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)

Chapitre 6 — Word embeddings / word2vec

1 Introduction à word2vec

word2vec est une méthode pour apprendre des représentations vectorielles de mots utilisant un réseau de neurones (perceptron multicouche). L'objectif est de créer un espace de dimension beaucoup plus faible que la taille du vocabulaire.

1.1 Deux modèles principaux

— **CBOW (Continuous Bag-of-Words)** : apprendre à prédire un mot étant donnés des mots du contexte
— **Skip-gram** : apprendre à prédire des mots voisins d'un mot donné

Une fois entraînés, l'intérêt porte sur la représentation des mots dans la couche cachée (plongement) plutôt que sur les prédictions.

2 Architecture et entraînement

2.1 Réseaux de neurones formels

Un neurone formel calcule la somme pondérée de ses entrées ($\sum_i w_i \cdot x_i$), éventuellement passée par une fonction d'activation (f), pour produire son activation de sortie (y).

2.2 Modèle Skip-gram

Objectif : trouver les mots voisins les plus probables pour un mot donné.

2.2.1 Architecture

— Couches d'entrée (I) et sortie (O) : taille du vocabulaire $|V|$ ($10^4 < N < 10^6$, N mots plus courants)
— Couche cachée (H) : dimension $d = 300$ (exemple)
— Connexions I vers H : matrice $\mathbf{W}$ de taille $|V| \times d$ (word embeddings)
— Connexions H vers O : matrice $\mathbf{C}$ de taille $d \times |V|$ (context embeddings)

2.2.2 Fonctionnement

— Entrée : vecteur 1-hot avec $x_j = 1$
— Activation de sortie k : $c_k \cdot w_j = \sum_i c_{k,i} w_{j,i}$
— Probabilité softmax : $P(w_{t+1} = w_k | w_t = w_j) = \dfrac{\exp(activation_k)}{\sum_p \exp(activation_p)}$

2.3 Apprentissage des matrices W et C

Objectif : maximiser la probabilité observée dans le corpus d'entraînement

$$\arg\max_{W,C} \prod_{(word,ctxt)\in T} P(ctxt|word)$$

2.3.1 Procédure d'entraînement

1. Initialisation aléatoire de W et C
2. Pour chaque exemple (w_t, w_{t+1}) :
   — Observer la différence entre sortie effective et désirée
   — Modifier les poids via gradients
   — Utiliser l'échantillonnage négatif (SGNS : Skip-gram with Negative Sampling)
3. Itérer jusqu'à convergence

3 Résultats et applications

3.1 Propriétés des embeddings

Les matrices W et C fournissent des représentations vectorielles où :
— À chaque mot w_i correspond un vecteur dans W et un dans C
— Généralement, on utilise les embeddings de W
— Possibilité d'additionner ou concaténer les vecteurs de W et C

3.1.1 Représentation de la couche cachée

Une fois word2vec entraîné, la couche cachée représente l'espace des **plongements de mots** (embeddings). Chaque unité de cette couche correspond à une dimension sémantique abstraite. L'ordre de grandeur typique est de **100 à 1000 unités** (souvent 300), soit une réduction drastique par rapport à la taille du vocabulaire (10^4 à 10^6 mots).

3.1.2 Prédiction du mot voisin le plus probable

Avec les matrices $W = (w_{ij})$ et $C = (c_{jk})$, le mot voisin k le plus probable d'un mot d'entrée i est trouvé par : $k^* = \arg\max_k P(w_k|w_i) = \arg\max_k \vec{c_k} \cdot \vec{w_i} = \arg\max_k \sum_j c_{jk} w_{ij}$

3.1.3 Comparaison activation / similarité du cosinus

Les méthodes sont similaires, car le produit scalaire est lié au cosinus par la formule $a \cdot b = \|a\| \, \|b\| \cos(a, b)$. Mais, si les vecteurs ne sont pas normalisés, maximiser le produit scalaire n'est pas la même chose que maximiser le cosinus (que ce soit avec softmax ou non).
Autre différence, les embeddings des mots seront choisis dans la même matrice alors que la méthode traditionnelle veut que le plongement du premier mot soit dans W et celui du second dans C. Conceptuellement, prédire un mot voisin n'est pas la même chose que trouver un mot similaire en termes d'embeddings.

3.2 Visualisation

Projection PCA en 2D montrant les relations pays-capitales avec des vecteurs parallèles reliant chaque pays à sa capitale.

3.3 Recherche pour une analogie

Si cuivre = Cu, zinc = v(Cu) − v(Cuivre) + v(Zinc).

4 Conclusion

4.1 Limitations

— Fonctionne bien avec mots fréquents et relations spécifiques
— **Biais culturels** reflétés dans les données
— Mots antonymes qui ont des embeddings proches
— Vocabulaire fixe → si faute dans un mot → « FastText »

4.1.1 FastText

Innovation : gestion des mots hors vocabulaire (OOV)
— Décomposition en n-grammes de caractères (n = 3, ..., 6)
— Vecteur OOV = moyenne des vecteurs de n-grammes connus

4.2 Autres modèles

— **GloVe** (Stanford) : statistiques de co-occurrences
— **ELMo** (AllenNLP) : représentations contextuelles, polysémie
— **BERT** (Google) : architecture Transformer, contextes bidirectionnels

Chapitre 7 — Désambiguïsation lexicale avec WordNet

1 Introduction aux approches sémantiques en TAL

Le traitement automatique du langage naturel utilise deux approches principales pour modéliser le sens des mots :

1.1 Approches distributionnelles (modèles statistiques)

- Principe : "des mots ayant souvent des voisins semblables sont semblables"
- Représentation des mots comme vecteurs dans un espace
- Méthodes de réduction dimensionnelle : LSA, PLSA, LDA, SVD, word2vec

1.2 Approches formelles (modèles logiques)

- Base de données explicite des sens : WordNet
- Méthodes de désambiguïsation lexicale (Word Sense Disambiguation - WSD)

2 Les sens des mots et leurs relations

2.1 Définition des sens lexicaux

Un sens lexical est une représentation discrète d'une gamme d'aspects possibles du sens d'un mot. Il est difficile de donner le nombre exact de sens d'un mot.

2.2 Polysémie et homonymie

2.2.1 Causes de la multiplicité des sens

1. **Homonymes** : mots différents avec étymologies distinctes
   - Français : 'avocat' (latin *advocatus* vs aztèque *ahuacatl*)
   - Anglais : 'bank', 'bat', 'ball'
2. **Homographes** : mots différents, même écriture, prononciation différente
   - Anglais : 'record' (/rkrd/ nom vs /r-kôrd/ verbe)
   - Français : 'couvent' (Les poules du couvent couvent)
3. **Homophones** : mots différents, même prononciation, écriture différente
   - Anglais : 'write'/'right', 'piece'/'peace'
   - Français : ver/verre/vert, bar/barre
4. **Mots polysémiques** : un seul mot avec plusieurs sens reliés
   - Exemple : 'bank' (institution financière vs bâtiment)
   - Relation de métonymie : extension du sens

2.3 Relations sémantiques entre mots

2.3.1 Types de relations

1. **Synonymie** : formes différentes, sens identique
   - Exemples : couch/sofa, big/large, automobile/car
   - Note : vrais synonymes rares (différences de niveau, connotations)
2. **Antonymie** : sens opposés sur un aspect, autres aspects identiques
   - Exemples : short/long, leader/follower
3. **Hyponymie/Hypéronymie** : relation hiérarchique de spécificité
   - Hyponyme : sens plus spécifique (voiture → véhicule)
   - Hypéronyme : sens plus général (véhicule voiture)
   - Définition logique : être X implique être Y
   - Relation transitive : cabriolet < voiture < véhicule < artéfact < objet

2.4 Difficultés pour le TAL

- Recherche d'information : 'bat care' → chauve-souris ou batte ?
- Traduction : 'bat' → 'murciélago' ou 'bate' ?
- Recherche d'images : ambiguïté visuelle

3 WordNet : base de données lexicales

3.1 Présentation générale

- Base de données des sens des mots anglais (années 1990, dernière MAJ 2011)
- Statistiques WordNet 3.1 :
  - Noms : 117'798
  - Verbes : 11'529
  - Adjectifs : 22'479
  - Adverbes : 4'481
- Intégration : NLTK, JWNL/extJWNL (Java)
- Consultation en ligne : `https://en-word.net`

3.2 Structure : les synsets

3.2.1 Définition d'un synset

Un synset (sens élémentaire) comprend :

- Index (lemme représentatif + POS + numéro)
- Définition textuelle
- Exemples d'usage
- Ensemble des mots synonymes ayant ce sens

3.2.2 Exemple : 'gull'

- `chump.n.01` : personne crédule → ['chump', 'fool', 'gull', 'mark', ...]
- `gull.n.02` : oiseau aquatique → ['gull', 'seagull']

3.3 Hiérarchie des synsets

- Organisation selon hyponymie/hypéronymie
- Exemple de hiérarchie : chump → dupe → actor → causal agency → physical entity → entity
- Permet de calculer la similarité sémantique via distance dans la hiérarchie

3.4 Relations sémantiques dans WordNet

3.4.1 Relations entre noms

- Hypernym/Hyponym : concepts généraux/spécifiques
- Instance Hypernym/Hyponym : instances vers concepts
- Member/Part/Substance Meronym/Holonym : parties/touts
- Antonym : opposition sémantique
- Derivationally Related Form : même racine morphologique

3.4.2 Relations entre verbes

- Hypernym/Troponym : événements généraux/spécifiques
- Entails : relations causales entre événements
- Antonym : opposition sémantique

3.5 Utilisation pour la similarité sémantique

- Calcul basé sur la distance dans la hiérarchie
- Question : prendre synsets les plus fréquents ou les plus proches ?

3.5.1 Amélioration de résultats

1. On peut enrichir la requête avec des synonymes de ses mots, en considérant tous les synsets possibles. Autrement dit, on cherchera des documents contenant des synonymes des termes de la requête. On peut se limiter aux mots fréquents. On peut faire cela si le système retourne peu de résultats.
2. On peut enrichir la requête avec des hypéronymes, ou substituer certains termes par leurs hypéronymes, sil y a peu de résultats.
3. Déclasser les meilleurs documents s'ils contiennent des antonymes de certains mots de la requête.

4 Désambiguïsation sémantique (WSD)

4.1 Position du problème

Données :

1. Occurrence d'un mot dans un contexte
2. Liste des sens possibles (synsets WordNet)

Objectif : Déterminer le sens correct de l'occurrence

Applications : traduction automatique, recherche d'information, synthèse vocale

4.2 Variantes de la tâche

1. **WSD d'un seul mot** : ex. 'interest' (6-7 sens)
   - Solutions : ML supervisé, méthodes non-supervisées
2. **WSD de tous les mots** porteurs de sens
   - Solutions : méthodes non-supervisées, embeddings

4.3 Méthodes de résolution

4.3.1 Méthode baseline : sens le plus fréquent

- WordNet ordonne les synsets par fréquence (corpus SemCor)
- Choisir systématiquement le premier synset
- Performance correcte mais limitée

4.3.2 Algorithme de Lesk simplifié (non-supervisé)

Principe :

1. Extraire définitions et exemples de chaque synset
2. Supprimer stopwords, lemmatiser
3. Compter mots communs entre définitions et contexte
4. Choisir synset avec plus grand overlap

Exemple : 'bank' dans « deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities »

- Synset financier : 2 mots communs (deposits, mortgage)
- Synset géographique : 0 mot commun
- Décision : sens financier

4.3.3 Apprentissage automatique supervisé

Représentation des attributs :

1. **Approche sac-de-mots**
   - Compter occurrences de chaque mot du voc.
   - Contexte fixe (ex. 10 mots avant/après)
   - Nombreux attributs, peu de valeurs
   - On supprime souvent les stopwords
2. **Approche positionnelle**
   - Noter le mot à chaque position relative
   - Inclure étiquettes POS
   - Peu d'attributs, nombreuses valeurs

Exemple positionnel : 'bass' dans « An electric guitar and bass player stand off... »

Attribut	mot_{i-2}	POS_{i-2}	mot_{i-1}	POS_{i-1}
Valeur	electric	JJ	guitar	NN

Attribut	mot_{i+1}	POS_{i+1}	mot_{i+2}	POS_{i+2}
Valeur	player	NN	stand	VB

Algorithmes : Naive Bayes, Regression Logistique, SVM, Réseaux de neurones

Chapitre 8 — Classification de documents

1 Introduction au problème de classification de textes

1.1 Définition de la tâche

La classification de textes consiste à assigner une classe (ou catégorie, étiquette) à un texte donné (document, paragraphe, message, phrase, fragment) parmi une liste de classes prédéfinies.

1.2 Types de classification

- **Classification binaire** (deux classes) : email → spam | non-spam (ham)
- **Classification ternaire** : critique de film → {positive, négative, neutre}
- **Classification multi-classe** : article de presse → {politique, sport, finance, ...}

2 Méthodes supervisées - Vue d'ensemble

La classification supervisée suit un pipeline en deux étapes :

1. **Vectorisation** : extraction d'attributs numériques
2. **Classification** : attribution de la catégorie

2.1 Trois approches principales

2.1.1 Vecteurs en dimension élevée (sparse vectors)

- Vectorisation : coefficients TF ou TF-IDF
- Classification : régression logistique, Naïve Bayes, SVM

2.1.2 Vecteurs de mots en dimension basse (dense vectors)

- Vectorisation : somme de vecteurs word2vec, GloVe, FastText
- Classification : régression logistique, Naïve Bayes, SVM, réseaux de neurones

2.1.3 Vecteurs de textes en dimension basse (dense vectors)

- Vectorisation : encodeurs Transformers (ex. BERT) pré-entraînés
- Classification : couche de classification à entraîner

3 Modèles bayésiens naïfs

3.1 Principe général

- **Naïfs** : présupposent l'indépendance des attributs lexicaux
- **Bayésiens** : s'appuient sur le théorème de Bayes

$$P(\text{classe}|\text{attributs}) = \frac{P(\text{attributs}|\text{classe}) \times P(\text{classe})}{P(\text{attributs})}$$

Classification par maximum a posteriori :

$$\arg \max_{\text{classe} \in C} P(\text{attributs}|\text{classe}) \times P(\text{classe})$$

3.2 Deux modèles de représentation

3.2.1 Modèle de Bernoulli

- Document = $(e_1, e_2, ..., e_v)$ où $e_i \in \{0, 1\}$
- Modélise la présence/absence des mots
- $P((e_1, ..., e_v)|\text{classe}) = \prod_{1 \leq i \leq v} P(\text{mot}_i|\text{classe})$

3.2.2 Modèle multinomial

- Document = $(f_1, f_2, ..., f_v)$ où $f_i \in \mathbb{N}$
- Modélise le nombre d'occurrences des mots
- $P((f_1, ..., f_v)|\text{classe}) = \prod_{1 \leq i \leq v} P(\text{mot}_i|\text{classe})^{f_i}$
- Meilleur pour les documents longs

3.3 Lissage de Laplace

Pour éviter les probabilités nulles

$$P(\text{mot}_i = \text{PRÉS}|C) = \frac{\text{nb textes classe C contenant mot}_i + 1}{\text{nb textes classe C} + \text{nb. val. mot}}$$

4 Modèles vectoriels en grande dimension

4.1 Formulation générale

1. **Données d'entraînement** : textes + classes correctes
2. **Transformation** : textes → vecteurs (TF ou TF-IDF)
3. **Classifieur** : régression logistique, SVM
4. **Évaluation** : précision, rappel, F1 par classe

4.2 Régression logistique

- Probabilité : $\frac{1}{1 + \exp(-\sum_{1 \leq i \leq n} w_i x_i + b)}$
- Entraînement : optimisation des poids $(w_1, ..., w_n, b)$
- Fonction de coût : entropie croisée (convexe)
- Algorithme : descente de gradient stochastique (SGD)

4.3 Différence avec les modèles bayésiens

- **Bayésiens naïfs** : génératifs (modélisent la génération du texte)
- **Régression logistique** : discriminative (identifie les traits pertinents)

5 Classification avec word2vec

5.1 Pipeline

1. Pré-traitement des textes
2. Embeddings des mots (word2vec, GloVe, fastText)
3. Représentation du texte : moyenne des vecteurs des mots
4. Classification (ex. régression logistique)

5.2 Limitations

- Approche sac-de-mots
- Aucune désambiguïsation contextuelle

6 Classification avec BERT

6.1 Principe

Utilisation d'un réseau encodeur basé sur les Transformers pour :

- Faire évoluer les embeddings non-contextualisés des mots
- Générer un embedding contextualisé pour chaque token
- Représenter le texte via l'embedding du token [CLS]

6.2 Architecture Transformer

- Réseau de neurones prenant des embeddings en entrée
- Produit des embeddings dépendant de tous les mots du contexte
- Empilement de plusieurs blocs Transformer (12, 24, 48, 100)

6.3 Pré-entraînement BERT

6.3.1 Masked Language Modeling (MLM)

- Prédire des tokens masqués avec [MASK]
- Couche linéaire + softmax → distribution sur le vocabulaire
- Fonction de coût : $-\log(P_{\text{modèle}}(\text{token}_{\text{correct}}))$

6.3.2 Next Sentence Prediction (NSP)

- Prédire si deux phrases sont consécutives
- 50% de paires consécutives (VRAI), 50% aléatoires (FAUX)
- Tokens spéciaux : [CLS] et [SEP]

6.4 Fine-tuning pour la classification

1. Ajout d'une couche de classification (head)
2. Dimensions : $\dim_{\text{embedding}} \times \dim_{\text{nb\_classes}}$
3. Utilise l'embedding du token [CLS]
4. Entraînement supervisé avec données annotées

6.5 Modèles disponibles

- Hugging Face : 71,330 modèles BERT
- Variants : RoBERTa, SpanBERT, CamemBERT (français), mBERT (multilingue)
- Modèles distillés : 50% moins de paramètres, performances similaires
- Repository sentence-BERT pour embeddings de textes

7 Données et évaluation

7.1 Corpus Reuters-21578

- Dépêches d'agence avec étiquettes de topics (118 catégories)
- Split standard : 9,603 entraînement, 3,299 test
- Utilisation : classification multi-classe ou binaire par étiquette
- Souvent limité aux 10 étiquettes les plus fréquentes

7.2 Mesures d'évaluation

7.2.1 Taux de correction (Accuracy)

Pourcentage de documents correctement classés.
Limitation : biaisé vers les classes fréquentes.

7.2.2 Moyennes

- **Micro-moyenne** : pondérée par la taille des classes (favorise les grandes classes)
- **Macro-moyenne** : même poids pour toutes les classes

8 Perspectives

La classification de textes est une tâche très générale permettant de :

- Classifier tout type de texte (email, post, review, etc.)
- Utiliser diverses représentations (surtout embeddings contextualisés)
- Traiter des classes variées (topic, sentiment, toxicité, etc.)

Les embeddings contextualisés basés sur les Transformers représentent l'état de l'art actuel pour cette tâche.

Chapitre 9 — Analyse des sentiments

1 Types de sentiments exprimés par le langage

1.1 Contenu objectif vs. subjectif

La distinction fondamentale s'établit entre :

- **Contenu objectif** : sens littéral figurant dans un dictionnaire (ex : "Le film dure 3h45")
- **Contenu subjectif** : opinion, résonance affective ou émotionnelle (ex : "La fin du film m'a beaucoup déçu")

La détection automatique du contenu subjectif pose des défis particuliers lorsque l'état affectif est exprimé implicitement.

1.2 Typologie des états affectifs (Klaus Scherer)

Les états affectifs se classifient selon leur durée temporelle :

1.2.1 Court terme

- **Émotions** : états brefs liés à des événements (joie, colère, surprise, dégoût, tristesse, peur)
- **Humeurs (mood)** : états diffus à plus long terme (irritabilité, sérénité)

1.2.2 Moyen terme

- **États interpersonnels** : attitudes durant une interaction (amical, distant, chaleureux, méprisant)

1.2.3 Long terme

- **Attitudes** : croyances ou dispositions envers des objets ou personnes (aimer, adorer, détester)
- **Personnalités** : traits stables, comportements typiques (anxieux, hostile, jaloux)

1.3 Définition formelle d'une opinion (Bing Liu, 2011)

Une opinion ou croyance subjective est un quintuplet (e, a, s, p, t) où :

- e = entité à propos de laquelle l'opinion est exprimée
- a = aspect de l'entité visé par l'opinion
- s = sentiment exprimé (positif ou négatif)
- p = personne qui exprime l'opinion
- t = moment où l'opinion a été formulée

1.4 Applications et utilité

L'analyse des sentiments permet de :

- Déterminer automatiquement l'attitude du public envers des produits, services, attractions, lieux, idées ou politiques
- Analyser les tendances d'opinion et leur diffusion
- Analyser les réponses du public à des campagnes publicitaires
- Créer des résumés d'opinions pour chaque aspect d'un item
- Détecter les critiques insincères (payées)

1.5 Données annotées disponibles

Plusieurs corpus de référence existent :

- **IMDB Movie Reviews** : 25'000 critiques polarisées pour l'entraînement et le test
- **Amazon Product Reviews** : 233.1 millions de critiques (34 Go)
- **Stanford Sentiment Treebank (SST-2)** : 11'855 phrases étiquetées, avec performances des meilleurs systèmes atteignant 95%-97%

2 Systèmes non-supervisés à base de lexiques

2.1 Principe général

Cette approche utilise une liste de mots (lexique) avec des informations de polarité p_{mot} de trois types :

- **Binaire** : mot POSITIF | mot NÉGATIF
- **Catégoriel** : TRÈS POSITIF, ASSEZ POSITIF, NEUTRE, ASSEZ NÉGATIF, TRÈS NÉGATIF (+2, +1, 0, -1, -2)
- **Numérique** : échelle de +10 à -10

2.2 Algorithme non-supervisé

Pour p_{mot} binaire : si un texte contient plus de mots positifs que négatifs, son sentiment est positif.

Pour p_{mot} numérique :

$$T_{pos} = \sum_{\{mot \in Texte | p_{mot} > 0\}} p_{mot}$$

$$T_{neg} = \sum_{\{mot \in Texte | p_{mot} < 0\}} p_{mot}$$

Si $T_{pos} > T_{neg}$, le sentiment est positif. Des variantes incluent des seuils : $T_{pos} - T_{neg} > \lambda$ ou $T_{pos}/T_{neg} > \lambda$.

2.3 Construction de lexiques

2.3.1 Lexiques existants

- **General Inquirer** : 1'915 mots positifs / 2'291 négatifs avec informations d'intensité
- **LIWC** : 2'300 mots en >70 classes émotionnelles (lexique payant)
- **MPQA Subjectivity Lexicon** : 2'718 mots positifs et 4'912 négatifs + intensité
- **Bing Liu's Opinion Lexicon** : 2'006 mots positifs et 4'783 négatifs
- **SentiWordNet** : tous les synsets de WordNet annotés automatiquement
- **VADER** : 7'500 mots incluant langage informel des médias sociaux (scores -4 à +4)
- **AFINN** : 2'477 mots avec scores dérivés de tweets annotés par des juges humains

2.3.2 Création automatique de lexiques

Méthode en 5 étapes utilisant les embeddings :

1. Utiliser des mots générateurs (positifs et négatifs)
2. Considérer leurs embeddings (word2vec, GloVe, FastText)
3. Grouper les embeddings : $S^+ = \{E(w_1^+), E(w_2^+), ..., E(w_n^+)\}$ et $S^- = \{E(w_1^-), E(w_2^-), ..., E(w_m^-)\}$
4. Calculer les vecteurs moyens et l'axe polaire : $\mathbf{V}_{axis} = \mathbf{V}^+ - \mathbf{V}^-$
5. Score de polarité : $score(w) = \cos(E(w), \mathbf{V}_{axis}) = \frac{E(w) \cdot \mathbf{V}_{axis}}{||E(w)|| \cdot ||\mathbf{V}_{axis}||}$

3 Systèmes utilisant l'apprentissage supervisé statistique

3.1 Classification de textes

Avec un corpus de textes annotés (phrases, messages, critiques), on peut entraîner des classifieurs :

- **Algorithmes** : Naïve Bayes, Logistic Regression, SVM, Decision Tree, Random Forest
- **Attributs** : "sac de n-grammes" (n=1,2,3) avec fréquences ou coefficients TF-IDF
- **Préprocessing** : filtrage des stopwords, sélection de mots polarisés

3.2 Forces et limitations

3.2.1 Forces

- Performants avec suffisamment de données d'entraînement
- Efficaces même avec des attributs simples (mots et bigrammes)
- La sélection d'attributs permet d'identifier des mots polarisés

3.2.2 Limitations

- Données de test très différentes des données d'entraînement
- Présence de mots modifieurs (négations)
- Complexité structurelle (constructions concessives)
- Sens non littéral (humour, ironie)

4 Systèmes utilisant les réseaux de neurones Transformer

4.1 L'encodeur BERT

BERT (Bidirectional Encoder Representations from Transformers) fonctionne en deux phases :

1. **Pré-entraînement auto-supervisé** : tâches de prédiction de mots masqués et de phrases consécutives
2. **Adaptation supervisée** : ajout d'une couche de classification pour la tâche spécifique

4.2 Architecture pour la classification

1. Tokens du texte + token spécial [CLS]
2. Embeddings non-contextualisés
3. Transformations par mécanisme d'attention
4. Embeddings finaux contextualisés
5. Classification basée sur l'embedding du token [CLS]

4.3 Adaptation pour l'analyse de sentiments

- Ajout d'une couche finale : $dim_{embedding} \times dim_{nombre\_de\_polarits}$
- Fine-tuning de l'ensemble du réseau avec données annotées
- Résultats sur SST-2 : 95% de précision

5 Généralisations et extensions

5.1 Tâches connexes

5.1.1 En amont

- **Détection de subjectivité** : distinguer contenu objectif vs. subjectif
- **Détection de tromperie** : identifier les fausses critiques

5.1.2 États transitoires

- **Détection d'émotions** : six émotions de base (Ekman) ou roue des émotions (Plutchik)
- **Applications** : analyse de l'état des clients, conducteurs, débats

5.1.3 États à long terme

- **Analyse de personnalité** : Big Five (extraversion, stabilité émotionnelle, amabilité, consciencieusité, ouverture d'esprit)

5.2 Recherches en personnalité

Études significatives corrélant traits linguistiques et personnalité :

- Pennebaker & King (1999) : 2'479 textes d'étudiants (1.9M mots)
- Mehl et al. (2006) : conversations de 100 participants (100k mots)
- Schwartz et al. (2013) : posts Facebook de 75k volontaires (309M mots)