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. Établir une règle de séparation;
3. Appliquer la règle à de nouveaux messages.

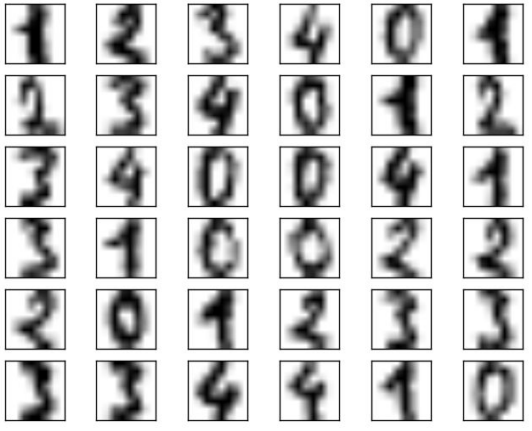
EX 2: RECONNAISSANCE DE CHIFFRES

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.

ENCODER DU TEXTE

INTRODUCTION

Dans notre 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

	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. Pour chaque mot, on compte le nombre de documents où ce 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").

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

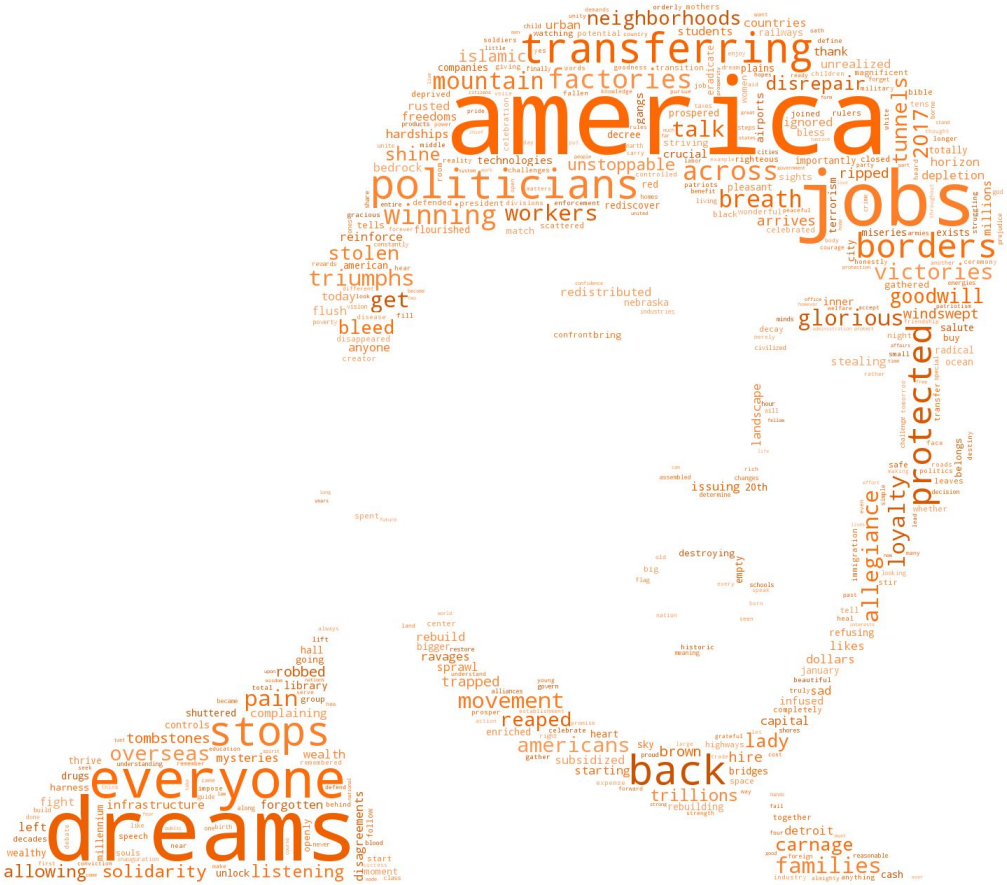
MODÈLE FINAL : TF-IDF

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

$$TfIdf(t, d) = TF(t, d) \times IDF(t)$$

	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

EXEMPLE



Source: http://keyonvafa.com/inauguration-wordclouds/

CONCLUSION SUR LE TF-IDF

- Il nous dit pour chaque document quels sont les mots qui apparaissent plus dans ce document que dans les autres.
- Même en soi, les résultats sont intéressants, mais l'idée est surtout de les utiliser dans des modèles prédictifs.
- Par contre, le TF-IDF considère que tous les ensembles de lettres séparés par un espace sont des mots qui ont un intérêt. Il ne corrige pas les fautes de frappe, et il ne se rend pas compte que "vends" et "vendre" ont un sens très lié.
- En fonction de la qualité des données en entrée, vous allez peut-être devoir faire quelques travaux manuels.

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 !).

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"

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

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", "..."

→ 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")

→ 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"

→ 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")

→ 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

"$", "@","%"

→ 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", ":/"

→ 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:

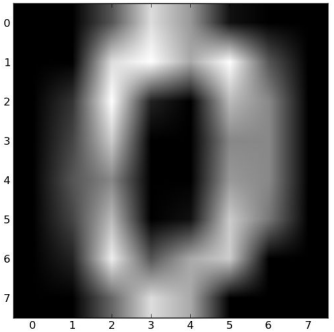
1. Apporter une nouvelle amélioration
2. Encoder
3. Faire les analyses/ prédictions

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

ENCODER DES IMAGES

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.

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

Si l'on a plusieurs images, chacune est représentée par une ligne (un

```
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.]])
```

```
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.])
```

DES IMAGES PLUS COMPLIQUÉES

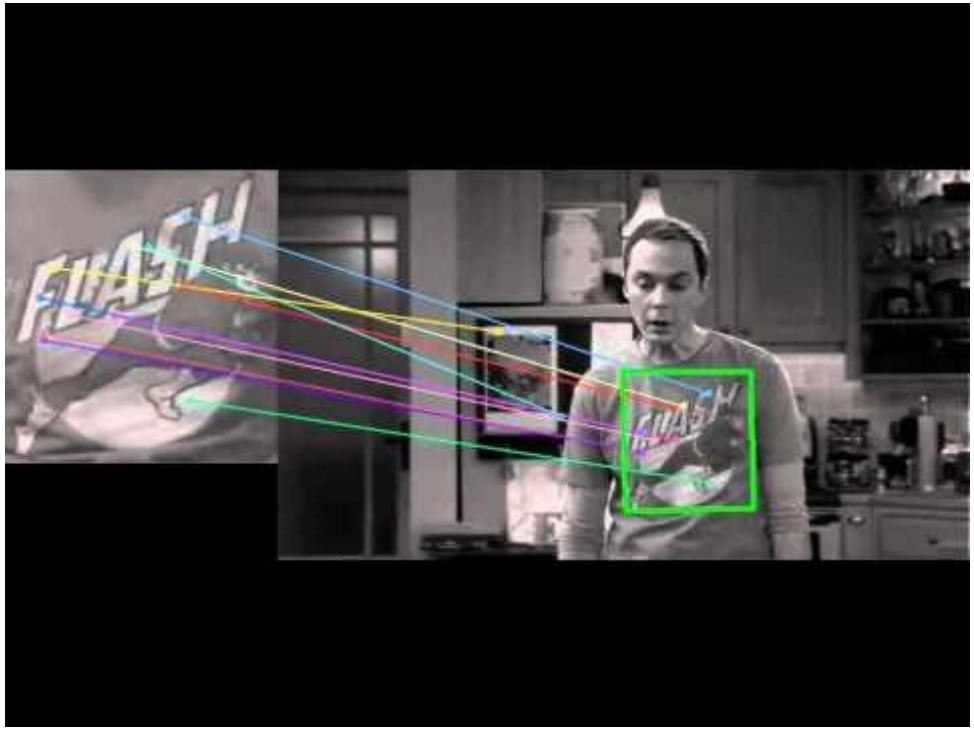
RÉSUMER L'INFORMATION 1: SIFT

SIFT (Scale Invariant Feature Transform) repère des points saillants dans l'image qui vont résumer cette image.

source: https://www.kieranoshea.com

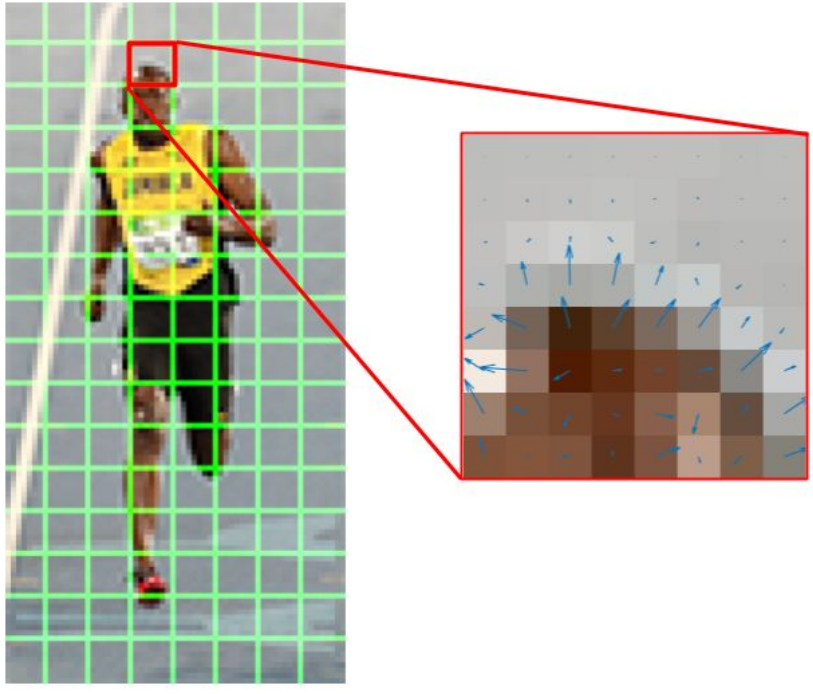
RÉSUMER L'INFORMATION 2: SURF

SURF (Speeded Up Robust Features) s'inspire de SIFT et des ondelettes, en théorie du signal.



RÉSUMER L'INFORMATION 3: HOG

HOG (Histogram of Oriented Gradients): s'intéresse aux courbes dans l'image, détectées par des gradients de couleur.



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

source: www.learnopencv.com

C'EST DU DEEP LEARNING?

Non! Ce n'est pas du machine learning, et donc pas du deep learning. Les autres images ne sont pas utilisées, et les labels des images non plus. C'est du traitement du signal.

L'idée du deep learning est que l'algorithme trouve les features intéressantes "tout seul". Cela a au moins 4 coûts: 1/ il va lui falloir beaucoup, beaucoup, beaucoup d'images. 2/ on ne sait pas interpréter ces modèles, 3/ ils sont très difficiles et longs à tuner, 4/ complexité en termes de calculs, 5/…. 6/…. (on les verra plus tard!)

Si vos projets traitent des images ou du texte, commencez maintenant avec les méthodes de ce cours-ci: on ne se lance pas dans du deep learning sans raison. C'est que tout le reste aura échoué.

EXEMPLE DE PROJET

Problème: classifier des champignons pour déterminer s'ils sont toxiques.

VARIABLES CATÉGORIELLES

- Jour de la semaine => transformer une colonne contenant “lundi”, “mardi”,... en 7 colonnes où une seule contient un 1 et les autres 0.
- “perdu, nul, gagné”: peut être encodé en 0, 1, 2, car les valeurs sont comparables (elles ont un ordre).
- Représenter des goûts: la personne 1 aime Batman et Superman, la personne 2 aime Batman et Iron Man. On va à nouveau utiliser des 0 et des 1, mais la somme n’est pas limitée à 1 par ligne.
- etc. Vous rencontrerez tous types de données! En cas de doute, nous demander!

CONCLUSION

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.