TRANSFORMER DU TEXTE ET DES IMAGES

EXEMPLE 1 : ANTI-SPAM TEXTOS

Données: messages SMS (en anglais)

- 747 spams. Exemple: "WINNER!! As a valued network customer you have been selected to receive a prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only."
- 4827 non-spams. Exemple: "Nah I don’t think he goes to usf, he lives around here though"
- 9663 mots distincts.

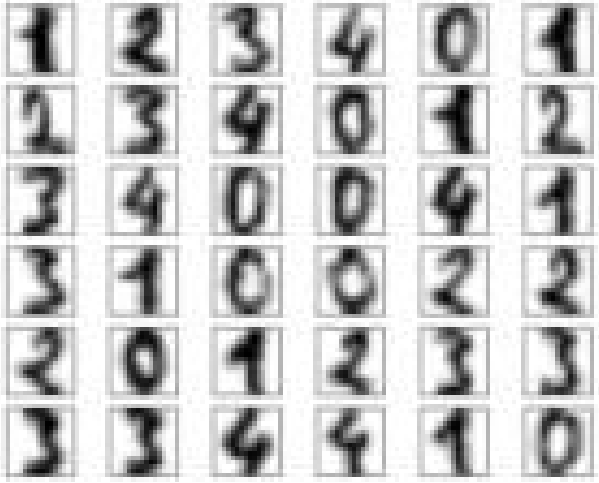
Objectifs :

1. Comprendre ce qui différencie les deux types de messages;
2. Etablir une règle de séparation;
3. Appliquer la règle à de nouveaux messages.

Exemple 2 : reconnaissance de chiffres manuscrits

Données:

- chiffres manuscrits de 0 à 4
- 901 exemples
- Environ 180 par classe.
- Chaque image : 8 pixels x 8 pixels



Objectif:

- être capable de classifier automatiquement ces images.
- établir une règle de séparation;
- appliquer la règle à un nouveau chiffre entrant. (application code postaux)

Pourquoi transformer les données ?

- Il faut **transformer l'entrée** (texte, image, vidéo) pour que la machine puisse la comprendre.
- On parle de "features" = nouvelles données obtenues en transformant des données brutes.
- Manière de résumer l'information contenue dans des données complexes.

Outline

Exemple

- chaque sms = un document ;
- chaque document = suite de mots (string) ;
- chaque mot = un identifiant unique ;
- l'ordinateur ne comprend pas les mots, il comprend des chiffres.

Il faut donc transformer les document en chiffres.

TERM FREQUENCIES (TF)

Chaque document peut être représenté par la fréquence de chacun de ses mots:

$$TF(t,d) = \frac{|t \in d|}{|d|} = \frac{\text{nombre de fois où le terme apparaît}}{\text{nombre de termes}}$$

Exemple:

Document 1: "voici un premier texte"

Document 2: "voici un second texte"

Document 3: "ce document contient ce texte".

Encodage:

Document 1: "voici": 1/4, "un": 1/4, "premier": 1/4, "texte": 1/4;

Document 2: "voici": 1/4, "un": 1/4, "second": 1/4, "texte": 1/4;

Document 3: "ce": 2/5, "document": 1/5, "contient": 1/5, "texte": 1/5;

Problème de ce modèle

Données finales:

	voici	un	premier	texte	second	ce	document	contient
doc1	1/4	1/4	1/4	1/4	0	0	0	0
doc2	1/4	1/4	0	1/4	1/4	0	0	0
doc3	0	0	0	1/5	0	2/5	1/5	1/5
	2	2	1	3	1	1	1	1

- On cherche ce qui différencie les documents.
- Les mots apparaissant dans beaucoup de documents sont moins discriminants.
- Exemple: le mot "texte" n'a aucun intérêt discriminant.

Il faut donner moins d'importance aux mots apparaissant souvent.

INVERSE DOCUMENT FREQUENCIES(IDF)

On considère N documents au total. On compte **le nombre de documents où un mot apparaît au moins une fois.** On divise N par ce nombre.

$$IDF_{tmp}(t) = \frac{N}{|d : t \in d|} = \frac{\text{nombre de documents}}{\text{nombre de documents où le terme apparaît}}$$

Plus le mot est courant, plus IDF_{tmp} est petit.

	voici	un	premier	texte	second	ce	document	contient	
doc1	1/4	1/4	1/4	1/4	0	0	0	0	
doc2	1/4	1/4	0	1/4	1/4	0	0	0	
doc3	0	0	0	1/5	0	2/5	1/5	1/5	
IDF_{tmp}	1, 5	1, 5	3	1	3	3	3	3	

IDF_{tmp} **est trop fort: il risque d'annuler l'effet de mots importants (ici: "voici" et "un").**

Les poids: Inverse Document Frequencies (IDF) - suite

IDF est le logarithme de IDF_{tmp} : il accorde **peu d'importance aux mots apparaissant très souvent**, mais il **n'annule pas l'effet des mots importants**.

$$IDF(t) = \ln\left(\frac{N}{|d : t \in d|}\right)$$

	voici	un	premier	texte	second	ce	document	contient	
doc1	1/4	1/4	1/4	1/4	0	0	0		0
doc2	1/4	1/4	0	1/4	1/4	0	0		0
doc3	0	0	0	1/5	0	2/5	1/5		1/5
IDF_{tmp}	1, 5	1, 5	3	1	3	3	3		3
IDF	0, 4	0, 4	1, 1	0	1, 1	1, 1	1, 1		1, 1

Modèle final : TF-IDF

Tfldf : fréquence du terme dans le document pondéré par l'importance inverse du terme dans l'ensemble des documents.

$$\text{Tfldf}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

	voici	un	premier	texte	second	ce	document	contient	
doc1	0, 1	0, 1	0, 275	0	0	0	0		0
doc2	0, 1	0, 1	0	0	0, 275	0	0		0
doc3	0	0	0	0	0	0, 44	0, 22		0, 22

Amélioration du modèle

Certains facteurs peuvent modifier crucialement le modèle:

- La casse : ("Yes", "YES", "yes")
- Les fautes d'orthographe ou de frappe ("helo", "hello", "helol")
- Les mots de la même famille ("write", "writing", "wrote", "written", "writer")
- La ponctuation ("U.S.A.", "<3", ...)
- Les chiffres ("0800555344", "0800655877", "4", "1000")
- Les URLs ("www.cashbin.co.uk", "www.b4utele.com",...)
- Les synonymes ("great", "amazing", "fantastic")
- Les caractères spéciaux ("$", "@","%")
- Les smileys et écritures "texto" ("LOL", "LMAO", ":-)")

C'est là que commence réellement le travail du data miner: que modifier? De quelle manière? Il n'y a pas de solution unique (mais certaines sont meilleures que d'autres !).

Amélioration du modèle - suite

Lesquels de ces points vous semblent importants? Comment régler les problèmes liés?

La casse

"Yes", "YES", "yes"

La casse

"Yes", "YES", "yes"

→ ces trois mots sont les mêmes, mais les spams ont tendance à utiliser plus de majuscules (qui se suivent). Les noms propres ("France", "Sharon", "Destiny") peuvent aussi poser problème.

Fautes d’orthographe / de frappe

"helo", "hello", "helol"

Fautes d’orthographe / de frappe

"helo", "hello", "helol"

→ on peut régler ce problème en passant un correcteur orthographique sur les textes.

Les mots de la même famille

"write", "writing", "wrote", "written", "writer"

Les mots de la même famille

"write", "writing", "wrote", "written", "writer"

→ on peut utiliser le **stemming** ou la **lemmatization**:

- Stemming: writ(e) = writ(ing) = writ(er) = writ(ten) = writ ; wrot(e) = wrot.
- Lemmatization : write = writing = wrote = written= writer si les mots sont connus. Sinon, pas de changement.

La ponctuation

"U.S.A.", "<3", "..."

La ponctuation

"U.S.A.", "<3", "..."

→ Afin de ne pas interpréter différemment deux sigles identiques ("USA" et "U.S.A."), on peut choisir de supprimer certains types de ponctuation. La ponctuation peut cependant avoir un sens (":-)", "...") et nous ne voulons pas perdre cette information. Il s'agit donc de créer un dictionnaire "à la main" permettant d'identifier les formes courantes "intéressantes".

Les chiffres

Les chiffres ("0800555344", "0800655877", "4", "1000")

Les chiffres

Les chiffres ("0800555344", "0800655877", "4", "1000")

→ Tous les chiffres ne se ressemblent pas! Il faut donc identifier les numéros de téléphone, les prix, les quantités... On peut alors regrouper les chiffres représentant la même chose.

Les URLs

"www.cashbin.co.uk", "www.b4utele.com"

Les URLs

"www.cashbin.co.uk", "www.b4utele.com"

→ Il sera souvent difficile de détecter si une URL provient d'un spam ou non. En revanche, on peut regrouper toutes les URL comme un même "mot".

Les synonymes

Les synonymes ("great", "amazing", "fantastic")

Les synonymes

Les synonymes ("great", "amazing", "fantastic")

→ Il est dommage d'encoder comme deux mots différents des mots qui ont le même sens. Une étape avancée peut donc consister à utiliser un dictionnaire de synonymes pour regrouper des mots très similaires sous un identifiant unique.

Les caractères spéciaux

"$", "@","%"

Les caractères spéciaux

"$", "@","%"

→ Certains de ces caractères spéciaux ont-ils un sens particulier? Par exemple, "$" peut-être associé à un chiffre et nous permettre d'identifier un prix...

Les smileys et l’écriture "texto"

":-P","lol", "LMAO", ":/"

Les smileys et l’écriture "texto"

":-P","lol", "LMAO", ":/"

→ Comme pour les smileys, on peut définir un dictionnaire d’expressions typiques "textos". Ce n’est pas une priorité.

Faut-il tout faire? Dans quel ordre?

Il y a potentiellement beaucoup de choses à modifier. Toutes n'ont pas la même importance et il est rare qu'on pense à tout immédiatement. On peut **procéder itérativement**:

Tant qu'on n'est pas satisfait:

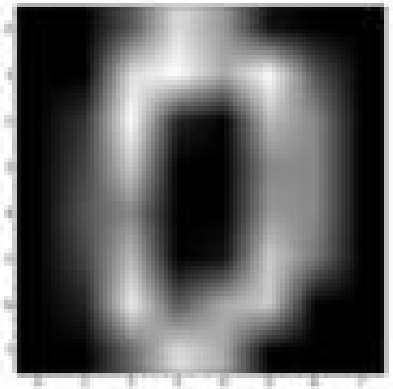
1. Apporter une nouvelle amélioration
2. Encoder
3. Faire les analyses/ prédictions

Rapport temps/qualité du modèle: préfère-t-on l'ultra-performance en 30 heures de travail ou quelque chose de correct en 1 heure?

Outline

Exemple

- Point de départ : une image N/B de 8 x 8 pixels.



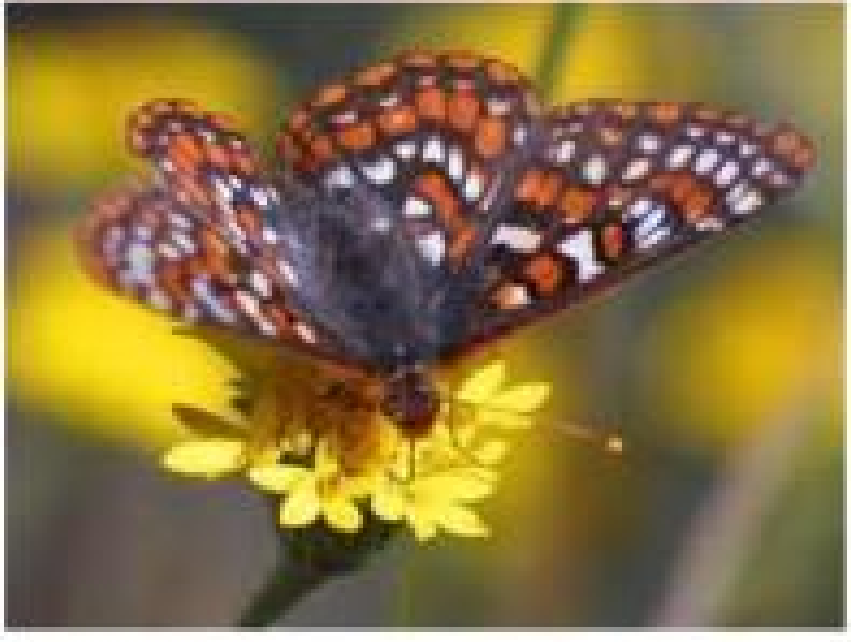
- Chaque **pixel** est transformé en une **valeur** entre 0 (noir) et 255 (blanc). Toutes les valeurs intermédiaires représentent des niveaux de gris.

```
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

- La **matrice** obtenue est transformée en **vecteur**, afin que l'image soit traitée par un algorithme.

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13.,
       15., 10., 15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,
        8.,  0.,  0.,  4., 12.,  0.,  0.,  8.,  8.,  0.,  0.,
        5.,  8.,  0.,  0.,  0.,  9.,  8.,  0.,  4., 11.,  0.,
        1., 12.,  7.,  0.,  0.,  2., 14.,  5., 10., 12.,  0.,
        0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

Des images plus compliquées



Quels éléments regarder ?

Une fois encore, ce qu'on regarde dépend de la question posée.

Exemples :

- Trouver les images dont la couleur principale est jaune : on va s'intéresser à la matrice de couleurs (cette fois en 3 ou 4 dimensions) et tenter de repérer les pixels jaunes.
- Trier les images d'insectes et les images de fleurs : la couleur peut aider, mais on s'interessera surtout aux formes des objets dans les images.

Nous ferons peu de traitement d'images en cours/TP car le problème est difficile et long à traiter. Mais cela peut faire l'objet d'un projet.

Autres types de données

Encoder les données consiste à créer des **indicateurs chiffrés**, des valeurs qui **résument** les données. Comment encoder:

- Un article de journal?
- Un film?
- Une offre d'emploi?
- Un match de foot?
- Un électeur?
- Une ville?
- Un utilisateur Facebook?
- Une maladie génétique ?

Conclusions sur l'encodage

- Encoder les données consiste entre autres à **choisir ce qui est important** pour la suite.
- Pas la partie la plus drôle.
- Probablement **la partie la plus déterminante.**
- En tout cas une partie où l'homme (vs machine) joue un rôle important.