composé de quatre termes auxquels on a associé les poids w_{qj} suivants :

{('machin' 0.362), ('intermedi' 0.656), ('optim' 0.393), ('code' 0.365)}

Pour le terme 'code', la partie ntf_{qj} de son poids est égale à 1 car il apparaît une seule fois dans la requête, et la fréquence maximale dans des termes dans la requête est aussi de 1 (chaque terme y apparaît une seule fois) : $ntf_{qj} = 1/1 = 1$.

Pour la composante $nidf_j$, on sait que le nombre de documents dans la collection est n = 3204, et le terme 'code' apparaît dans 168 documents. Ainsi on obtient
$nidf_j = (\log(3204) - \log(168)) / \log(3204) = 0.365$.

Ainsi, le terme 'code' étant beaucoup plus fréquent dans la collection que le terme 'intermediate' (16 documents), son importance dans la requête est plus petite.

Parmi les termes d'indexation de la requête, le document en question contient seulement le terme 'code'. Sa composante $nidf_j$ reste la même (0.365). Pour ce document, la valeur ntf_{ij} se calcule comme $ntf_{ij} = 1/5 = 0.2$. Cela car le document #1613 contient une occurrence du terme 'code', et son terme de fréquence maximale est 'compil' avec 5 occurrences. Ainsi, le poids du terme 'code' dans le document #1613 est :
$t_{ij} = 0.365*0.2 = 0.073$, avec i = 1613 et j = 'code'.

La similarité entre ce document et la requête est de

$$sim(1613,17) = \frac{0.365 * 0.073}{\sqrt{(0.362^2 + 0.656^2 + 0.393^2 + 0.365^2) * 1.17}} = 0.027$$

Dans cette dernière formule, la somme $\sum t_{hj}^2$ est égale à 1.17, mais nous ne l'avons pas détaillé car le document a été indexé avec 29 termes.

Les documents trouvés seront classés et présentés à l'usager en débutant par le document possédant la plus forte similarité avec la requête. L'usager peut évaluer chaque document retourné et spécifier s'il répond ou non à ses souhaits.

Deux mesures principales servent à mesurer la qualité de la réponse: la **précision**, calculé comme le pourcentage de documents pertinents retrouvés, et le **rappel**, mesurant le pourcentage de documents pertinents retournés parmi tous les documents pertinents (Salton 1989).

Par exemple, si la recherche nous retourne une liste de 10 documents dont 4 sont pertinents pour la requête, la valeur de la précision sera de 4/10 = 0.4. Si l'on sait que dans la collection il y a 12 documents pertinents pour cette requête, on peut calculer la valeur du rappel comme 4/12 = 0.33.

En pratique, et dans cet article, on utilise une mesure plus complexe, appelée la "**précision à 11 points de rappel**", qui permet de comparer deux systèmes de recherche avec plus d'objectivité (Salton 1989, Vrajitoru 1997).

2.2. Codage du problème en recherche d'informations

Parmi les applications de l'algorithme génétique en recherche d'informations (Chen 1995, Yang 1992, Gordon 1988, 1991), notre recherche se rapproche du modèle développé par Gordon et en constitue une extension.

Toute la difficulté de la recherche d'informations réside dans la recherche de la sémantique associée aux documents et à la requête. En effet, le même terme peut posséder des sens très différents et le même concept peut s'exprimer par divers mots. Une indexation n'est donc jamais parfaite et, idéalement, il faudrait permettre à l'ordinateur d'apprendre à mieux représenter les documents de base à partir de l'information fournie par les jugements de pertinence. La solution que l'algorithme génétique recherche dans nos expériences est, donc, une indexation des documents présentant une meilleure performance que l'approche sans apprentissage.

Dans le contexte présenté, l'espace de recherche de l'algorithme génétique appliqué en recherche d'information est constitué par l'ensemble de tous les descripteurs possibles des documents. Pour un document donné d_h, avec $h = 1..n$ et un ensemble de termes T_j avec $j = 1..m$, le descripteur associé à d_h possède la forme suivante:

$$d_h = \langle t_{1h}, t_{2h}, \ldots, t_{mh} \rangle$$

Le gène t_{hj} a la valeur calculée selon l'équation (1).

Un individu s'obtient par concaténation des descripteurs de tous les documents d_h, pour $h = 1, 2, \ldots, n$:

$$i = \langle d_1, d_2, \ldots, d_n \rangle = \langle t_{11}, \ldots t_{m1}, t_{12}, \ldots t_{m2}, \ldots, t_{1n}, \ldots t_{mn} \rangle \qquad (3)$$

Pour arriver à une représentation binaire de ces vecteurs, les valeurs t_{hj} sont discrétisées et remplacées par des entiers entre 0 et 10 qui seront représentés de manière sur 4 bits. La fonction de performance est calculée comme la précision moyenne à 11 points de rappel par rapport aux requêtes disponibles.

2.3. Population initiale et résultats

Un des buts de notre recherche est d'utiliser les jugements de pertinence des requêtes passées afin d'améliorer la performance du système pour des requêtes futures. Nous allons en tenir compte dans la construction de la population initiale.

Le système de dépistage de l'information décrit dans le paragraphe 2.1 sera la base de comparaison lors de l'évaluation de l'algorithme génétique.

La population de base contient un individu obtenu par le même processus d'indexation et 7 autres construits à partir des jugements de pertinence repartis équitablement (Vrajitoru 1997)

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 (méthode d'évaluation "leave-one-out", Efron 1986, Vrajitoru 1997).

Le nombre d'individus qui forment la population initiale s'avère être un premier paramètre important pour l'algorithme génétique. Le second est constitué par le nombre de générations. Pour en déterminer les bonnes valeurs, nous avons effectué des expériences où le nombre d'individus multiplié par le nombre de génération reste constant.

Les résultats essentiels de ces évaluations sont repris dans le tableau 1 et, pour une meilleure illustration, la figure 5 les représente graphiquement. Les deux meilleurs scores obtenus sont écrits en gras

Selon le critère statistique reconnu dans le domaine, une différence supérieure à 5% sera considérée comme significative (Spark Jones 1977). Ainsi, on notera une amélioration significative de la performance qui confirme l'importance de l'apprentissage. Les résultats suggèrent aussi qu'il vaut mieux avoir une population de départ plus importante, quitte à réduire le nombre de générations.

Tableau 1 Résultats par variation de la taille de la population initiale

Collection	**CACM**	**CISI**
Individus/générations	**Précision**	**(%changement)**
approche classique	32.70	19.83
4 / 20	37.37 (+14.28)	21.71 (+9.48%)
8 / 10	38.16 (+16.7)	**24.90 (+25.57%)**
10 / 8	39.43 (+20.58)	24.14 (+21.73%)
14 / 6	**40.62 (+24.22)**	24.77 (+24.91%)
20 / 4	**41.61 (+27.25)**	**24.96 (+25.87%)**

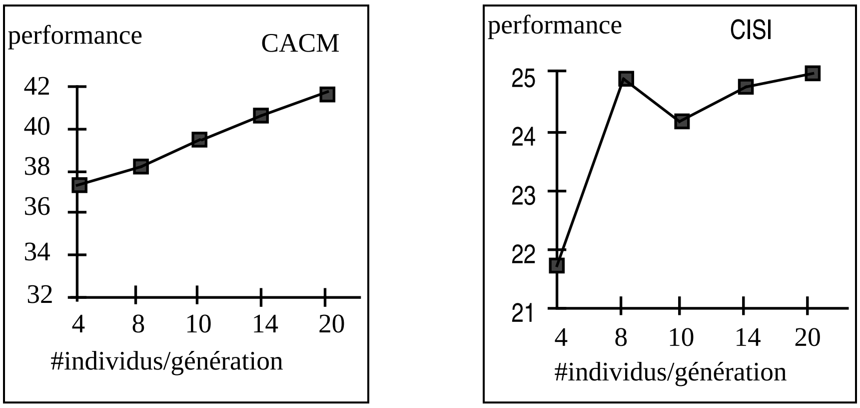

Figure 1 Illustration graphique du tableau 1

3. Amélioration de l'algorithme génétique pour la recherche d'informations

Lors de l'opération de croisement, le but recherché par l'algorithme génétique est de créer au moins un nouvel individu dont la performance sera plus grande que celle de ses parents. En analysant l'évolution de la performance au fil des générations, nous avons souvent constaté le contraire.

Une analyse mathématique nous a montré que l'opération de croisement classique présente un phénomène bien connu par les mathématiciens concernant les combinaisons affines (Vrajitoru 1997). Ainsi, la performance des fils peut s'exprimer souvent comme

$$f(fils) = \alpha \cdot f(parent_1) + (1-\alpha) \cdot f(parent_2)$$

ce qui implique

$$f(parent_1) \leq f(fils) \leq f(parent_2)$$

Pour éviter ce phénomène, nous proposons une nouvelle opération de croisement, nommée **croisement dissocié**, qui consiste à faire une distinction entre les sites de croisement des deux parents. Ainsi la performance des fils devient

$$f(fils) = \alpha \cdot f(parent_1) + \beta \cdot f(parent_2).$$

Puisque $\beta \neq (1 - \alpha)$, la performance du fils peut dépasser la performance des parents. La réalisation concrète de cette idée consiste à appliquer un croisement distinct à chaque parent, comme l'illustre la figure 6.

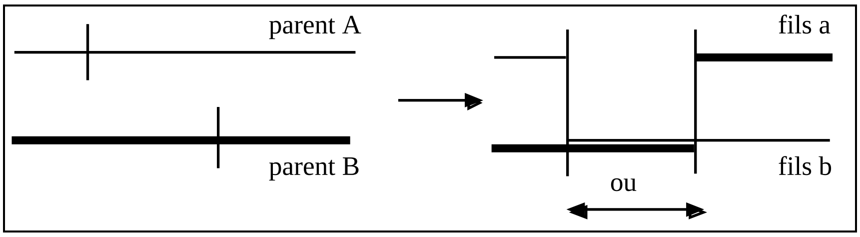

Figure 2 Croisement dissocié

L'évaluation de l'algorithme génétique utilisant le croisement dissocié comparé à l'algorithme classique est présentée dans le tableau 2 (paramètres: 8 individus par génération et 10 générations).

Tableau 2 Résultats du nouvel opérateur

Collection	**Départ**	**Classique**	**Dissocié**
CACM	32.70	38.16	43.95 (+15.17%)
CISI	19.83	24.90	25.74 (+3.37%)

4. Conclusion

Les conclusions que l’on peut tirés de nos expériences sont de trois ordres. D'abord, l'algorithme génétique présente un paradigme intéressant pour l'apprentissage automatique. Ensuite, le choix des valeurs des différents paramètres s'avère très important pour la performance. Enfin, notre nouvel opérateur de croisement apporte une amélioration significative comparée à d'autres algorithmes génétiques.

Remerciements: Cette recherche a été soutenue en partie par le Fonds National Suisse pour la Recherche Scientifique (subvention 21-37'345.95).

Références

Chen, H. (1995). Machine Learning for Information Retrieval : Neural Networks, Symbolic Learning, and Genetic Algorithms. *JASIS.* 46(3), 194-216.

De Jong, K.A. (1975). An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Doctoral disertation, University of Michigan.

Efron, B. (1986). How Biased Is the Apparent Error Rate of a Prediction Rule. *JASA,* 81 (394), 461-470.

Gordon, M. (1988). Probabilistic and Genetic Algorithms for Document Retrieval. *CACM.* 31(10), 1208-1218.

Gordon, M. (1991). User-Based Document Clustering by Redescribing Subject Descriptions with a Genetic Algorithm. *JASIS*. 42(5), 311-322.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press, Ann Arbor, 1975.

Salton, G., Fox, E., Wu, U. (1983). Extended Boolean Information Retrieval. *CACM,* 26(12), 1022-1036.

Salton G. (1989). *Automatic Text Processing, The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading (MA).

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

Vrajitoru, D. (1997). *Apprentissage en recherche d'informations.* Thèse de doctorat, Université de Neuchâtel, Faculté des Sciences.

Yang, J.-J., Korfhage, R.R., Rasmussen, E. (1992). Query Improvement in Information Retrieval Using Genetic Algorithms. *TREC'1.* NIST, Gaitherburgs (MD), 31-58 .