

Introduction

François HU

Data Scientist au DataLab de la Société Générale Assurances - 13/10/2020

Les cours se trouvent ici : <https://curiousml.github.io/>

Sommaire

1. Quelques prérequis

- Machine Learning
- Deep Learning

2. Définitions et preprocessing

- Tokenization
- Normalisation des tokens
- Stop-words

3. Vectorisation des données textuelles

- Données textuelles en données tabulaires
- Modèles de vectorisation des textes

Programme

Introduction

Représentations vectorielles

Deep Learning pour NLP

Active Learning

1. Quelques prérequis

Machine Learning

Machine Learning (ML) (Apprentissage machine ou automatique) : la science de programmer les ordinateurs de sorte qu'ils puissent apprendre à partir des données (que nous notons X)

Machine Learning

Machine Learning (ML) (Apprentissage machine ou automatique) : la science de programmer les ordinateurs de sorte qu'ils puissent apprendre à partir des données (que nous notons X)

Apprentissage supervisé : apprentissage machine où les données d'entraînement sont étiquetées $\rightarrow (X, y)$

- Classification (régression logistique, régression softmax, ...)
- Régression, ...

Apprentissage non-supervisé : apprentissage machine où les données d'entraînement **ne** sont **pas** étiquetées $\rightarrow X$

- Partitionnement (clustering)
- Réduction de la dimensionalité, ...

Machine Learning

Machine Learning (ML) (Apprentissage machine ou automatique) : la science de programmer les ordinateurs de sorte qu'ils puissent apprendre à partir des données (que nous notons X)

Apprentissage supervisé : apprentissage machine où les données d'entraînement sont étiquetées $\rightarrow (X, y)$

- Classification (régression logistique, régression softmax, ...)
- Régression, ...

Apprentissage non-supervisé : apprentissage machine où les données d'entraînement **ne** sont **pas** étiquetées $\rightarrow X$

- Partitionnement (clustering)
- Réduction de la dimensionalité, ...

Phase d'un système de Machine Learning : phase d'apprentissage, phase de prédiction

Machine Learning

Machine Learning (ML) (Apprentissage machine ou automatique) : la science de programmer les ordinateurs de sorte qu'ils puissent apprendre à partir des données (que nous notons X)

Apprentissage supervisé : apprentissage machine où les données d'entraînement sont étiquetées $\rightarrow (X, y)$

- Classification (régression logistique, régression softmax, ...)
- Régression, ...

Apprentissage non-supervisé : apprentissage machine où les données d'entraînement **ne** sont **pas** étiquetées $\rightarrow X$

- Partitionnement (clustering)
- Réduction de la dimensionalité, ...

Phase d'un système de Machine Learning : phase d'apprentissage, phase de prédiction

Comment entraîner un modèle ML ?

choisir un modèle prédictif puis calibrer ses paramètres (souvent via minimisation d'une **métrique** par **descente de gradient**)

Machine Learning

Machine Learning (ML) (Apprentissage machine ou automatique) : la science de programmer les ordinateurs de sorte qu'ils puissent apprendre à partir des données (que nous notons X)

Apprentissage supervisé : apprentissage machine où les données d'entraînement sont étiquetées $\rightarrow (X, y)$

- Classification (régression logistique, régression softmax, ...)
- Régression, ...

Apprentissage non-supervisé : apprentissage machine où les données d'entraînement **ne** sont **pas** étiquetées $\rightarrow X$

- Partitionnement (clustering)
- Réduction de la dimensionalité, ...

Phase d'un système de Machine Learning : phase d'apprentissage, phase de prédiction

Comment entraîner un modèle ML ?

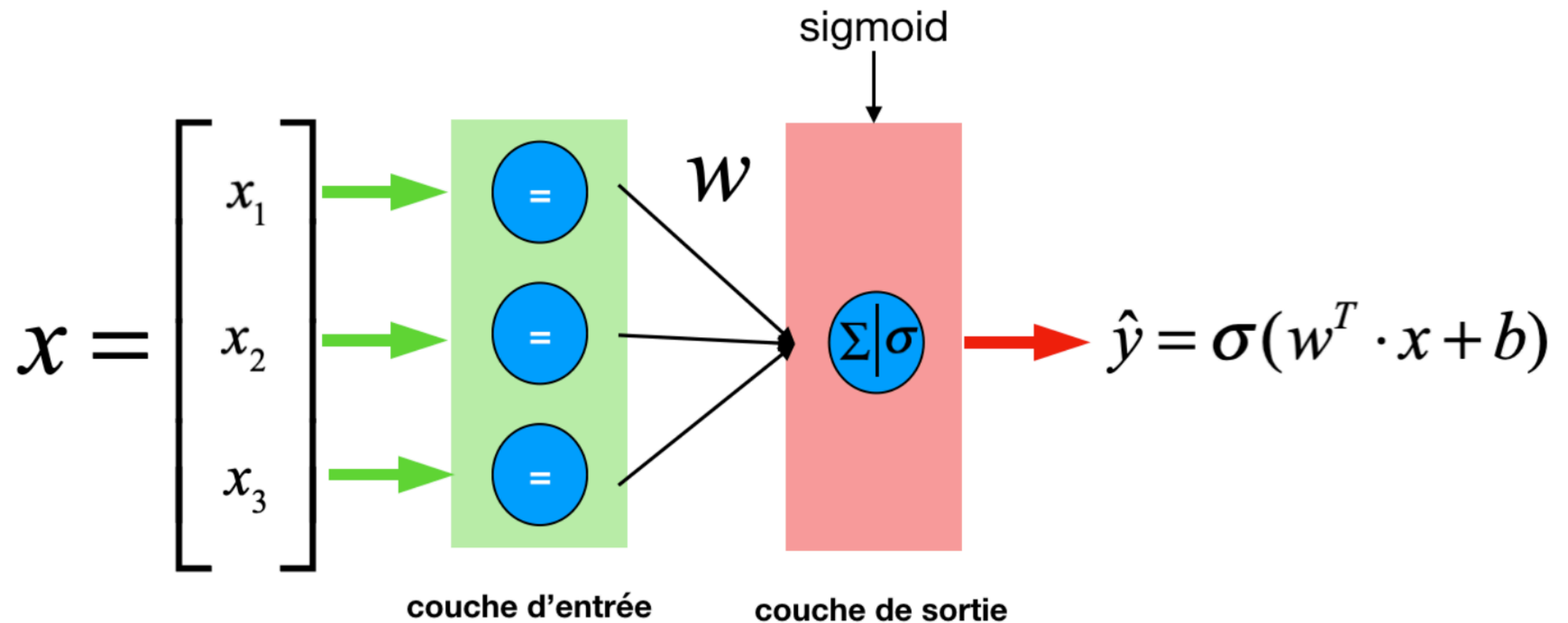
choisir un modèle prédictif puis calibrer ses paramètres (souvent via minimisation d'une **métrique** par **descente de gradient**)

Natural Language Processing (NLP) ?

La science de programmer les ordinateurs à comprendre le langage humain

Deep Learning (1/3)

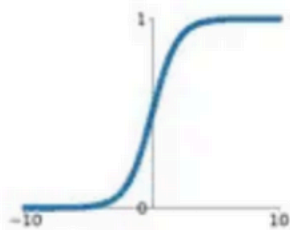
Régression logistique



Quelques fonctions d'activation

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



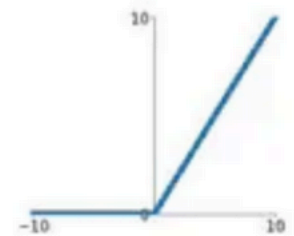
tanh

$$\tanh(x)$$



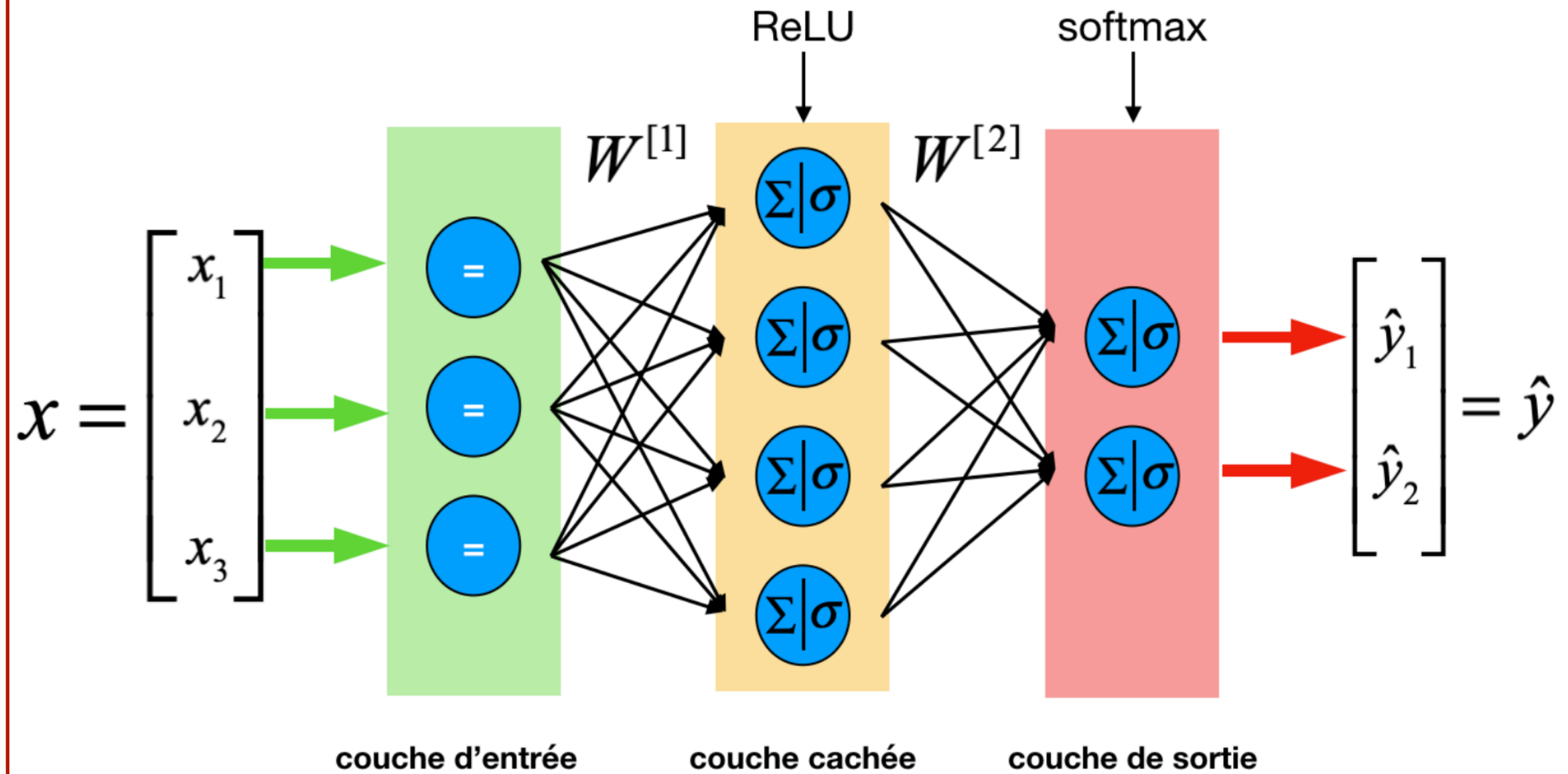
ReLU

$$\max(0, x)$$



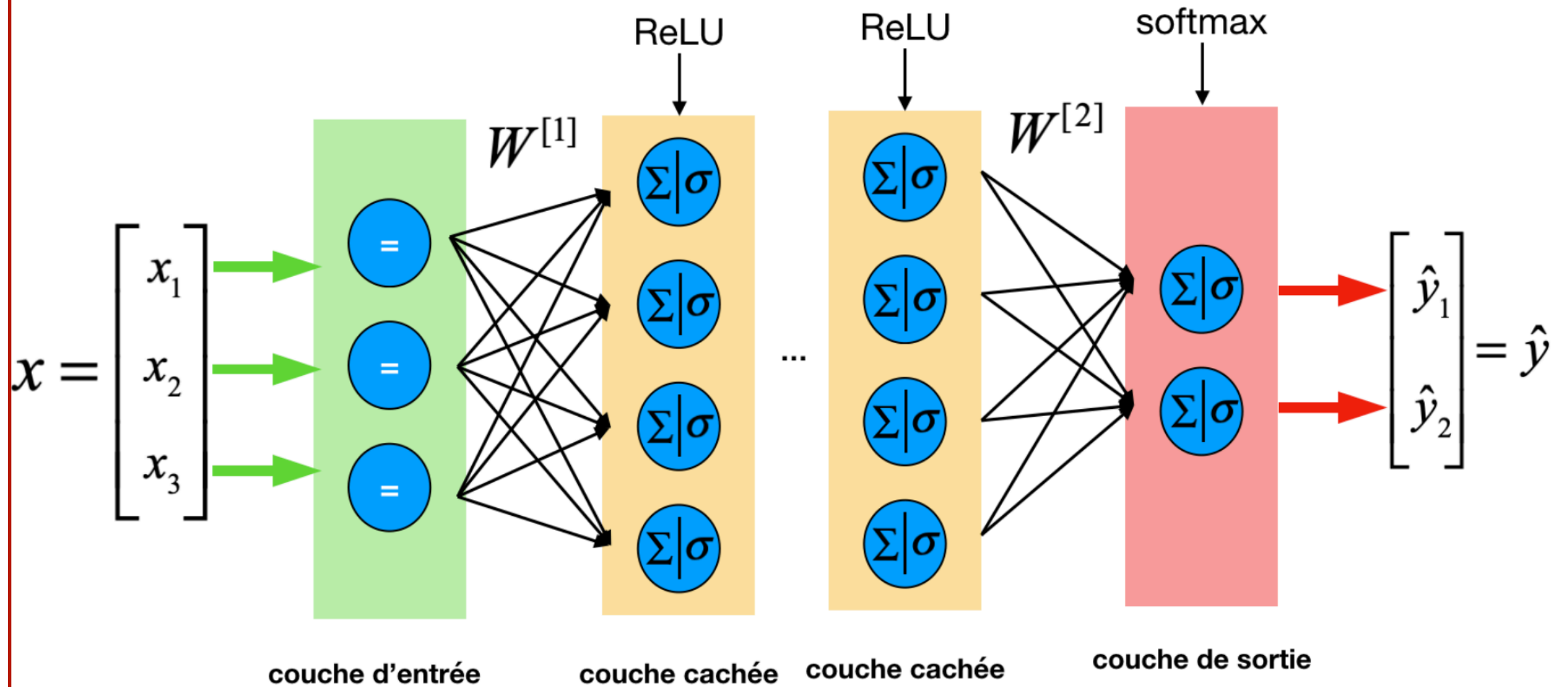
Deep Learning (2/3)

Réseau de neurones



Deep Learning (3/3)

Deep Learning



2. Définitions et Préprocessing

Intuitions :

- formater les textes pour se rapprocher des données tabulaires
(structurer)
- rendre les textes simple à traiter pour les modèles ML
(normaliser et/ou supprimer des mots)

Tokenization : définition

Texte : séquence de mots

Mot : séquence logique de caractères

Tokenization : processus qui sépare une séquence (texte) en une liste de tokens (mots)

Question : comment trouver les limites d'un mot ?

Réponse : En français / anglais, nous pouvons séparer les mots par les **espaces** et les **ponctuations**

Input : 'Devons-nous,
mon collègue et moi,
vous recontacter ?'

Tokenization

Output : ['Devons',
'nous', 'mon', 'collègue',
'et', 'moi', 'vous',
'recontacter']

implémentation python

```
from nltk.tokenize import WhitespaceTokenizer, WordPunctTokenizer, word_tokenize
text = "J'ai une question : devons-nous, mon collègue et moi, vous recontacter?"
```

Tokenization : code

implémentation python

- **exemple 1** : tokenization par les espaces

```
tokenizer = WhitespaceTokenizer()  
print(tokenizer.tokenize(text))
```

```
["J'ai", 'une', 'question', ':', 'devons-nous,', 'mon', 'collègue', 'et', 'moi,', 'vous', 'recontacter?']
```

- **exemple 2** : tokenization par les ponctuations

```
tokenizer = WordPunctTokenizer()  
print(tokenizer.tokenize(text))
```

```
['J', '"', 'ai', 'une', 'question', ':', 'devons', '-', 'nous', ',', 'mon', 'collègue', 'et', 'moi', ',', 'vous', 'recontacter', '?']
```

- **exemple 3** : tokenization par un ensemble de règles

```
print(word_tokenize(text, language='french'))
```

```
["J'ai", 'une', 'question', ':', 'devons-nous', ',', 'mon', 'collègue', 'et', 'moi', ',', 'vous', 'recontacter', '?']
```

Normalisation : Racinisation et lemmatisation

Stemming (Racinisation) : garder la racine d'un terme (souvent, couper à partir d'un caractère)

- **exemple** : continua^a, continuait^{ait}, continuant^{ant}, continuation^{ation}, continuation^{ation}, continue^e → continu

implémentation python

```
from nltk.stem import SnowballStemmer
fr = SnowballStemmer('french')
" ".join(fr.stem(token) for token in word_tokenize(text))
```

```
"j ' ai une question : devon - nous , mon collègu et moi , vous recontact ?"
```

Lemmatisation : processus qui sépare une séquence (texte) en une liste de tokens (mots)

- **exemple** : continua, continuait, continuant → continuer
continuations, continuation → continuation
continue → continu / continuer (adjectif / verbe)

implémentation python

```
import spacy
nlp = spacy.load('fr_core_news_md')
" ".join(token.lemma_ for token in nlp(text))
```

```
'je avoir un question : devoir - nous , mon collègue et moi , vous recontacter  
'
```

Autres types de normalisation

Les **expressions régulières** en Python nécessitent d'importer le module natif **re** : `import re`
voir https://fr.wikibooks.org/wiki/Programmation_Python/Regex pour la documentation.

Exemple de texte :

`j'ai une question : devons-nous, mon collègue et moi, vous recontacter?`

- supprimer les **balises** : `<...> ... </...>`

implémentation python

```
text_b = re.sub("<.*?>", " ", text_b) # balises <...>
print(text_b)
```

j'ai une question : devons-nous, mon collègue et moi, vous recontacter?

- supprimer les **ponctuations** et les **grands espaces** : `-, !, ?, :, ...`

implémentation python

```
import string
text_b = re.sub(r"[" + string.punctuation + r"]", " ", text_b) # ponctuations
text_b = re.sub(r"\s+", " ", text_b) # grands espaces
print(text_b)
```

j ai une question devons nous mon collègue et moi vous recontacter


Stop-words

Les **stop-words** : ensemble de mots fréquemment utilisés dans une langue et qui n'apportent pas de signification importante

- exemple : a, à, des, de, et, est, un, ...

→ **objectif** : supprimer ces stop-words

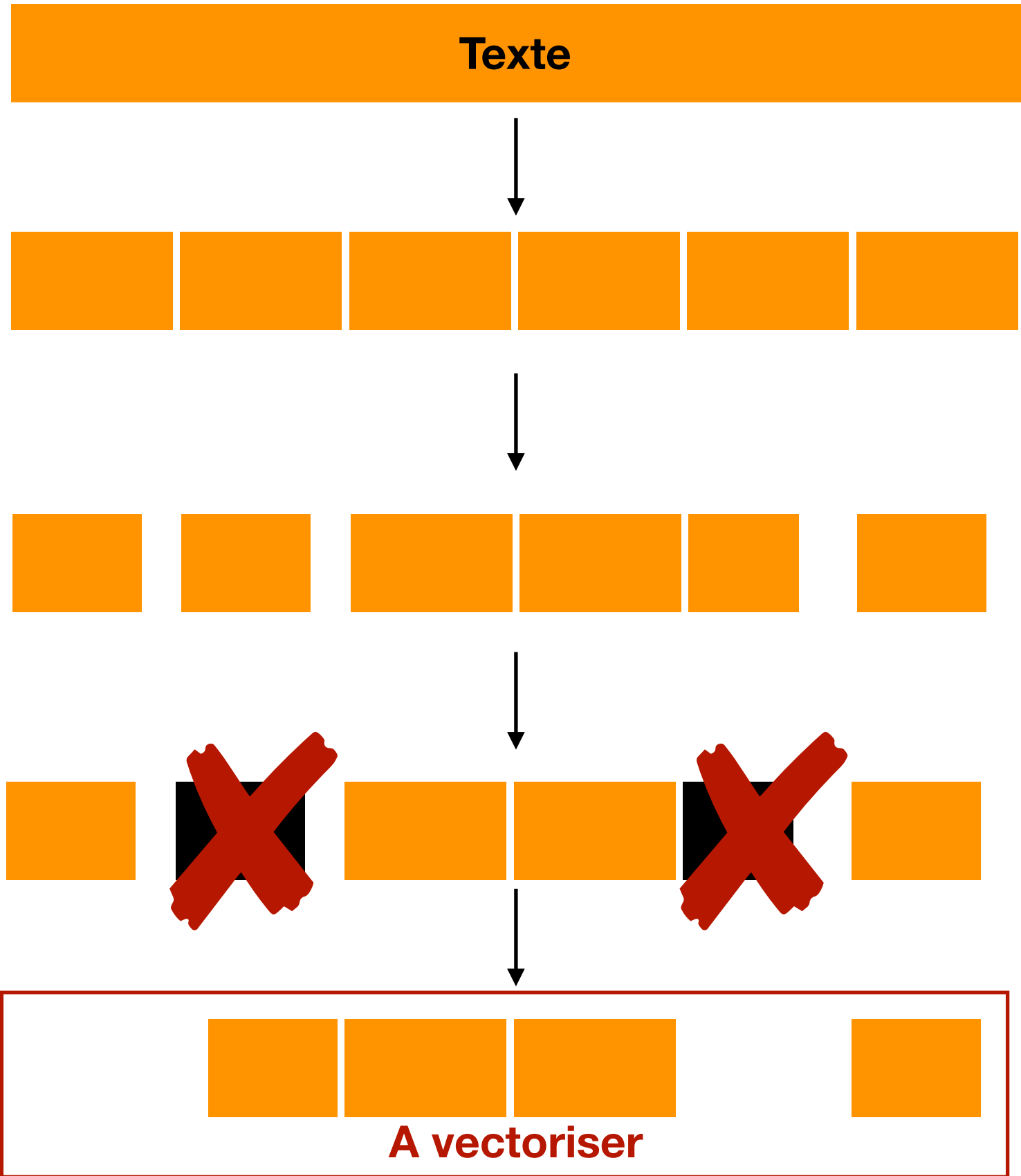
implémentation python



```
from nltk.corpus import stopwords
# Chargement des stopwords pour la langue française avec NLTK
stopwords_lst = list(stopwords.words("french"))
re.sub(r"(\s+|^)(" + r"|".join(stopwords_lst) + r")(\s+|$)", " ", text)
```

"J'ai question : devons-nous, collègue moi, recontacter?"

Résumé



3. Vectorisation des données textuelles

Intuitions :

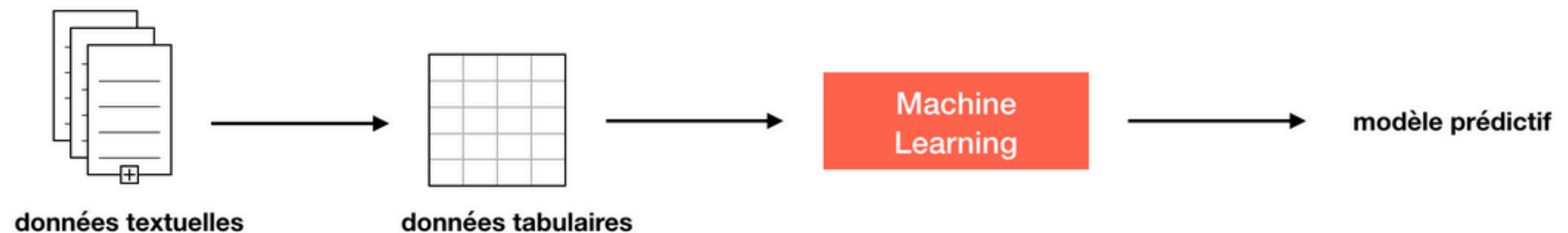
- Transformer des documents textuels en données tabulaires
(partir des données textuelles tokenisées et normalisées)
- Données textuelles => données séquentielles (tenir compte de l'ordre des mots)

Données textuelles en données tabulaires

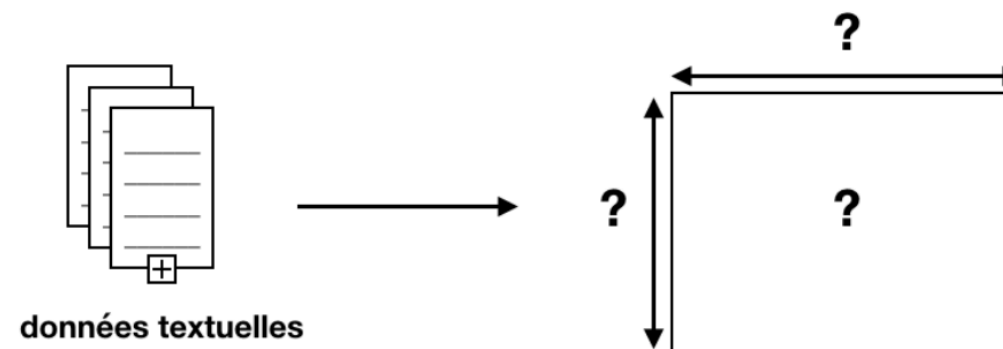
processus d'apprentissage pour des données **tabulaires**



processus d'apprentissage pour des données **textuelles**



Problèmes



- approches : (**section suivante**) TF-IDF et (**cours 1**) word embedding

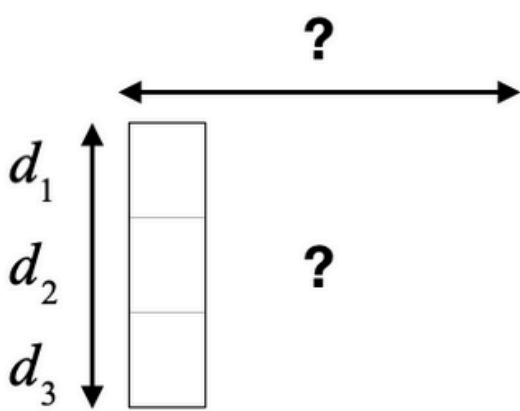
Vectorisation des textes



d_1	trouver bonne assurance
d_2	contrat satisfaisant
d_3	changement contrat assurance

$V = \{$
 'assurance' : 1,
 'bonne' : 2,
 'changement' : 3,
 'contrat' : 4,
 'satisfaisant' : 5,
 'trouver' : 6
 $\}$

assurance	contrat
1	0
0	0
0	0
0	1
0	0
0	0




	assurance	bonne	changement	...
d_1				
d_2			?	
d_3				

	assurance	bonne	changement	...		
d_1	1	1	0	0	0	1
d_2	0	0	0	1	1	0
d_3	1	0	1	1	0	0

Vectorisation des textes : approche bag-of-words

- vectorise un document en comptant le **nombre d'occurrences** d'un token t dans le document d : $f_{t,d}$
- **exemple** : méthode de comptage

trouver bonne assurance		trouver	contrat	assurance	...
contrat satisfaisant		1	0	1	...
changement contrat assurance		0	1	0	...
		0	1	1	...

- **problème** : pas d'ordre entre les mots
- **solution** : approche n-grammes

Vectorisation des textes : approche n-grammes

- vectorise un document en comptant le **nombre d'occurrences** des paires de tokens (2-grams), des triplets de tokens (3-grams), ...
- **exemple** : méthode de comptage (1,2)-grammes

trouver bonne assurance
contrat satisfaisant
changement contrat assurance



trouver	assurance	contrat assurance	...
1	1	0	...
0	0	0	...
0	1	1	...

- **problème** : trop de variables
- **solution** : supprimer les stop-words et qq n-grammes (très **hautes** et très **basses** fréquences)

Vectorisation des textes : approche TF-IDF (1/2)

- **Term Frequency** : nombre d'occurrences d'un token / n-grams t dans le document d

$$\text{tf}(t, d) = f_{t,d}$$

- **variantes** : $\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$ ou $\mathbb{1}(t \in d)$ ou $(1 + \log f_{t,d})$, ...

- **Inverse Document Frequency** : mesure l'importance du token / n-grams dans l'ensemble du corpus

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d \mid t \in d\}|} = \log \frac{\# \text{ documents}}{\# \text{ documents contenant le terme } t}$$

- **Term Frequency - Inverse Document Frequency (TF-IDF)** :

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

Vectorisation des textes : approche TF-IDF (2/2)

- **exemple** : approche TF-IDF (1,2)-grammes

trouver bonne assurance		trouver	assurance	contrat assurance	...
contrat satisfaisant	→	0.10	0.41	0	...
changement contrat assurance		0	0	0	...
		0	0.41	0.10	...

implémentation python

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1,2))
```

```
features = tfidf.fit_transform(texts)
```

- voir **cours 1** pour des techniques de vectorisation plus avancées

Résumé

trouver bonne assurance		trouver	contrat	assurance	...
contrat satisfaisant	→	1	0	1	...
changement contrat assurance		0	1	0	...
		0	1	1	...

trouver bonne assurance		trouver	assurance	contrat assurance	...
contrat satisfaisant	→	1	1	0	...
changement contrat assurance		0	0	0	...
		0	1	1	...

trouver bonne assurance		trouver	assurance	contrat assurance	...
contrat satisfaisant	→	0.10	0.41	0	...
changement contrat assurance		0	0	0	...
		0	0.41	0.10	...

Méthode 1 :
vectorisation par comptage +
bag-of-words

Méthode 2 :
vectorisation par comptage +
n-grammes

Méthode 3 :
vectorisation par TF-IDF