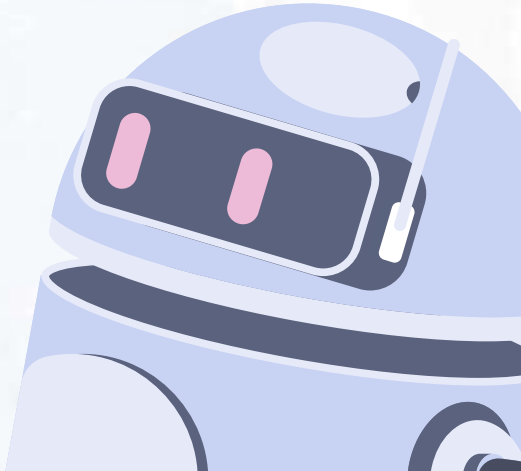
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Larbi Ben M'Hidi Oum El Bouaghi

Cours Traitement du langage naturel

Chapitre 2
Prétraitement du texte



Dr.Belhouchette K

Contenu du deuxiéme chapitre

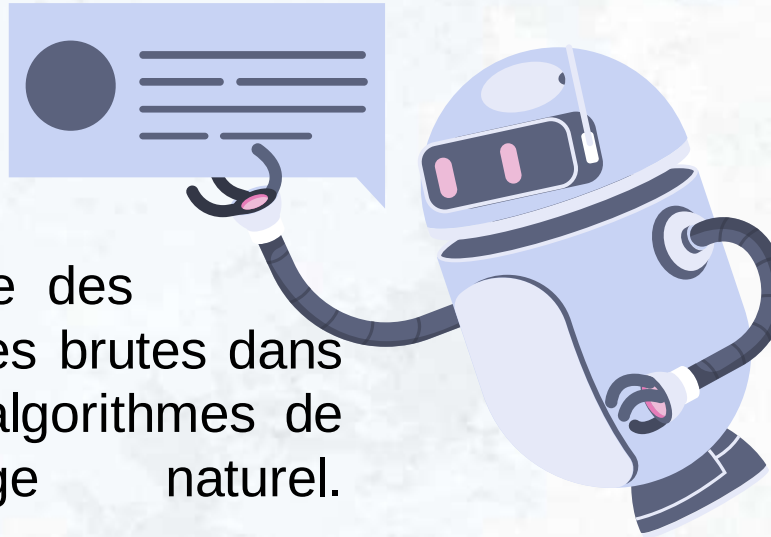
01 ⟶ Techniques de préparation de données textuelles

02 ⟶ Techniques de normalisation de texte

03 ⟶ Nettoyage de données textuelles

Techniques de préparation de données textuelles

Définition



- Le prétraitement du texte désigne l'ensemble des opérations réalisées sur des données textuelles brutes dans le but de les rendre exploitables par des algorithmes de traitement automatique du langage naturel.

- Ces étapes visent à transformer les données textuelles souvent désordonnées et bruitées en une structure cohérente et normalisée, adaptée à une analyse automatique.

Pourquoi le prétraitement est essentiel ?

- Les textes issus de sources réelles, comme les réseaux sociaux, les documents numérisés ou les bases de données textuelles

- Ces derniers présentent généralement une grande variabilité et contiennent des éléments inutilisables.

Pourquoi le prétraitement est essentiel ?

Le prétraitement est une étape fondamentale qui permet :

1. **D’assurer la cohérence des données** : Harmoniser la structure et le format des textes pour éviter les problèmes liés à la casse, aux fautes de frappe ou aux variations d'écriture.

2. **De faciliter le traitement automatique :** Convertir les données textuelles en formats exploitables tout en éliminant les informations superflues.

3. **D’améliorer les performances des modèles :** Une donnée de meilleure qualité se traduit souvent par une meilleure précision des algorithmes utilisés.

Problématiques fréquentes dans les textes bruts

Bruit linguistique : Caractères inutiles, ponctuation non pertinente ou contenu comme les balises HTML et les hyperliens.

Variabilité linguistique : Les différences dans l'écriture comme "analyse" et "analyses" ou dans les styles d'expression: abréviations, contractions: compliquent le traitement.

Données inutiles ou redondantes : Les doublons, les mots vides : "le", "la", "un« ou les informations hors contexte peuvent alourdir inutilement les analyses.

01 →

Techniques de préparation de données textuelles

01

Les techniques préparation de données textuelles

Les techniques de prétraitement de données textuelles visent à structurer et à transformer les données en un format exploitable

01

Les techniques préparation de données textuelles

1. Tokenization (Segmentation)

- Est le processus de division d'un texte en unités fondamentales appelées **tokens**.

- Ces tokens peuvent représenter des mots, des phrases, ou des sous-mots en fonction de l'approche choisie.

01

Les techniques préparation de données textuelles

1. Tokenization (Segmentation)

Pourquoi la tokenization est-elle importante ?

- Elle simplifie le traitement en décomposant les textes en unités distinctes.
- Elle constitue la base des représentations numériques par exemple: TF-IDF, Word Embeddings

01

Les techniques préparation de données textuelles

1. Tokenization (Segmentation)

01

Les techniques préparation de données textuelles

1. Tokenization (Segmentation)

Méthodes de tokenization

- **Tokenization en mots**
- Tokenization en phrases
- Tokenization en sous-mots

Diviser le texte en mots :

"Le traitement du langage naturel" → ["Le", "traitement", "du", "langage", "naturel"]).

01

Les techniques préparation de données textuelles

1. Tokenization (Segmentation)

Méthodes de tokenization

- Tokenization en mots
- **Tokenization en phrases**
- Tokenization en sous-mots

Découper le texte en phrases distinctes à l'aide de délimiteurs comme les points ou les points d'interrogation.

01

Les techniques préparation de données textuelles

1. Tokenization (Segmentation)

Méthodes de tokenization

- Tokenization en mots
- Tokenization en phrases
- **Tokenization en sous-mots**

Utilisée dans les modèles modernes comme **BERT** pour capturer les relations entre fragments de mots.

Exemple : "Traitement" → ["Trait", "ement"].

01

Les techniques préparation de données textuelles

2. Extraction des entités nommées (Named Entity Recognition, NER)

Cette technique identifie les entités importantes dans un texte, comme les noms de personnes, lieux, organisations ou dates.

Les techniques préparation de données textuelles

2. Extraction des entités nommées (Named Entity Recognition, NER)

Exemple :
Texte brut : Younes est né en Algérie

Résultat : { " Younes " : PERSON, " Algérie " : LOCATION }

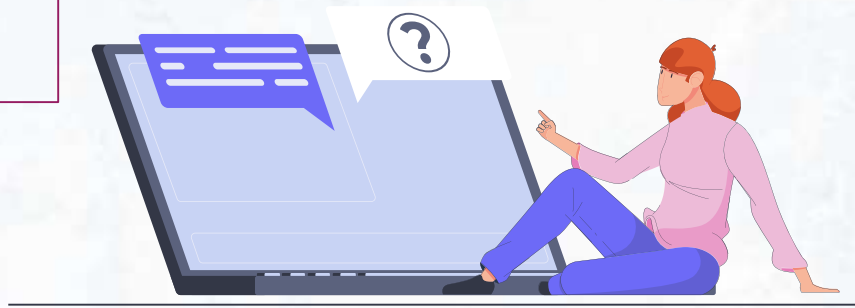
01

Les techniques préparation de données textuelles

3. Part-of-Speech Tagging (Étiquetage morphosyntaxique)

Le POS tagging consiste à annoter chaque mot d'un texte avec sa catégorie grammaticale (nom, verbe, adjectif, etc.).

Texte : "Le chat dort."
Résultat : [(Le, DET), (chat, NOUN), (dort, VERB)]



01

Les techniques préparation de données textuelles

3. Part-of-Speech Tagging (Étiquetage morphosyntaxique)

Pourquoi c’est utile ?

- Améliore l’analyse syntaxique et sémantique.
- Permet de différencier les mots ayant des significations multiples selon leur rôle "train" peut être un verbe ou un nom

01

Les techniques préparation de données textuelles

4. Vectorisation des données textuelles

transformer le texte brut en une représentation numérique utilisable par les algorithmes

- **Approches classiques**
- **Approches avancées**

Les techniques préparation de données textuelles

4. Vectorisation des données textuelles

Approches classiques

1. **Bag-of-Words (BoW)** : Chaque document est représenté par un vecteur contenant la fréquence de ses mots, sans tenir compte de l'ordre.

2. **TF-IDF (Term Frequency - Inverse Document Frequency)** : Pondère les mots en fonction de leur importance dans un document tout en réduisant l'impact des termes communs dans le corpus.

01

Les techniques préparation de données textuelles

4. Vectorisation des données textuelles

Approches avancées

1. **Word Embeddings** : Les mots sont représentés dans un espace vectoriel où des mots similaires ont des vecteurs proches comme Word2Vec, GloVe

2. **Contextual Embeddings** : Modèles modernes comme BERT, ELMo ou GPT qui prennent en compte le contexte des mots.

01

Les techniques préparation de données textuelles

Technique	Objectif	Outil couramment utilisé
Tokenization	Diviser le texte en unités manipulables	NLTK, SpaCy, WordPiece
NER	Identifier les entités importantes	SpaCy, BERT
POS Tagging	Annoter les mots avec des catégories	NLTK, Stanford NLP
Vectorisation	Représenter le texte en vecteurs numériques	TF-IDF, Word2Vec, BERT

02 →

Techniques de normalisation de texte

02

Les techniques de normalisation de texte

Ces techniques visent à uniformiser et standardiser le texte pour réduire la complexité et la variabilité

02

Les techniques de normalisation de texte

1. Conversion en minuscules

uniformiser les casses des caractères pour éviter les ambiguïtés dues à la distinction entre majuscules et minuscules.

```python
def convert_to_lowercase(text):
    return text.lower()
```

Texte brut : "Bonjour Monde, BONJOUR monde !"
Texte normalisé : "bonjour monde, bonjour monde !"

Pourquoi c’est utile ?

- Éviter que "Bonjour" et "bonjour" soient traités comme deux mots différents.
- Simplifier les comptages de fréquences et les comparaisons lexicales.

02

Les techniques de normalisation de texte

2. Suppression des accents et caractères spéciaux

- Les accents ou caractères spéciaux peuvent poser problème dans les systèmes non conçus pour les traiter correctement.

- Leur suppression permet une meilleure compatibilité et une uniformité des données.

```
phrase = ''.join(c for c in unicodedata.normalize('NFD', phrase) if unicodedata.category(c) != 'Mn')
resultat.set(phrase)
```

02

Les techniques de normalisation de texte

```
phrase = ''.join(c for c in unicodedata.normalize('NFD', phrase) if unicodedata.category(c) != 'Mn')
resultat.set(phrase)
```

unicodedata.normalize('NFD', phrase) :
Convertit les lettres accentuées en une combinaison de lettre de base + accent séparé.
Ex : "é" devient "é" (lettre e + un caractère séparé pour l'accent).

unicodedata.category(c) != 'Mn' :
Mn signifie Mark Nonspacing (marques diacritiques = accents)
On les exclut, ce qui supprime tous les accents

Les techniques de normalisation de texte

3. Suppression de la ponctuation

La ponctuation, bien qu'utile pour la structure grammaticale, peut souvent être retirée pour les analyses où elle n'apporte pas de valeur significative.

```
def supprimer_ponctuation():
    phrase = entree.get()
    phrase_sans_ponctuation = phrase.translate(str.maketrans('', '', string.punctuation))
    resultat.config(text="Résultat : " + phrase_sans_ponctuation)
```

```
str.maketrans(x, y, z) crée une table de traduction :
x : caractères à remplacer (ici, vide '').
y : caractères de remplacement (ici, vide '').
z : caractères à supprimer ici, string.punctuation, qui contient tous les signes de
ponctuation en python
```

Texte brut : "Bonjour, comment allez-vous ?"

Texte normalisé : "Bonjour comment allez vous"

02

Les techniques de normalisation de texte

4. Gestion des mots vides (stop words)

- Les mots vides, tels que "le", "de", "et", "à", apportent peu ou pas de valeur sémantique mais sont très fréquents dans les textes.

- Leur suppression permet de réduire la taille des données tout en conservant les informations importantes.

```
def supprimer_stopwords():
    phrase = entree.get()
    stop_words = set(stopwords.words('french'))
    mots = phrase.split()
    mots_sans_stopwords = [mot for mot in mots if mot.lower() not in stop_words]
```

Les techniques de normalisation de texte

4. Gestion des mots vides (stop words)

```python
def supprimer_stopwords():
    phrase = entree.get()
    stop_words = set(stopwords.words('french'))
    mots = phrase.split()
    mots_sans_stopwords = [mot for mot in mots if mot.lower() not in stop_words]
```

stopwords.words('french') :
Cette commande appartient à NLTK: Natural Language Toolkit une bibliothèque Python puissante et populaire pour NLP. Elle permet d'analyser, manipuler et comprendre le texte en utilisant des techniques avancées.
Elle charge une liste de mots vides en français comme: "le", "la", "de", "et", "est", "aux", "en", "du"
set(...) permet de convertir cette liste en ensemble (set) pour une recherche plus rapide.

Texte brut : "Le chat est sur le tapis."
Texte normalisé sans stop words : "chat tapis"

02

Les techniques de normalisation de texte

5. Stemming (Racisation)

Le stemming consiste à réduire un mot à sa racine en supprimant ses suffixes et préfixes, souvent sans tenir compte de la grammaire.

02

Les techniques de normalisation de texte

5. Stemming (Racisation)

```
stemmer = SnowballStemmer("french")
mots_stemmes = [stemmer.stem(mot) for mot in mots]
```

Création d'un objet stemmer basé sur l'algorithme Snowball pour la langue française

Application du stemming sur chaque mot de la liste

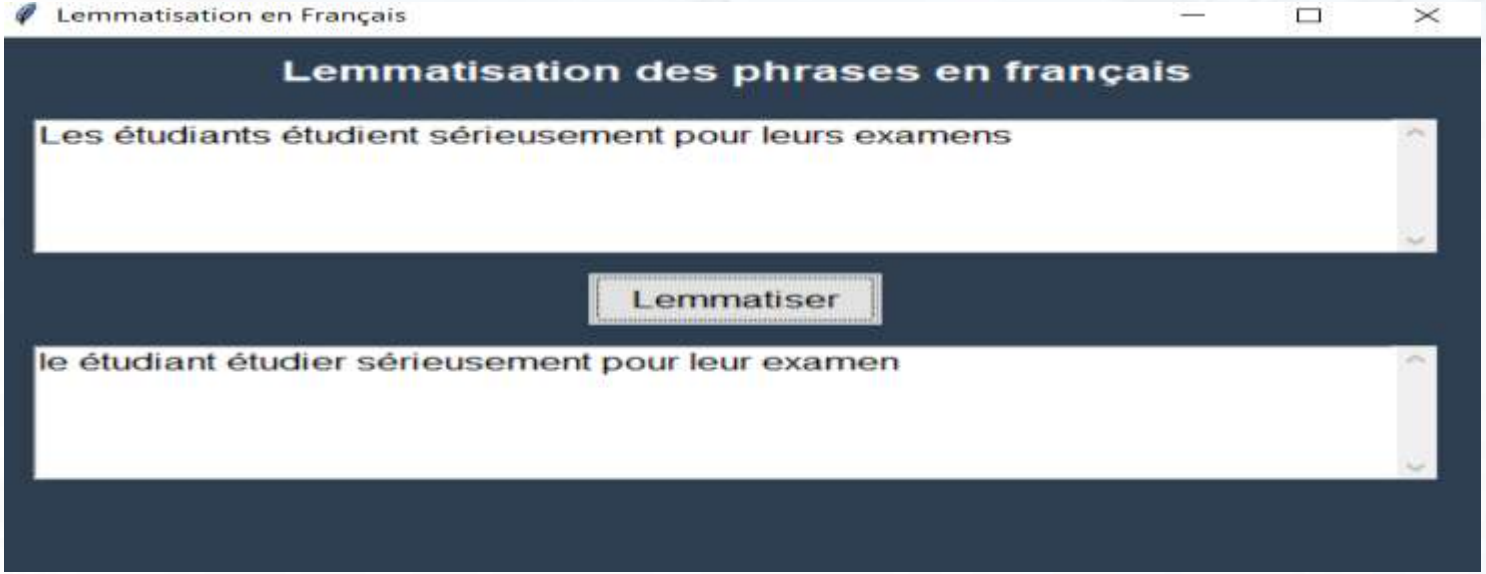
La boucle [stemmer.stem(mot) for mot in mots] applique stemmer.stem(mot) à chaque mot de la liste mots et stocke les résultats dans mots_stemmes.

02

Les techniques de normalisation de texte

6. Lemmatisation

Contrairement au stemming, la lemmatisation utilise une analyse morphologique pour réduire un mot à sa forme canonique ou lexicale (lemme).



02

Les techniques de normalisation de texte

6. Lemmatisation

```
nlp = spacy.load("fr_core_news_sm")

def effectuer_lemmatisation():
    phrase = entree_texte.get("1.0", tk.END).strip()
    doc = nlp(phrase)
    lemmes = " ".join([token.lemma_ for token in doc])
```

nlp(phrase) transforme phrase en objet doc, qui contient les tokens avec leurs propriétés linguistiques: lemmes, types grammaticaux ….

token.lemma_ → Récupère le lemme de chaque mot
[token.lemma_ for token in doc] → Liste des lemmes
" ".join(...) → Convertit cette liste en chaîne de texte en séparant les lemmes par des espaces

Les techniques de normalisation de texte

7. Correction orthographique

Cette étape consiste à corriger les fautes d'orthographe dans le texte brut pour améliorer la qualité des données.

Texte brut : "Je veus aprendre le traitement du lengage."
Texte corrigé : "Je veux apprendre le traitement du langage."

SpellChecker est une bibliothèque Python de correction orthographique qui utilise un algorithme de correction basé sur un dictionnaire pour détecter et corriger les fautes d'orthographe dans un texte

02

Les techniques de normalisation de texte

7. Correction orthographique

```
def correction_orthographique():
    texte = entry_text.get()

    # Initialiser le correcteur orthographique pour le français
    spell = SpellChecker(language='fr')

    mots = texte.split()
    mots_corriges = []

    for mot in mots:
        correction = spell.correction(mot)
        if correction is None:
            mots_corriges.append(mot)
        else:
            mots_corriges.append(correction)
```

Les techniques de normalisation de texte

8. Détection et remplacement des abréviations

Les textes bruts, en particulier issus des réseaux sociaux ou SMS, contiennent souvent des abréviations qu'il convient d'expliciter.

Texte brut : "Je vais bientôt te MP pr avoir + d'infos."
Texte normalisé : "Je vais bientôt te message privé pour avoir plus d'informations."

Les techniques de normalisation de texte

8. Détection et remplacement des abréviations

```
abbreviations = {
    "svp": "s'il vous plaît",
    "rdv": "rendez-vous",
    "msg": "message",
    "bcp": "beaucoup",
    "tt": "tout",
    "mdr": "mort de rire",
    "pcq": "parce que",
    "tkt": "t'inquiète",
    "vrmt": "vraiment"
}

def remplacer_abreviations():
    texte = entree_texte.get("1.0", tk.END).strip()
    mots = texte.split()
    texte_corrige = " ".join([abbreviations.get(mot.lower(), mot) for mot in mots])
```

Les techniques de normalisation de texte

9. Uniformisation des unités et formats

Les textes peuvent contenir des dates, unités ou formats hétérogènes qu’il est nécessaire d’uniformiser.

Date : "12/31/2024" → "31 décembre 2024"

02

Les techniques de normalisation de texte

9. Uniformisation des unités et formats

Les textes peuvent contenir des dates, unités ou formats hétérogènes qu'il est nécessaire d'uniformiser.

```
# Fonction de conversion de la date
def convertir_date():
    try:
        date_str = entry_date.get()
        if not date_str:
            messagebox.showerror("Erreur", "Veuillez entrer une date.")
            return
        # Convertir la date en objet datetime
        date_obj = datetime.strptime(date_str, "%Y/%m/%d")
        # Extraire le jour, mois et année
        jour = date_obj.day
        mois = mois_francais[date_obj.month - 1]
        annee = date_obj.year

        label_resultat.config(text=f"Date : \"{date_str}\" → \"{jour} {mois} {annee}\"")
    except ValueError:
        messagebox.showerror("Erreur", "Le format de la date est invalide. Veuillez entrer au format année/mois/jour.")
```

02

Les techniques de normalisation de texte

"Salut, je t'écris pr te demander si tu as lu l'article du 12/05/23 sur l'IA générative ? Il est TOP ! À bientôt. ☺"

Conversion en minuscule	"salut, je t'écris pr te demander si tu as lu l'article du 12/05/23 sur l'ia générative ? il est top ! à bientôt. ☺"
Suppression des accents	"salut, je tecris pr te demander si tu as lu larticle du 12/05/23 sur lia generative ? il est top ! a bientot. ☺"
Suppression de la ponctuation	"salut je tecris pr te demander si tu as lu larticle du 12 05 23 sur lia generative il est top a bientot"
Détection et remplacement des abréviations	"salut je tecris pour te demander si tu as lu larticle du 12 05 23 sur l'intelligence artificielle il est top a bientot"
Lemmatisation	"écris" → "écrire""demandé" → "demander""générative" → "génératif"
Uniformisation des formats (dates)	"salut je écrire pour te demander si tu avoir lire l'article du 12 mai 2023 sur l'intelligence artificielle il être top à bientôt"
Suppression des mots vides (stop words)	"salut écrire demander lire article 12 mai 2023 intelligence artificielle être top bientôt"
Texte final normalisé	"salut écrire demander lire article 12 mai 2023 intelligence artificielle être top bientôt"

03 →

Nettoyage de données textuelles

03

Nettoyage de données textuelles

Supprimer les éléments indésirables et réduire le bruit dans le texte.

Techniques :

- Suppression de la ponctuation, des chiffres, des symboles spéciaux
- Suppression des espaces blancs inutiles, des doublons ou des éléments non pertinents comme les balises HTML
- Élimination des "stop words" (mots vides) qui n'apportent pas d'informations pertinentes pour l'analyse
- Correction d'orthographe, suppression des caractères spéciaux et autres erreurs typographiques

03

Nettoyage de données textuelles VS la normalisation

Le nettoyage vise à éliminer les éléments non pertinents ou perturbateurs dans le texte

La normalisation vise à rendre le texte plus uniforme, en harmonisant les formes des mots et en standardisant les éléments du texte