Introduction au Traitement Automatique des Langues

4 – Les niveaux de traitement – Le niveau Lexical

Introduction au traitement automatique des langues

Contenu de la matière :

1) Introduction Générale
2) Les applications du TAL
3) Les niveaux de traitement - Traitements de «bas niveau»
4) **Les niveaux de traitement - Le niveau lexical**
5) Les niveaux de traitement - Le niveau syntaxique
6) Les niveaux de traitement - Le niveau sémantique
7) Les niveaux de traitement - Le niveau pragmatique

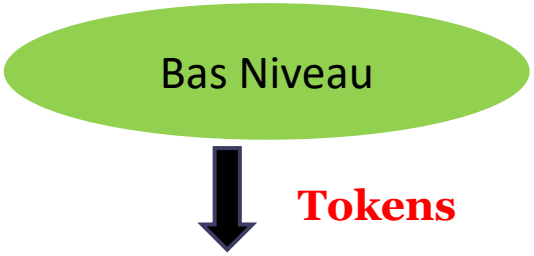
Plan du cours

1. Définitions
2. Nature de mots (POS Tagging) et Trait grammatical
3. Morphologie
4. Morphèmes et leurs typologie
5. Morphologie flexionnelle
6. Morphologie dérivationnelle
7. En Pratique – Normalisation et filtrage du texte
8. Lemmatisation et Stemming - Algorithmes

Définitions

- L'analyse lexicale comprend l'**identification** et l'**analyse** de la **structure** des **tokens** dans les phrases.

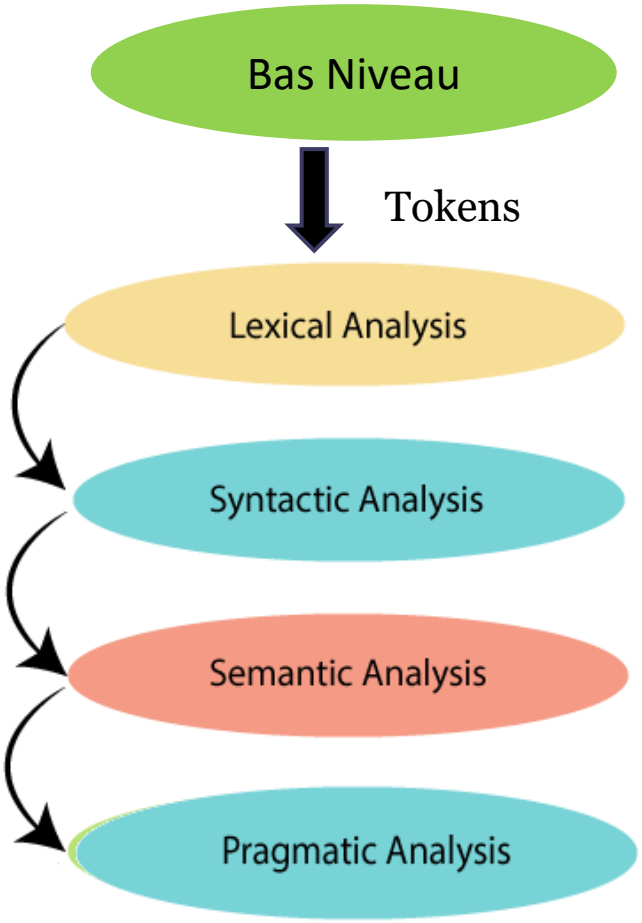
- Le but de cette étape de traitement lexicale est de passer des formes atomiques (**tokens**) identifiées par le segmenteur aux mots, c'est-à-dire de **reconnaître** dans chaque chaîne de caractère une (ou plusieurs) unité(s) linguistique(s), dotée(s) de **caractéristiques** propres.



Définitions

- L'analyse lexicale comprend l'**identification** et l'**analyse** de la **structure** des **tokens** dans les phrases.

- Le but de cette étape de traitement lexicale est de passer des formes atomiques (**tokens**) identifiées par le segmenteur aux mots, c'est-à-dire de **reconnaître** dans chaque chaîne de caractère une (ou plusieurs) unité(s) linguistique(s), dotée(s) de **caractéristiques** propres.



Définitions

- Identifier les composants lexicaux, et leurs propriétés : c'est l'étape de **traitement lexical**

- Exemple énoncé : ***Le président des conseils mangeait une pomme.***

Définitions

- Identifier les composants lexicaux, et leurs propriétés : c'est l'étape de **traitement lexical**

- Exemple énoncé : *Le président des conseils mangeait une pomme.*

  le

  président

  des

  conseils

  mangeait

  pomme

Définitions

- Identifier les composants lexicaux, et leurs propriétés : c'est l'étape de **traitement lexical**

- Exemple énoncé : *Le président des conseils mangeait une pomme.*

  le - **det**. masc. sing.; / **pron**. pers. masc. sing.

  président - **vrb** 3pers. plur. prés. ind.;/ subjonctif ;/ **nom** masc. sing.

  des - **det**. masc./fem. plur. ; / **prep**.

  conseils - **nom**. masc. plur.

  mangeait - **vrb** (1,3) pers. sing. imp. ind., [mang+e+ait].

  pomme - **nom** fem. sing.

Trait grammatical

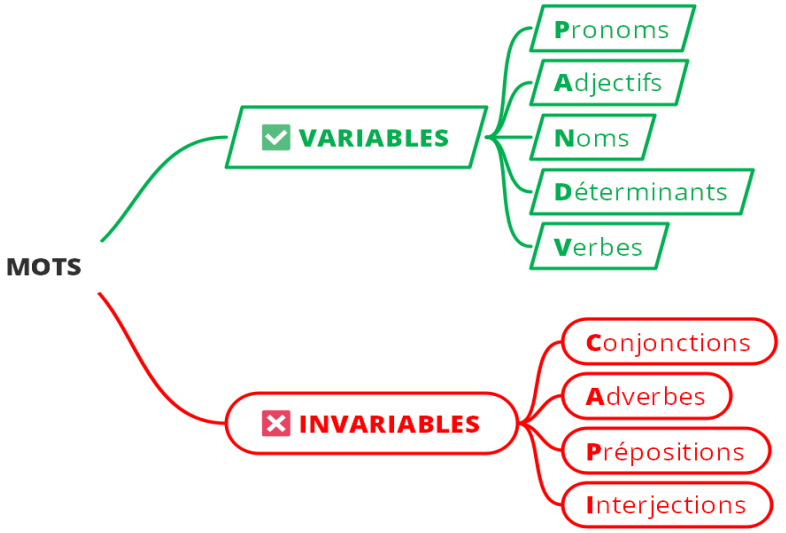
- Le **trait grammatical** est une catégorie d'ordre grammatical et linguistique permettant de décrire les flexions morphologiques des mots variables d'une langue.

- Les traits ne sont pas tous représentés dans toutes les langues, et chaque trait se subdivise en catégories de nombre variable dans les langues concernées.

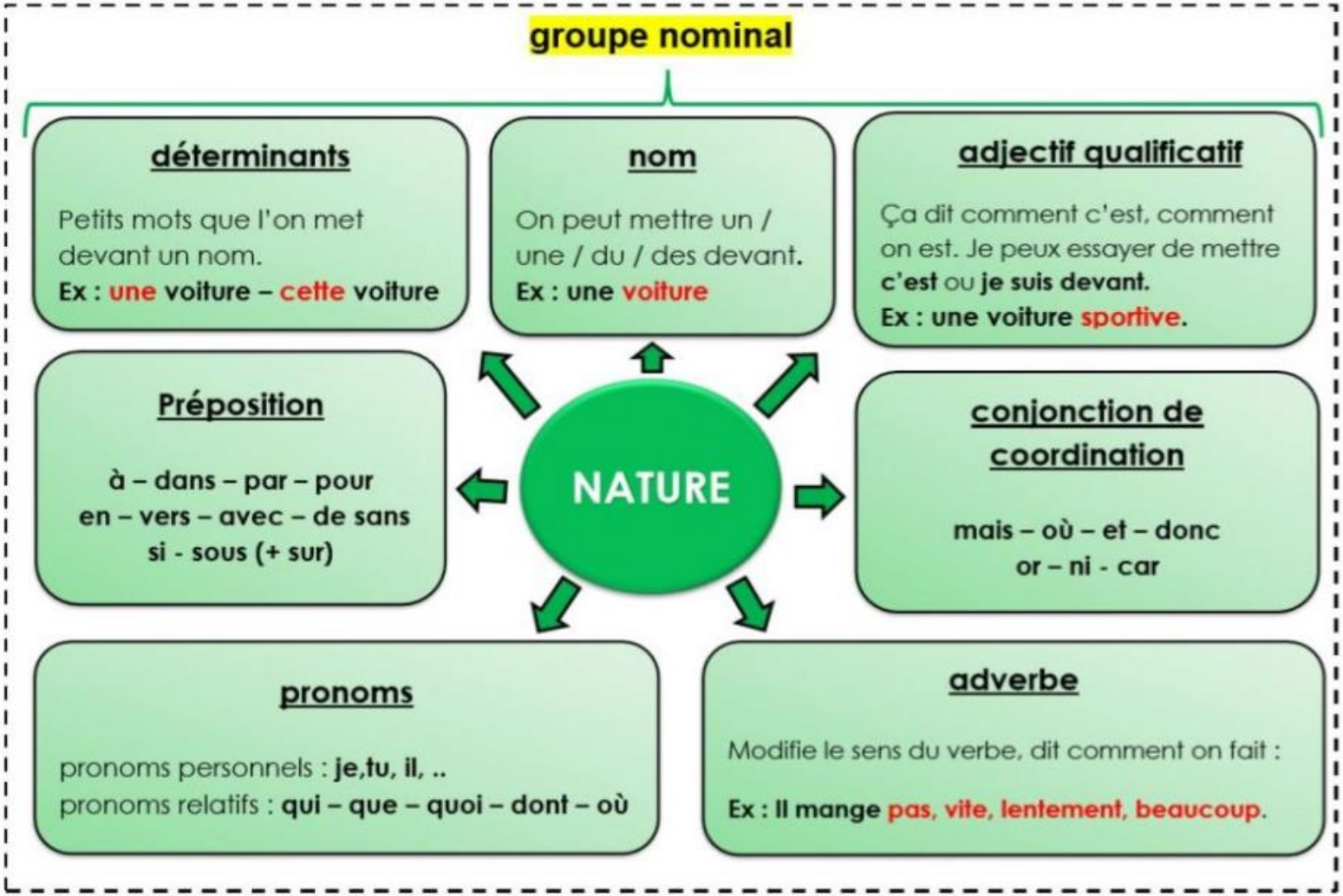
- Les principaux traits grammaticaux sont les suivants :
  - la **nature** ou classe grammaticale (nom, verbe, adjectif, etc.) ;
  - le genre (masculin, féminin, neutre, etc.) ;
  - le nombre (singulier, pluriel, etc.) ;
  - la personne (1re, 2e et 3e) ;
  - le cas (nominatif, accusatif, régime, etc.) ;
  - le temps (imparfait, présent, futur antérieur, etc.) ;
  - le mode (indicatif, subjonctif, infinitif, etc.) ;
  - la fonction (sujet, COD, COI, etc.) ;
  - la voix (active, passive) ;

Nature d'un mot - Part of speech

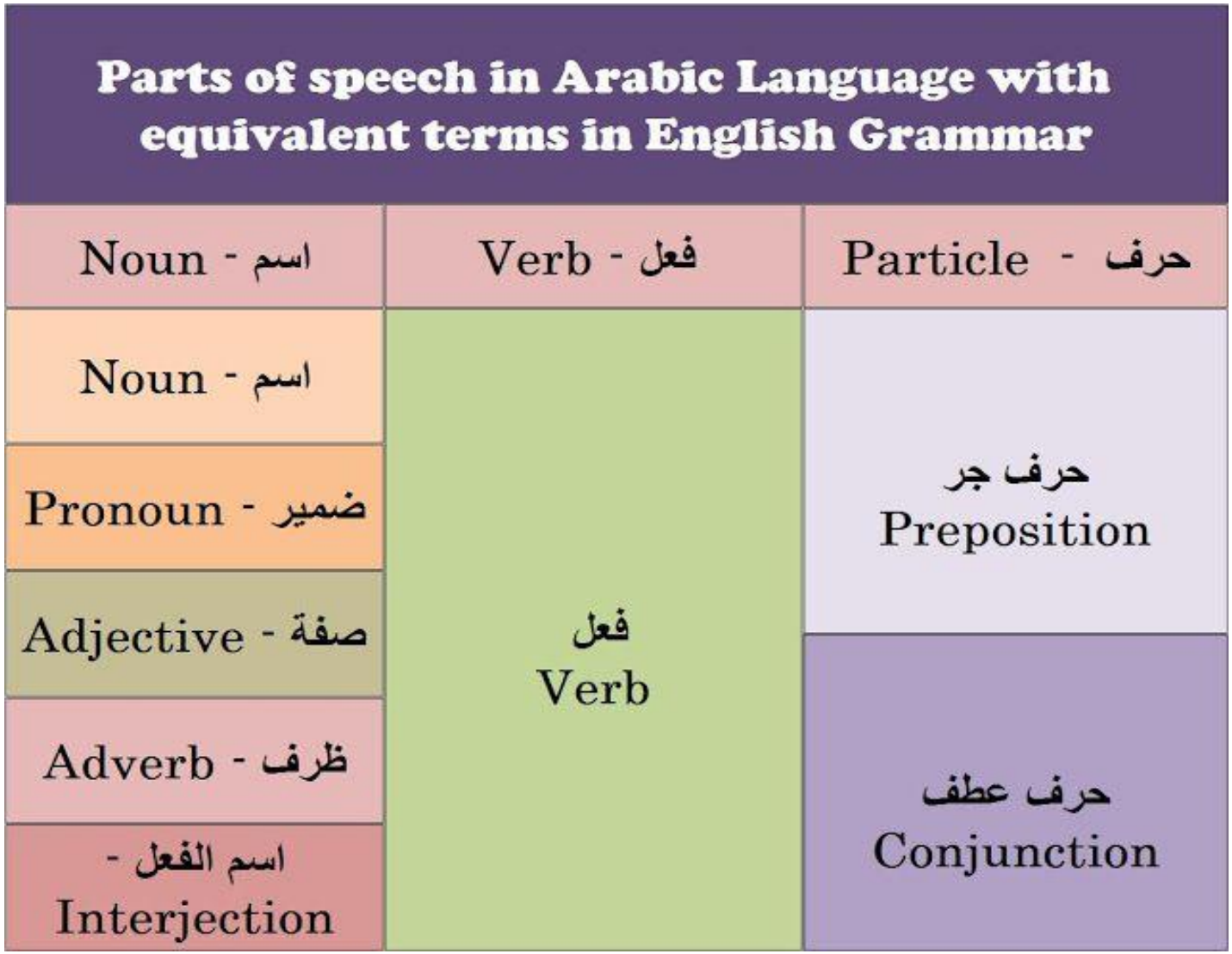
- La **nature** d'un mot est la catégorie/classe grammaticale de mots à laquelle il appartient. En Anglais : **part-of-speech** (POS).
- Une catégorie de mots est la réunion de mots d'un certain type, ayant des traits grammaticaux en commun.
- Catégories grammaticale, Lexicale, Classes grammaticale, etc. :
  - ❖ English / Français :
    - – Noun (names)
    - – Pronoun (replaces)
    - – Adjective (describes, limits)
    - – Verb (states action or being)
    - – Adverb (describes, limits)
    - – Preposition (relates)
    - – Conjunction (connects)
    - – Article (describes, limits)
    - – Interjection (expresses feelings and emotions)



Nature d'un mot - Part of speech



Nature d’un mot - Part of speech



1) Noun
2) Pronoun
3) Adjective
4) Adverb

} اِسْم

5) Verb → فِعْل

6) Preposition
7) Conjunction
8) Article

} حَرْف

Nature d'un mot - Part of speech

Les catégories grammaticales principales sont :

- noms (propre ou communs) : couvrent normalement les personnes, endroits et choses. Ex : ship, relationship, IBM
- verbes : font normalement référence à des actions. Ex : draw, provide, eating.
- adjectifs : décrivent normalement des propriétés ou qualités (good, bad, beautiful)
- adverbes (here, downhill, slowly, yesterday)
- auxiliaires (can, may, should, are) sont une sous-famille des verbes, qui se combinent à d'autres verbes.
- préprosition marquent généralement des relations spatiales ou temporelles (on, under, over, near, by, at, from, to, with)
- déterminants (a, an, the, this, that)
- pronoms (she, who, I, others, what)
- conjonctions (and, but, or, as, if, when) (mais, ou, et, donc, car, ni et or)
- particules ressemblent à une préposition ou à un adverbe et sont utilisées en combinaison avec un verbe (up, down, on, off, in, out, at, by)
- adjectifs numéraux ou numerals (one, two, three, first, second, third)

Nature d’un mot

- Catégorie moderne: Mots pleins (**open-class**) et mots outils (**closed-class**)

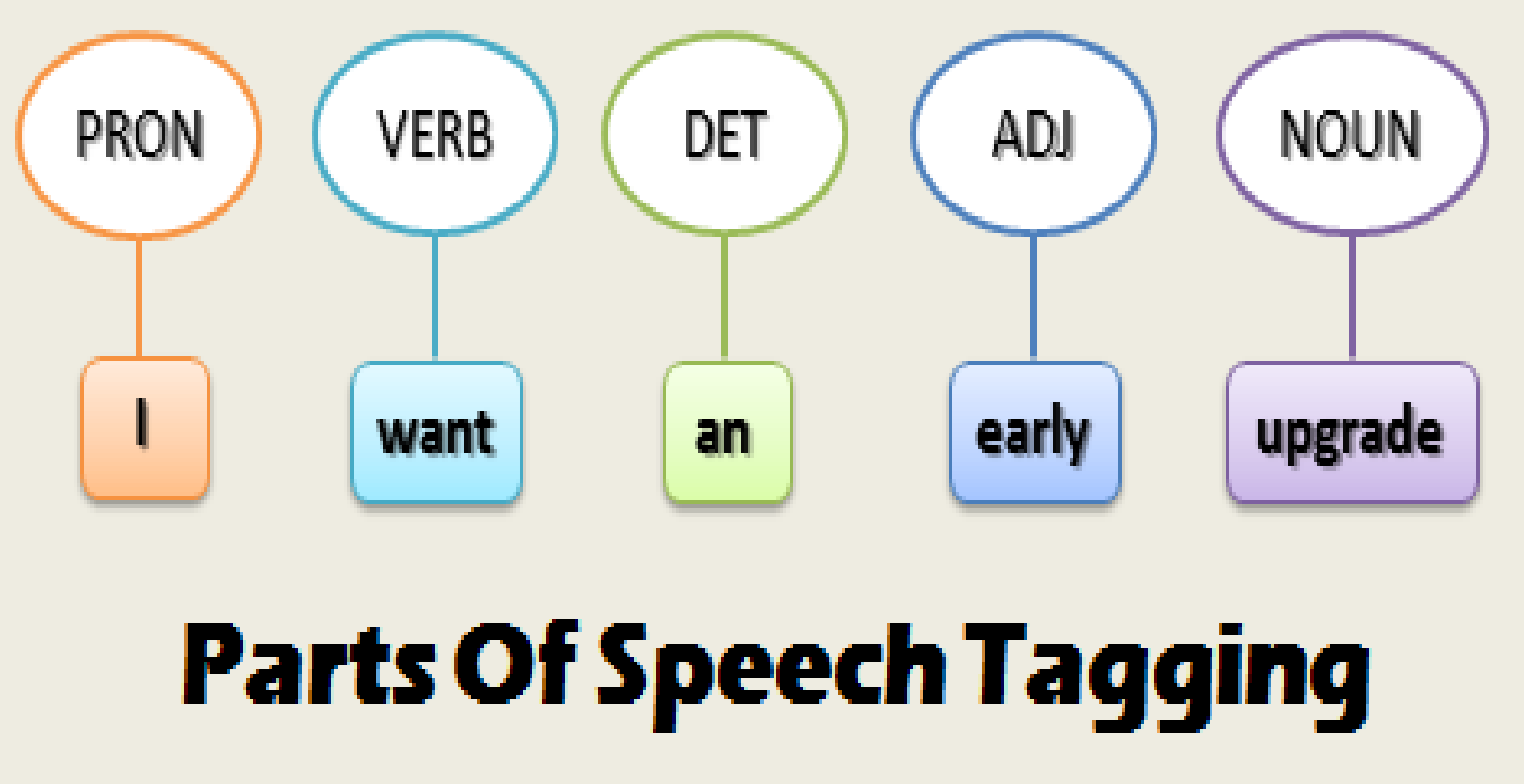
- **Classe ouverte**, mots pleins, est celle qui accepte généralement l'ajout de nouveaux mots.
- La classes ouverte contiennent normalement un grand nombre de mots.
- On retrouve : les noms , les verbes (à l'exclusion des verbes auxiliaires , s'ils sont considérés comme une classe distincte), les adjectifs , les adverbes et les interjections.

- **Classe fermée**, mots outils, est celle à laquelle de nouveaux éléments sont très rarement ajoutés.
- La classe fermée est beaucoup plus petites et contient un nombre limité de mots.
- On retrouve : les prépositions, les déterminants , les conjonctions, et les pronoms .

Nature d’un mot

Nature d'un mot - En Pratique – POS Tagging

- **Part of Speech Tagging – Etiquetage Morpho-Syntaxique:**

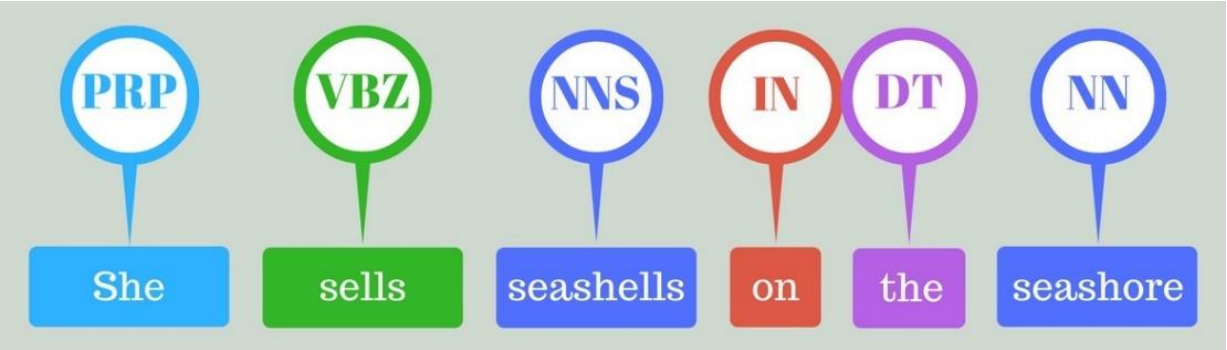


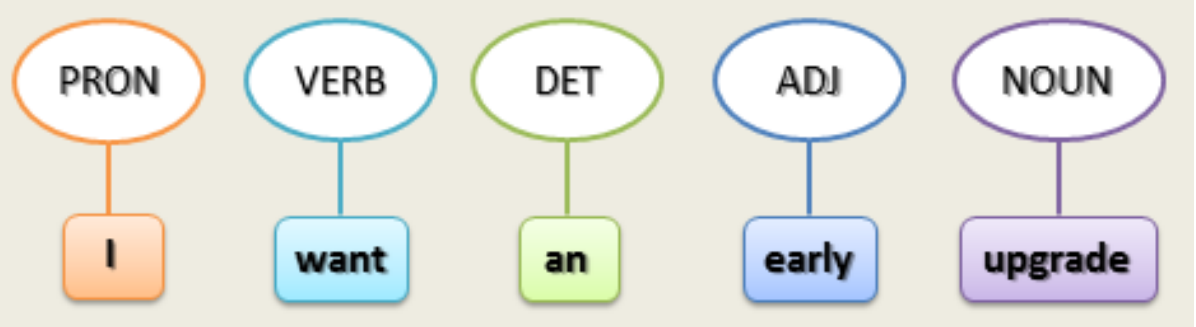
Etiquetage Morpho-syntaxique

- Part-of-Speech Tagging – **POS Tagging**
- Est le processus de classification automatique des mots dans leurs catégorie grammaticale, et de les étiqueter (annoter) en conséquence.
- Assigne chaque mot d'un texte à sa catégorie grammaticale.
- Par exemple, le mot ferme peut être un **verbe** (**V**) dans « il ferme la porte » et un **nom** (**N**) dans « il va à la ferme ».
- Par exemple : drink peut être brevage (nom) ou boire (verbe).

Etiquetage Morpho-syntaxique

Etiquetage Morpho-syntaxique





Etiquetage Morpho-syntaxique

Ensemble d'étiquettes – Tagset

- La collection de **tags** utilisés pour une tâche d'étiquetage morpho-syntaxique est connue sous le nom **tagset**.
- Dans un **corpus**, on représente les **classes grammaticales** à partir d'ensembles d'étiquettes ou **tagset**.
- Le choix du tagset détermine quelles classes grammaticales on va distinguer dans nos données.
- Exemple – Corpus :

The/**DT** grand/**JJ** jury/**NN** commented/**VBD** on/**IN** a/**DT** number/**NN** of/**IN** other/**AP** topics/**NNS** ,/, AMONG/**IN** them/PPO the/AT Atlanta/NP and/CC Fulton/NP-tl County/NN-tl purchasing/VBG departments/NNS which/WDT it/PPS said/VBD "/" ARE/BER well/QL operated/VBN and/CC follow/VB generally/RB accepted/VBN practices/NNS which/WDT inure/VB to/IN the/AT best/JJT interest/NN of/IN both/ABX governments/NNS "/" ./.

Etiquetage Morpho-syntaxique

Ensemble d'étiquettes – Tagset

- **Universal Part-of-Speech Tagset – Universal Dependencies :**

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Etiquetage Morpho-syntaxique

Ensemble d'étiquettes – Tagset

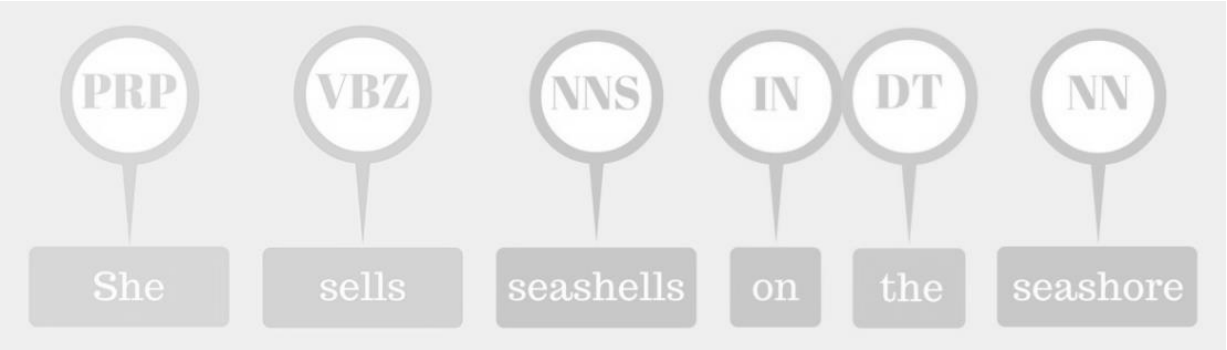
- **Universal Part-of-Speech Tagset** :
- https://universaldependencies.org/u/pos/all.html

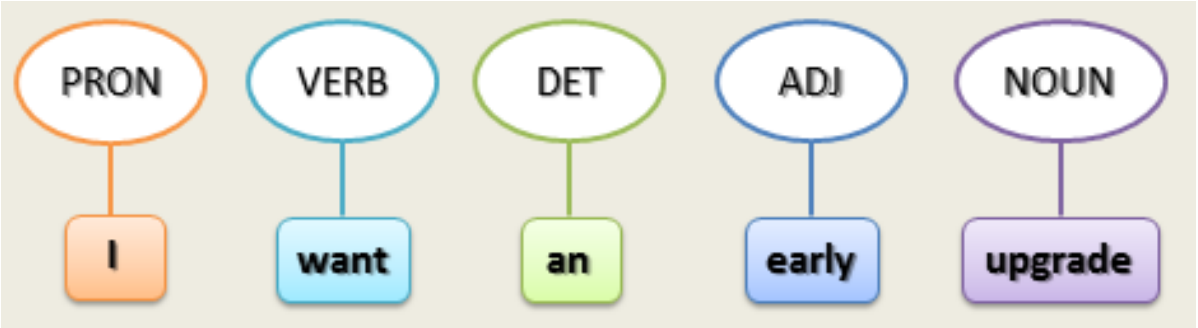
Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Alphabetical listing

- ADJ: adjective
- ADP: adposition
- ADV: adverb
- AUX: auxiliary
- CCONJ: coordinating conjunction
- DET: determiner
- INTJ: interjection
- NOUN: noun
- NUM: numeral
- PART: particle
- PRON: pronoun
- PROPN: proper noun
- PUNCT: punctuation
- SCONJ: subordinating conjunction
- SYM: symbol
- VERB: verb
- X: other

Etiquetage Morpho-syntaxique





Etiquetage Morpho-syntaxique

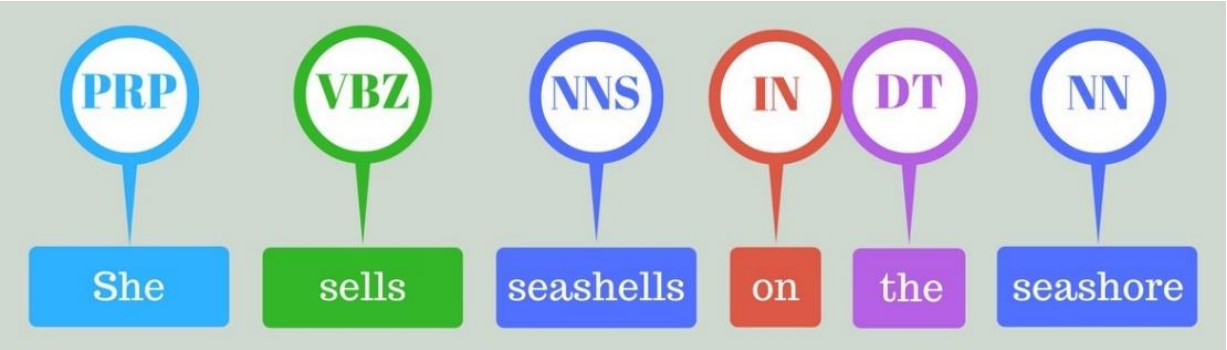
Ensemble d'étiquettes – Tagset

- **Penn Treebank tagset** : Le tagset le plus populaire. **45** étiquettes. Utilisé pour étiqueter de nombreux corpus en anglais.

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	"	left quote	' or "
LS	list item marker	*1, 2, One*	TO	"to"	*to*	"	right quote	' or "
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

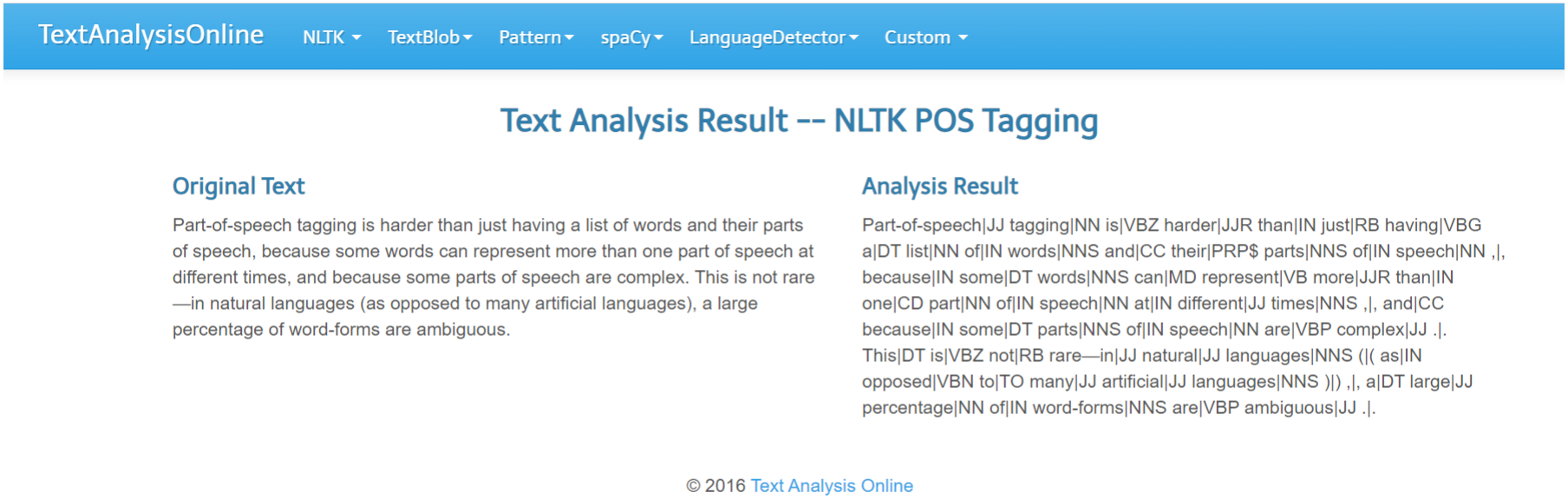
- Il existe d'autres tagsets, plus exhaustifs: le **Brown** tagset, **87** étiquettes.

Etiquetage Morpho-syntaxique



Etiquetage Morpho-syntaxique

- A tester en ligne: par exemple, NLTK POS-Tagging
- https://textanalysisonline.com/nltk-pos-tagging
- http://text-processing.com/demo/tag/



Etiquetage Morpho-syntaxique

- **Comment** cette identification est-elle réalisée ?
- La solution la plus simple pour les mots/tokens les plus fréquents est de rechercher la forme dans un **lexique** pré-compilé : Accès lexical direct.
- **Wordnet** est une grande base de données lexicale, disponible gratuitement et publiquement pour la langue anglaise (et autres), visant à établir des relations sémantiques structurées entre les mots.
- http://wordnetweb.princeton.edu/perl/webwn

Part of Speech	Tag
Noun	n
Verb	v
Adjective	a
Adverb	r

POS Tagging - WordNet

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: helping [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

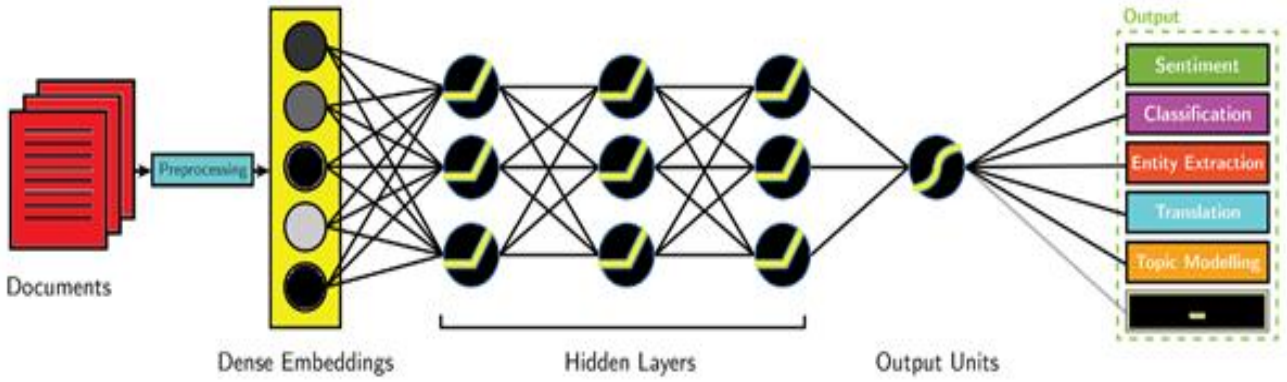
- S: (n) **helping**, portion, serving (an individual quantity of food or drink taken as part of a meal) *"the helpings were all small"; "his portion was larger than hers"; "there's enough for two servings each"*

Verb

- S: (v) help, assist, aid (give help or assistance; be of service) *"Everyone helped out during the earthquake"; "Can you help me carry this table?"; "She never helps around the house"*
- S: (v) help, aid (improve the condition of) *"These pills will help the patient"*
- S: (v) help, facilitate (be of use) *"This will help to prevent accidents"*
- S: (v) help oneself, help (abstain from doing; always used with a negative) *"I can't help myself--I have to smoke"; "She could not help watching the sad spectacle"*
- S: (v) serve, help (help to some food; help with food or drink) *"I served him three times, and after that he helped himself"*
- S: (v) help (contribute to the furtherance of) *"This money will help the development of literacy in developing countries"*
- S: (v) avail, help (take or use) *"She helped herself to some of the office supplies"*
- S: (v) help (improve; change for the better) *"New slipcovers will help the old living room furniture"*

POS Tagging

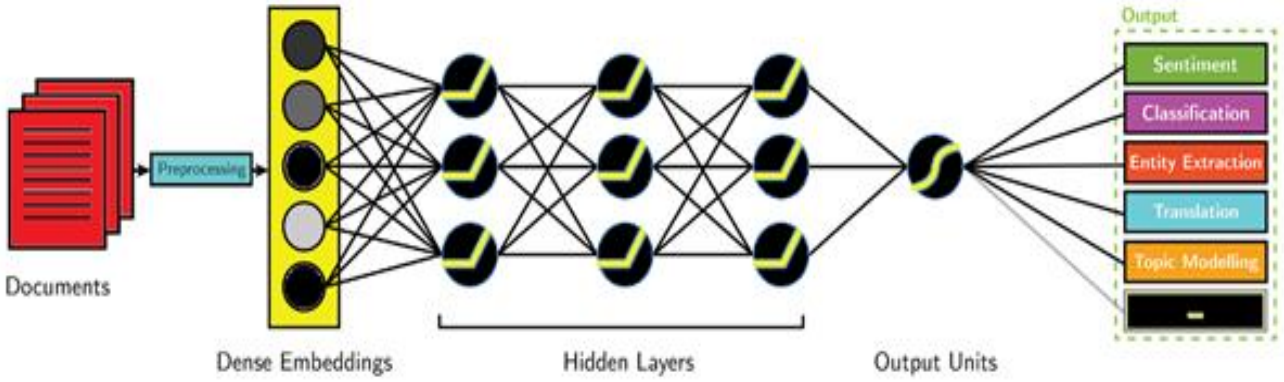
- **Comment** cette identification est-elle réalisée ?
- La solution la plus simple pour les mots/tokens les plus fréquents est de rechercher la forme dans un **lexique** pré-compilé : Accès lexical direct.
- Cette solution **ne résout pas** tous les problèmes. Le langage est création, et de nouvelles formes/tokens surgissent tous les jours.
- Mise en œuvre d'autres **approches**, de manière à traiter aussi les formes hors-lexique.



Etiquetage Morpho-syntaxique

- **Comment** cette identification est-elle réalisée ?
- **Approches** :

- Règles manuelles
- Apprentissage automatique:
  - ✓ Apprendre les règles : Transformation-Based Learning (TBL)
  - ✓ Modèles statistiques : **Hidden Markov Models**
  - ✓ Sur l'entropie maximale, champs aléatoires conditionnels (CRF)



POS Tagging

POS Tagging

- **Comment** cette identification est-elle réalisée ?
- La solution la plus simple pour les mots/tokens les plus fréquents est de rechercher la forme dans un **lexique** pré-compilé : Accès lexical direct.
- Cette solution ne résout pas tous les problèmes. Le langage est création, et de nouvelles formes/tokens surgissent tous les jours.
- Mise en œuvre d'autres **approches**, de manière à traiter aussi les formes hors-lexique.
- ➔ **Morphologie :** « *étude des **formes** sous lesquelles se présentent les **mots** dans une langue, des changements dans la forme des mots pour exprimer leurs relations à d'autres mots de la phrase, des processus de **formation de mots** nouveaux, etc.* ». G. Mounin
- Analyse lexicale ou analyse morphologique.

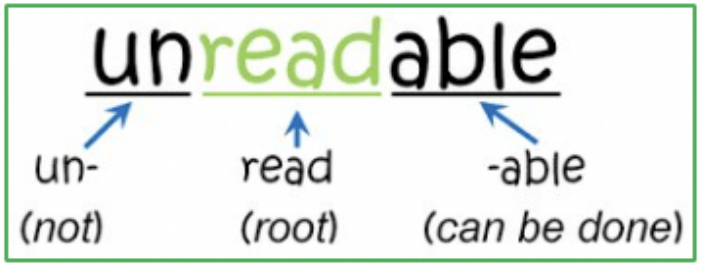
Morphologie

➔ **Morphologie :** La morphologie est la branche de la linguistique et de la grammaire qui s'intéresse à la **formation** du mot.

- Exemple : Pour comprendre une forme telle que *Parisien*, être capables de reconnaître dans cette forme des **composants** plus **petits**, nommément une **racine**, Paris, qui réfère au nom d'une ville, et un **suffixe**, ien, qui permet de manière régulière de construire des adjectifs à partir de noms propres.

- La linguistique traditionnelle appelle ces composants plus petits les **morphèmes**, et l'étude de leurs combinaisons la morphologie.

- Morphème: le plus petit élément significatif individualisé dans un énoncé, isolé par segmentation d'un mot.



Morphèmes et leurs typologie

- **Morphème**: le plus petit élément significatif individualisé dans un énoncé, isolé par segmentation d'un mot. Concept de base en morphologie.

- Par exemple, le mot *chanteurs* est composé de trois morphèmes : *chant-* « chant », *-eur-* « celui qui fait » et *-s* (marque du pluriel).

- Exemple 2 : *base de données* n'est pas composé de trois morphèmes mais bien d'un seul morphème qui contient la signification « base de données ».

- Typologie : Morphèmes lexicaux ou Morphèmes grammaticaux.

❖ Les morphèmes **lexicaux** sont en nombre illimité (*ouverte*), tels que *lave*, *vite*, *lune*, etc. Il s'agit de noms, adjectifs, verbes ou adverbes.

❖ Les morphèmes **grammaticaux** sont en nombre limité (*fermée*), tels que *tu*, *à*, *et*, etc. Il s'agit de pronoms, prépositions, conjonctions, déterminants : des listes de mots qui ne varient pratiquement jamais.

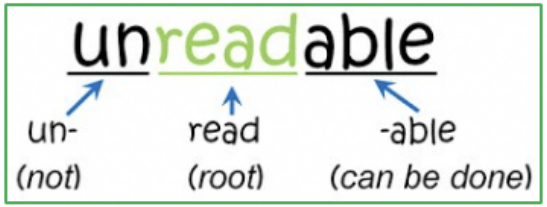
Morphèmes et leurs typologie

- Typologie : Morphèmes liés ou Morphèmes libres.

❖ **Lié** s'il ne se manifeste pas comme mot et n'existe jamais à l'état libre mais est toujours rattaché à un autre morphème appelé base.
- Ex : -ons dans ouvr-ons, ou re- dans re-faire, ou un radical comme -cevoir (re-cevoir, per-cevoir, dé-cevoir, etc.) qui n'existe pas non plus à l'état libre.

❖ **Libre** s'il peut constituer un mot à lui seul, racine: le ou beau sont libres.

- Un morphème **libre** peut se rencontrer seul
  - petit, maison, jadis, et, avec
- Un morphème **lié** est toujours associé à un autre
  - petite = petit (libre) + -e (lié) *écrit*
    - petite, partons
  - des **affixes** (suffixes, préfixes) :
    - impossible, peureux, ex-ministre, mangeable



Morphèmes et leurs typologie

- Typologie : Morphèmes flexionnels ou Morphèmes dérivationnels.

❖ Les morphèmes **flexionnels**, ou flexions, indiquent la relation que la base à laquelle ils s'ajoutent entretient avec les autres unités de l'énoncé.

- Deux types principaux de flexions selon la catégorie de la base :

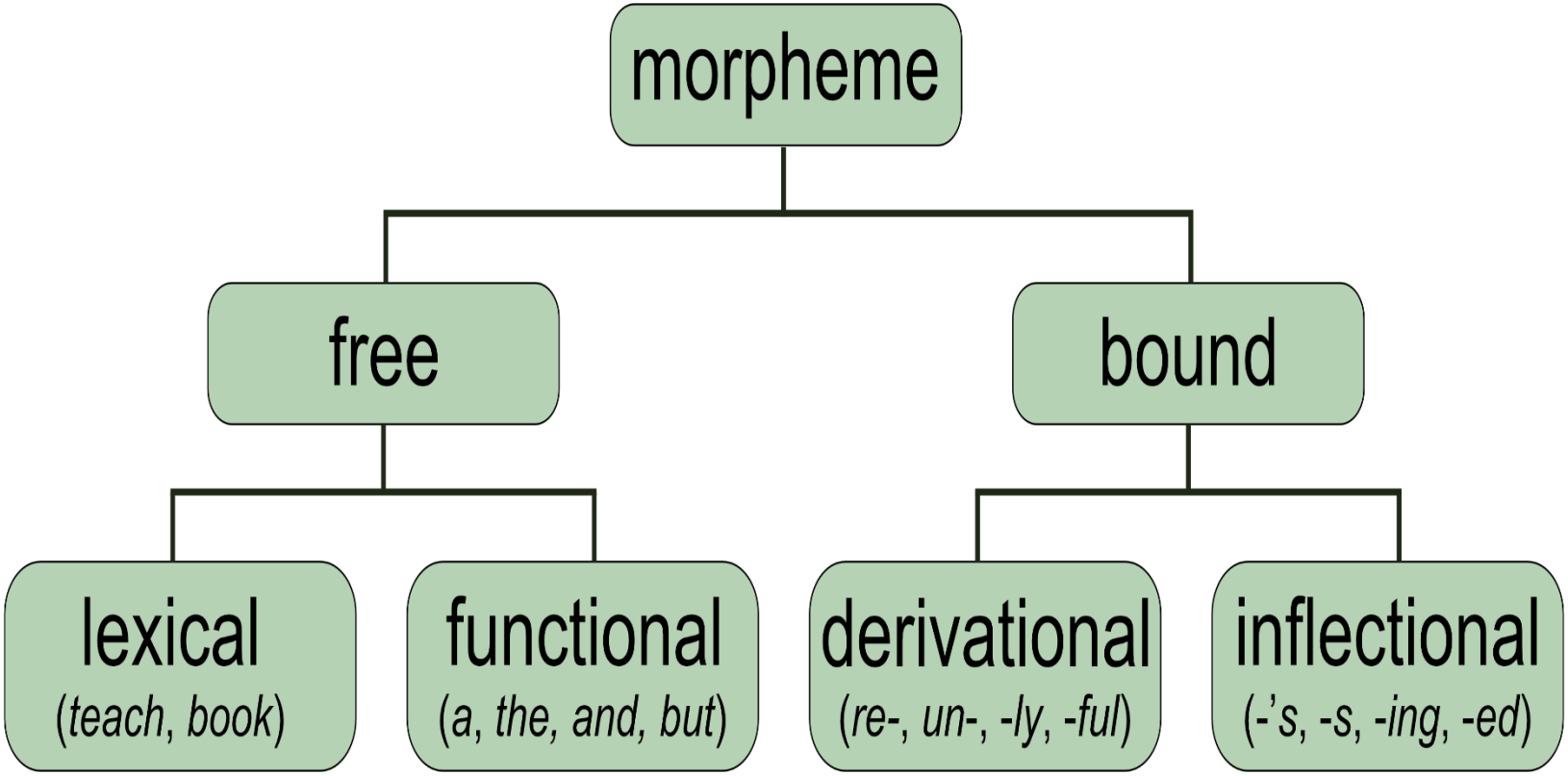
✓ les flexions qui concernent les bases nominales, adjectivales, et pronominales. Elles sont de trois sortes en français : le genre, le nombre, et les cas.

✓ les flexions verbales qui correspondent à la conjugaison des verbes. Elles ont pour fonction de marquer la personne, le nombre, le temps, le mode et la voix.

❖ Les morphèmes **dérivationnels**, ou affixes, servent à la création/formation de nouveaux mots lexicaux par dérivation.

- Deux types principaux de morphèmes dérivationnels : préfixes et suffixes.

Morphèmes et leurs typologie



Morphologie

- Deux **types** de **morphologie**:

1) La morphologie flexionnelle (en interne) التصريف: les processus d'ajustement et de variation de mots imposés par les conditions/traits grammaticaux d'utilisation du mot, sans changer son sens ou sa catégorie grammaticale. Ex :

   – étudiant => étudiant**s (pluriel)**
   – Petit => petit**e (féminin)**

   – أكل=> يأكل، أكلنا
   – طالب => طالبة، طلبة

2) La morphologie dérivationnelle (en externe) الاشتقاق: les processus de création de nouveaux mots à partir de mots existants. Les processus dérivationnels entraînent le plus souvent un changement de la catégorie grammaticale: un nom se transforme en verbe, un verbe en adjectif. Ex:

   – Jouer => joue**ur**

   – طلب => طالب، مطلوب

Morphologie flexionnelle

1 - **La morphologie flexionnelle**:

- En morphologie, on nomme **flexion** l'ensemble des modifications subies aux mots d'une langue flexionnelle pour dénoter les traits grammaticaux voulus.
- À la différence de la dérivation, la flexion ne crée pas de nouveaux mots, mais différentes formes d'un même mot.
- Il existe **2 grandes catégories de flexions**: nominale (et adjectivale) et verbale.

❖ La déclinaison pour le système nominal: Les noms y changent généralement de forme selon le **genre**, le **cas,** ou le **nombre**;

❖ La conjugaison pour les verbes: qui varient généralement en **personne**, **nombre**, **temps**, **voix,** et **mode**.

Morphologie flexionnelle

1 - **La morphologie flexionnelle**:

Exemple de déclination

La déclination de mot chat en espagnole

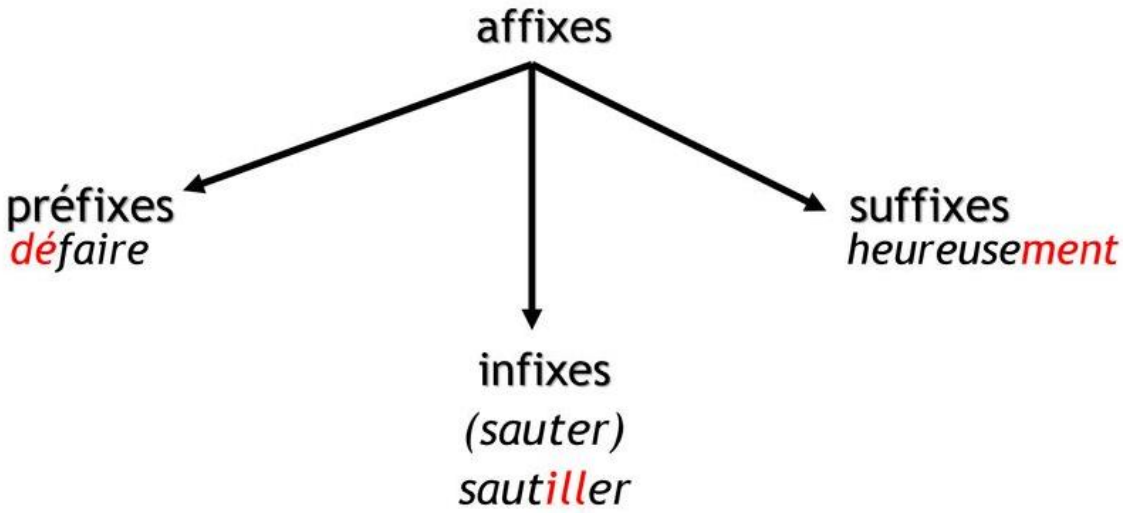
Conjugaison

•المضارع	الماضي
•أنا أكتبُ	أنا كتبتُ
•أنت تكتبُ	أنت كتبتَ
•أنت تكتبين	أنت كتبتِ
•هو يكتب	هو كتبَ
•هي تكتب	هي كتبتْ

Morphologie flexionnelle

1 - **La morphologie flexionnelle**:

- La génération des nouvelles formes se fait par les méthodes suivantes :
  - L'affixation: l'ajout des affixes (préfixes, infixes, et des suffixes)
    - ✓ Préfixes, placés avant leur base
    - ✓ Infixes, placés à l'intérieur d'une base, ou entre une base et un affixe d'un terme déjà lexicalisé
    - ✓ Suffixes, placés après la base



Morphologie flexionnelle

1 - **La morphologie flexionnelle**:

- La génération des nouvelles formes se fait par les méthodes suivantes :
  - Duplication : Dupliquer le mot ou bien une partie de mot. Ex:

- ✓ زلزل
- ✓ الشدة
- ✓ كيف كيف
- ✓ Ping-pong
- ✓ Bye-bye
- ✓ zigzag

https://www.almaany.com/ar/dict/ar-ar/%D8%B9%D9%82%D8%AF/

Morphologie flexionnelle

1 - **La morphologie flexionnelle**:

- La génération des nouvelles formes se fait par les méthodes suivantes :
  - Altération : Changer une lettre ou bien une partie du mot. Ex :

    ✓ Franc => Franche
    ✓ Verlan : pourri > ripou

  - Variation super segmentale : Changer l'intonation du mot

    ✓ رايب
    ✓ Azekka

Morphologie dérivationnelle

2 - **La morphologie dérivationnelle**:

- La dérivation lexicale, ou encore dérivation, sert à créer des nouveaux mots ; on peut former de mots nouveaux en ajoutant des morphèmes dérivationnels à des mots existants.

- Elle affecte la signification du mot et à l'occasion la catégorie grammaticale.

- Ex : danser, dans-eur
- Ex :علم, عالم, معلومة, استعلام, إعلام

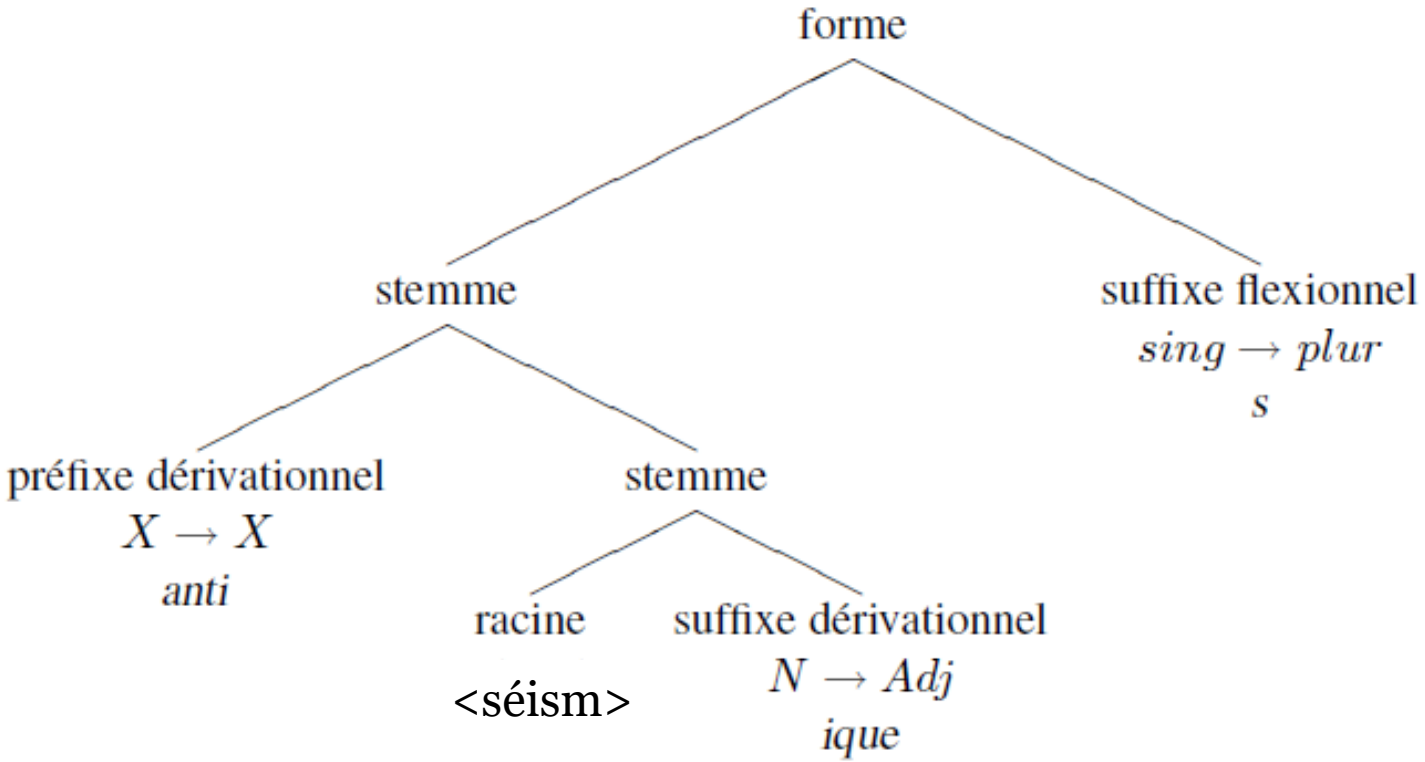
- Les méthodes de dérivation :

❖ L'affixation: l'ajout des affixes (préfixes et suffixes) et patterns dérivationnels:
Exemples en anglais:

- adjective-to-noun: -ness (slow → slow**ness**)
- adjective-to-adverb: -ly (personal → personal**ly**)

Morphologie

- Une décomposition **arborescente** de la forme : **antiséismiques**



Quelques méthodes de formation de mots

= Quelques types de combinaisons de morphèmes :

- **Dérivation**.
- **Flexion**.
- **Composition** : permet de jumeler deux items lexicaux (racines) pour en former un nouveau. Ex : Plate-forme, hors-la-loi, Smartphone, en effet, etc.
- **Mot-valise** : consiste à coller le début d'un mot à la fin d'un autre pour créer un nouveau mot. Ex: autobus, franglais, transistor (transfer resistor), etc.
- **Troncation** : permet de raccourcir un item lexical. Ex: fac, ricain, foot, etc.
- **Acronyme** : un sigle passé dans le langage courant : ovni, radar, covid, etc.
- Etc.

L’emprunt lexicale

- Un emprunt est un mot ou une expression qu’un locuteur ou une communauté emprunte à une autre langue, sans le traduire, mais en l’adaptant généralement aux règles morphosyntaxiques, phonétiques et prosodiques de sa langue (dite «langue d’accueil»).

- Est le procédé consistant pour une langue à adopter dans son lexique un mot ou une expression d’une autre langue.

- L’emprunt peut être : – direct (une langue emprunte directement à une autre langue), – indirect (une langue emprunte à une autre langue via une ou plusieurs langues vecteurs).

- Ex : قهوة => kahve => caffè => café
- Ex : Internet
- Ex : Espagnol => Dardja : Suma => “ سُّومَة ”

En Pratique – Normalisation et Filtrage du texte

Filtrage du texte

- PROBLEMATIQUE : le texte peut contenir des caractères, des mots et des expressions qui peuvent entraver son traitement.
- SOLUTION : suppression, modification, conversion
- Stopwords removal, lowercase conversion, etc.

Normalisation du texte

- PROBLEMATIQUE : Un texte peut contenir des variations du même terme. Aussi, dans des tâches comme la recherche d'information, on n'a pas besoin d'avoir le contenu exacte du texte.
- SOLUTION : transformer le texte à une **forme canonique**.
- **Lemmatisation, Stemming, etc.**

Normalisation du texte

- La **normalisation du texte** est le processus de transformation du texte en une seule forme canonique qu'il n'aurait peut-être pas eu auparavant.
- Normaliser le texte avant de le stocker ou de le traiter permet de séparer les problèmes, car l'entrée est garantie d'être cohérente avant que les opérations ne soient effectuées sur celui-ci.
- La normalisation de texte nécessite de savoir quel type de texte doit être normalisé et comment il doit être traité par la suite.
- Normaliser les diverses manières d'écrire un même mot; convertir tout le texte en minuscule; corriger les fautes d'orthographe évidentes ou les incohérences typographiques et à expliciter certaines informations lexicales (ex: l’ => le/la);
- **lemmatisation; stemming**; etc.

Normalisation du texte

Description du traitement	Étape
1. (Grefenstette et Tapanainen, 1994)	
1. Supprimer les étiquettes SGML	I2
2. Recoller les césures	I3
3. Marquer les nombres et les abréviations	A1
4. Segmenter le texte en phrases	E1
2. Multext multilingual segmenter tools (v 1.3.1), 1997	
1. Séparer le texte selon les espaces	A2
2. Isoler les ponctuations	A1
3. Fusionner les ponctuations composées	A1
4. Séparer les expressions avec ponctuations	A1
5. Fusionner les abréviations composées	A1
6. Identifier les abréviations	A1
7. Recombiner les unités multimots	E2
8. Identifier les dates	A1
9. Identifier les nombres	A1
10. Identifier les énumérations	A2
11. Détecter les frontières des phrases	E2

Description du traitement	Étape
4. Gate 3.0 (Cunningham et al., 2002)	
1. Segmentation en mots	E2
2. Lemmatisation	E2
3. Identification des entités nommées	A1
4. Découpage en phrases	E1
5. SXPIPE 1.0 (Sagot et Boullier, 2005)	
1. Identification des courriels, dates, adresses, etc.	A1
2. Détection des frontières des phrases	A2
3. Identification des mots inconnus	A1
4. Identification des acronymes, noms propres	A1
5. Identification des mots étrangers	A1
6. Découpage en mots et correction orth.	E2/I3
7. Identification des nombres en lettres	A1
8. Identification des mots composés	E2
9. Réaccentuation et recapitalisation	I3

http://lexicometrica.univ-paris3.fr/jadt/jadt2006/PDF/043.pdf

Normalisation du texte

3. (Adda et al., 1997)	
0. Encodage des accents et autres diacritiques	I1
0. Prétraitement des nombres et unités	I3
0. Correction du formatage et des ponctuations	I3
0. Traitement des ponctuations non ambiguës	I3
0. Séparation en articles, paragraphes, phrases	E2
1. Traitement des ponctuations ambiguës	A1
2. Traitement des débuts de phrase capitalisés	A1
3. Traitement des nombres	A1
4. Traitement des acronymes	A1
5. Traitement des capitales emphatiques	A1
6. Décomposition	E2
7. Pas de distinction de casse	I3
8. Pas de diacritiques	I1

6. Notre modèle (Amrani et al., 2004)	
1. Remplace les caractères non ASCII et entités	I1
2. Convertit le document en format texte linéaire	I2
3. Met un paragraphe par ligne	A2
4. Normalise le paragraphe - Incohérences	I3
5. Normalise le paragraphe - Ambiguïtés	A1
6. Recherche des frontières des phrases	A1, A2
7. Met une phrase par ligne	E1
8. Normalise les phrases	E2

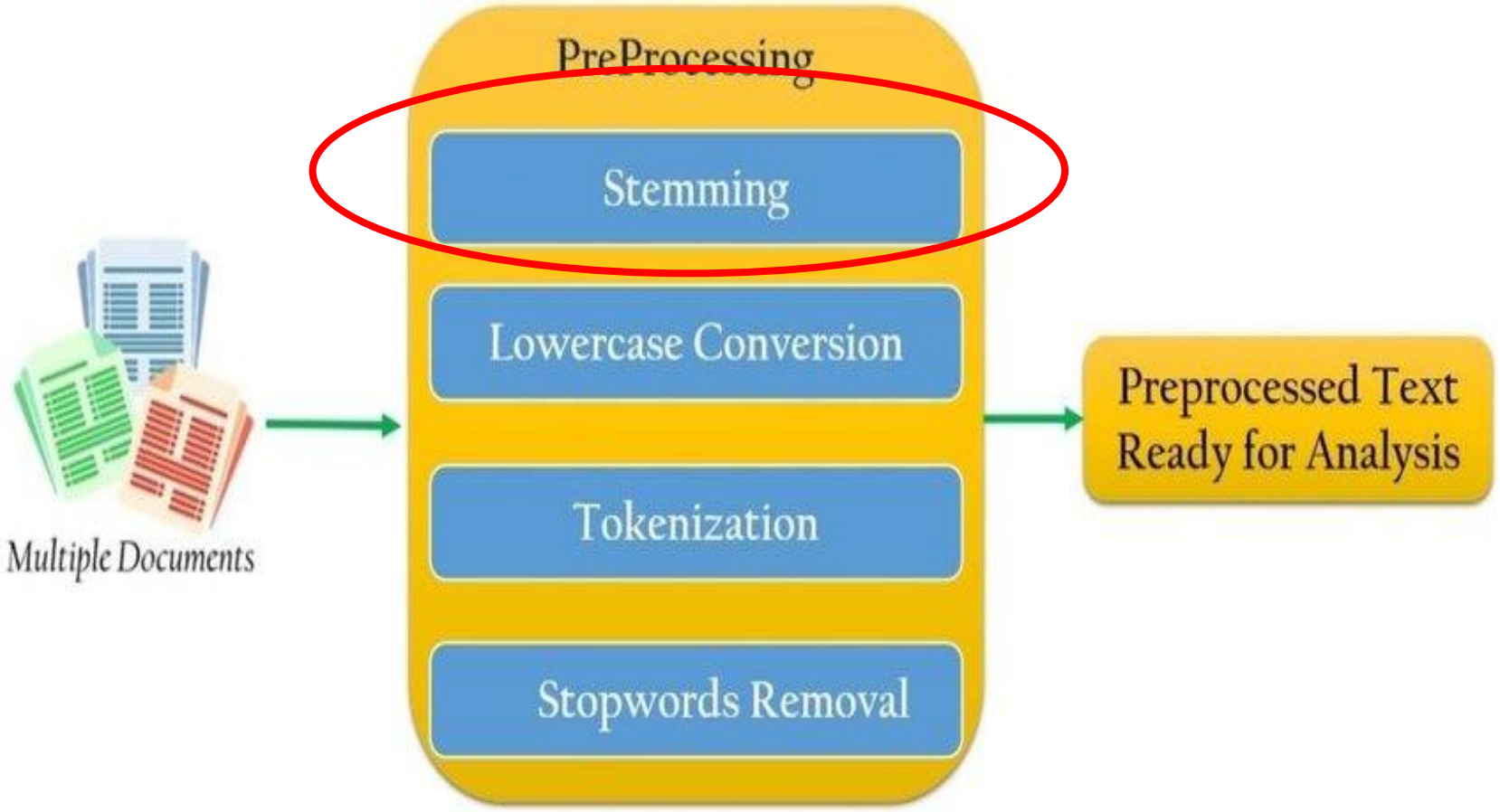
7. (Mikheev, 2000)	
1. Classer les expressions terminées par un point	A1
2. Classer les mots capitalisés après un point	A1
3. Assigner les fins de phrases	E1

Tab. 1 - *Enchaînements des traitements proposés dans différentes chaînes de prétraitement des textes.*

http://lexicometrica.univ-paris3.fr/jadt/jadt2006/PDF/043.pdf

En Pratique - Normalisation et Filtrage du texte

- Etapes de prétraitement (PreProcessing) d'un énoncé textuel :

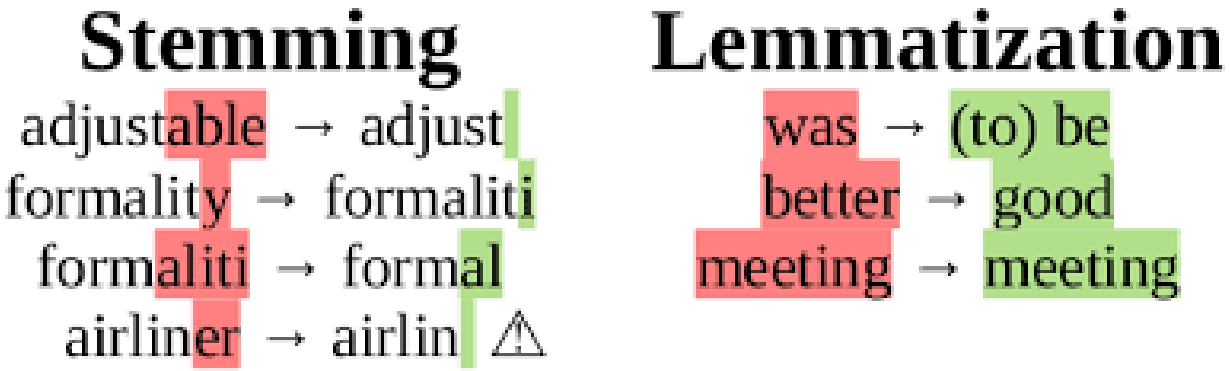


Lemmatisation et Stemming

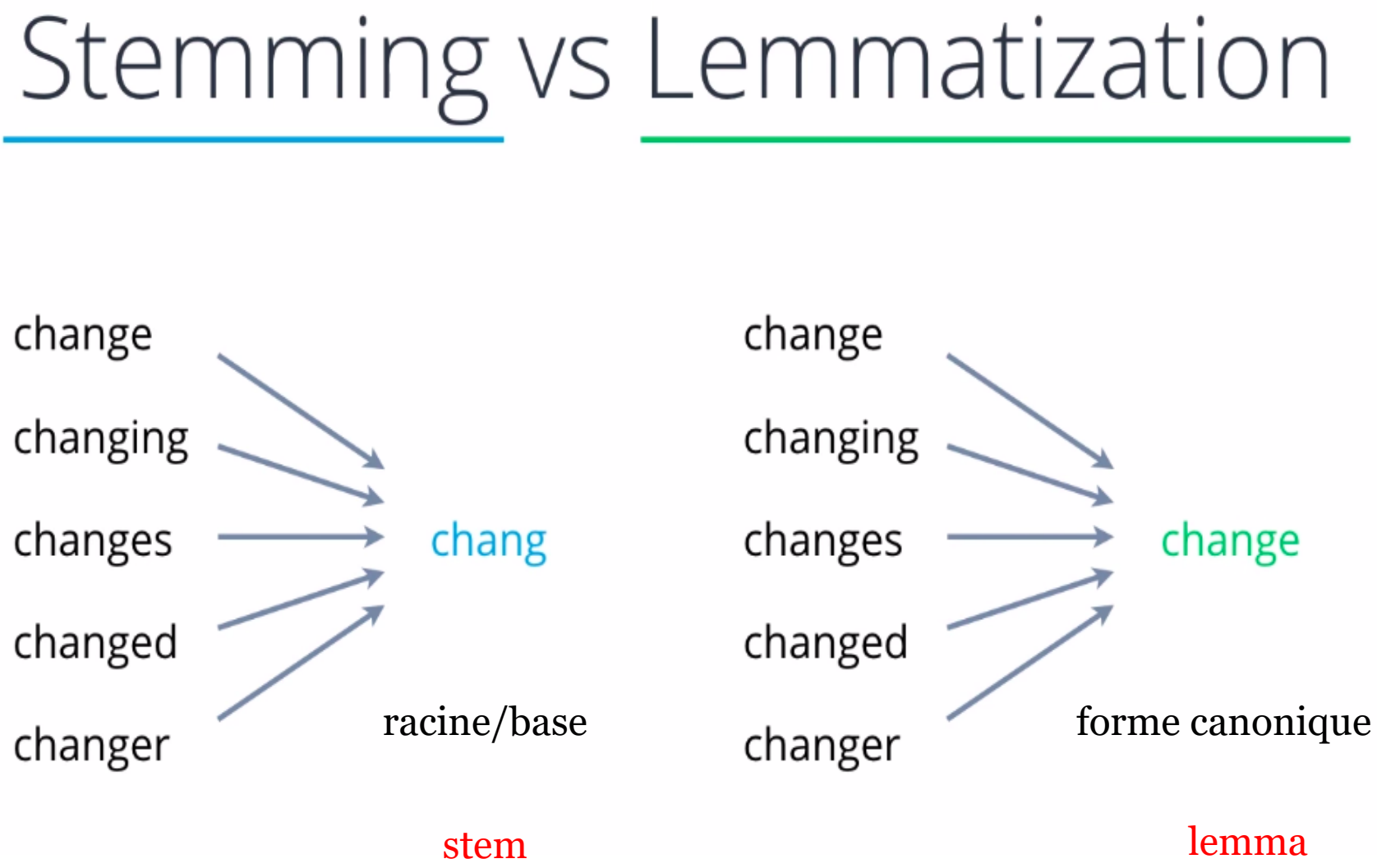
- **Stemming** consiste à retrouver le **stemme** d'une forme/mot. = la racinisation ou désuffixation, un processus qui consiste à réduire les mots par leur radical (appelés également stemmes, bases ou racines). Pas nécessairement un mot dans le dictionnaire.

- **Lemmatisation** consiste à retrouver le **lemme** d'un mot (i.e. forme canonique / forme dictionnaire d'un mot), c'est-à-dire à lui retirer ses suffixes. Regrouper les mots d'une même famille. On doit avoir son contexte.

- La lemmatisation est plus informative que le stemming. Mais prend plus de temps à s'éxécuter.

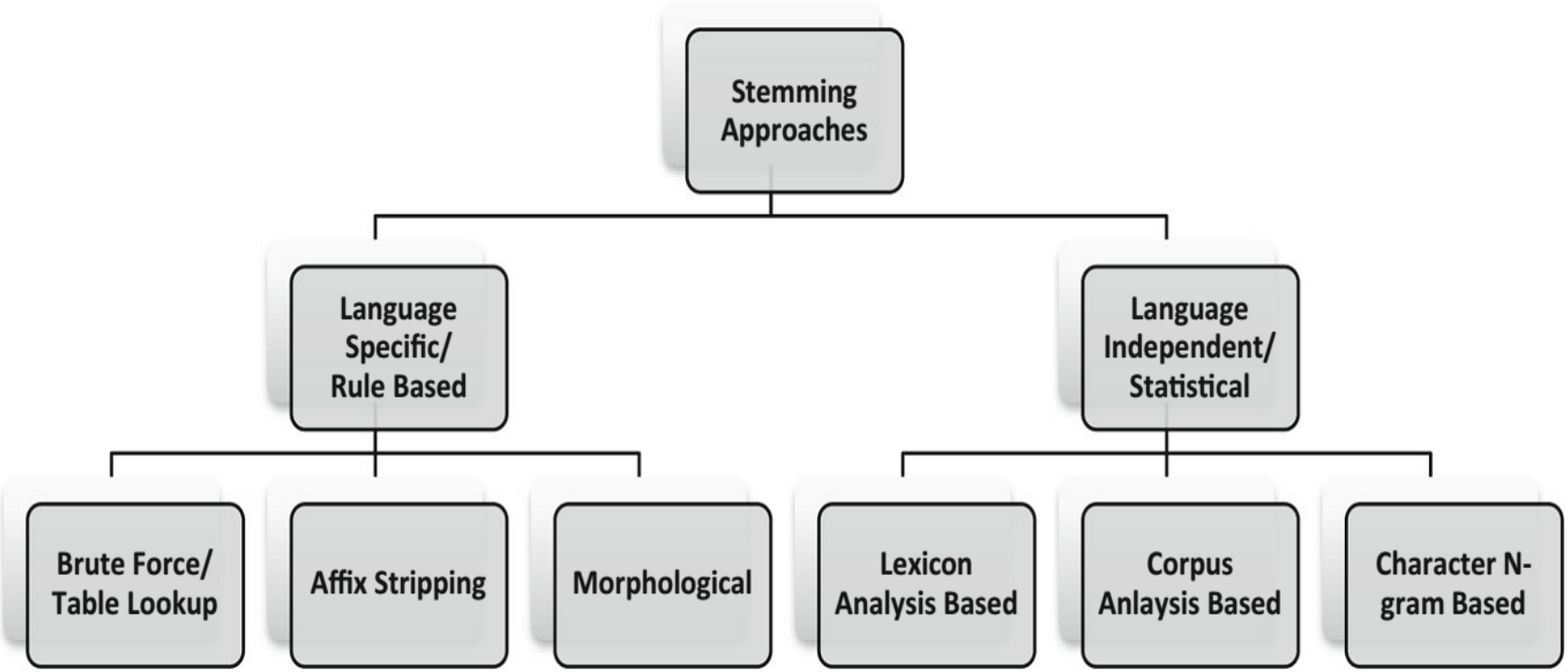


Lemmatisation et Stemming



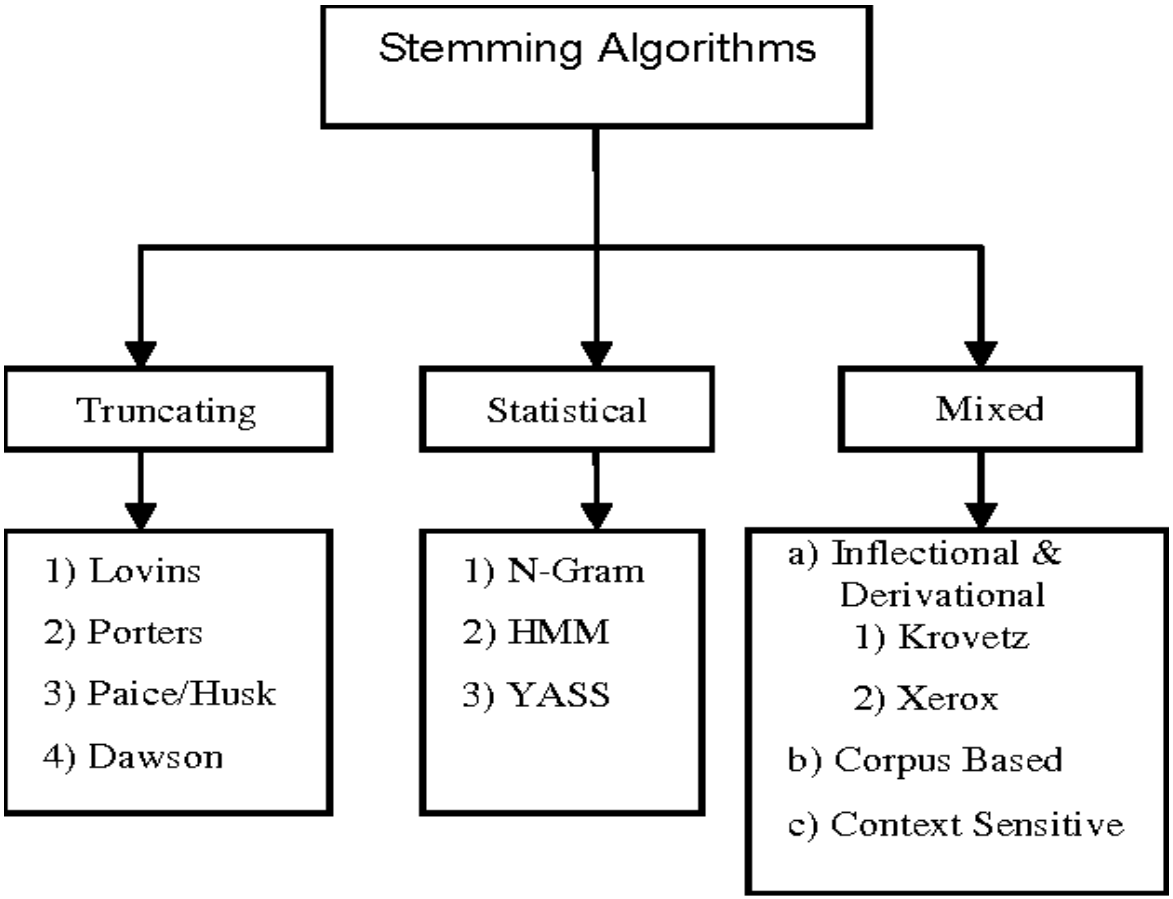
Lemmatisation et Stemming

- **Stemming** : quelques algorithmes et approches – Stemmers



Lemmatisation et Stemming

- **Stemming** : quelques algorithmes et approches – Stemmers



Lemmatisation et Stemming

- **Stemming** : **Porter's Stemmer algorithm**
- C'est l'un des algorithmes de stemming le plus populaire et le plus utilisé développé en 1980 par Martin Porter.
- Suffix removal algorithm – **Suffix** Stripping. Langue anglaise.
- Il est divisé en un certain nombre **d'étapes** linéaires qui sont utilisées pour produire le stemme final. **Rule based** algorithm.
- Une consonne est une lettre autre que A, E, I, O, U, et autre que Y précédé d'une consonne. Ex: Dans le mot 'TOY', les consonnes sont T and Y, et dans le mot 'SYZYGY' elles sont S, Z and G.
- Une voyelle est une lettre qui n'est pas une consonne.
- Une liste de consonnes supérieures ou égales à la longueur 1 sera désignée par un C et une liste similaire de voyelles par un V.

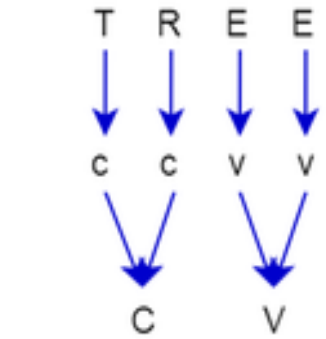
Lemmatisation et Stemming

- **Stemming** : **Porter's Stemmer algorithm**
- N'importe quel mot peut être représenté par la forme unique :

$$[C]\,(VC)^m\,[V]$$

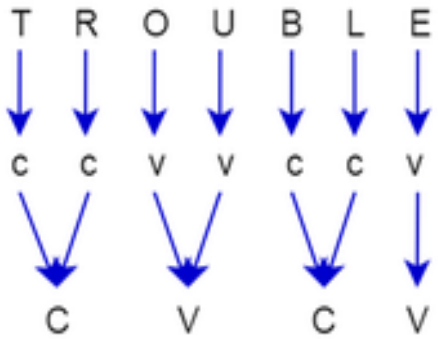
- **m** : désigne m répétitions de VC. La valeur **m** est appelée la mesure d'un mot et peut prendre toute valeur supérieure ou égale à zéro, et est utilisée pour décider si un suffixe donné doit être supprimé.
- **[]** dénotent la présence facultative de leur contenu.
- Les règles de suppression de suffixes sont de la forme **(condition) S1 -> S2.**
- Cela signifie que si un mot se termine par le suffixe S1 et que la racine avant S1 satisfait la condition donnée, S1 est remplacé par S2.

$[C]\ (VC)^m\ [V]$



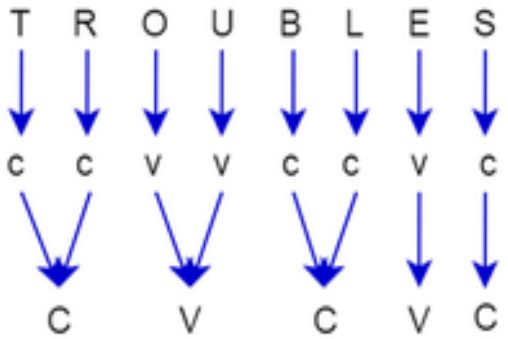
CV

$[C]\ (VC)^m\ [V] \longrightarrow m = 0$



C (VC) V

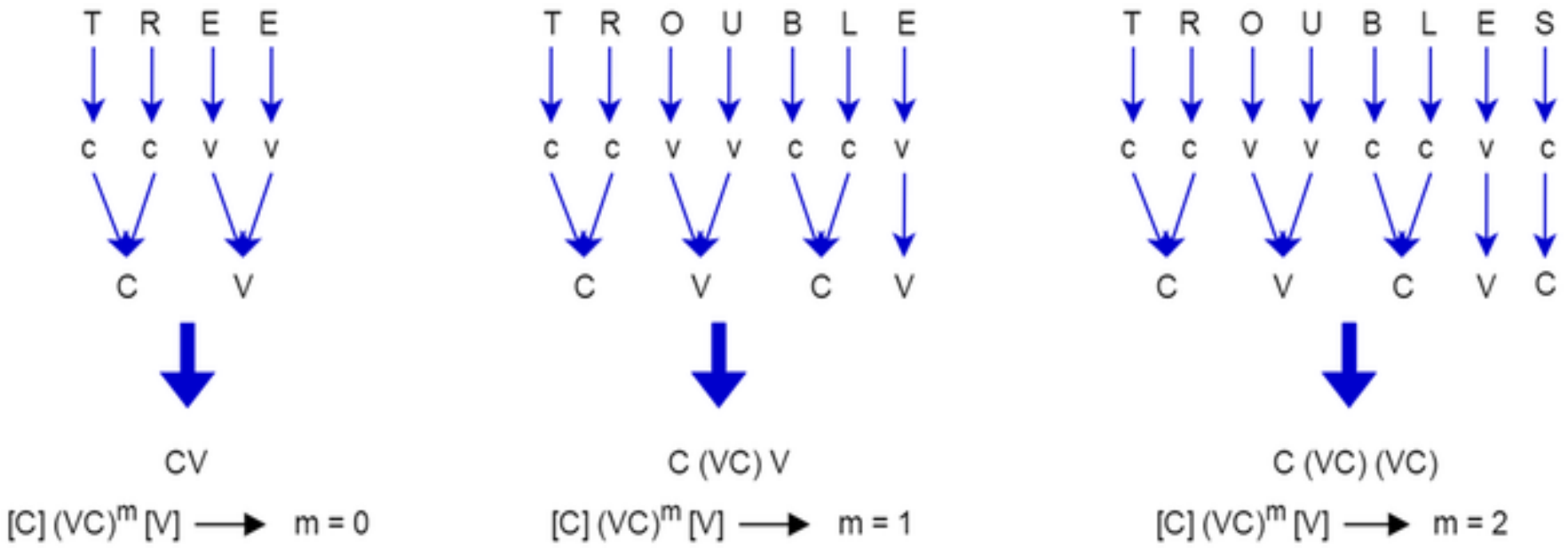
$[C]\ (VC)^m\ [V] \longrightarrow m = 1$



C (VC) (VC)

$[C]\ (VC)^m\ [V] \longrightarrow m = 2$

[C] (VC)^m [V]



Porter Stemming Algorithm

Règle : **(m> 1) EMENT →**
(S1 est «EMENT» et S2 est nul)

REPLACEMENT devient REPLAC, puisque m = 2 dans la racine avant S1.

(m>0) ATIONAL → ATE
(m>0) TIONAL → TION
(m>0) ENCI → ENCE
(m>0) ANCI → ANCE

Lemmatisation et Stemming

- **Stemming** : **Porter's Stemmer algorithm**

- Les **conditions** peuvent contenir : **(condition) S1 -> S2**

✓ *S - la racine se termine par S (et de même pour les autres lettres).
✓ * v* - la racine contient une voyelle.
✓ *d - la racine se termine par une double consonne (par exemple -TT, -SS).
✓ *o - la racine se termine par cvc, où le deuxième c n'est pas W, X ou Y (par exemple -WIL, -HOP).

- Et la partie condition peut également contenir des expressions: and, or, et not:

✓ (m> 1 and (*S or*T)) teste une racine avec m> 1 se terminant par S ou T.
✓ (*d and not (*L or*S or *Z)) teste une racine se terminant par une double consonne et ne se terminant pas par les lettres L, S ou Z.

- Dans un ensemble de **règles** écrites, une seule est respectée, et ce sera celle avec le plus long S1 correspondant pour le mot donné.

Porter's Stemmer algorithm - 5 Steps

Step 1a

1. SSES → SS
2. IES → I
3. SS → SS
4. S →

Step 1b

1. (m>0) EED → EE
2. (*v*) ED →
3. (*v*) ING →

If the second or third of the rules in Step 1b is successful, the following is performed.

1. AT → ATE
2. BL → BLE
3. IZ → IZE
4. (*d and not (*L or *S or *Z)) → single letter
5. (m=1 and *o) → E

Step 1c

1. (*v*) Y → I

Porter's Stemmer algorithm - 5 Steps

Step 2

1. (m>0) ATIONAL	→	ATE
2. (m>0) TIONAL	→	TION
3. (m>0) ENCI	→	ENCE
4. (m>0) ANCI	→	ANCE
5. (m>0) IZER	→	IZE
6. (m>0) ABLI	→	ABLE
7. (m>0) ALLI	→	AL
8. (m>0) ENTLI	→	ENT
9. (m>0) ELI	→	E
10. (m>0) OUSLI	→	OUS
11. (m>0) IZATION	→	IZE
12. (m>0) ATION	→	ATE
13. (m>0) ATOR	→	ATE
14. (m>0) ALISM	→	AL
15. (m>0) IVENESS	→	IVE
16. (m>0) FULNESS	→	FUL
17. (m>0) OUSNESS	→	OUS
18. (m>0) ALITI	→	AL
19. (m>0) IVITI	→	IVE
20. (m>0) BILITI	→	BLE

Porter's Stemmer algorithm - 5 Steps

Step 3

1. (m>0) ICATE	→	IC
2. (m>0) ATIVE	→	
3. (m>0) ALIZE	→	AL
4. (m>0) ICITI	→	IC
5. (m>0) ICAL	→	IC
6. (m>0) FUL	→	
7. (m>0) NESS	→	

Porter's Stemmer algorithm - 5 Steps

Step 4

1. (m>1) AL →
2. (m>1) ANCE →
3. (m>1) ENCE →
4. (m>1) ER →
5. (m>1) IC →
6. (m>1) ABLE →
7. (m>1) IBLE →
8. (m>1) ANT →
9. (m>1) EMENT →
10. (m>1) MENT →
11. (m>1) ENT →
12. (m>1 and (*S or *T)) ION →
13. (m>1) OU →
14. (m>1) ISM →
15. (m>1) ATE →
16. (m>1) ITI →
17. (m>1) OUS →
18. (m>1) IVE →
19. (m>1) IZE →

Step 5a

1. (m>1) E →
2. (m=1 and not *o) E →

Step 5b

1. (m > 1 and *d and *L) → single letter

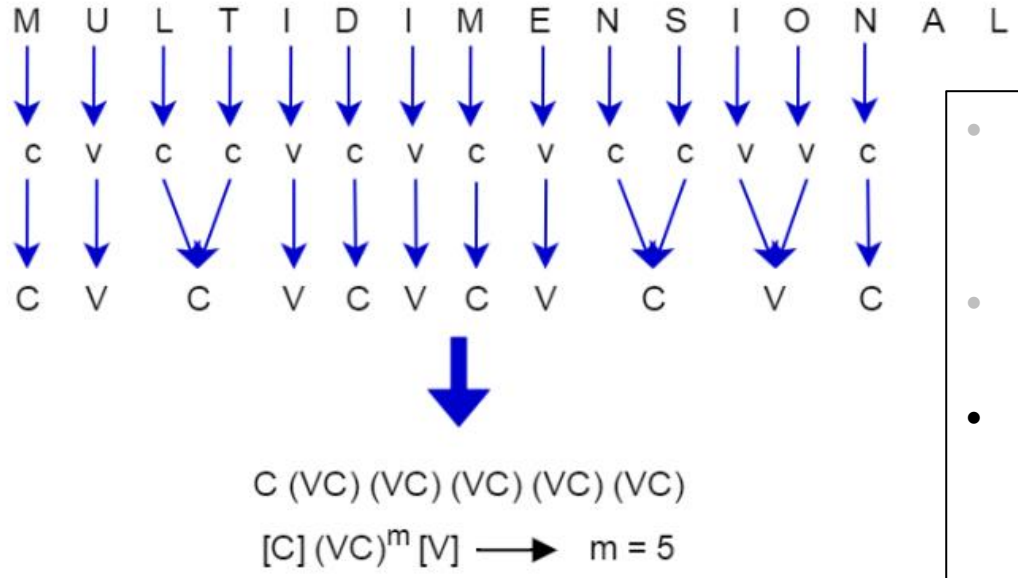
Porter’s Stemmer algorithm – 5 Steps

Exemple 1 : mot = **MULTIDIMENSIONAL**

- Le suffixe ne correspondra à aucun des cas trouvés aux étapes 1, 2 et 3.
- Ensuite, il s'agit de l'étape **4**.

Porter's Stemmer algorithm - 5 Steps

Exemple 1 : mot = **MULTIDIMENSIONAL**



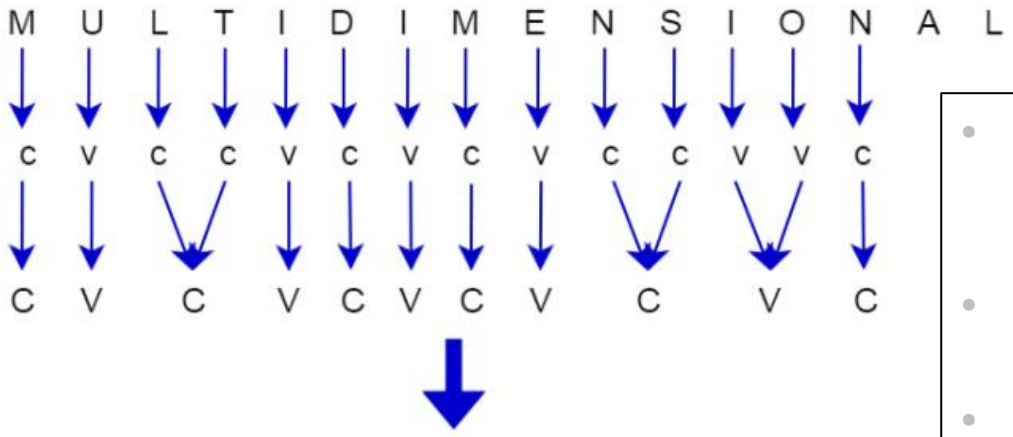
- Le suffixe ne correspondra à aucun des cas trouvés aux étapes 1, 2 et 3.
- Ensuite, il s'agit de l'étape **4**.
- La racine du mot a **m> 1** (puisque m = 5) et se termine par «**AL**».
- Par conséquent, à l'étape 4, «AL» est supprimé (remplacé par null).

Step 4

1. (m>1) AL →

Porter's Stemmer algorithm - 5 Steps

Exemple 1 : mot = **MULTIDIMENSIONAL**

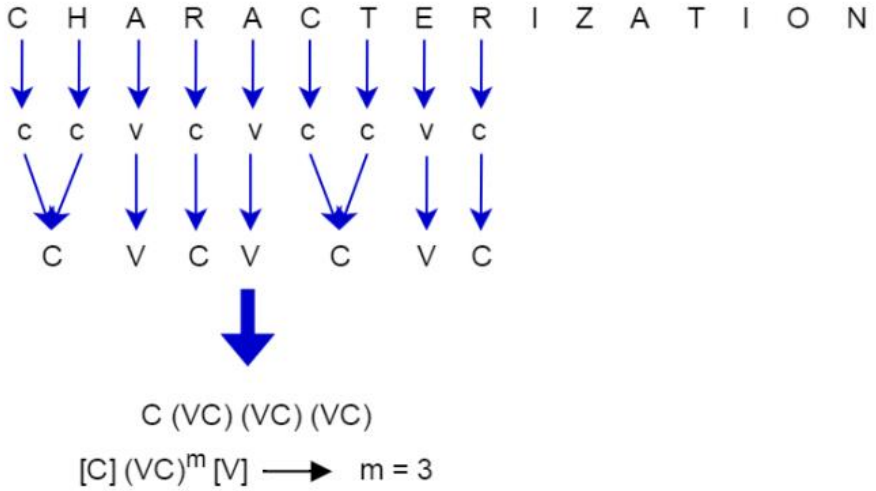


- Le suffixe ne correspondra à aucun des cas trouvés aux étapes 1, 2 et 3.
- Ensuite, il s'agit de l'étape **4**.
- La racine du mot a **m> 1** (puisque m = 5) et se termine par «**AL**».
- Par conséquent, à l'étape 4, «AL» est supprimé (remplacé par null).
- L'appel de l'étape 5 ne changera pas davantage la racine.

Enfin, la sortie sera :
MULTIDIMENSIONAL → MULTIDIMENSION

Porter's Stemmer algorithm - 5 Steps

Exemple 1 : mot = **CHARACTERIZATION**

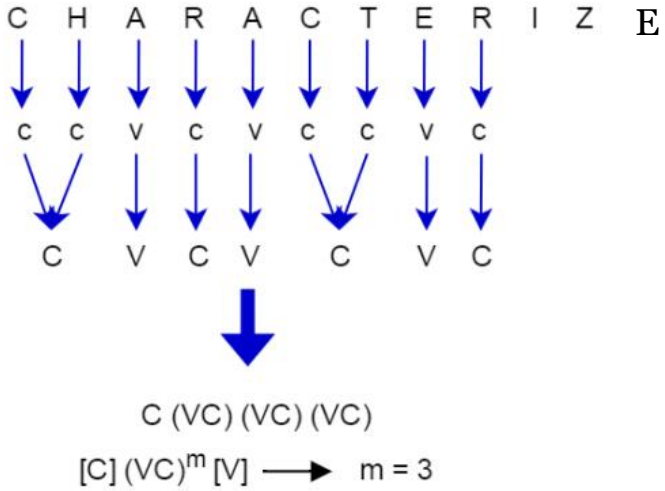


	11. (m>0) IZATION	→	IZE

- Le suffixe ne correspondra à aucun des cas trouvés à l'étape 1.
- Il passera donc à l'étape **2**.
- La racine du mot a **m> 0** (puisque m = 3) et se termine par «**IZATION**».
- Par conséquent, à l'étape 2, «IZATION» sera remplacé par «IZE».
- Ensuite, la nouvelle racine sera **CHARACTERIZE**.

Porter's Stemmer algorithm - 5 Steps

Exemple 1 : mot = ~~**CHARACTERIZATION**~~ **-> CHARACTERIZE**



19. (m>1) IZE →

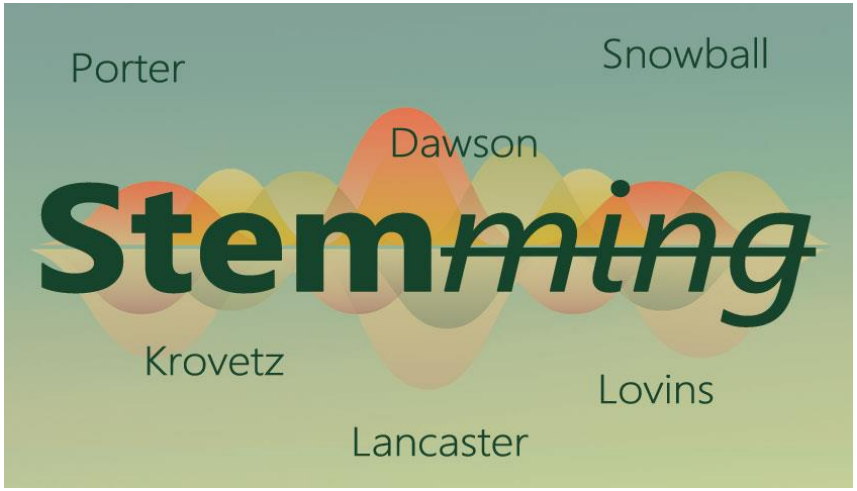
CHARACTER~~IZATION~~ ->

CHARACTER~~IZE~~ → CHARACTER

- Ensuite, la nouvelle racine sera CHARACTERIZE.
- L'étape 3 ne correspondra à aucun des suffixes et passera à l'étape **4**.
- Maintenant **m> 1** (puisque m = 3) et la racine se termine par «**IZE**».
- Ainsi, à l'étape 4, «IZE» sera supprimé (remplacé par null).
- Aucun changement ne se produira à la racine dans les autres étapes.
- Enfin, la sortie sera CHARACTER.

Lemmatisation et Stemming

- **Stemming** : quelques algorithmes et techniques – Stemmers
- **Liste** d'algorithmes: https://snowballstem.org/algorithms/
- **Demo** : https://snowballstem.org/demo.html
- **Demo** NLTK : https://text-processing.com/demo/stem/
- Porter Stemmer **Online** : https://9ol.es/porter_js_demo.html



Lemmatisation et Stemming

- **Lemmatization**: Quelques approches - Lemmatizers

- Deux principales approches :

- **Méthode basée sur des règles** – Rule Based: utilise un ensemble de règles qui indiquent comment un mot doit être modifié pour extraire son lemme. Exemple: si le mot est un verbe, et qu'il se termine par -ing, faire quelques substitutions… Cette méthode est très délicate et ne donne probablement pas les meilleurs résultats (difficile à généraliser en anglais, ex : bring).

- **Méthode basée sur le corpus** – Corpus Based: utilise un corpus taggé (ou un ensemble de données annoté) pour fournir le lemme de chaque mot. Fondamentalement, il s'agit d'une énorme liste de mots (lexicon) et de leur lemme associé pour chaque PoS (ou non, une approche naive). Ceci, bien sûr, nécessite l'accès à un corpus annoté.

Lemmatisation et Stemming

- **Lemmatization**: Quelques approches - Lemmatizers
- Il existe de nombreuses façons d'aider une machine à gérer les lemmes :

✓ Méthode Rule Based

✓ Méthode Copus Based

✓ **Neural lemmatizers** : utilise sequence-to-sequence (seq2seq) neural networks.

- Quelques Implémentations :
- WordNet Lemmatizer NLTK, SpaCy Lemmatizer, Gensim Lemmatizer, Stanford CoreNLP Lemmatization, TextBlog Lemmatizer, etc.
- Demo WordNet NLTK : http://textanalysisonline.com/nltk-wordnet-word-lemmatizer

Lemmatisation et Stemming

- **Lemmatization**: **WordNet Lemmatizer**

- **Wordnet** est une grande base de données lexicale, disponible gratuitement et publiquement pour la langue anglaise (et autres), visant à établir des relations sémantiques structurées entre les mots.

- Il offre également des capacités de lemmatisation et est l'un des lemmatiseurs les plus anciens et les plus couramment utilisés.

- **WordNet Lemmatizer** utilise la base de données WordNet pour rechercher (mapping) des lemmes de mots. Corpus based method.

- Doit aussi prendre en compte le **contexte** dans lequel on souhaite lemmatiser : Part-Of-Speech (POS). Car le même mot peut avoir plusieurs lemmes en fonction du sens / du contexte.

Lemmatisation et Stemming

- **Lemmatization**: **WordNet Lemmatizer**

Mapping from text-word to lemma
help (verb) ***help (noun), helping (noun)***

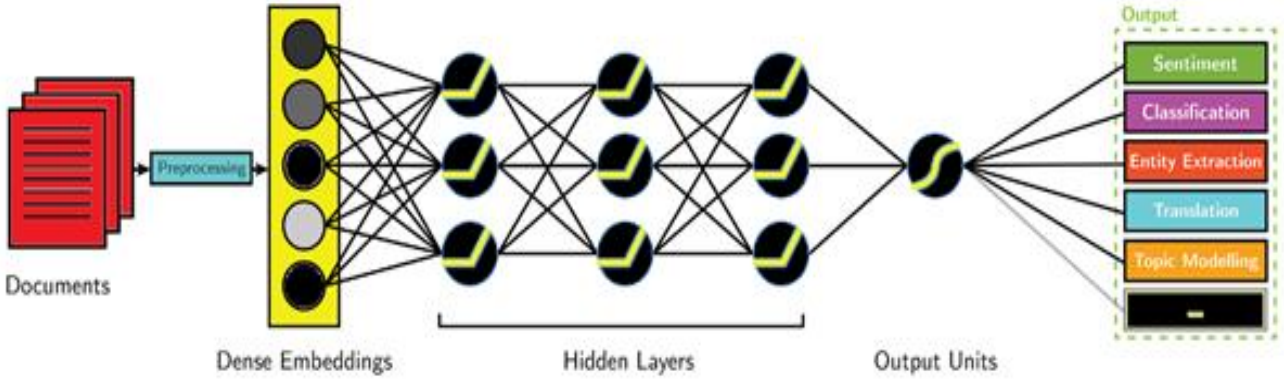
text-word	to	**lemma**
help		help (v), help (n)
helps		help (v), helps (n)**
helping		help (v), helping (n)
helped		help (v)
helpings		helping (n)

help (n):* usually a mass noun, but part of compound ***home help which is a count noun, taking the "s" ending.

Etiquetage Morpho-syntaxique

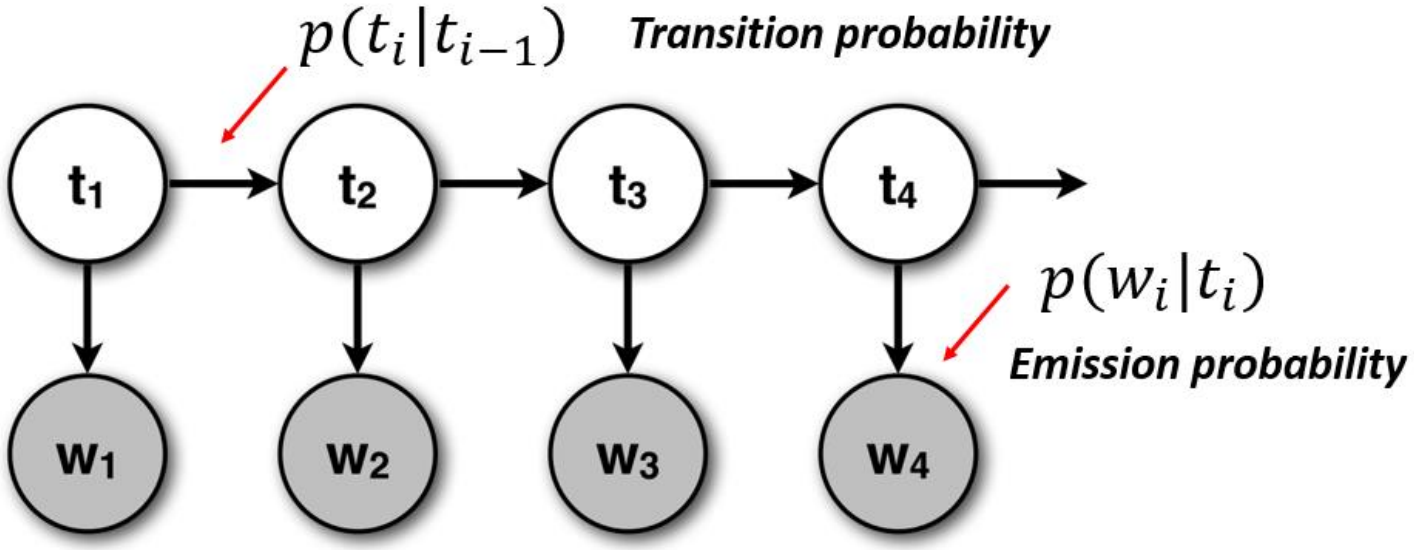
- **Comment** cette identification est-elle réalisée ?
- **Approches** :

- Règles manuelles
- Apprentissage automatique:
  - ✓ Apprendre les règles : Transformation-Based Learning (TBL)
  - ✓ **Modèles statistiques : Hidden Markov Models (HMM)**
  - ✓ Sur l'entropie maximale, champs aléatoires conditionnels (CRF)



HMM Part-of-Speech Tagging

- Hidden Markov Model (**HMM**) **POS Tagger**.
- Il s'agit d'une méthode **probabiliste** qui attribue des tags POS en fonction de la vraisemblance des mots et des probabilités de transition.
- La séquence POS la plus probable est prédite en fonction de la **vraisemblance** (Emission) des mots et des probabilités de **transition**.



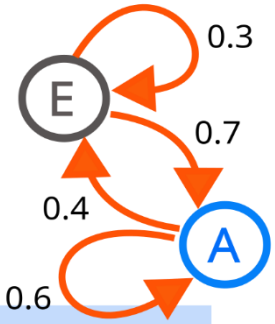
HMM Part-of-Speech Tagging

- HMM POS Tagger se base sur les chaînes de Markov.
- Une **chaîne de Markov** est un modèle qui nous renseigne sur les probabilités de séquences de variables aléatoires, appelées *états*.
- Une chaine de Markov est un processus stochastique possédant la propriété de Markov : l'information utile pour la prédiction du futur est entièrement contenue dans l'état présent du processus et n'est pas dépendante des états antérieurs (le système n'a pas de « mémoire »).
- Markov Assumption sur les probabilités d'une séquence : lors de la prédiction du futur, le passé n'a pas d'importance, seul le présent compte.

Markov Assumption: $P(q_i = a | q_1 \ldots q_{i-1}) = P(q_i = a | q_{i-1})$

HMM Part-of-Speech Tagging

- Les **états** sont représentés sous forme de nœuds dans le **graphe** et les transitions, avec leurs **probabilités**, sous forme d'arêtes.

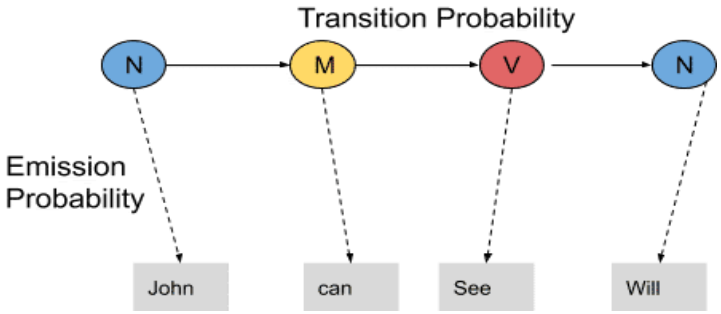


Formally, a Markov chain is specified by the following components:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{N1} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

HMM Part-of-Speech Tagging

- Une chaîne de Markov est utile lorsque nous devons calculer une probabilité pour une séquence d'événements **observables**. Ex: les mots.
- Dans de nombreux cas, cependant, les événements qui nous intéressent sont **cachés** : nous ne les observons pas directement. Ex: POS Tags.
- Nous **voyons plutôt des mots** et devons **déduire les POS Tags** à partir de la séquence de mots.
- Un modèle de Markov caché (HMM) nous permet de parler à la fois d'événements observés (comme les mots que nous voyons dans la séquence) et d'événements cachés (comme les POS Tags) que nous considérons comme des facteurs causaux dans notre modèle probabiliste.



HMM Part-of-Speech Tagging

- An HMM is specified by the following components:

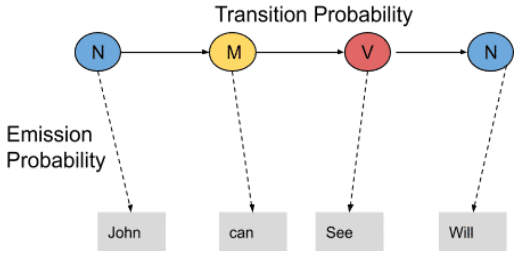
$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t (drawn from a vocabulary $V = v_1, v_2, ..., v_V$) being generated from a state q_i
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

The HMM is given as input $O = o_1 o_2 \ldots o_T$: a sequence of T **observations**, each one drawn from the vocabulary V.

HMM Part-of-Speech Tagging

- Here are the general **steps** involved in **HMM POS tagging**:

1. **Define the Problem**:
   - **Input**: A sequence of words $W = w_1, w_2, \ldots, w_n$.
   - **Output**: A sequence of POS tags $T = t_1, t_2, \ldots, t_n$ corresponding to each word in W.
2. **Model Components**:
   - **States**: The POS tags t_i are the hidden states.
   - **Observations**: The words w_i are the observed outputs.
   - **Transition Probabilities**: The probability of transitioning from one POS tag to another, $P(t_i|t_{i-1})$.
   - **Emission Probabilities**: The probability of a word being generated by a particular POS tag, $P(w_i|t_i)$.
   - **Initial Probabilities**: The probability of starting with a particular POS tag, $P(t_1)$.



HMM Part-of-Speech Tagging

- Here are the general **steps** involved in **HMM POS tagging**:

3. **Training the HMM**:
   - **Estimate Transition Probabilities**: Calculate $P(t_i|t_{i-1})$ from a tagged corpus by counting the frequency of tag transitions.
   - **Estimate Emission Probabilities**: Calculate $P(w_i|t_i)$ by counting how often a word w_i is tagged with t_i in the corpus.
   - **Estimate Initial Probabilities**: Calculate $P(t_1)$ by counting how often a sentence starts with each POS tag.

HMM Part-of-Speech Tagging

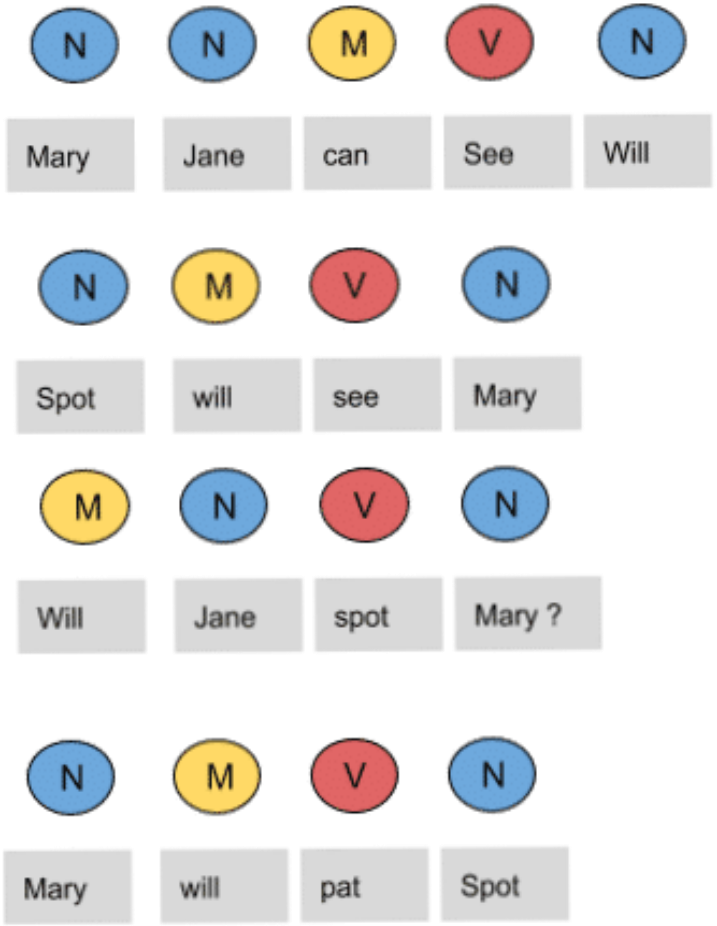
- Here are the general **steps** involved in **HMM POS tagging**:

4. **Decoding**:
   - **Objective**: Find the most likely sequence of POS tags T given the sequence of words W.
   - **Algorithm**: Use the Viterbi algorithm to efficiently compute the most likely sequence of hidden states (POS tags) given the observed sequence (words).
     - **Initialization**: Compute the initial probabilities for the first word.
     - **Recursion**: For each subsequent word, compute the maximum probability of reaching each state (POS tag) from the previous states, considering both transition and emission probabilities.
     - **Termination**: Identify the sequence of states that maximizes the overall probability.
5. **Output**:
   - The sequence of POS tags T that maximizes the probability $P(T|W)$.

HMM Part-of-Speech Tagging

- **Exemple - Corpus:**

1. Mary Jane can see Will
2. Spot will see Mary
3. Will Jane spot Mary?
4. Mary will pat Spot

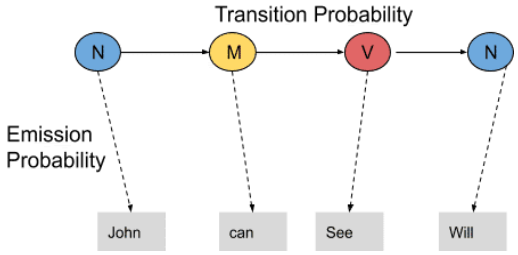


Source: https://www.mygreatlearning.com/blog/pos-tagging/

HMM Part-of-Speech Tagging

- Here are the general **steps** involved in **HMM POS tagging**:

1. **Define the Problem**:
   - **Input**: A sequence of words $W = w_1, w_2, \ldots, w_n$.
   - **Output**: A sequence of POS tags $T = t_1, t_2, \ldots, t_n$ corresponding to each word in W.
2. **Model Components**:
   - **States**: The POS tags t_i are the hidden states.
   - **Observations**: The words w_i are the observed outputs.
   - **Transition Probabilities**: The probability of transitioning from one POS tag to another, $P(t_i|t_{i-1})$.
   - **Emission Probabilities**: The probability of a word being generated by a particular POS tag, $P(w_i|t_i)$.
   - **Initial Probabilities**: The probability of starting with a particular POS tag, $P(t_1)$.



HMM Part-of-Speech Tagging

- **Exemple - Corpus:**

1. States (POS Tags)

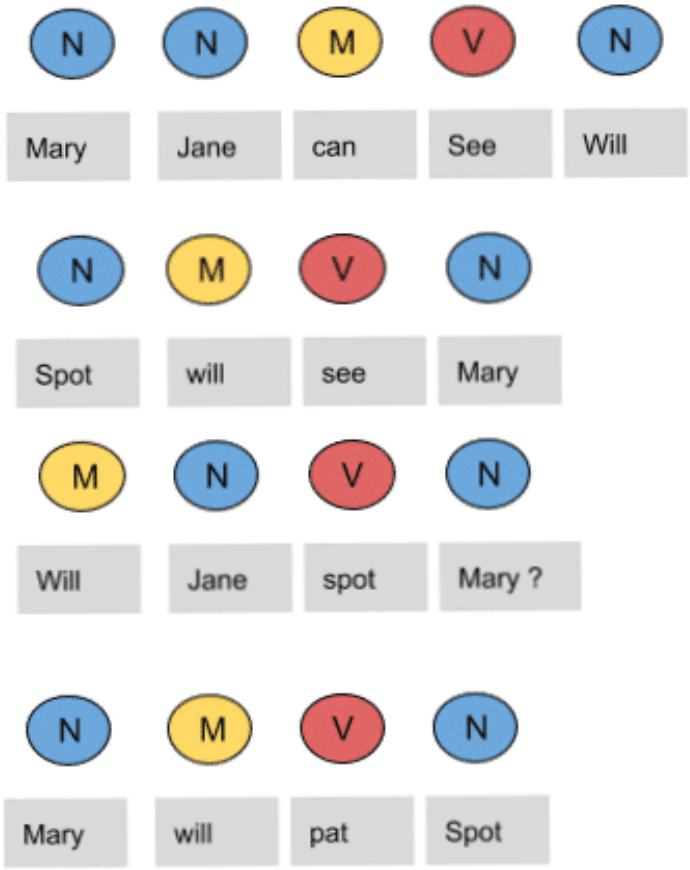
The possible parts of speech (**hidden states**) are:

- **Noun (N)**
- **Verb (V)**
- **Modal (M)** (for modal verbs like *can, will*)

2. Observations (Words in Sentences)

The given sentences contain the following words (**observations**):

- **Mary**
- **Jane**
- **Can**
- **See**
- **Will**
- **Spot**
- **Pat**



HMM Part-of-Speech Tagging

- Here are the general **steps** involved in **HMM POS tagging**:

3. **Training the HMM**:
   - **Estimate Transition Probabilities**: Calculate $P(t_i|t_{i-1})$ from a tagged corpus by counting the frequency of tag transitions.
   - **Estimate Emission Probabilities**: Calculate $P(w_i|t_i)$ by counting how often a word w_i is tagged with t_i in the corpus.
   - **Estimate Initial Probabilities**: Calculate $P(t_1)$ by counting how often a sentence starts with each POS tag.

HMM Part-of-Speech Tagging

- **Exemple - Corpus:**

Transition Probabilities:

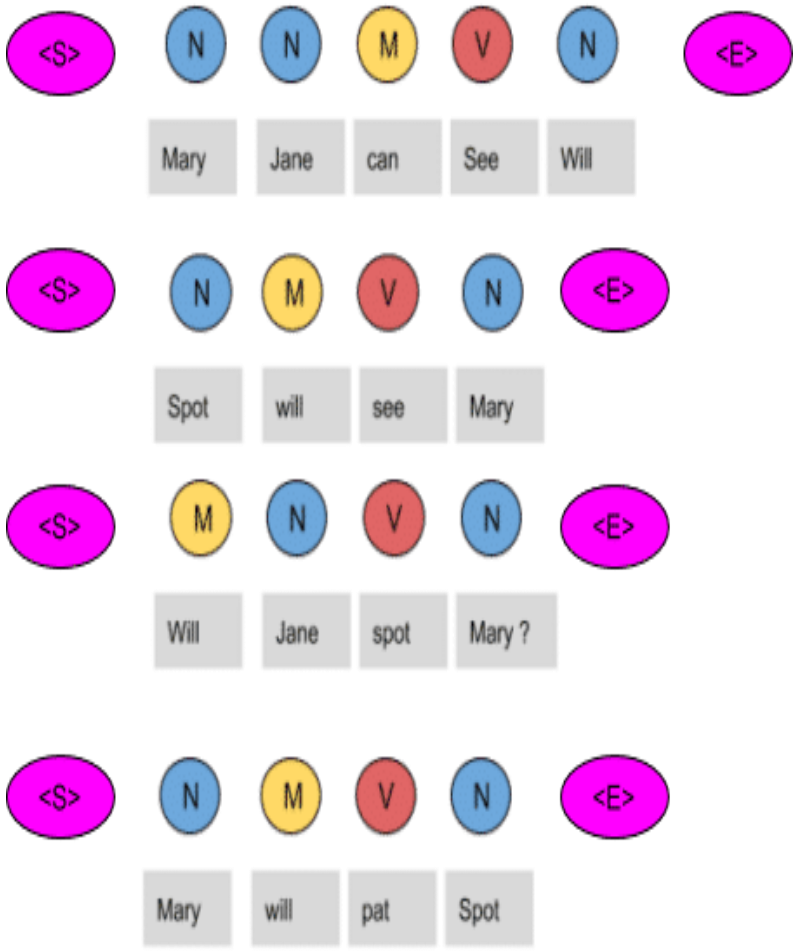
Count how often one POS tag follows another in the sentences.

t_{i-1}	$t_i = V$	$t_i = N$	$t_i = M$	$t_i =< E >$
$< S >$	$P(V \mid < S >)$	$P(N \mid < S >)$	$P(M \mid < S >)$	0
V	$P(V \mid V)$	$P(N \mid V)$	$P(M \mid V)$	$P(< E > \mid V)$
N	$P(V \mid N)$	$P(N \mid N)$	$P(M \mid N)$	$P(< E > \mid N)$
M	$P(V \mid M)$	$P(N \mid M)$	$P(M \mid M)$	$P(< E > \mid M)$

HMM Part-of-Speech Tagging

Transition Probabilities:

	N	M	V	<E>
<S>	3/4	1/4	0	0
N	1/9	3/9	1/9	4/9
M	1/4	0	3/4	0
V	4/4	0	0	0

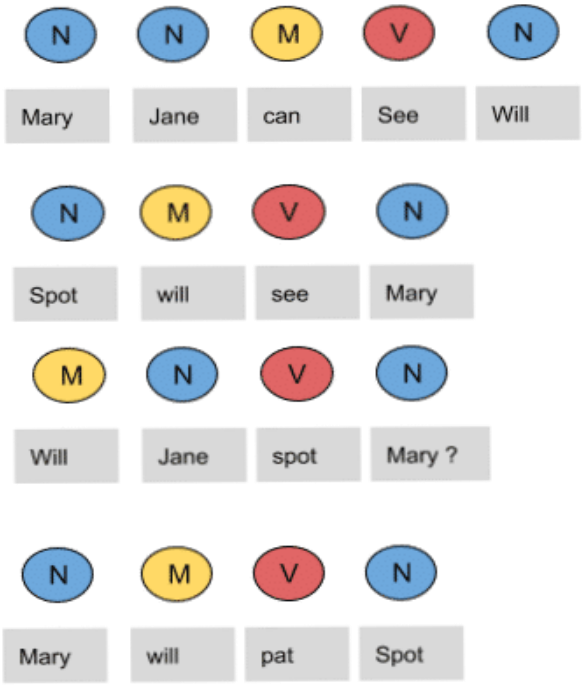


HMM Part-of-Speech Tagging

Emission Probabilities:

Count how often a word is associated with a specific POS tag.

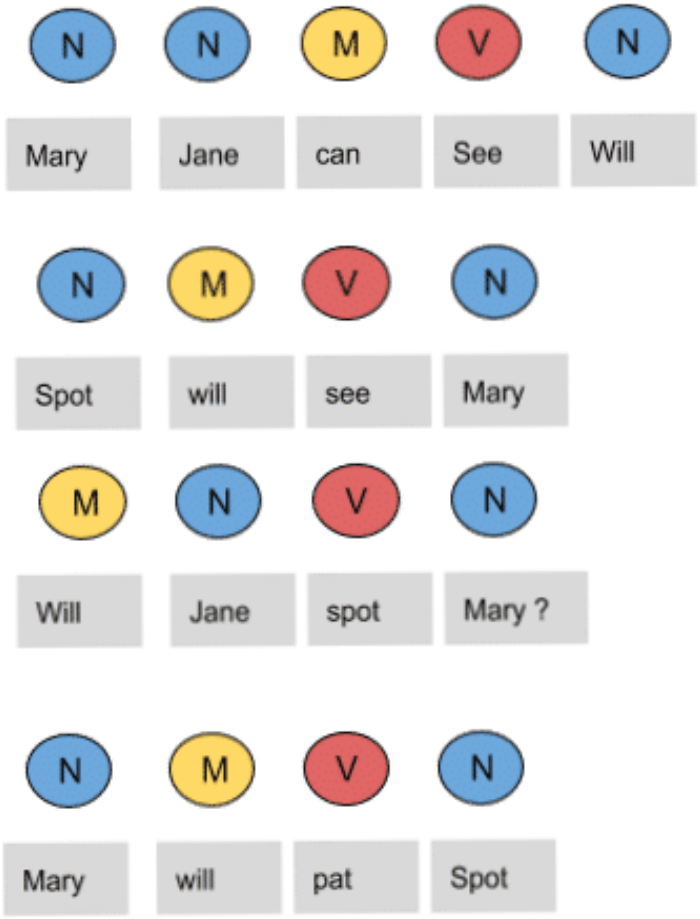
Word	$P(\text{Word} \mid V)$	$P(\text{Word} \mid N)$	$P(\text{Word} \mid M)$
Mary	$P(\text{Mary} \mid V)$	$P(\text{Mary} \mid N)$	$P(\text{Mary} \mid M)$
Jane	$P(\text{Jane} \mid V)$	$P(\text{Jane} \mid N)$	$P(\text{Jane} \mid M)$
can	$P(\text{can} \mid V)$	$P(\text{can} \mid N)$	$P(\text{can} \mid M)$
see	$P(\text{see} \mid V)$	$P(\text{see} \mid N)$	$P(\text{see} \mid M)$
Will	$P(\text{Will} \mid V)$	$P(\text{Will} \mid N)$	$P(\text{Will} \mid M)$
Spot	$P(\text{Spot} \mid V)$	$P(\text{Spot} \mid N)$	$P(\text{Spot} \mid M)$
will	$P(\text{will} \mid V)$	$P(\text{will} \mid N)$	$P(\text{will} \mid M)$
pat	$P(\text{pat} \mid V)$	$P(\text{pat} \mid N)$	$P(\text{pat} \mid M)$



HMM Part-of-Speech Tagging

Emission Probabilities:

Words	Noun	Model	Verb
Mary	4/9	0	0
Jane	2/9	0	0
Will	1/9	3/4	0
Spot	2/9	0	1/4
Can	0	1/4	0
See	0	0	2/4
pat	0	0	1



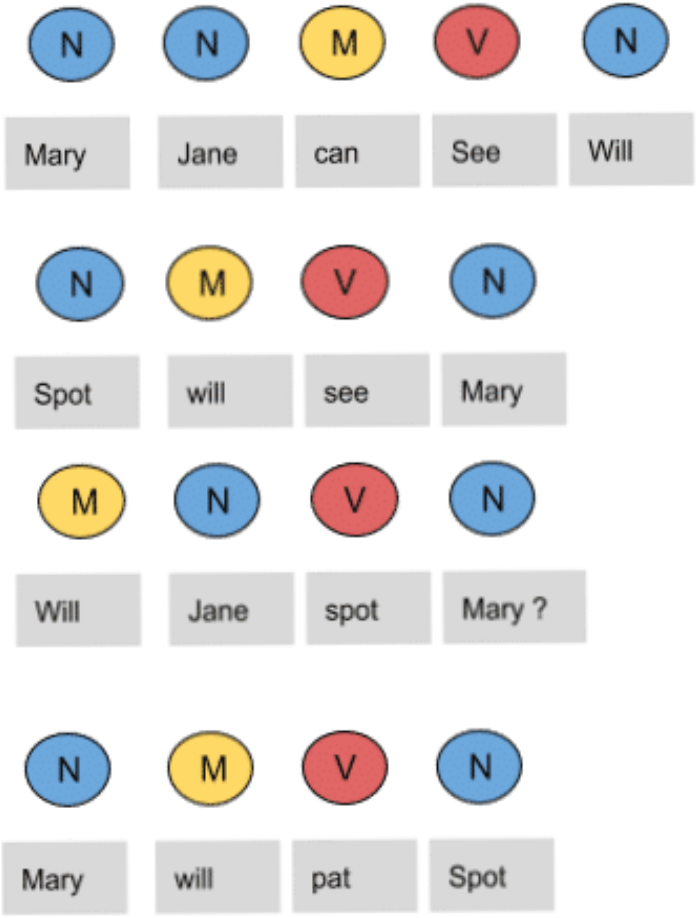
HMM Part-of-Speech Tagging

Initial Probabilities:

Count how often a sentence starts with each POS tag.

POS Tag	$P(t_1)$
V	$P(V)$
N	$P(N)$
M	$P(M)$

POS Tag	$P(t_1)$
V	0
N	$\frac{3}{4}$
M	$\frac{1}{4}$



HMM Part-of-Speech Tagging

- Here are the general **steps** involved in **HMM POS tagging**:
  - **Decoding**: the task of determining the hidden variables sequence corresponding to the sequence of observations

4. **Decoding**:
   - **Objective**: Find the most likely sequence of POS tags T given the sequence of words W.
   - **Algorithm**: Use the Viterbi algorithm to efficiently compute the most likely sequence of hidden states (POS tags) given the observed sequence (words).
     - **Initialization**: Compute the initial probabilities for the first word.
     - **Recursion**: For each subsequent word, compute the maximum probability of reaching each state (POS tag) from the previous states, considering both transition and emission probabilities.
     - **Termination**: Identify the sequence of states that maximizes the overall probability.
5. **Output**:
   - The sequence of POS tags T that maximizes the probability $P(T|W)$.

HMM Part-of-Speech Tagging

- **Decoding** to POS Tag the sentence : **Will can spot Mary**

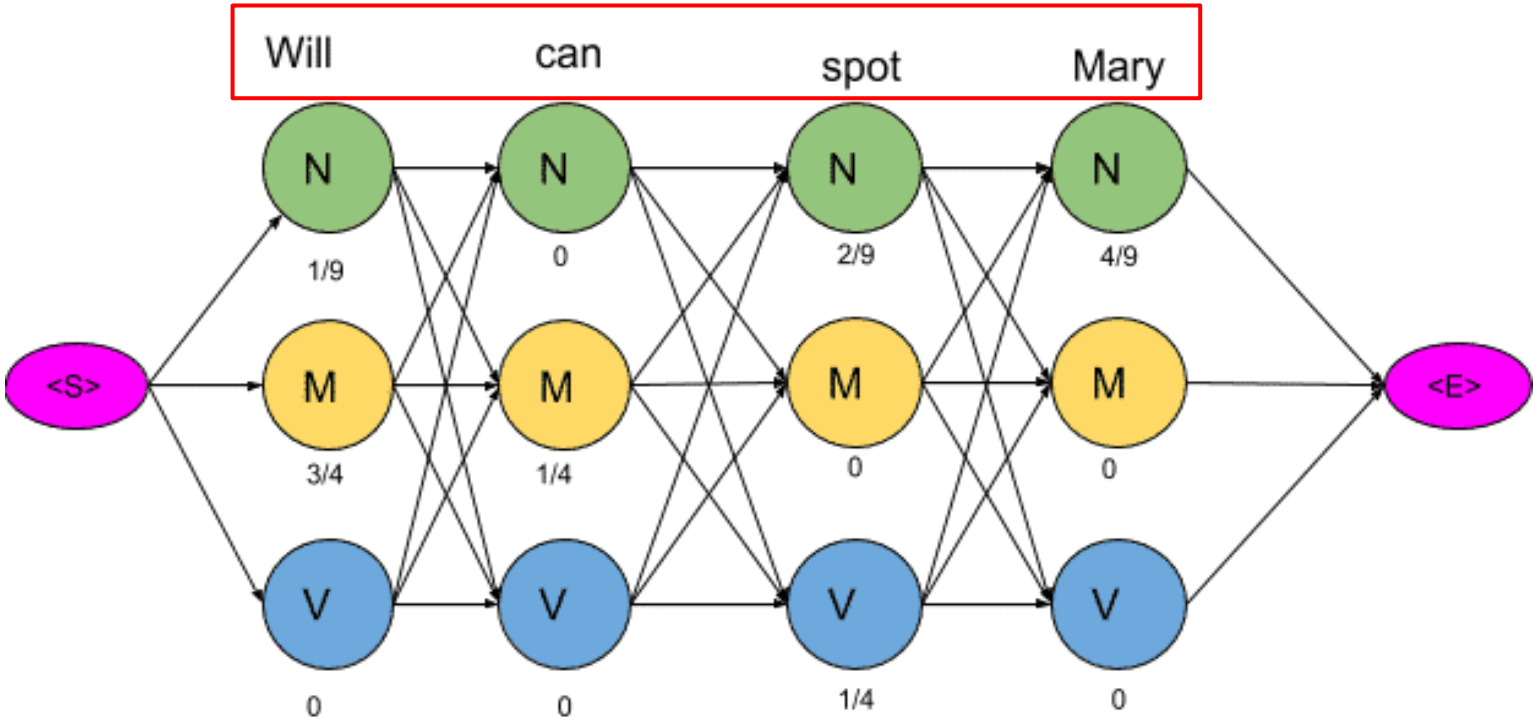
For part-of-speech tagging, the goal of HMM decoding is to choose the tag sequence $t_1 \ldots t_n$ that is most probable given the observation sequence of n words $w_1 \ldots w_n$:

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(t_1 \ldots t_n | w_1 \ldots w_n)$$

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(t_1 \ldots t_n | w_1 \ldots w_n) \approx \underset{t_1 \ldots t_n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{\text{emission}} \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

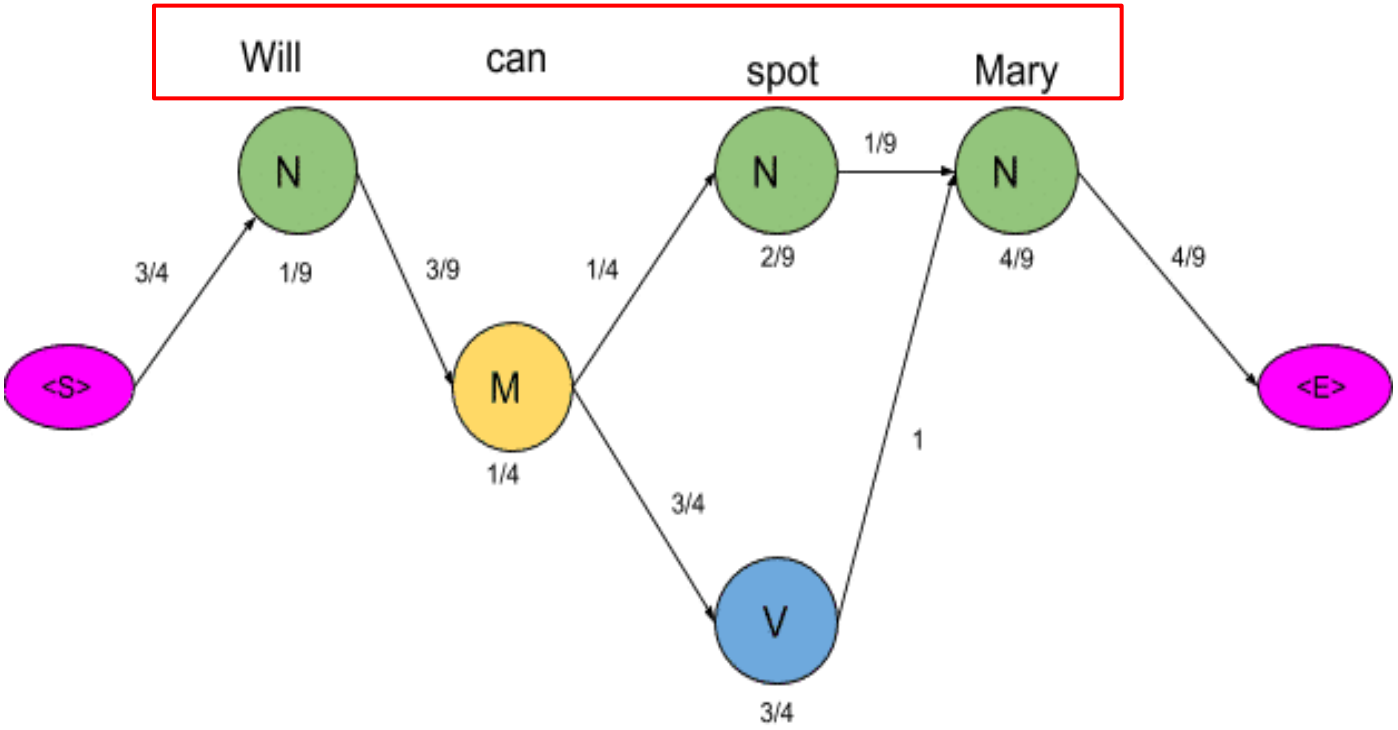
HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Brute-force: 81 combinations as paths and using the **emission** and **transition** probability mark each **node** and **edge** as shown below.



HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Possible solution: delete all the nodes and edges with probability zero, also the nodes which do not lead to the endpoint.



HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi*$[N,T]$
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
$bestpath \leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Figure 17.10 Viterbi algorithm for finding the optimal sequence of tags. Given an observation sequence and

In the Viterbi Algorithm, at each step, we compute probabilities **for all possible previous states**, but we only keep the **maximum probability for each current state**.

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

How it works:

1. For each word in the sequence, we consider all possible tags (states) it could have.
2. We compute the probability of reaching each tag by considering **all possible previous tags**.
3. We multiply:
   - The probability of being in the previous tag (computed in the last step).
   - The **transition probability** from the previous tag to the current tag.
   - The **emission probability** of the current word given the current tag.
4. **We take the maximum probability** among all possible previous tags and store the best previous tag for backtracking.

- We calculate probabilities for all possible previous states.
- Then, we only keep the maximum probability for each tag.
- This ensures that at the end, we have the most likely sequence of POS tags.

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

From the emission probabilities table:

Word	$P(\text{Word} \mid V)$	$P(\text{Word} \mid N)$	$P(\text{Word} \mid M)$
Will	0	$\frac{1}{9}$	$\frac{3}{4}$

Step 1: Initialization (First Word: "Will")

- Possible tags: N (Noun), M (Modal).
- Compute $\delta_1(N) = P(N \mid < S >) \cdot P(\text{Will} \mid N)$.

$$\delta_1(N) = \frac{3}{4} \cdot \frac{1}{9} = \frac{3}{36} = \frac{1}{12}$$

- Compute $\delta_1(M) = P(M \mid < S >) \cdot P(\text{Will} \mid M)$.

$$\delta_1(M) = \frac{1}{4} \cdot \frac{3}{4} = \frac{3}{16}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

$$\overbrace{P(w_i|t_i)}^{\text{emission}}\ \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

Step 1: Initialization (First Word: "Will")

- Possible tags: N (Noun), M (Modal).
- Compute $\delta_1(N) = P(N \mid < S >) \cdot P(\text{Will} \mid N)$.

$$\delta_1(N) = \frac{3}{4} \cdot \frac{1}{9} = \frac{3}{36} = \frac{1}{12}$$

- Compute $\delta_1(M) = P(M \mid < S >) \cdot P(\text{Will} \mid M)$.

$$\delta_1(M) = \frac{1}{4} \cdot \frac{3}{4} = \frac{3}{16}$$

HMM Part-of-Speech Tagging

Transition Probabilities:

	N t (i)	M t (i)	V t (i)	<E>
<S> t (i-1)	3/4	1/4	0	0
N	1/9	3/9	1/9	4/9
M	1/4	0	3/4	0
V	4/4	0	0	0

$$P(N \mid < S >)$$

δ

$$: P(M \mid < S >)$$

$$\overbrace{P(t_i \mid t_{i-1})}^{\text{transition}}$$

HMM Part-of-Speech Tagging

Emission Probabilities:

Words	Noun	Model	Verb
Mary	4/9	0	0
Jane	2/9	0	0
Will	1/9	3/4	0
Spot	2/9	0	1/4
Can	0	1/4	0
See	0	0	2/4
pat	0	0	1

$\cdot P(\text{Will} \mid N).$

$\cdot P(\text{Will} \mid M).$

$\cdot \overbrace{P(w_i|t_i)}^{\text{emission}}$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

$$\overbrace{P(w_i|t_i)}^{\text{emission}}\ \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

Step 1: Initialization (First Word: "Will")

- Possible tags: N (Noun), M (Modal).
- Compute $\delta_1(N) = P(N \mid < S >) \cdot P(\text{Will} \mid N)$.

$$\delta_1(N) = \frac{3}{4} \cdot \frac{1}{9} = \frac{3}{36} = \frac{1}{12}$$

- Compute $\delta_1(M) = P(M \mid < S >) \cdot P(\text{Will} \mid M)$.

$$\delta_1(M) = \frac{1}{4} \cdot \frac{3}{4} = \frac{3}{16}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

$$\overbrace{P(w_i|t_i)}^{\text{emission}}\ \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

Step 1: Initialization (First Word: "Will")

- Possible tags: N (Noun), M (Modal).
- Compute $\delta_1(N) = P(N \mid < S >) \cdot P(\text{Will} \mid N)$.

$$\delta_1(N) = \frac{3}{4} \cdot \frac{1}{9} = \frac{3}{36} = \frac{1}{12}$$

- Compute $\delta_1(M) = P(M \mid < S >) \cdot P(\text{Will} \mid M)$.

$$\delta_1(M) = \frac{1}{4} \cdot \frac{3}{4} = \frac{3}{16}$$

- **Store the best probabilities**:
  - $\delta_1(N) = \frac{1}{12}$
  - $\delta_1(M) = \frac{3}{16}$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

Step 2: Recursion (Second Word: "can")

Can	0	1/4	0

- Possible tags: M (Modal).
- From N:

$$P(M, \text{can} \mid N) = \delta_1(N) \cdot P(M \mid N) \cdot P(\text{can} \mid M) = \frac{1}{12} \cdot \frac{3}{9} \cdot \frac{1}{4} = \frac{3}{432} = \frac{1}{144}$$

- From M:

$$P(M, \text{can} \mid M) = \delta_1(M) \cdot P(M \mid M) \cdot P(\text{can} \mid M) = \frac{3}{16} \cdot 0 \cdot \frac{1}{4} = 0$$

- Store the best probability:

$$\delta_2(M) = \max\left(\frac{1}{144}, 0\right) = \frac{1}{144}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

Step 2: Recursion (Second Word: "can")

$$\overbrace{P(w_i|t_i)}^{\text{emission}}\ \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

- Possible tags: M (Modal).
- From N:

$$P(M, \text{can} \mid N) = \delta_1(N) \cdot P(M \mid N) \cdot P(\text{can} \mid M) = \frac{1}{12} \cdot \frac{3}{9} \cdot \frac{1}{4} = \frac{3}{432} = \frac{1}{144}$$

- From M:

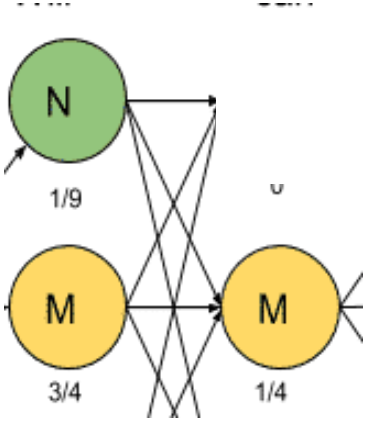
$$P(M, \text{can} \mid M) = \delta_1(M) \cdot P(M \mid M) \cdot P(\text{can} \mid M) = \frac{3}{16} \cdot 0 \cdot \frac{1}{4} = 0$$

- Store the best probability:

$$\delta_2(M) = \max\left(\frac{1}{144}, 0\right) = \frac{1}{144}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**



Step 2: Recursion (Second Word: "can")

- Possible tags: M (Modal).
- From N:

$$P(M, \text{can} \mid N) = \delta_1(N) \cdot P(M \mid N) \cdot P(\text{can} \mid M) = \frac{1}{12} \cdot \frac{3}{9} \cdot \frac{1}{4} = \frac{3}{432} = \frac{1}{144}$$

- From M:

$$P(M, \text{can} \mid M) = \delta_1(M) \cdot P(M \mid M) \cdot P(\text{can} \mid M) = \frac{3}{16} \cdot 0 \cdot \frac{1}{4} = 0$$

- Store the best probability:

$$\delta_2(M) = \max\left(\frac{1}{144}, 0\right) = \frac{1}{144}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

Step 2: Recursion (Second Word: "can")

$$\delta_t(s) = \max_{s'}\left(\delta_{t-1}(s') \times P_{\text{trans}}(s' \to s)\right) \times P_{\text{emit}}(\text{word}|s)$$

- Possible tags: M (Modal).
- From N:

$$P(M, \text{can} \mid N) = \delta_1(N) \cdot P(M \mid N) \cdot P(\text{can} \mid M) = \frac{1}{12} \cdot \frac{3}{9} \cdot \frac{1}{4} = \frac{3}{432} = \frac{1}{144}$$

- From M:

$$P(M, \text{can} \mid M) = \delta_1(M) \cdot P(M \mid M) \cdot P(\text{can} \mid M) = \frac{3}{16} \cdot 0 \cdot \frac{1}{4} = 0$$

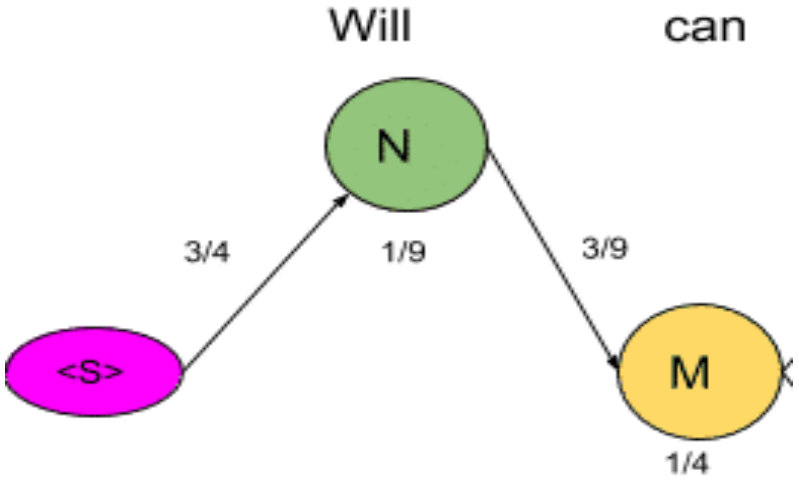
- Store the best probability:

$$\delta_2(M) = \max\left(\frac{1}{144}, 0\right) = \frac{1}{144}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

- Possible tags: M (Modal).
- From N:



$$P(M, \text{can} \mid N) = \delta_1(N) \cdot P(M \mid N) \cdot P(\text{can} \mid M) = \frac{1}{12} \cdot \frac{3}{9} \cdot \frac{1}{4} = \frac{3}{432} = \frac{1}{144}$$

$$\left(\delta_{t-1}(s'\right) \overbrace{P(t_i|t_{i-1})}^{\text{transition}} \overbrace{P(w_i|t_i)}^{\text{emission}}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

Step 2: Recursion (Second Word: "can")

- Possible tags: M (Modal).
- From N:

$$P(M, \text{can} \mid N) = \delta_1(N) \cdot P(M \mid N) \cdot P(\text{can} \mid M) = \frac{1}{12} \cdot \frac{3}{9} \cdot \frac{1}{4} = \frac{3}{432} = \frac{1}{144}$$

- From M:

$$P(M, \text{can} \mid M) = \delta_1(M) \cdot P(M \mid M) \cdot P(\text{can} \mid M) = \frac{3}{16} \cdot 0 \cdot \frac{1}{4} = 0$$

- Store the best probability:

$$\delta_2(M) = \max\left(\frac{1}{144}, 0\right) = \frac{1}{144}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

Step 3: Recursion (Third Word: "spot")

Spot	2/9	0	1/4

- Possible tags: N (Noun), V (Verb).
- From M:

$$P(N, \text{spot} \mid M) = \delta_2(M) \cdot P(N \mid M) \cdot P(\text{spot} \mid N) = \frac{1}{144} \cdot \frac{1}{4} \cdot \frac{2}{9} = \frac{2}{5184} = \frac{1}{2592}$$

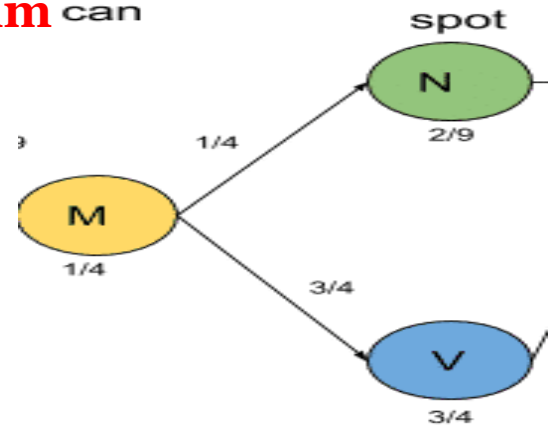
$$P(V, \text{spot} \mid M) = \delta_2(M) \cdot P(V \mid M) \cdot P(\text{spot} \mid V) = \frac{1}{144} \cdot \frac{3}{4} \cdot \frac{1}{4} = \frac{3}{2304} = \frac{1}{768}$$

- Store the best probabilities:

$$\delta_3(N) = \frac{1}{2592}, \quad \delta_3(V) = \frac{1}{768}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**



Step 3: Recursion (Third Word: "spot")

- Possible tags: N (Noun), V (Verb).
- From M:

$$P(N, \text{spot} \mid M) = \delta_2(M) \cdot P(N \mid M) \cdot P(\text{spot} \mid N) = \frac{1}{144} \cdot \frac{1}{4} \cdot \frac{2}{9} = \frac{2}{5184} = \frac{1}{2592}$$

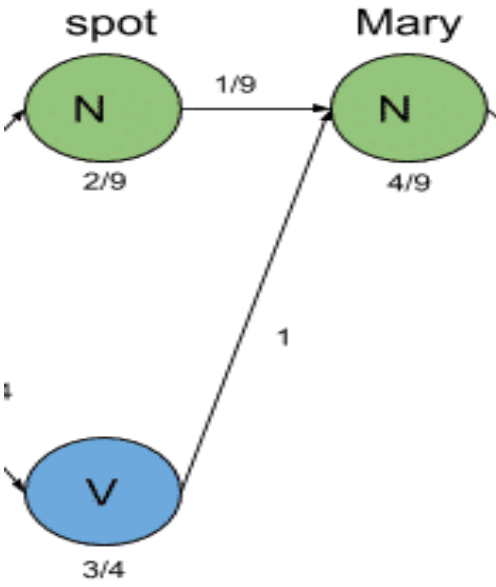
$$P(V, \text{spot} \mid M) = \delta_2(M) \cdot P(V \mid M) \cdot P(\text{spot} \mid V) = \frac{1}{144} \cdot \frac{3}{4} \cdot \frac{1}{4} = \frac{3}{2304} = \frac{1}{768}$$

- Store the best probabilities:

$$\delta_3(N) = \frac{1}{2592}, \quad \delta_3(V) = \frac{1}{768}$$

HMM Part-of-Speech Tagging

- Decoding to POS Tag : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**



Step 4: Recursion (Fourth Word: "Mary")

- Possible tags: N (Noun).
- From N:

$$P(N, \text{Mary} \mid N) = \delta_3(N) \cdot P(N \mid N) \cdot P(\text{Mary} \mid N) = \frac{1}{2592} \cdot \frac{1}{9} \cdot \frac{4}{9} = \frac{4}{209952} = \frac{1}{52488}$$

- From V:

$$P(N, \text{Mary} \mid V) = \delta_3(V) \cdot P(N \mid V) \cdot P(\text{Mary} \mid N) = \frac{1}{768} \cdot 1 \cdot \frac{4}{9} = \frac{4}{6912} = \frac{1}{1728}$$

- Store the best probability:

$$\delta_4(N) = \max\left(\frac{1}{52488}, \frac{1}{1728}\right) = \frac{1}{1728}$$

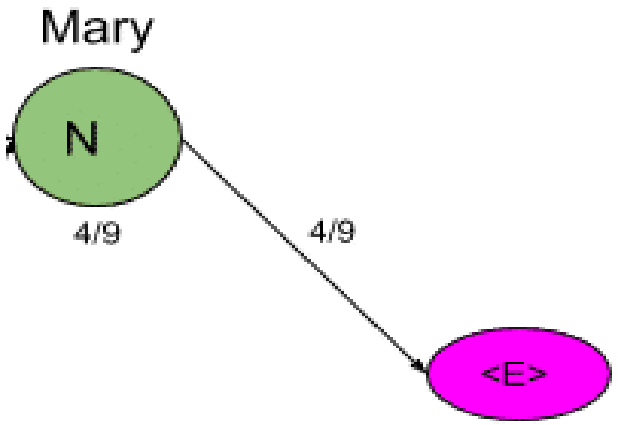
HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

Step 5: Termination

We now transition to <E>:

$$P(< E >) = P(N) \times P(N \rightarrow < E >) = \frac{1}{1728} \times \frac{4}{9}$$



HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

Step 5: Termination

- The most likely sequence ends with N (Mary).
- Backtrack to find the most likely sequence of POS tags
  - $\text{Mary} \rightarrow N$
  - $\text{spot} \rightarrow V$ (since $\delta_3(V) > \delta_3(N)$)
  - $\text{can} \rightarrow M$
  - $\text{Will} \rightarrow M$ (since $\delta_1(M) > \delta_1(N)$)

Summary of Probabilities:

- $\delta_1(N) = \frac{1}{12}$
- $\delta_1(M) = \frac{3}{16}$
- $\delta_2(M) = \frac{1}{144}$
- $\delta_3(N) = \frac{1}{2592}$
- $\delta_3(V) = \frac{1}{768}$
- $\delta_4(N) = \frac{1}{1728}$

HMM Part-of-Speech Tagging

- Decoding to POS Tag the sentence : **Will can spot Mary**
- Solution: **Optimizing HMM with Viterbi Algorithm**

Best POS Tag Sequence:

1. Will → Modal (M)
2. Can → Modal (M)
3. Spot → Verb (V)
4. Mary → Noun (N)

The highest probability sequence is:

M → M → V → N, meaning "**Will can spot Mary**" is tagged as:

> Will/M Can/M Spot/V Mary/N

Références

Speech and Language Processing - Livre de Dan Jurafsk - https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

Cours - *François Yvon* – Une petite introduction au Traitement Automatique des Langues Naturelles, https://perso.limsi.fr/anne/coursM2R/intro.pdf

Article – Marcel Cori - Des méthodes de traitement automatique aux linguistiques fondées sur les corpus
- https://www.cairn.info/revue-langages-2008-3-page-95.htm

Article - Pascale Sébillot - Le traitement automatique des langues face aux données textuelles volumineuses et potentiellement dégradées : qu'est-ce que cela change ?
- https://hal.archives-ouvertes.fr/hal-01056396/document

Cours – Morphologie - https://elc.hypotheses.org/155

A Comparative Study of Stemming Algorithms - A. Jivani, 2011.

Porter Stemming Algorithm - https://vijinimallawaarachchi.com/2017/05/09/porter-stemming-algorithm/