Info407 – Langage Fonctionnel – TP n°2

Classification automatique de documents par apprentissage supervisé

(séances 2, 3 et 4 - du 3 au 18 avril 2025)

Consignes :

- Mettez des commentaires pour faciliter la lecture de votre code. Vous délimiterez notamment les différentes « zones » d'abstraction (zone de la définition des types ; zone où sont définis les constructeurs et sélecteurs ; zone où sont définies les fonctions de manipulation de base ; etc.)
- Indiquez systématiquement le profil des fonctions que vous écrivez et décrivez par un commentaire – précis et concis - ce qu'elles font.

Le but des 3 séances de TPs restant est d'écrire un programme en CAML capable de classer automatiquement des documents.Vous procéderez conformément à ce qui est indiqué dans la partie « travail à réaliser » ci-dessous. Les parties suivantes expliquent ce que sont la classification automatique et l'apprentissage par l'exemple, méthode de classement retenue pour ce TP, que vous aurez à programmer donc.

Plan du document

1. Travail à réaliser

Analysez le problème dans le cadre d'une démarche par décomposition fonctionnelle. Vous écrirez les fonctions correspondant à votre analyse. Il vous faudra définir les types nécessaires et les utiliser dans le cadre d'une approche « types abstraits ».

Vous procéderez par étapes :

1. Définition des types et fonctions de base pour la manipulation des documents et ensemble de documents
2. Construction d'un arbre de décision non optimisé ; détermination de sa capacité à prédire correctement le classement sur les exemples fournis.
3. Même chose avec un arbre de décision optimisé

Une fois ce travail réalisé, vous mènerez une réflexion sur les points suivants :

- Que pensez-vous du principe de cette méthode : quels en sont les limites, les qualités, les défauts ? Quelles améliorations peut-on y apporter ?
- Quels sont les problèmes posés par la construction des vecteurs de mots qui représentent les documents ? Comment les traiter ?

2. La classification automatique de documents

2.1 Définition

La classification automatique est un processus qui permet par exemple à une application de messagerie électronique de ranger automatiquement un message reçu dans tel ou tel dossier, selon qu'il appartient - ou pas - à une catégorie donnée, de faire le tri entre les messages qui sont du spam et ceux qui n'en sont pas, etc. C'est également une des techniques à laquelle un navigateur peut avoir recours pour trouver les pages web cherchées par un utilisateur, celles qu'il jugera « intéressantes » versus les « non intéressantes ».

Toutes sortes de filtres peuvent être appliqués aux documents à classer : spam / pas spam, intéressant / pas intéressant, contenu haineux / contenu ok, etc. A chaque fois cependant, on distingue entre 2 catégories (qui peuvent s'apparenter à du « oui/non »).

2.2 L'apprentissage par l'exemple

L'apprentissage par l'exemple est une des méthodes possibles pour faire de la classification automatique, considérée comme faisant partie des méthodes d'intelligence artificielle. Elle enchaîne trois étapes :

1. la première consiste à fournir à l'ordinateur des exemples de documents avec leur catégorie, « intéressant » ou « pas intéressant ». En analysant le contenu de chaque document, l'ordinateur construit une ***fonction de décision***. C'est la ***phase d'apprentissage*** ; l'ensemble des documents utilisé pour cette phase est appelé ***Doc_appr*** (documents d'apprentissage);
2. la deuxième a pour objectif de valider la fonction de décision obtenue ; Il s'agit ici de réaliser des tests avec des documents différents de ceux qui ont servi pour l'apprentissage, dont on connaît la catégorie également. C'est la ***phase de test.*** La fonction de décision est considérée comme « bonne » si elle trouve correctement les catégories des documents de validation. L'ensemble de ces documents est appelé ***Doc_test ;***
3. si le test est concluant, il ne reste plus alors qu'à utiliser la fonction de décision pour classer tout nouveau document. C'est la ***phase d'utilisation***.

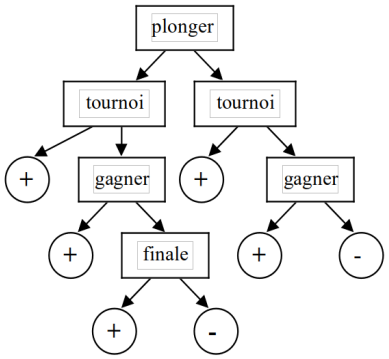
Plusieurs techniques existent pour construire la fonction de décision : réseaux de neurones, K plus proches voisins, réseaux Bayésiens, algorithmes génétiques, système à base de règles, etc. Nous vous

proposons d'utiliser ici une méthode fondée sur la construction d'un arbre de décision. La fonction de décision aura donc la forme d'un tel arbre.

3. Les données à manipuler

3.1 Forme d'un arbre de décision

L'exemple ci-dessous montre un arbre de décision qui pourrait être utilisé pour repérer des documents traitant de sport (sur le web par exemple).



Un arbre de décision est tel que :

- chaque nœud non terminal de l'arbre contient un mot,
- les feuilles de l'arbre contiennent une décision, représentée par les symboles + (intéressant) ou – (pas intéressant).

3.2 Forme des documents et ensembles de documents.

Un document est un assemblage complexe de textes, de symboles et d'images. Dans les problèmes de classification automatique avec arbre de décision, ils sont pré-traités pour en extraire une liste de mots signifiants qui les représentera et en facilitera la manipulation. Ce pré-traitement étant complexe, vous supposerez ici qu'il a d'ores et déjà été réalisé et qu'un document est un vecteur de mots.

> Ex : un article intitulé « Cocorico ! La France remporte haut la main le tournoi des 6 nations » pourra être représenté par ("Cocorico", "France", "remporter", "tournoi", "6 nations")

De fait, les ensembles de documents utilisés pour les phases d'apprentissage et de test seront de la forme {doc1, decision1, doc2, decision2, …}, les doc-i étant des documents (des vecteurs de mots), les décisions-i étant des décisions (intéressant / pas intéressant ; + / - ; D_pos/D_neg ;…). L'exemple qui suit montre un ensemble qui pourrait être un *Doc_test* ou un *Doc_appr*.

```
Doc_sports  =
        {
                ( ("tournoi", "Irlande","retrouver","goût", "victoire"),  +),
                ( ("ligue", "champion","Paris","demi-finale"),  +),
                ( ("AMD", "gagner","course","processeur"),  -),
                ( ("JO", "Paris","Romain Ntamak","porter", "flamme", "olympique", "Occitanie"),  +),
                ( ("JO", "Paris","menace","grève", "SNCF"),  -)
        }
```

4. Utilisation d'un arbre de décision pour classer un document

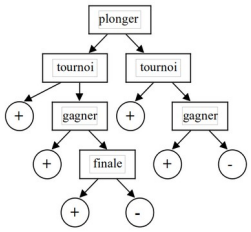
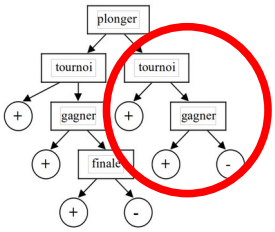
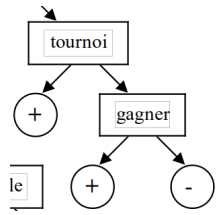
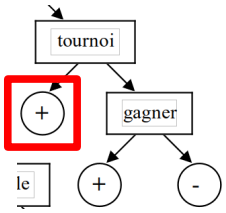
4.1 Description du processus

On suppose que l'on dispose de l'arbre de décision *ArbreD* construit à partir d'un ensemble d'apprentissage donné. Soit un document *doc* à classer en utilisant cet arbre. La décision se construit en parcourant l'arbre comme suit :

- on considère le mot à la racine de *ArbreD*. Si ce mot est dans *doc* alors on descend dans l'arbre à gauche (exploration de la branche gauche), sinon on descend à droite (exploration de la branche droite) ;
- on recommence avec la branche retenue (gauche ou droite) et, ce, récursivement, jusqu'à arriver sur une feuille qui donnera la décision.

Exemple :

Soit *ArbreD* l'arbre de décision donné ci-dessus (arbre « sports »).
Soit à classer le document ("tournoi", "Irlande", "retrouver", "goût", "victoire").

Arbre considéré	racine	test	décision
ArbreD 	"plonger"	"plonger" n'est pas dans *doc*	On part à droite 
	"tournoi"	"tournoi" est dans *doc*	On part à gauche 
		On est sur une feuille qui donne la décision	+

Cette méthode est utilisée aussi bien en phase de test qu'en phase d'utilisation. Sur cet exemple, le document est bien classé. Ce n'est pas toujours le cas.

4.2 Qualité de la prédiction

Un arbre de décision est construit à partir d'un ensemble de documents d'apprentissage *Doc_appr*. Une fois l'arbre construit, il faut vérifier qu'il est capable de classer correctement des documents. On considère pour cela un ensemble de documents *Doc_test* disjoint de *Doc_appr* (l'intersection entre les deux est vide) qui va servir à la validation :

- si l'arbre construit permet de classer correctement une majorité des documents de doc_test, il est considéré comme une « bonne » fonction de classification et peut être mis en exploitation, à la disposition des usagers ;
- si ce n'est pas le cas, c'est que l'ensemble d'apprentissage n'est pas suffisant : les documents qu'il contient ne suffisent pas à distinguer la catégorie de documents qu'ils sont sensés représenter, ne couvrent pas suffisamment l'espace sémantique de cette catégorie. Il faut alors enrichir doc_test avec d'autres documents et recommencer le processus.

La fonction de décision correspondant à l'arbre sera considérée comme « bonne» si le taux de prédictions correctes atteint ou dépasse les 70%.

Exemple :

Soit *ArbreD* l'arbre de décision donné ci-dessus (arbre « sports ») construit à partir d'un ensemble de documents *doc_appr*. Nous allons évaluer la qualité de la prédiction à l'aide de *Doc_sports* (cf. la page 3, partie « forme des documents ») qui fait office ici d'ensemble de documents de test.

```
Doc_sports =
{       ( ("tournoi", "Irlande","retrouver","goût", "victoire"),  +),
        ( ("ligue", "champion","Paris","demi-finale"),  +),
        ( ("AMD", "gagner","course","processeur"),  -),
        ( ("JO", "Paris","Romain Ntamak","porter", "flamme", "olympique", "Occitanie"),  +),
        ( ("JO", "Paris","menace","grève", "SNCF"),  -)     }
```

ArbreD pronostique les résultats suivants pour *Doc_test* :

(+ , - , + , - , -)

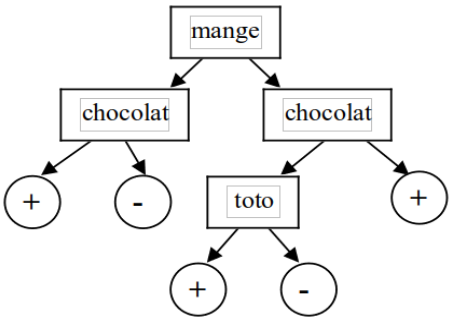
soit 2 prédictions correctes sur 5. On est loin des 70 % de prédictions correctes. L'arbre de décision ne convient pas ; l'ensemble de documents d'apprentissage est à revoir.

5. Construction d'un arbre de décision

5.1 Arbre de décision non optimisé

Soit *Doc_appr* l'ensemble d'apprentissage ci-dessous. L'arbre de décision correspondant est présenté à côté.

```
{ (("mange", "chocolat", "toto", "bras"),  + ),
  (("école", "maîtresse", "toto"), + ),
  (("mange", "chocolat", "toto", "maîtresse"), + ),
  (("maîtresse", "mange"), - ),
  (("toto", "chocolat", "maîtresse"), + ),
  (("chocolat", "maîtresse", "bras"), - )
}
```



L'arbre de décision est construit à partir de *Doc_appr* et de l'ensemble de tous les mots présents dans les documents d'apprentissage. Nous appellerons *listeMots* cette liste de mots. Sur l'exemple ci-dessus, il s'agit de :

("mange", "chocolat", "toto", "bras", "école", "maîtresse").

Le principe de construction repose sur l'appartenance ou la non appartenance des mots de *listeMots* aux différents documents de *Doc_appr :*

1. On considère le premier mot de *listeMots* : c'est "mange" sur l'exemple. Ce mot devient la racine de l'arbre de décision à construire.

2. Les branches sont construites comme suit :

   a) On découpe *Doc_appr* en 2 ensembles :
      - l'ensemble *Doui* des documents de *Doc_appr* qui contiennent le premier mot ("mange") :

```
Doui = { (("mange", "chocolat", "toto", "bras"),  + ),
         (("mange", "chocolat", "toto", "maîtresse"), + ),
         (("maîtresse", "mange"), - ) }
```

- l'ensemble *Dnon* des documents de *Doc_appr* qui ne contiennent pas ce premier mot :
  Dnon = { (("école", "maîtresse", "toto"), +),
  (("toto", "chocolat", "maîtresse"), +),
  (("chocolat", "maîtresse", "bras"), -) }

Doui permet de construire la branche gauche de l'arbre, *Dnon* la branche droite.

b) Si les documents appartenant à *Doui* sont tous associés à la même décision, qu'elle soit positive (+) ou négative (-), cela signifie qu'il suffit que le premier mot ("mange") soit dans un document pour avoir une décision. La branche gauche de l'arbre est alors une feuille correspondant à cette décision.

Si ce n'est pas le cas, la branche gauche est un arbre de décision construit à partir de *Doui* comme ensemble d'apprentissage et *listeMots* privée de son premier mot ("mange" - il a déjà été traité) comme liste de mots.

c) On fait de même avec *Dnon* pour construire la branche droite de l'arbre.

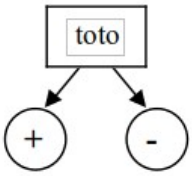
L'arbre est de fait construit récursivement en divisant l'ensemble d'apprentissage en 2 et en réduisant la liste des mots , jusqu'à tomber sur des ensembles d'apprentissage homogènes (i.e . avec une même décision : + ou - pour tous les documents qu'ils contiennent).

5.2 Arbre de décision optimisé

5.2.1 Le principe de l'optimisation

L'arbre de l'exemple ci-dessus a été construit en prenant les mots au hasard (ordre dans lequel ils apparaissent dans la liste des mots : cela dépend de la façon dont la liste est construite). Or les mots n'ont pas tous le même impact dans le partitionnement des documents d'apprentissage en 2.

Ainsi, si l'on considère le mot « toto » plutôt que « mange » dans l'exemple ci-dessus, on obtient un arbre moins profond permettant de prendre les mêmes décisions. Il s'agit de :



L'arbre étant moins profond, sa construction mais aussi son utilisation seront plus rapides. Nous allons donc améliorer le processus de construction en sélectionnant, à chaque étape de ce processus, non pas un mot au hasard mais le mot qui sépare « au mieux » les documents qui restent à utiliser pour construire l'arbre.

L'identification du « meilleur mot » suppose de calculer, pour chaque mot candidat, ce qui est appelé un « gain de séparation ». Le mot ayant le plus fort gain sera retenu.

5.2.2 Calcul du gain

Le gain peut être calculé en ayant recours à une des fonctions mathématiques définies en théorie de l'information : l'entropie de Shannon. Intuitivement, cette fonction mesure la quantité d'information contenue ou délivrée par une source d'information. Le calcul du gain pour un mot donné se fait alors comme décrit dans ce qui suit.

1) On commence par calculer l'entropie de l'ensemble de documents d'apprentissage doc_appr.

- soit **N** le nombre de documents présents dans *doc_appr*

- dans *doc_appr*, il y a par ailleurs :
  - **Dpos** documents étiquetés « + » (intéressants),
  - **Dneg** documents étiquetés « - » (pas intéressants)

L'entropie de *doc_appr* est alors donnée par la formule :

$$Entropie = \left[\frac{-D_{pos}}{N} * \log 2\left(\frac{D_{pos}}{N}\right)\right] + \left[\frac{-D_{neg}}{N} * \log 2\left(\frac{D_{neg}}{N}\right)\right]$$

A noter : log2 est le logarithme base 2 (et pas népérien). Par convention, log2(0) = 0

2) Soit *mot* le mot considéré. Il permet de découper *doc_appr* en 2 ensembles :
- ***doc_P***, rassemblant les documents de *doc_appr* où *mot* est présent,
- et ***doc_A***, rassemblant les documents de *doc_appr* où *mot* est absent.

Nous allons ici calculer l'entropie de ces 2 ensembles :
- *doc_P* contient **NbP** documents dont **p_pos** sont intéressants (étiquetés « + »), **p_neg** ne le sont pas ;
- *doc_A* contient **NbA** documents dont **a_pos** sont intéressants, **a_neg** ne le sont pas.

Les entropies correspondant à *doc_P* et *doc_A* sont alors données par la même formule que ci-dessus :

$$Entropie_{present} = \left[\frac{-p_{pos}}{NbP} * \log 2\left(\frac{p_{pos}}{NbP}\right)\right] + \left[\frac{-p_{neg}}{NbP} * \log 2\left(\frac{p_{neg}}{NbP}\right)\right]$$

$$Entropie_{absent} = \left[\frac{-a_{pos}}{NbA} * \log 2\left(\frac{a_{pos}}{NbA}\right)\right] + \left[\frac{-a_{neg}}{NbA} * \log 2\left(\frac{a_{neg}}{NbA}\right)\right]$$

Au final, la mesure du gain pour *mot* est obtenue en appliquant le calcul suivant :

$$Gain = Entropie - Entropie_{present} - Entropie_{absent}$$

5.2.3 Exemple

Application à la construction de l'arbre de décision associé à l'ensemble de documents *doc_appr* de la page 5.

	total	p_pos	p_neg	entropie
doc_appr	**6**	**4**	**2**	0,918295834

mot	doc_P				doc_A				
	P	p_pos	p_neg	entropie	A	a_pos	a_neg	entropie	Gain
toto	4	4	0	**0,00**	2	0	2	**0,00**	**0,92**
mange	3	2	1	**0,92**	3	2	1	**0,92**	-0,92
chocolat	4	3	1	**0,81**	2	1	1	**1,00**	-0,89
bras	2	1	1	**1,00**	4	3	1	**0,81**	-0,89
école	1	1	0	**0,00**	5	3	2	**0,97**	-0,05

Le mot « toto » est donc sélectionné. Il devient ainsi la racine de l'arbre de décision.