Chapitre 1 — Prétraitements

1 Segmentation en mots et phrases

1.1 Étapes préalables du TAL

— Segmentation en phrases (sentence splitting) : analyse du rôle des ponctuations
— Segmentation en mots/tokens (tokenization) : analyse des ponctuations également nécessaire
— Normalisation des mots (plus ou moins avancée)

1.2 Difficultés de segmentation

— Phrases : format initial des textes, ponctuations, majuscules
— Points d'interrogation et d'exclamation fiables comme fins de phrase, contrairement aux points
— Solution : analyse des ponctuations préalable à la segmentation en phrases

1.3 Tokens versus types

— Token = occurrence d'un mot | Type = forme d'un mot (dictionnaire)
— Lemme = forme de base (ex : pars, partir, partîmes PARTIR)
— Le nombre de formes (types) augmente avec le nombre de tokens (approximation : $|V| > N^{1/2}$)

1.4 Tokenization et normalisation

— Difficultés : ponctuations, abréviations, contractions, élisions, mots composés
— Convention Penn Treebank : séparer apostrophes (doesn't does n't) et ponctuations
— Normalisation : capitalisation (lowercase, préservation, truecasing)
— Byte-Pair Encoding (BPE) : solution hybride pour traiter les mots rares

2 Niveaux d'analyse en TAL

2.1 Analyse lexicale

— Segmentation en mots, analyse morphosyntaxique (POS tagging), identification des entités nommées

2.2 Analyse syntaxique

— Regroupement des syntagmes (chunking), identification des fonctions grammaticales, création d'arbres syntaxiques

2.3 Analyse sémantique

— Désambiguïsation sémantique, probabilités de co-occurrence, rôles sémantiques, forme logique

2.4 Analyse pragmatique

— Thèmes, sentiments, pronoms, actes de langage, structures argumentatives

3 Alphabets et encodage informatique

3.1 Systèmes d'écriture

— Alphabétiques : dizaines de signes, phonétiques (latin, grec, arabe, hébreu)
— Syllabiques : centaine de signes (japonais hiragana/-katakana, inuit)
— Idéographiques : dizaines de milliers de signes (chinois/japonais)

3.2 Encodages

— Jeu de caractères : correspondance entre caractères et nombres (points de code)
— Encodage : représentation machine des points de code (suite de bits)
— ASCII : 128 points de code (alphabet anglais standard) → 7 bits
— ISO Latin/8859 : 256 points de code, plusieurs variantes (ISO 8859-1, 8859-2, etc.) → 8 bits

3.2.1 Unicode

Standard international pour tout alphabet, idéogramme, symbole
— Conçu pour ne plus limiter le nombre de caractères (plus d'un million de points de code)
— Version 15.1 : 149'813 caractères dans 161 scripts
— UTF-8 : variable (1-4 octets), char ASCII = 1 octet

3.3 Problèmes pratiques

— Compatibilité imparfaite entre ISO 8859 et UTF-8
— Déclaration d'encodage explicite en XML et HTML
— Byte Order Mark (BOM) : caractère FEFF au début des fichiers texte
— Attention en TAL à l'encodage correct des données (utiliser UTF-8)

Chapitre 2 — Part-of-Speech tagging

1 Définition et jeux d'étiquettes

— Détermine pour chaque mot sa catégorie grammaticale (nom, adjectif, verbe, etc.)
— Utilise un jeu d'étiquettes (tagset) fixé à l'avance, plus ou moins détaillé selon les projets
— Exemples de tagsets importants :
  — Penn Treebank : 36 étiquettes + ponctuation, standard en anglais
  — Universal Dependencies : 17 catégories universelles avec attributs additionnels
  — French Treebank : 13 catégories avec sous-catégorisation et traits morphologiques
— Difficultés liées aux ambiguïtés : beaucoup de mots peuvent appartenir à plusieurs catégories selon le contexte

2 Applications du POS tagging

— Étape préliminaire pour l'analyse syntaxique
— Détection des groupes nominaux (utiles comme mots-clés)
— Extraction d'information et systèmes de question-réponse
— Traits (attributs) utilisés pour l'apprentissage automatique

3 Approche probabiliste Markovienne

— Détermine la séquence de tags qui maximise $P(t_{1\ldots n}|m_{1\ldots n})$ où t = tags et m = mots
— Application du théorème de Bayes :

$$P(t_{1\ldots n}|m_{1\ldots n}) = \frac{P(m_{1\ldots n}|t_{1\ldots n}) \cdot P(t_{1\ldots n})}{P(m_{1\ldots n})}$$

— Hypothèses simplificatrices :
  — Indépendance des mots entre eux
  — Probabilité d'un mot indép. des tags voisins
  — Probabilité d'un tag **dépend seulement du tag précédent** (chaîne de Markov)
— Modèle : $\prod_{k=1}^{n} P(m_k|t_k)P(t_k|t_{k-1})$
— Apprentissage des probabilités à partir de corpus annotés :
  — $P(m^x|t^y) = \dfrac{\text{fois où } m^x \text{ possède le tag } t^y}{\text{apparitions du tag } t^y}$
  — $P(t^y|t^x) = \dfrac{\text{fois où } t^y \textbf{ suit } \text{le tag } t^x}{\text{apparitions du tag } t^x}$

Calcul de la probabilité conjointe de deux mots et tags

Pour exprimer la probabilité qu'une séquence de deux mots m_1 et m_2 soit étiquetée respectivement avec t_1 et t_2, notée $P(t_1, t_2|m_1, m_2)$, on procède ainsi :

$$\begin{aligned} P(t_1, t_2|m_1, m_2) &\propto P(m_1, m_2|t_1, t_2) \times P(t_1, t_2) \\ &= P(m_1|t_1) \times P(m_2|t_2) \times P(t_1, t_2) \quad \text{(ind. mots)} \\ &= P(m_1|t_1) \times P(m_2|t_2) \times P(t_2|t_1) \times P(t_1) \\ &= P(m_1|t_1) \times P(m_2|t_2) \times P(t_2|t_1) \times P(t_1|\hat{}) \end{aligned}$$

Où $P(t_1|\hat{})$ représente la probabilité que le tag t_1 apparaisse en début de phrase. Cette formulation permet de calculer la probabilité conjointe en combinant :
— Les probabilités d'émission $P(m_i|t_i)$ pour chaque mot
— La probabilité de transition entre tags $P(t_2|t_1)$
— La probabilité initiale du premier tag $P(t_1|\hat{})$

3.1 Raffinements de l'approche Bayes-Markov

3.1.1 Gestion des mots inconnus

— Créer un nouveau type m_{inc} et affecter à chaque tag t_k une probabilité $P(m_{inc}|t_k)$
— Utiliser des caractéristiques morphologiques (suffixes, préfixes, majuscules)
— Adapter les probabilités selon les catégories (plus élevées pour noms, nulles pour pronoms)

3.1.2 Utilisation de n-grammes de tags

— Étendre aux trigrammes : $P(t_k|t_{k-1}, t_{k-2})$ au lieu des bigrammes $P(t_k|t_{k-1})$
— Interpolation pour gérer les n-grammes jamais vus dans l'entraînement

3.1.3 Maximum Entropy Markov Model

— Estimer directement $P(t_k|m_k, t_{k-1})$ sans passer par le théorème de Bayes
— Permet d'intégrer facilement des traits supplémentaires :
  — Dépendances à différentes distances (1, 2, ..., n tokens)
  — Propriétés lexicales : préfixes, suffixes, présence de tirets, majuscules, chiffres
  — Contexte des mots environnants

3.1.4 Lissage des probabilités

— Interpolation linéaire :

$$P_{lisse}(t_i|t_{i-1}) = \lambda_1 P(t_i|t_{i-1}) + \lambda_2 P(t_i)$$

— Lissage de Good-Turing ou technique de Kneser-Ney pour éviter les probabilités nulles

4 Lemmatisation et racinisation

— **Lemmatisation** : déterminer la forme canonique (lemme) d'un mot
  — Ex : "Les ventilateurs sont mieux disposés" "[le] [ventilateur] [être] [bien] [disposer]"
  — Utilise souvent un dictionnaire comme LEFFF pour le français
— **Stemming** : extraire la racine approximative d'un mot
  — Plus rapide et plus simple que la lemmatisation
  — Empirique, à base de règles de désuffixation
  — Algorithme de Porter très répandu pour l'anglais
  — But : réduire la diversité des mots pour permettre des généralisations

Chapitre 3 — Parsing

1 Analyse syntaxique des langages de programmation

— Les erreurs de syntaxe (*syntax error*) surviennent quand le code ne respecte pas la structure définie
— Les instructions acceptables sont définies par : mots-clés, forme des noms de variables, combinaison d'opérations
— Le formalisme BNF (Backus-Naur Form) permet de spécifier formellement la syntaxe
— Yacc/Lex ou GNU Bison/Flex : générateurs d'analyseurs syntaxiques (compilateurs de compilateurs)

2 Grammaires formelles et hiérarchie

Une grammaire formelle contient :
— V_t : symboles terminaux (vocabulaire)
— N : symboles non-terminaux (catégories grammaticales)
— S : symbole de départ (proposition)
— R : règles transformant non-terminaux

Hiérarchie de Chomsky (1956) : types 0, 1, 2, 3 (du plus général au plus restrictif)
— Type 0 : grammaires générales, sans restriction
— Type 1 : grammaires dépendantes du contexte
— Type 2 : grammaires indépendantes du contexte (CFG, ex : BNF)
— Type 3 : grammaires régulières

Les langues naturelles sont généralement décrites par des grammaires de type 2

3 Grammaires formelles pour langues naturelles

— Grammaires hors-contexte (CFG) : règles de forme $N \to \alpha$ où N est non-terminal
— Types de symboles : terminaux (mots), préterminaux (POS tags), catégories (constituants)
— L'arbre syntaxique représente les règles de dérivation et l'analyse en constituants
— Différence entre phrase bien formée (syntaxiquement correcte) et phrase ayant du sens
— Difficultés : accord, compléments obligatoires vs. optionnels
— Alternative : grammaires de dépendances (relations binaires entre mots)

4 Analyse syntaxique avec grammaires formelles

— Parsing : trouver la série de règles dérivant une phrase depuis S
— Types d'analyseurs :
  — Par direction : top-down vs. bottom-up
  — Par technique : profondeur vs. largeur
— Analyse descendante récursive : décomposer les objectifs en sous-objectifs
— Autres algorithmes : shift-reduce, chart parsing (CKY, Earley)

5 Machine learning et grammaires probabilistes

— Problèmes des parsers formels : explosion combinatoire, priorités des règles
— Solution : grammaires probabilistes (PCFG)
  — Annoter manuellement corpus (Penn Treebank)
  — Extraire règles syntaxiques et affecter probabilités selon fréquence
  — Guider l'analyse en fonction des probabilités
— Calcul de probabilité d'un arbre : produit des probabilités de toutes les règles utilisées

Chapitre 4 — Named entities

1 Définition et représentation

1.1 Entités nommées

Les entités nommées (EN) sont principalement des noms propres désignant des entités uniques, classées en types :
— PERSONNE : Marcel Proust
— LIEU : Yverdon-les-Bains
— ORGANISATION : HEIG-VD

S'y ajoutent d'autres termes relativement fixes :
— TEMPS : dates, heures, noms de fêtes
— NOMBRE : montants, pourcentages
— PRODUIT/MARQUE : substances chimiques, etc.

1.2 Reconnaissance d'entités nommées (NER)

La NER comporte deux composantes :
1. Délimiter les groupes de mots constituant des EN
2. Étiqueter chaque groupe avec son type

1.3 Représentation : système d'étiquettes

— IOB (Inside-Outside-Beginning) : indique frontières et types
  — B : début d'une EN
  — I : continuation d'une EN
  — O : pas une EN
— IO : simplifié, remplace B par I (moins d'étiquettes mais perd la distinction entre entités consécutives du même type)

2 Principales méthodes

2.1 Attributs utilisés pour la NER

— Mots à étiqueter et mots voisins
— Présence dans des listes prédéfinies (gazetteers)
— Plongements (embeddings) du mot et voisins
— POS tags et étiquettes syntaxiques
— Forme (shape) : majuscules, préfixes, suffixes, tirets
— Étiquettes IOB précédentes

2.2 Méthodes de reconnaissance

— Expressions régulières (solution élémentaire)
— Classifieurs : considèrent les tokens indépendamment

2.2.1 Modèles de séquence

— Hidden Markov Model (HMM) : modèle génératif
— Conditional Random Fields (CRF) : modèle discriminatif permettant plus d'attributs
— Transition-Based Parser : modèle à états finis
— Réseaux de neurones :
  — Utilisation pour attributs (Word2vec, GloVe, LSTM, Transformer)
  — Encodeur BERT + couche de classification

2.3 Approche bayésienne-markovienne pour la NER

— Application du modèle HMM (Hidden Markov Model) à la NER similaire au POS tagging :
  — Variables observables : les mots du texte
  — États cachés : tags IOB avec types d'entités
  — Transitions : probabilités entre états (tags)
— Inférence sur nouvelles données :
  — Objectif : trouver la série de tags $t_1, ..., t_n$ qui maximise $\prod_{k=1}^{n} P(m_k|t_k)P(t_k|t_{k-1})$
  — Algorithme de Viterbi pour déterminer la séquence optimale de tags

3 Évaluation

3.1 Métriques

— Précision (p) : $\dfrac{\text{tokens correctement identifiés (TP)}}{\text{tokens proposés (TP + FP)}}$
— Rappel (r) : $\dfrac{\text{tokens correctement identifiés (TP)}}{\text{tokens de référence (TP + FN)}}$
— F1-score : moyenne harmonique de p et r : $\dfrac{2pr}{p+r}$
— Score total :
  — Micro-average : pondérée fréquence des tags
  — Macro-average : même poids à chaque type

4 Prolongements

4.1 Named Entity Linking

Association des EN reconnues à des identifiants uniques (pages Wikipédia) pour résoudre l'ambiguïté :

— George Bush plusieurs personnes possibles
— Champagne région viticole ou communes

4.2 Extraction de mots-clés (KPE)

Mots ou expressions caractéristiques d'un texte, basés sur :

— Fréquence des mots (TF-IDF) ou n-grammes
— Statistiques de co-occurrence (PPMI)
— Syntaxe/sémantique (groupes nominaux, patrons)
— Terminologie du domaine

Algorithme RAKE :

1. Identifier termes candidats délimités par stopwords
2. Construire le graphe de co-occurrence des mots
3. Calculer le score de chaque mot (degré/fréquence)
4. Score des candidats = somme des scores de leurs mots

Chapitre 5 — Représentation vectorielle

1 Modélisation du sens en TAL

Applications : recherche d'information, désambiguïsation, classification de textes, calcul de similarité

1.1 Approches sémantiques

— *Sens statistique* (approche distributionnelle) : modélisation par vecteurs et contextes
— *Sens logique* (approche formelle) : dictionnaires, relations (WordNet), désambiguïsation

1.2 Principe de sémantique distributionnelle

— Harris (1954) : « Si A et B ont des environnements presque identiques, ils sont synonymes »
— Firth (1957) : « You shall know a word by the company it keeps »
— Deux mots sont semblables s'ils apparaissent dans des contextes semblables

2 Représentation vectorielle

2.1 Vecteurs et mesures de similarité

— Similarité du cosinus :

$$\text{sim}_{\cos}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} {v_i}^2}\sqrt{\sum_{i=1}^{N} {w_i}^2}}$$

— Valeurs : +1 (colinéaires), 0 (orthogonaux), -1 (opposés)

2.2 Représentation one-hot

vecteur de dimension $|V|$ avec un seul 1, reste = 0
— Limite : tous les mots sont différents (similarité cosinus = 0), pas de sémantique

3 Vecteurs de cooccurrences

3.1 Matrice termes-documents

— Chaque mot = vecteur ligne (occurrences dans chaque document)
— Chaque document = vecteur colonne (occurrences de chaque mot)
— Similarité entre mots/documents = similarité cosinus entre leurs vecteurs

3.2 Coefficient tf-idf

$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$
— $\text{tf}_{t,d}$ = fréquence du terme t dans le document d
— $\text{idf}_t = \log(|D|/\text{df}_t)$ avec df_t = nombre de documents contenant t
— Diminue l'importance des mots très fréquents dans tous les documents

4 Réduction de dimensionnalité avec SVD

4.1 Latent Semantic Analysis (LSA)

décomposition en valeurs singulières (SVD)
— Décomposition de la matrice $M_{|V|\times|D|}$ en $M = U\Sigma V^T$
— SVD tronquée : conserver les k plus grandes valeurs singulières ($50 \leq k \leq 1000$)
— Plongement (embedding) = lignes de U_t (matrice U tronquée)
— Avantages : dimension réduite, vecteurs denses, meilleure généralisation

5 Le modèle GloVe

5.1 Principe

— Exploiter les rapports de fréquences de cooccurrences
— Objectif : produit scalaire $w_i \cdot w_j$ proportionnel à $\log(X_{ij})$
— X_{ij} = nombre de cooccurrences des mots i et j

5.2 Méthode

— Construction de la matrice de cooccurrences
— Initialisation aléatoire des vecteurs de dimension réduite
— Optimisation par régression (AdaGrad) d'une fonction de coût J
— $J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$

5.3 Implémentation

— Dimensions : 50-300
— Corpus : Wikipedia, Gigaword, Common Crawl
— Fenêtre de contexte : 10 mots, pondérée par 1/distance

6 Évaluation des représentations vectorielles

6.1 Méthodes d'évaluation

— *Extrinsèque* : performance sur des tâches (classification, réponses aux questions)
— *Intrinsèque* : corrélation avec jugements humains de similarité, tests d'analogie

6.2 Tests d'analogie

— Trouver d tel que w_d est le plus proche de $w_b - w_a + w_c$
— Exemple : « Si le code postal d'Anaheim est 92804, quel est celui d'Honolulu ? »

7 Synthèse des types de vecteurs de mots

— **Représentations creuses (sparse)** :
  — Vecteurs one-hot
  — Matrices de cooccurrences avec tf-idf
— **Représentations denses (embeddings)** :
  — LSA/SVD tronquée
  — GloVe
  — word2vec (Skip-gram, CBOW)
  — Modèles contextuels (BERT)