Traitement Automatique du Langage Naturel/Vision par Ordinateur

NLP/VISION, ML/DL, Python

Cheikh Brahim EL VAIGH

Institut Supérieur du Numérique (SupNum) - M1

Cheikh-Brahim.Elvaigh@supnum.mr

Traitement Automatique du Langage Naturel

Plan

Objectifs du cours

Objectifs du cours

- Étudier le TALN/Vision par Ordinateur sous différents angles;
- Découvrir un langage de programation → Python;
- Comprendre les base du TALN;
- Comprendre les base de la Vision par Ordinateur;
- Étudier quelques tâches en exemples;
- Manipuler des cas pratiques avec des corpus de données

Organisation du cours

- 18h CM + TD et 18H TPs/Projets
- Des TP notés/Projets
- Évaluation Finale (sur table)

Traitement Automatique du Langage Naturel

Qu'est-ce que le TALN ?

Le Traitement Automatique du Langage Naturel (TALN), ou NLP en anglais, est un domaine de l'intelligence artificielle qui se concentre sur l'interaction entre les ordinateurs et le langage humain[8]. Il vise à :

- Analyser et comprendre le langage naturel
- Générer du texte humainement lisible
- Faciliter la communication homme-machine

I'm making a presentation. Can you help me create an image for NLP introduction ?

Pourquoi le Traitement Automatique du Langage Naturel (TALN) ?

À quoi sert le langage ?

- Nous communiquons en utilisant le langage
- Nous pensons (en partie) avec le langage
- Nous racontons des histoires en langage
- Nous construisons des théories scientifiques avec le langage
- Nous faisons des amis/construisons des relations

Pourquoi le TALN ?

- Accéder à la connaissance (moteur de recherche, système de recommandation, etc.)
- Communiquer (par exemple, la traduction)
- Linguistique et sciences cognitives (analyser les langues elles-mêmes)

Quantité de données textuelles en ligne. . .

- 70 milliards de pages web en ligne (1,9 milliard de sites internet)
- 55 millions d'articles sur Wikipédia

Internetlivestats

Pourquoi le Traitement Automatique du Langage Naturel (TALN) ?

. . . Croissant à un rythme rapide

- 9000 tweets/seconde
- 3 millions d'e-mails/seconde (dont 60 % de spam)

Internetlivestats

Statistiques des utilisateurs

- 7,9 milliards de personnes utilisent une forme de langage (janvier 2022)
- 4,7 milliards d'utilisateurs d'internet (janvier 2021) (59 %)
- 4,2 milliards d'utilisateurs des réseaux sociaux (janvier 2021) (54 %)

Datareportal

Quels Produits ?

- Recherche: +2 milliards d'utilisateurs de Google, 700 millions d'utilisateurs de Baidu
- Réseaux Sociaux: +3 milliards d'utilisateurs des réseaux sociaux
- Assistant Vocal: +100 millions d'utilisateurs (Alexa, Siri, Google Assistant)
- Traduction Automatique: 500 millions d'utilisateurs de Google Traduction

Datareportal

Les 5 Défis du Traitement Automatique du Langage Naturel (TALN)

1. Productivité
   - ≪ Construire et comprendre un nombre indéfiniment grand d'énoncés, y compris des énoncés jamais rencontrés auparavant. ≫
2. Ambiguïté
   - Résoudre les multiples sens et interprétations possibles des mots et phrases.
3. Variabilité
   - Gérer les différentes façons dont les concepts peuvent être exprimés en langage naturel.
4. Diversité
   - S'adapter à la diversité des langues, des dialectes et des styles d'écriture.
5. Sparsité
   - Gérer les cas où certaines données linguistiques sont rares ou absentes dans les corpus.

Productivité linguistique: Préfixation

Définition

La productivité linguistique est la capacité des locuteurs à créer et comprendre un nombre illimité de nouvelles expressions.

French: Prefix "re-"

- faire (to do) → refaire (to redo)
- lire (to read) → relire (to reread)
- commencer (to begin) → recommencer (to begin again)

English: Prefix "un-"

- do → undo
- tie → untie
- lock → unlock

Nouvelle création

Un enfant pourrait créer un nouveau mot comme : Jouer → Jouable même s'il ne l'a jamais entendu auparavant.

Ambiguïté linguistique

Définition

L'ambiguïté linguistique se produit lorsqu'un mot, une phrase ou une expression peut être interprétée de plusieurs façons.

Exemple : Ambiguïté syntaxique

J'ai vu un homme avec un télescope.

Interprétations possibles

1. J'ai utilisé un télescope pour voir un homme.
2. J'ai vu un homme qui avait un télescope.

Ambiguïté Sémantique

Polysémie

- La tête de la Nouvelle-Zélande est une femme.

Entité Nommée

- Exemple : ≪ Michael Jordan ≫
- Michael Jordan est professeur à Berkeley.

Objet/Couleur

- Exemple : ≪ cerise ≫
- Ton manteau cerise.

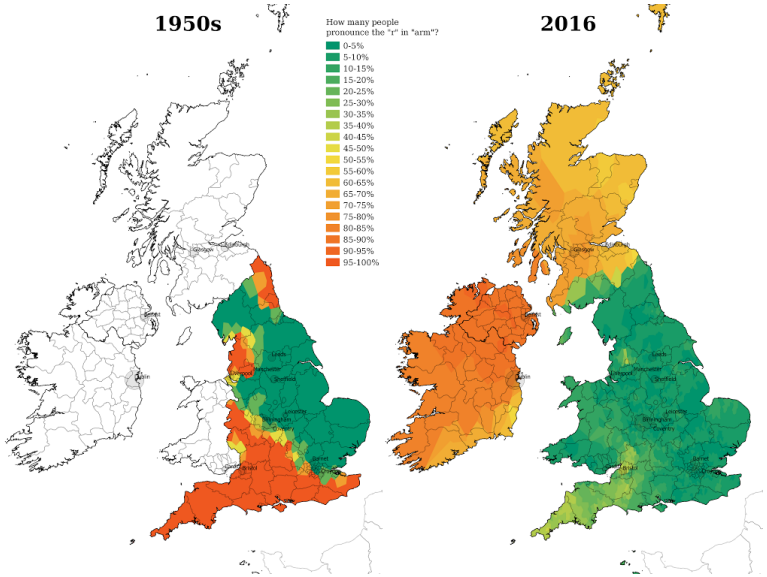
Variabilité linguistique

Définition

Fait référence aux différences systématiques dans la façon dont une langue est utilisée par différents groupes ou dans différents contextes.

Exemple : Variabilité dialectale

Do you pronounce the 'r' in 'arm' ?:



Impact

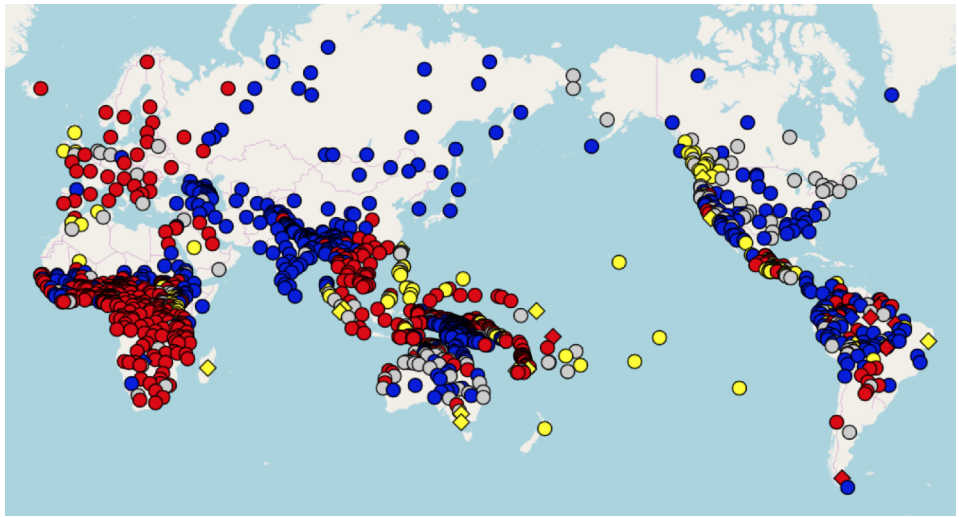
Peut affecter la prononciation, le vocabulaire, la grammaire et l'usage de la langue.

Diversité linguistique

Définition

La diversité linguistique se réfère à la variété des langues parlées dans une région ou un contexte donné.

Exemple : Diversité Syntaxique [5]



Importance

La diversité linguistique est cruciale pour la préservation des cultures et des connaissances uniques à chaque communauté linguistique.

Sparsité linguistique

Définition

En linguistique computationnelle, la sparsité fait référence au phénomène où de nombreuses combinaisons possibles de mots ou de caractéristiques linguistiques sont rares ou absentes dans les données disponibles.

Exemple : Sparsité lexicale

Dans un corpus de 1 million de mots :

- Mots fréquents (ex: "le", "être") : des milliers d'occurrences
- Mots rares : une seule occurrence
- Nombreuses combinaisons possibles : jamais observées

Défi

La sparsité pose des défis importants pour les modèles statistiques et d'apprentissage automatique en traitement du langage naturel.

Applications du TALN

Le TALN trouve de nombreuses applications dans notre vie quotidienne[12] :

- Traduction automatique
- Assistants vocaux
- Analyse de sentiments
- Résumé automatique de textes
- Systèmes de question-réponse

Techniques fondamentales

Les techniques de base du TALN incluent[1] :

- Tokenisation
- Représentations vectorielles de mots
- Analyse morphologique
- Étiquetage grammatical
- Analyse syntaxique
- Désambiguïsation sémantique

Défis actuels

Malgré les progrès significatifs, le TALN fait face à plusieurs défis[6] :

- Compréhension du contexte et des nuances
- Gestion de l'ambiguïté du langage
- Traitement des langues peu dotées
- Interprétation des expressions idiomatiques

Les débuts du TALN

Années 1950-1960

- Alan Turing propose le "test de Turing"[16]
- L'expérience Georgetown-IBM, une des premières tentatives de traduction automatique[7]
- Développement des premières théories linguistiques formelles

L'ère des systèmes basés sur des règles

Années 1960-1990

- Développement de SHRDLU par Terry Winograd[18] → LISP
- Accent mis sur les systèmes basés sur des règles
- Création du Hidden Markov Model Toolkit (HTK) pour la reconnaissance vocale

L'essor des approches statistiques

Années 1990-2010

- Introduction des modèles probabilistes
- Théorie de l'Apprentissage Statistique (SVM, Forêt Aléatoire)
- Modèles Graphiques Probabilistes (par exemple, LDA, HMM)
- Intérêt croissant pour l'apprentissage automatique en TALN
- Développement de corpus annotés à grande échelle

L'ère de l'apprentissage profond

Années 2010 à aujourd'hui

- 2013 : Introduction de Word2Vec[13]
- 2014 : Google Neural Machine Translation
- 2017 : Modèle Transformer et BERT[17]
- 2020 : GPT, un des plus grands modèles de langage à ce jour[3]

Plan

Les Bases du TALN

Objectifs principaux

- Comprendre le langage naturel
- Générer du langage naturel
- Traduire automatiquement entre les langues

Composants clés

- Analyse morphologique
- Analyse syntaxique
- Analyse sémantique
- Analyse pragmatique

Exemple d'application du TALN

Analyse d'une phrase

Considérons la phrase : "L'esprit est fort mais la chair est faible."

Étapes de traitement:

1. Tokenisation : Séparation en mots

```
# Using python and NLTK
raw = "L'esprit est fort mais la chair est faible."
tokens = nltk.word_tokenize(text)
print(tokens)
['L', '\'', 'esprit', 'est', 'fort', 'mais', 'la', 'chair', 'est', '
    faible', '.']
```

2. Étiquetage morphosyntaxique : Attribution des catégories grammaticales

Mot	L'	esprit	est	fort	mais	la	chair	est	faible
Étiquette	DET	NC	V	ADJ	CC	DET	NC	V	ADJ
Description	Déterminant	Nom commun	Verbe	Adjectif	Conjonction de coordination	Déterminant	Nom commun	Verbe	Adjectif

3. Analyse syntaxique : Identification des relations entre les mots
4. Analyse sémantique : Compréhension du sens de la phrase

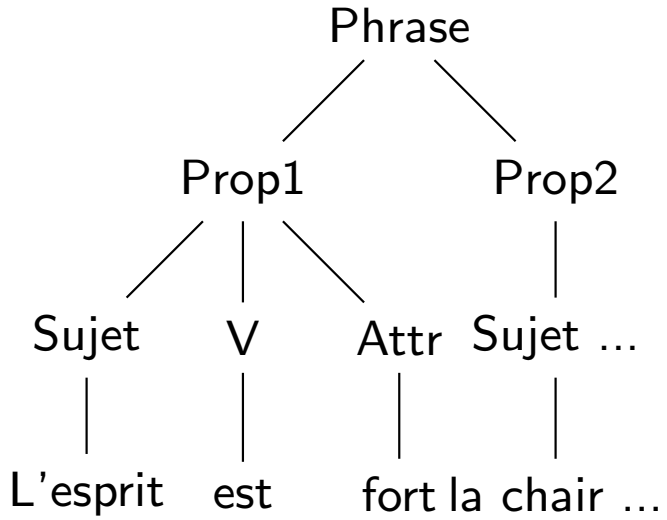
Analyse syntaxique

Phrase d'exemple

L'esprit est fort mais la chair est faible.

Structure syntaxique

- Proposition principale 1 : L'esprit est fort
  - Sujet : L'esprit
  - Verbe : est → relation
  - Attribut du sujet : fort
- Conjonction de coordination : mais
- Proposition principale 2 : la chair est faible
  - Sujet : la chair
  - Verbe : est → relation
  - Attribut du sujet : faible



Observations

- Structure parallèle des deux propositions
- Utilisation de "mais" pour contraster les idées
- Verbe "être" utilisé comme copule dans les deux propositions → relation

Analyse sémantique : Reconnaissance d'entités nommées (NER)

Phrase d'exemple

"L'esprit est fort mais la chair est faible."

Reconnaissance d'entités nommées (NER)

La NER vise à identifier et classifier les entités nommées dans un texte. Dans notre exemple :

- Esprit : Concept abstrait
- Chair : Concept physique

Analyse sémantique approfondie

- 'Esprit' et 'chair' sont identifiés comme des concepts opposés
- 'Fort' et 'faible' sont reconnus comme des attributs contrastés
- La structure "X est Y mais Z est W" indique une opposition

Pré-traitement : Exemple

Étapes du pré-traitement

```
# Phrase d'origine
raw = L"\'"esprit est fort mais la chair est faible.

# 1. Tokenisation --> raw.split()
tokens = [L"\'", esprit, est, fort, mais, la, chair, est, faible, .]


# 2. Normalisation --> ntokens = [t.lower() for t in tokens]
n_tokens = [l"\'", esprit, est, fort, mais, la, chair, est, faible, .]

# 3. Suppression des mots vides --> s_tokens = [t for t in n_tokens if s not
    in stopwords]
s_tokens = [esprit, fort, chair, faible]

# 4. Lemmatisation
l_tokens = [esprit, fort, chair, faible]
```

Tokenisation

Définition

La tokenisation est un processus qui consiste à diviser un texte en unités plus petites appelées tokens. Ces tokens peuvent être des mots, des phrases, des symboles, des *n-gramme* ou d'autres éléments significatifs, selon l'application.

Objectifs

- Faciliter l'analyse et le traitement du texte
- Préparer les données pour d'autres tâches de TALN
- Réduire la complexité du traitement linguistique

Algorithme de Tokenisation : Basé sur les Espaces

Divise le texte en utilisant les espaces comme séparateurs.

Exemple

```
Entrée : "Le chat mange une souris."
Sortie : ["Le", "chat", "mange", "une", "souris."]
```

Avantages et Inconvénients:

- Simple et rapide
- Ne gère pas bien la ponctuation et les cas spéciaux

Algorithme de Tokenisation : Expressions Régulières

Principe

Utilise des expressions régulières pour identifier les tokens.

Exemple

Une ou plisuers lettres ou de la ponctuation

```python
import re
# Regex : \w+|[^\w\s]
# Entree : "L'oiseau vole, rapidement!"
# Sortie : ["L", "'", "oiseau", "vole", ",", "rapidement", "!"]
def tokenize(text):
    return re.findall(r'\w+|[^\w\s]', text)

sample_text = "L'oiseau vole, rapidement!"
tokens = tokenize(sample_text)
```

Avantages et Inconvénients:

- Plus flexible que la méthode basée sur les espaces
- Peut gérer la ponctuation et les cas spéciaux (la contraction)
- Traitement plus complexe

Algorithme de Tokenisation : Basé sur les Règles

Principe

Appliquer un ensemble de règles linguistiques prédéfinies.

Exemple de Règles

- Séparer les contractions (n'est $\rightarrow$ n' + est)
- Traiter les abréviations comme des unités
- Gérer les nombres et les dates

Avantages et Inconvénients:

- Précis pour des langues spécifiques
- Peut gérer des cas complexes
- Traitement plus complexe
- Necessite les connaissances d'experts

Normalisation de Textes en TALN

Définition

La transformation d'un texte en une forme standard ou canonique pour faciliter son traitement ultérieur dans les applications de TALN.

Objectifs

- Réduire la variabilité du texte
- Améliorer la cohérence des données
- Faciliter l'analyse et la comparaison des textes

Les étapes courantes

- Conversion en minuscules
- Suppression des accents
- Élimination de la ponctuation et des espaces superflus
- Standardisation des formats (dates, nombres, etc.)
- Correction orthographique
- Expansion des abréviations

Suppression des Mots Vides (Stop Words)

Définition

Les mots vides sont des mots très communs qui n'apportent généralement pas de valeur sémantique significative à l'analyse du texte.

Exemples de mots vides en français

- Articles : le, la, les, un, une, des
- Prépositions : à, de, pour, par, en, dans
- Pronoms : je, tu, il, elle, nous, vous, ils, elles
- Conjonctions : et, ou, mais, donc, car
- Verbes auxiliaires : être, avoir (formes conjuguées)

Objectifs

- Réduire la dimensionnalité des données
- Améliorer l'efficacité des algorithmes de TALN
- Concentrer l'analyse sur les mots porteurs de sens

Suppression des Mots Vides: Méthodes

Méthodes

- Utilisation de listes prédéfinies de mots vides
- Sélection basée sur la fréquence des mots dans le corpus
- Approches adaptatives selon le domaine ou la tâche

Considérations

- Le choix des mots à supprimer peut varier selon la langue et le contexte
- Certaines applications peuvent nécessiter de conserver certains mots vides
- Risque de perte d'information dans certains cas (ex: analyse de sentiment)

Étiquetage Morphosyntaxique (POS Tagging)

Définition

L'étiquetage morphosyntaxique consiste à attribuer à chaque mot d'un texte une étiquette indiquant sa catégorie grammaticale et ses caractéristiques morphologiques dans son contexte[11].

Importance

- Étape fondamentale dans le traitement automatique du langage naturel
- Utilisé dans la correction grammaticale, l'analyse syntaxique, et la traduction automatique
- Aide à la désambiguïsation des mots selon leur contexte

Méthodes Basées sur les Règles: Brill 1992

Principe

Utilisation de règles linguistiques définies manuellement par des experts pour déterminer la catégorie grammaticale des mots[2].

Étiqueteur de Brill

- Attribue d'abord l'étiquette la plus fréquente à chaque mot
- Applique ensuite des règles de transformation pour corriger les erreurs
- Exemple de règles :
  - Si un mot se termine par "-ment" et est précédé d'un ADJ, changer l'étiquette en ADV
  - Si un mot étiqueté NOM est précédé de DET, ne pas changer l'étiquette

Étapes principales

1. Initialisation : Attribuer l'étiquette la plus fréquente à chaque mot
2. Application de règles : Modifier les étiquettes selon des règles contextuelles
3. Itération : Répéter l'application des règles jusqu'à convergence

Exemple d'application de l'Étiqueteur de Brill

Étape 1 : Initialisation

L'	abeille	butine	joyeusement	des	fleurs	colorées	dans	le	jardin	ensoleillé
DET	VER	VER	ADV	DET	NOM	ADJ	PRP	DET	NOM	ADJ

Étape 2 : Application de règles

- Si un mot précédé d'un déterminant est étiqueté comme VERB, transformer en NOM.
- Si un mot se termine par "-ment" et est précédé d'un ADJ, changer l'étiquette en ADV
- Si un mot précédé d'un adverbe est étiqueté comme NOM, transformer en ADJ.

L'	abeille	butine	joyeusement	des	fleurs	colorées	dans	le	jardin	ensoleillé
DET	NOM	VER	ADV	DET	NOM	ADJ	PRP	DET	NOM	ADJ

Méthodes Statistiques : HMM

Principe des Modèles de Markov Cachés (HMM)

Utilise les probabilités de transition entre étiquettes et les probabilités d'émission des mots pour chaque étiquette[10].

Fonctionnement

- Calcule la séquence d'étiquettes la plus probable pour une séquence de mots donnée
- Utilise l'algorithme de Viterbi pour trouver le chemin optimal

Avantages et Inconvénients

- (+) Performant avec un corpus d'entraînement suffisant
- (+) Rapide à l'exécution
- (-) Nécessite un grand corpus annoté pour l'entraînement
- (-) Difficultés avec les mots inconnus

TreeTagger : Un Étiqueteur Basé sur les Arbres de Décision

Principe de TreeTagger

TreeTagger est un outil d'étiquetage morphosyntaxique et de lemmatisation développé par Helmut Schmid à l'Institut de Linguistique Computationnelle de l'Université de Stuttgart. TreeTagger Utilise des arbres de décision pour estimer les probabilités de transition entre les étiquettes[15].

Fonctionnement

- Construit un arbre de décision à partir d'un corpus d'entraînement
- Utilise le contexte (mots précédents et suivants) pour prendre des décisions
- Combine l'arbre de décision avec un modèle trigram pour l'étiquetage final

Caractéristiques principales

- Multilingue : applicable à diverses langues
- Paramétrable : peut être entraîné sur des corpus spécifiques
- Bonne gestion des mots inconnus grâce à l'utilisation des suffixes
- Efficace : traitement rapide des textes
- Précis : taux d'exactitude élevé pour l'étiquetage (précision $>$ 96% pour l'anglais)

Fonctionnement de TreeTagger

Processus d'étiquetage

1. Tokenisation : découpage du texte en unités lexicales
2. Consultation du lexique : recherche des étiquettes possibles
3. Application de l'arbre de décision : désambiguïsation
4. Attribution de l'étiquette et du lemme

Arbre de décision

TreeTagger utilise un arbre de décision pour choisir l'étiquette la plus probable en fonction du contexte. Chaque nœud de l'arbre représente un test sur les caractéristiques du mot et de son environnement, tandis que les feuilles contiennent les probabilités des différentes étiquettes.

Étapes d'étiquetage avec TreeTagger

Phrase d'exemple

L'abeille butine joyeusement des fleurs colorées dans le jardin ensoleillé.

Étapes détaillées

1. Tokenisation
2. Consultation du lexique
3. Application de l'arbre de décision
4. Attribution des étiquettes et lemmes

Étapes 1 et 2

Étape 1: Tokenisation

[L'] [abeille] [butine] [joyeusement] [des] [fleurs] [colorées] [dans] [le] [jardin] [ensoleillé] [.]

Étape 2 : Consultation du lexique

Pour chaque token, TreeTagger consulte son lexique interne pour déterminer les étiquettes morphosyntaxiques possibles[1].

Exemple pour quelques mots

- L' : DET:ART
- abeille : NOM
- butine : VER:pres
- joyeusement : ADV

Cette étape permet de réduire l'ambiguïté en limitant les choix possibles pour chaque mot.

Étape 3 : Application de l'arbre de décision

Principe de l'arbre de décision

TreeTagger utilise un arbre de décision pour désambiguïser les mots ayant plusieurs étiquettes possibles. L'arbre prend en compte le contexte du mot (mots précédents et suivants) pour choisir l'étiquette la plus probable.

Exemple simplifié d'arbre de décision pour "des"

- Si le mot précédent est un verbe :
  - Si le mot suivant est un nom : PRP:det
  - Sinon : PRP
- Si le deux mots précédents sont des adjectifs : DET:ART
- Sinon : PRP:det

Étape 4 : Attribution des étiquettes et lemmes

Résultat final de l'étiquetage

- L' : DET:ART (le)
- abeille : NOM (abeille)
- butine : VER:pres (butiner)
- joyeusement : ADV (joyeusement)
- des : PRP:det (de)
- fleurs : NOM (fleur)
- colorées : ADJ (coloré)
- dans : PRP (dans)
- le : DET:ART (le)
- jardin : NOM (jardin)
- ensoleillé : ADJ (ensoleillé)
- . : SENT (.)

Chaque token reçoit une étiquette morphosyntaxique et un lemme.

Arbre de décision pour l'étiquetage

Structure générale de l'arbre

L'arbre de décision de TreeTagger est construit lors de la phase d'apprentissage sur un corpus annoté[5]. Chaque nœud de l'arbre représente un test sur les caractéristiques du mot et de son contexte.

Exemple de structure d'arbre (simplifié)

- Racine : Mot actuel
  - Branche 1 : Mot précédent
    - Sous-branche 1.1 : Suffixe du mot actuel
    - Sous-branche 1.2 : Mot suivant
  - Branche 2 : Deux mots précédents
    - Sous-branche 2.1 : Préfixe du mot actuel
    - Sous-branche 2.2 : Catégorie du mot précédent

Méthodes Récentes : Réseaux de Neurones

Principe

Utilisation de réseaux de neurones profonds pour apprendre automatiquement les caractéristiques pertinentes pour l'étiquetage[4].

Types de Réseaux Utilisés

- Réseaux récurrents (RNN, LSTM, BiLSTM)
- Transformers (BERT, GPT)

Avantages

- Capacité à capturer des dépendances à long terme
- Très performants sur de grands corpus
- Possibilité d'apprentissage multi-tâches

Inconvénients

- Nécessitent de grandes quantités de données annotées
- Coûteux en ressources de calcul pour l'entraînement

Lemmatisation : Introduction

Définition

La lemmatisation est le processus de réduction des mots à leur forme canonique, appelée lemme.

Objectif

- Réduire la complexité lexicale $\rightarrow$ POS Tagging
- Améliorer les performances des systèmes de TALN
- Faciliter l'analyse sémantique

Lemmatisation basée sur les dictionnaires

Principe

Utilisation de dictionnaires de lemmes précompilés[15].

Étapes

1. Recherche du mot dans le dictionnaire
2. Récupération du lemme associé
3. Gestion des mots hors vocabulaire

Avantages

- Rapidité d'exécution
- Précision pour les mots connus

Lemmatisation basée sur les règles

Principe

Utilisation de règles linguistiques prédéfinies pour transformer les mots en leurs lemmes[9].

Étapes

1. Identification de la catégorie grammaticale
2. Application de règles de transformation spécifiques à la langue
3. Gestion des exceptions

Avantages

- Précision pour les langues bien documentées
- Contrôle fin sur le processus de lemmatisation

Approche statistique

Principe

Utilisation de modèles probabilistes pour déterminer le lemme le plus probable.

Méthodes

- Modèles de Markov cachés
- Apprentissage automatique supervisé

Avantages

- Adaptabilité à différentes langues
- Capacité à gérer des cas ambigus

Approche hybride

Principe

Combinaison d'approches basées sur les règles et statistiques[14].

Composants

- Règles linguistiques pour les cas courants
- Modèles statistiques pour les cas ambigus ou rares

Avantages

- Meilleure précision globale
- Flexibilité accrue

Exemple de Traitement de Texte en TALN avec Python

Code python

```python
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Phrase originale
texte = "L abeille butine joyeusement des fleurs colorees dans le jardin ensoleille."
texte ="The sun is shining today."
# Tokenisation
tokens = word_tokenize(texte)

tagged = nltk.pos_tag(tokens)
print("tagged: ", tagged)
# Normalisation
tokens_norm = [token.lower() for token in tokens if token.isalnum()]

# Suppression des mots vides
stop_words = set(stopwords.words('english'))
tokens_sans_stop = [mot for mot in tokens_norm if mot not in stop_words]

# Lemmatisation
lemmatizer = WordNetLemmatizer()
lemmes = [lemmatizer.lemmatize(mot) for mot in tokens_sans_stop]

print("Tokens:", tokens)
print("Normalises:", tokens_norm)
print("Sans mots vides:", tokens_sans_stop)
```

Résultats du Traitement

Tokens

['L'', 'abeille', 'butine', 'joyeusement', 'des', 'fleurs', 'colorées', 'dans', 'le', 'jardin', 'ensoleillé', '.']

Tokens Normalisés

['l', 'abeille', 'butine', 'joyeusement', 'des', 'fleurs', 'colorées', 'dans', 'le', 'jardin', 'ensoleillé']

Tokens Sans Mots Vides

['abeille', 'butine', 'joyeusement', 'fleurs', 'colorées', 'jardin', 'ensoleillé']

Lemmes

['abeille', 'butiner', 'joyeusement', 'fleur', 'coloré', 'jardin', 'ensoleillé']

Reconnaissance d'entités nommées (NER)

Définition

La reconnaissance d'entités nommées (NER) est une tâche de traitement du langage naturel qui vise à identifier et classifier les entités nommées dans un texte en catégories prédéfinies.

Catégories courantes

- Personnes
- Organisations
- Lieux
- Dates
- Quantités

Importance

La NER est cruciale pour l'extraction d'information, la recherche d'information, et d'autres tâches avancées de traitement du langage naturel.

Approche basée sur les règles

Description

Cette approche utilise des règles linguistiques et des listes prédéfinies pour identifier les entités nommées.

Avantages

- Simple à mettre en œuvre
- Efficace pour des domaines spécifiques
- Ne nécessite pas de données d'entraînement

Inconvénients

- Manque de flexibilité
- Difficile à maintenir pour de grands ensembles de règles
- Performances limitées sur des textes variés

Approche statistique

Description

Utilise des modèles statistiques comme les HMM ou CRF pour apprendre à reconnaître les entités à partir de données annotées.

Avantages

- Plus flexible que l'approche basée sur les règles
- Peut capturer des motifs complexes
- Bonnes performances sur des données variées

Inconvénients

- Nécessite des données d'entraînement annotées
- Peut être sensible au bruit dans les données

Approche par apprentissage profond

Description

Utilise des réseaux de neurones profonds, comme les LSTM ou les Transformers, pour apprendre à reconnaître les entités.

Avantages

- Performances état de l'art
- Peut capturer des dépendances à long terme
- Adaptable à différents domaines

Inconvénients

- Nécessite de grandes quantités de données annotées
- Coût computationnel élevé
- Peut être difficile à interpréter

Format de balisage IOB

Définition

IOB (Inside-Outside-Beginning) est un format de balisage utilisé pour annoter les entités nommées dans un texte.

Étiquettes

- B-[TYPE] : Début d'une entité
- I-[TYPE] : À l'intérieur d'une entité
- O : En dehors de toute entité

Exemple

Mot	**Étiquette**
Jean	B-PER
Dupont	I-PER
travaille	O
à	O
Paris	B-LOC

Étapes du NER avec IOB tagging

Phrase d'exemple

Jean Dupont travaille a Paris pour Apple Inc.

Étape 1 : Tokenisation

[Jean] [Dupont] [travaille] [a] [Paris] [pour] [Apple] [Inc] [.]

Étape 2 : Identification des entités

- Jean Dupont : Personne (PER)
- Paris : Lieu (LOC)
- Apple Inc : Organisation (ORG)

Cette étape consiste à repérer les entités nommées dans la phrase.

Étape 3 : Application des tags IOB

Tags IOB

- B-PER : Début d'une entité Personne
- I-PER : Suite d'une entité Personne
- B-LOC : Début d'une entité Lieu
- I-LOC : Suite d'une entité Lieu
- B-ORG : Début d'une entité Organisation
- I-ORG : Suite d'une entité Organisation
- O : Outside (hors entité)

Étape 4 : Résultat final

Résultat du NER avec IOB tagging

- Jean B-PER
- Dupont I-PER
- travaille O
- a O
- Paris B-LOC
- pour O
- Apple B-ORG
- Inc I-ORG
- . O

Exemple avec spaCy

Code Python utilisant spaCy pour la NER

```python
import spacy

# Charger le modele francais
nlp = spacy.load("fr_core_news_sm")

# Texte d exemple
texte = "Jean Dupont travaille a Paris pour Apple Inc."

# Traitement du texte
doc = nlp(texte)

# Affichage des entites nommees
for entite in doc.ents:
    print(f"{entite.text} - {entite.label_}")
```

Résultat

Le résultat affichera les entités nommées identifiées dans le texte, avec leur type et leur position.

Plan

Bird, S., Klein, E., and Loper, E.
Natural language processing with Python: analyzing text with the natural language toolkit.
O'Reilly Media, Inc., 2009.

Brill, E.
A simple rule-based part of speech tagger.
Proceedings of the third conference on Applied natural language processing (1992), 152–155.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.
Language models are few-shot learners.
Advances in neural information processing systems 33 (2020), 1877–1901.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P.
Natural language processing (almost) from scratch.
Journal of machine learning research 12, Aug (2011), 2493–2537.

Dryer, M. S.
Order of subject, object and verb (v2020.3).
In *The World Atlas of Language Structures Online*, M. S. Dryer and M. Haspelmath, Eds. Zenodo, 2013.

Goldberg, Y.
Neural network methods for natural language processing.
Morgan & Claypool Publishers, 2017.

Hutchins, J.
The georgetown-ibm experiment demonstrated in january 1954.
Machine translation: from real users to research (2004), 102–114.

Jurafsky, D., and Martin, J. H.
Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.
Prentice Hall, 2000.

Korenius, T., Laurikkala, J., Järvelin, K., and Juhola, M.
Stemming and lemmatization in the clustering of finnish text documents.
In *Proceedings of the thirteenth ACM international conference on Information and knowledge management* (2004), pp. 625–633.

Kupiec, J.
Robust part-of-speech tagging using a hidden markov model.
Computer speech & language 6, 3 (1992), 225–242.

Manning, C. D.
Part-of-speech tagging from 97% to 100%: is it time for some linguistics?
Computational linguistics and intelligent text processing (2011), 171–189.

Manning, C. D., and Schütze, H.
Foundations of statistical natural language processing.
MIT press, 1999.

Mikolov, T., Chen, K., Corrado, G., and Dean, J.
Efficient estimation of word representations in vector space.
arXiv preprint arXiv:1301.3781 (2013).

Pal, S., Pakray, P., and Naskar, S. K.
A hybrid approach to word segmentation of thai texts using corpus statistics and heuristic rules.
In *International Conference on Intelligent Text Processing and Computational Linguistics* (2013), Springer, pp. 467–478.

Schmid, H.
Probabilistic part-of-speech tagging using decision trees.
New methods in language processing (2013), 154.

Turing, A. M.
Computing machinery and intelligence.
Mind 59, 236 (1950), 433–460.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I.
Attention is all you need.

In *Advances in neural information processing systems* (2017), pp. 5998–6008.

Winograd, T.
Understanding natural language.
Cognitive psychology 3, 1 (1972), 1–191.