CHAPITRE 9

COHÉRENCE DU DISCOURS

LES LANGUES

En lisant un texte, nous ne devons pas sentir une interruption dans l'enchaînement des idées. On ne doit pas sauter d'une idée à une autre dans le même paragraphe. On appelle ça : la cohérence ; un texte cohérent doit garder la même idée ou passer à une autre idée en gardant une sorte de continuité. L'analyse de la cohérence nous permet de vérifier qu'un texte est facile à lire et à comprendre. Plusieurs théories ont été proposées pour représenter les relations de cohérence et donc permettre son analyse. Dans ce chapitre, on va présenter quelques théories de cohérence ainsi que des méthodes pour l'analyser.

Prenons le texte "L'hiver est un des quatre saisons de l'année. Le rapport était bien exposé. La pluie est une des caractéristiques de cette saison. J'ai lu un rapport sur cette saison.". On peut comprendre chaque phrase à part ; mais, il est difficile de comprendre l'idée du texte. Il est clair que les phrases sont mal-ordonnées. Donc, comprendre les phrases à part n'est pas une condition suffisante afin de comprendre un discours ; il faut que ce dernier soit cohérent. L'analyse de la cohérence peut aider dans plusieurs tâches :

- Génération du texte : lorsqu'on génère un texte automatiquement, il faut que les phrases soient cohérentes. Un exemple plus simple est la génération des résumés automatiques par extraction. Une fois les phrases importantes sont sélectionnées, on doit les réordonner pour avoir un résumé cohérent. Un exemple d'une méthode pour ordonner les phrases dans le contexte du résumé automatique est proposé par Oufaida et al. (2019).
- Compréhension du texte : comme illustré par l'exemple précédent, on ne peut pas bien comprendre un texte que lorsqu'il est cohérent.
- Évaluation du texte : on peut utiliser l'analyse de la cohérence afin d'évaluer automatiquement les expressions écrites des enfants. Aussi, on peut l'utiliser comme outil d'aide de rédaction (Personnellement, je n'ai pas vu un tel outil ; mais il va être d'une grande utilité pour les écrivains).
- Détection de plagiat : lorsqu'on copie et colle des parties de plusieurs origines, on va avoir un texte incohérent.

1 Relations de cohérence

En général, les phrases proches se partagent quelques mots ou sens. Ce phénomène est appelé "**Cohésion lexicale**" où la relation entre deux phrases consécutives est la similarité lexicale ou sémantique. Une autre façon de voir la cohérence est en se basant sur le sujet du texte ; les phrases d'un discours forment une seule entité (elles discutent le même sujet). On appelle ça "**Cohérence basée sur l'entité**", où la relation entre les phrase est le sujet. Parmi les théories dans cette direction, on peut mentionner : Centering theory et Entity grid model. Dans cette section, on va présenter des relations plus concrètes qui relient les phrases l'une à l'autre. Dans ce cas, on parle de la **cohérence basée sur les relations**. Deux représentations du discours sont bien connues : Rhetorical Structure Theory (RST) et Penn Discourse TreeBank (PDTB).

1.1 Rhetorical Structure Theory (RST)

RST modélise le discours sous forme des relations entre deux unités (des phrases ou parties des phrases) : **Noyau (N)** et **Satellite (S)**. Le noyau est une unité indépendante, peut être interprétée indépendamment des autres unités du texte. Le satellite est une unité dépendante, ne peut être interprétée qu'en utilisant le noyau. Chaque relation de cohérence est définie en se basant sur deux acteurs : **Lecteur (L)** ; qui a lit le script et **Scripteur (S)** ; qui a rédigé le script. Elle fournit des restrictions sur le noyau, sur le satellite et/ou sur les deux. L'effet de la relation peut prendre lieu dans un ou les deux acteurs. Voici quelques définitions des relations **RST** (Cornish, 2006) :

- **Élaboration** : S donne des informations additionnelles sur la situation présentée dans N
  - **Contraintes sur N + S :** S présente des informations supplémentaires vis-à-vis de la situation ou de quelqu'aspect du thème présenté(e) dans N
  - **Effet :** L reconnaît la situation présentée dans S comme fournissant des informations supplémentaires au sujet de N.
  - **Lieu de l'effet :** N + S
  - Ex. [$_N$ L'examen est facile.] [$_S$ Il ne prend qu'une heur.]

- **Évidence** : S donne des informations additionnelles sur la situation présentée dans N dans le but de convincre L à accepter les informations de N
  - **Contraintes sur N :** L pourra ne pas croire en N à un degré suffisant pour Sc
  - **Contraintes sur S :** L croit S ou le trouve crédible
  - **Contraintes sur N + S :** La compréhension de S par L augmente sa croyance en N
  - **Effet :** La croyance de L en N est effectivement augmentée
  - **Lieu de l'effet :** N
  - Ex. [$_N$ Kevin doit être ici.] [$_S$ Sa voiture est garée à l'extérieur.]

- **Contraste** : Une relation d'opposition entre deux noyaux
  - **Contraintes sur N :** multi-noyaux
  - **Contraintes sur N + N :** pas plus de deux noyaux ; les situations présentées dans ces deux noyaux sont (a) comprises comme les mêmes à beaucoup d'égards, (b) comprises comme différant par quelques aspects et (c) comparées par rapport à l'une ou plus d'une de ces différences
  - **Effet :** L reconnaît la comparabilité et les différences fournies par la comparaison effectuée
  - **Lieu de l'effet :** noyaux multiples
  - Ex. [$_N$ Il a voté "Non" à la nouvelle constitution.] [$_N$ Son frère a voté "Oui".]

- **Antithèse** : Une relation d'opposition entre N et S dans le but que L ait une attitude positive envers N
  - **Contraintes sur N :** Sc a une attitude positive par rapport à la situation présentée dans N
  - **Contraintes sur N + N :** es situations présentées dans N et S sont en opposition (cf. CONTRASTE)
  - **Effet :** L'attitude positive de L vis-à-vis de N est augmentée
  - **Lieu de l'effet :** N
  - Ex. [$_N$ J'ai défendu la République jeune ;] [$_S$ Je ne l'abandonnerai pas maintenant que je suis vieux.]

Les relations **RST** peuvent être classifiées selon la distinction thématique/présentationnelle (voir le tableau 9.1). Une relation est dite "thématique" si le lecteur peut l'identifier. Elle est utilisée afin d'exprimer certains aspects de la thématique. Par contre, une relation présentationnelle vise à augmenter une disposition chez le lecteur, comme le désir d'agir ou le degré de son attitude positive envers, ou croyance en, ou encore acceptation de la proposition noyau.

thématique		présentationnelle
Élaboration	But	Motivation
Circonstance	Condition	Antithèse
Problème-Solution	Interprétation	Arrière-plan
Cause intentionnelle	Évaluation	Facilitation
Résultat intentionnel	Reformulation	Évidence (« indice »)
Cause non-intentionnelle	Résumé	Justification
Résultat non-intentionnel	Séquence	Concession
	Contraste	

Tableau 9.1 : *Classification des relations RST selon le point de vu lecteur (Cornish, 2006)*

Les relations de cohérence peuvent être classifiées selon le niveau de traitement du langage (voir le tableau 9.2). La catégorie "thématique" correspondait à la catégorie "sémantique" dans cette classification. La catégorie "présentationnelle" peut être divisée sur deux catégories : "pragmatique" et "textuelle".

Sémantique		Pragmatique	Textuelle
Élaboration	But	Motivation	Arrière-plan
Circonstance	Condition	Antithèse	Reformulation
Problème-Solution	Interprétation	Facilitation	Résumé
Cause intentionnelle	Évaluation	Évidence (« indice »)	
Résultat intentionnel	Séquence	Justification	
Cause non-intentionnelle	Contraste	Concession	
Résultat non-intentionnel			

Tableau 9.2 : *Classification des relations RST selon les niveaux de traitement du langage (Cornish, 2006)*

1.2 Penn Discourse TreeBank (PDTB)

PDTB est un dataset annoté avec un autre modèle de cohérence. Dans ce modèle, on ne s'intéresse pas par la structure en arbre ; on s'intéresse seulement par les relations binaires. Les unités en relation sont annotées comme arguments : "ARG1" et "ARG2". L'unité marquée par "ARG2" est annotée par un connective du discours (explicitement ou implicitement). Le tableau 9.3 représente des statistiques sur les connectives et leurs sens. Les connectives du discours peuvent être :

- **Conjonctions de subordination** : temporelle (Ex., "when", "as soon as"), causale (Ex., "because"), concessive (Ex., "although", "even though"), objectif (Ex., "so that", "in order that") et conditionnelle (Ex., "if", "unless").
- **Conjonctions de coordination** : "and", "but", "or".
- **Conjonctives adverbiales** : des adverbes qui expriment une relation de discours entre des évènements ou des états. Ex., "however", "therefore", "then", etc. Des syntagmes prépositionnels sont inclus aussi dans cette classe. Ex. "as a result", "in addition", "in fact", etc.
- **Connectives implicites** : identifiées entre deux phrases adjacentes qui ne sont pas reliées par des connectives explicites.

Les relations **PDTB** sont divisées en quatre familles : temporelle, comparaison, contingence et extension. Chaque relation appartient à une famille d'un façon hiérarchique comme indiqué dans la figure 9.1. Par exemple, la relation de succession indiquée par des connectives comme "after" est une relation synchrone qui est à son tour une relation temporelle.

Connective	Sens
after	succession (523), succession-reason (50), other (4)
since	reason (94), succession (78), succession-reason (10), other (2)
when	Synchrony (477), succession (157), general (100), succession-reason (65), Synchrony-general (50), Synchrony-reason (39), hypothetical (11), implicit assertion (11), Synchrony-hypothetical (10), other (69)
while	juxtaposition (182), Synchrony (154), Contrast (120), expectation (79), opposition (78), Conjunction (39), Synchrony-juxtaposition (26), Synchrony-Conjunction (21), Synchrony-Contrast(22), COMPARISON (18), Synchrony-opposition (11), other (31)
meanwhile	Synchrony-Conjunction (92), Synchrony (26), Conjunction (25), Synchrony-juxtaposition (15), other(35)
but	Contrast (1609), juxtaposition (636), contra-expectation (494), COMPARISTON (260), opposition (174), Conjunction (63), Conjunction-Pragmatic contrast (14), Pragmatic-contrast (14), other (32) however Contrast (254), juxtaposition (89), contra-expectation (70), COMPARISON (49), opposition (31), other (12)
although	expectation (132), Contrast (114) juxtaposition (34), contra-expectation (21), COMPARISON (16), opposition (9), other (2)
and	Conjunction (2543), List (210), result-Conjunction (138), result (38), precedence-Conjunction (30), juxtaposition (11), other(30)
if	hypothetical (682), general (175), unreal present (122), factual present (73), unreal past (53), expectation (34), implicit assertion (29), relevance (20), other (31)

Tableau 9.3 : *Quelques connectives et des statistiques sur leurs sens (Prasad et al., 2008)*

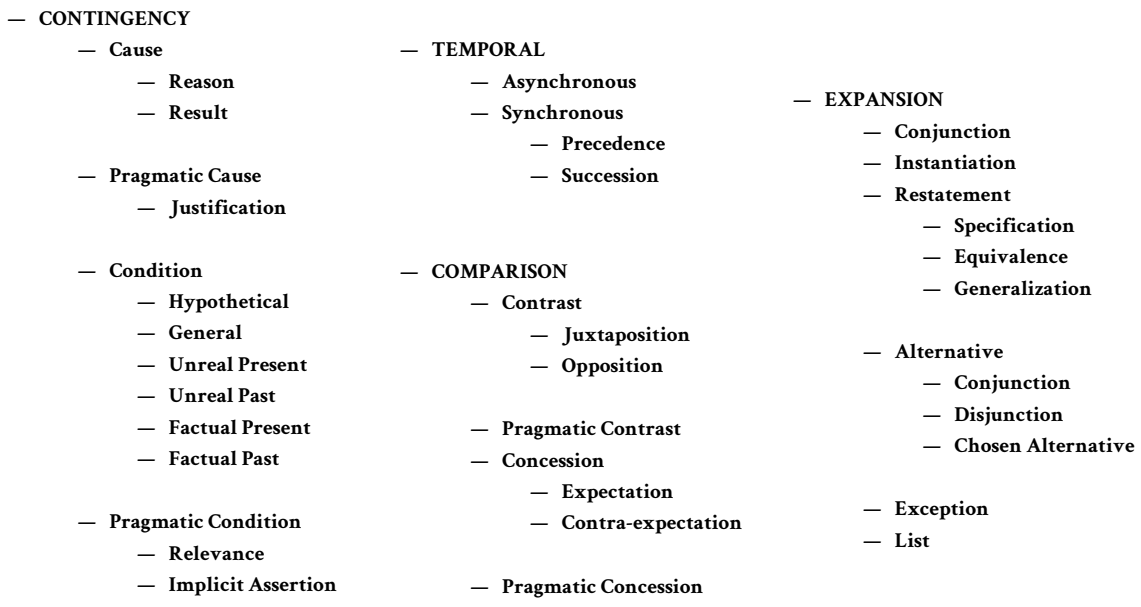


Figure 9.1 : *Hiérarchie des sens dans PDTB (Prasad et al., 2008)*

2 Analyse basée structure de discours

Un discours est cohérent s'il existe des relations entre ses parties. Analyser un discours veut dire chercher une structure du texte. Dans le cas de **RST**, cette structure est un arbre indiquant les relations de cohérence entre les différentes parties. Contrairement à **RST**, **PDTB** ne génère pas un arbre. Plutôt, c'est un

ensemble des relations binaires entre les parties du texte ; surtout les phrases consécutives.

2.1 Analyse RST

L'analyse **RST** consiste à construire un arbre de relations de cohérence entre les parties d'un texte. Cette tâche passe par deux étapes : détection des unités élémentaires de discours et classification des relations. La figure 9.2 représente une analyse **RST** d'un texte. On remarque que l'unité n'est pas toujours une phrase, mais elle peut être une partie d'une phrase (une clause).

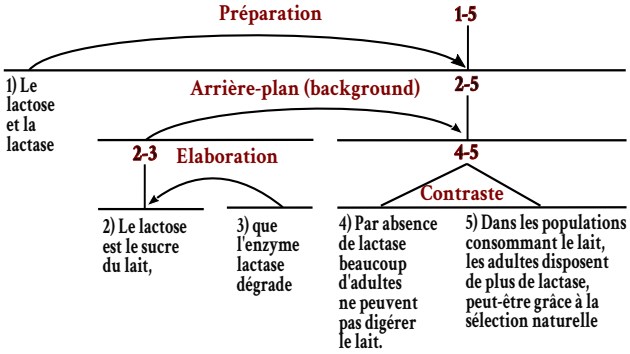


Figure 9.2 : *Exemple d'un arbre RST*

Détection des unités élémentaires

Une unité élémentaire de discours, en anglais Elementary Discourse Unit (EDU), est une phrase ou une clause de la phrase qui représente un sens. Exemple d'un texte divisé en EDU : [Mr. Rambo says]$_{e1}$ [that a 3.2-acre property]$_{e2}$ [overlooking the San Fernando Valley]$_{e3}$ [is priced at $4 million]$_{e4}$ [because the late actor Erroll Flynn once lived there.]$_{e5}$. La détection des EDU consiste à trouver ces unités qui sont en général des clauses. Une des méthodes les plus anciennes utilise l'analyse syntaxique afin de trouver ces clauses. Des méthodes statistiques peuvent être utilisées afin de détecter les limites des EDUs. Comme caractéristiques, on peut utiliser les informations syntaxiques et des indices de surface. Ce problème peut être vu comme annotation des séquences. Dans (Wang et al., 2018), les auteurs ont proposé un système neuronal [1] qui utilise le **embedding** des mots. La figure 9.3 représente leur architecture où l'entrée est une concaténation entre deux **embeddings** : **GloVe** et **BERT**. On utilise un réseau **Bi-LSTM** afin d'avoir le contexte en avant et en arrière pour chaque mot. Ensuite, chaque mot est classé comme : début d'un EDU (1) ou continuation d'un EDU (0).

Classification des relations

Une fois les EDUs sont extraits, on doit les lier en utilisant des relations de cohérence. La méthode la plus utilisée pour trouver la structure **RST** est SHIFT-REDUCE. Celle-ci se base sur une machine abstraite ayant la configuration $C = (\sigma, \beta, A)$ où σ est une pile, β est le tampon (buffer) d'entrée et A est la liste des arcs créés (relations). La figure 9.4 représente l'architecture de cette machine qui est similaire à celle utilisée dans l'analyse syntaxique des dépendances par transition. La différence est que cette machine utilise les EDUs et pas les mots comme éléments d'analyse. Au début, la pile est vide, la liste des relations est vide et le tampon d'entrée contient tous les EDU ordonnés d'où $C_{initiale} = (\varnothing, w, \emptyset)$. A la fin, la pile et le tampon d'entrée doivent être vides d'où $C_{finale} = (\varnothing, \varnothing, A)$.

Les opérations permises par cette machine sont :

1. EDU avec réseau de neurones : https://github.com/PKU-TANGENT/NeuralEDUSeg [visité le 2021-09-15]

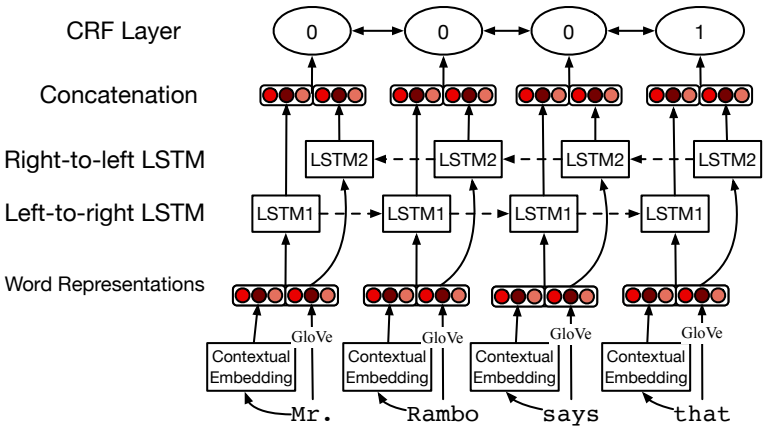


Figure 9.3 : *Segmentation en EDUs proposée par Wang et al. (2018) ; figure prise de (Jurafsky et Martin, 2019)*

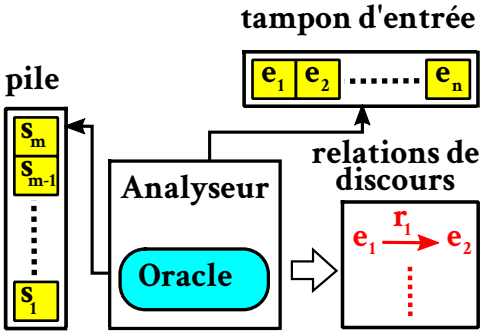
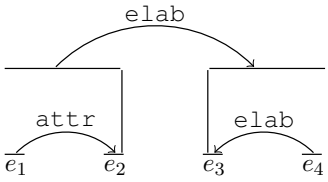


Figure 9.4 : *Architecture SHIFT-REDUCE pour la résolution de la structure RST*

— **Shift** : mettre le premier élément du buffer dans la pile
— **Reduce(l, d)** : fusionne les deux sous-arbres supérieurs de la pile, où **l** est l'étiquette de relation de cohérence, et **d** est la direction de nucléarité : **d** ∈ **{NN, NS, SN}**.
— **Pop Root** : enlever la racine de l'arbre final de la pile.

Un exemple d'une analyse **RST** en utilisant la méthode SHIFT-REDUCE est illustré das la figure 9.5.



e_1: American Telephone & Telegraph Co. said it
e_2: will lay off 75 to 85 technicians here , effective Nov. 1.
e_3: The workers install , maintain and repair its private branch exchanges,
e_4: which are large intracompany telephone networks.

Step	Stack	Queue	Action	Relation
1	∅	e_1, e_2, e_3, e_4	SH	∅
2	e_1	e_2, e_3, e_4	SH	∅
3	e_1, e_2	e_3, e_4	RD (attr, SN)	∅
4	$e_{1:2}$	e_3, e_4	SH	$\widehat{e_1 \mathbf{e_2}}$
5	$e_{1:2}$, e_3	e_4	SH	$\widehat{e_1 \mathbf{e_2}}$
6	$e_{1:2}$, e_3, e_4	∅	RD (elab, NS)	$\widehat{e_1 \mathbf{e_2}}$
7	$e_{1:2}$, $e_{3:4}$	∅	RD (elab, SN)	$\widehat{e_1 \mathbf{e_2}}$, $\widehat{\mathbf{e_3} e_4}$
8	$e_{1:4}$	∅	PR	$\widehat{e_1 \mathbf{e_2}}$, $\widehat{\mathbf{e_3} e_4}$, $\widehat{e_{1:2} \mathbf{e_{3:4}}}$

Figure 9.5 : *Exemple d'analyse RST en utilisant Shif-Reduce (Yu et al., 2018)*

Le composant "Oracle" doit être entraîné afin de décider l'opération suivante. On peut utiliser des caractéristiques afin d'entraîner un algorithme d'apprentissage comme on a vu dans l'analyse des dépendances

(chapitre 5). On peut aussi utiliser les **embeddings** avec un réseau de neurone comme la méthode proposée dans (Yu et al., 2018) [2]. Les auteurs proposent d'utiliser une architecture encodeur-décodeur afin de décider l'opération suivante. L'encodeur a comme but de représenter tous les EDUs comme des vecteurs. Le décodeur utilise les représentations de quelques EDUs afin de décider l'opération suivante.

On commence par l'encodeur qui prend en entrée une représentation x_i^w de chaque mot w_i. Cette représentation et la concaténation du **embedding** du mot w_i et de sa catégorie grammaticale t_i comme indiqué par l'équation 9.1

$$x_i^w = embedding(w_i) \oplus embedding(t_i) \tag{9.1}$$

Les mots d'une phrase $w_1 w_2 \dots w_m$ sont passés par un réseau récurrent **Bi-LSTM** pour avoir une représentation séquentielle comme indiqué par l'équation 9.2.

$$\{h_1^w, h_2^w, \dots, h_m^w\} = biLSTM(\{x_1^w, x_2^w, \dots, x_m^w\}) \tag{9.2}$$

Nous avons déjà sélectionné les EDUs ; donc on sais chaque début et fin d'un EDU. Afin de représenter un EDU $\{w_s, w_{s+1}, \dots, w_t\}$, on cherche le vecteur central des vecteurs représentant les mots qui lui composent comme indiqué dans l'équation 9.3

$$x^e = \frac{1}{t-s+1} \sum_{k=s}^{t} h_k^w \tag{9.3}$$

Une fois la représentation individuelle de chaque EDU x_i^e est trouvée, on cherche leurs représentations séquentielles. Pour ce faire, on utilise un autre réseau récurrent **Bi-LSTM** comme indiqué par l'équation 9.4

$$\{h_1^e, h_2^e, \dots, h_n^e\} = biLSTM(\{x_1^e, x_2^e, \dots, x_n^e\}) \tag{9.4}$$

Le décodeur est une couche neuronale feed-forward W qui vise à inférer l'action suivante o. En entrée, il prend la représentation séquentielle du premier UED dans le buffer h_e^{q0} et les représentations séquentielles des trois sous-arbres i au sommet de la pile h_{si}^{sbt}. Le décodeur est représenté par l'équation 9.5.

$$o = W(h_{s0}^{sbt} \oplus h_{s1}^{sbt} \oplus h_{s2}^{sbt} \oplus h_{q0}^{e}) \tag{9.5}$$

Un sous arbre peut être représenté par plusieurs EDUs $s = \{e_i, \dots, e_j\}$. Sa représentation peut être calculée par la moyenne des représentations des UEDs couverts, comme indiqué dans l'équation 9.6.

$$h_s^{sbt} = \frac{1}{j-i+1} \sum_{k=i}^{j} h_k^e \tag{9.6}$$

2.2 Analyse PDTB

Dans l'analyse PDTB, on essaye de séparer les parties du texte et les annoter deux à deux par : "ARG1" et "ARG2". Dans la partie annotée par "ARG2", on essaye de l'annoter par une connective. Nous avons vu qu'il y a deux types de connectives : explicites (existantes dans le texte) et implicites (qu'on peut inférer). Dans le cas des connectives explicites, on peut facilement les chercher et les trouver en utilisant leur liste. Mais, il existe des connectives qui ne représentent pas des relations de cohérence. Pour résoudre ce problème, on peut utiliser un algorithme de désambigüisation qui classe un connective donnée comme : discours ou non. Une fois une connective est marquée comme discours, on marque la partie le contenant comme "ARG2". On cherche la partie en relation "ARG1" en utilisant un algorithme d'apprentissage

2. Embedding pour RST : https://github.com/yunan4nlp/NNDisParser [visité le 2021-09-15]

automatique. L'algorithme prend "ARG2" et une autre partie adjacente comme entrée et comme sortie il estime si elles sont en relation ou non. Ensuite, on marque la relation entre les deux en utilisant la connective.

Dans le cas où le connective est implicite, on doit inférer la connective étant donnée deux parties sans connectives. Une méthode est d'utiliser **BERT** avec les parties séparées par "[SEP]" comme entrée. Dans la sortie "[CLS]", on attache un réseau de neurones à propagation avant afin d'inférer la connective. Une autre méthode similaire est l'utilisation de deux réseaux **LSTM** pour représenter chacune des deux parties. Cette méthode a été proposée par Liang et al. (2020), où l'architecture est illustrée dans la figure 9.6. Le "Max-Pool" est utilisé sur les représentations récurrentes des mots afin de composer le sens d'une partie et aussi pour réduire les paramètres du modèle. Dans chaque position dans les vecteurs des mots, il prend le nombre max pour avoir un autre vecteur de même taille.

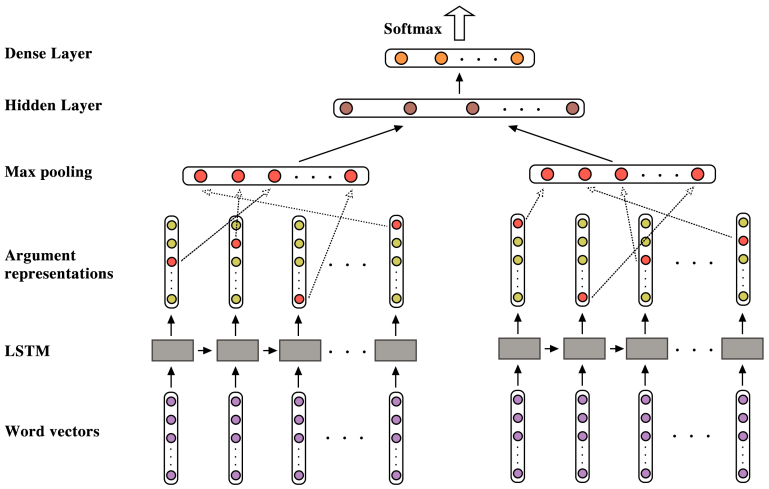


Figure 9.6 : *Architecture pour la détection de relations PDTB implicites (Liang et al., 2020)*

3 Analyse basée sur l'entité de discours

Un discours est cohérent s'il discute le même sujet. Ce dernier peut être représenté par une entité. Quelque soit sa position dans le discours, cette entité doit rester la plus importante. Prenons l'exemple "John went to his favorite music store to buy a piano [John]. He had frequented the store for many years [John]. He was excited that he could finally buy a piano [John]. He arrived just as the store was closing for the day [John].". Dans cet exemple, l'entité centrale de chaque phrase est "John". Maintenant, prenons l'exemple : "John went to his favorite music store to buy a piano [John]. It was a store John had frequented for many years [The store]. He was excited that he could finally buy a piano [John]. It was closing just as John arrived [The store].". On peut sentir que le texte est absurde ; il est un peut difficile à comprendre. Cela est dû au fait que l'entité centrale se bascule entre les phrases ; des fois "John", d'autres "The store". Dans cette section, on va discuter deux théories/méthodes pour analyser un texte en se basant sur l'entité et pas la structure.

3.1 Centering theory

Nous avons vu que les phrases (énoncés) doivent maintenir la même entité centrale. Cette intuition est réalisée dans cette théorie en maintenant deux représentations pour chaque énoncé U_n. La première est l'entité saillante actuelle ; celle sur laquelle se concentre le discours dans l'énoncé U_{n-1}. Elle s'appelle "**Backward-looking center**" (centre rétrospectif), et elle est dénotée par $C_b(U_n)$. La deuxième est un

ensemble des entités potentiellement saillantes dans le futur ; celles candidates pour être $C_b(U_{n+1})$. Il s'appelle "**Forward-looking center**" (centres prospectifs), et il est dénoté par $C_f(U_n)$. On score les entités de cet ensemble en se basant sur leurs rôles grammaticaux (sujet plus important que l'objet qui est plus important que le reste), l'ordre (En Arabe, ce qui est en premier est plus important), etc. L'entité avec le plus grand score est choisie comme candidate pour être $C_b(U_{n+1})$. Elle s'appelle "**Prefered center**" (centre préféré), et elle est dénotée par $C_p(U_n)$.

Cette théorie se base sur l'hypothèse que le discours est plus facile à traiter lorsque les énoncés successifs parlent de la même entité. Cette hypothèse est formalisée comme une classification des énoncés selon la transition qu'ils induisent dans le centre local. Il existe trois types de transitions (Poesio et al., 2004) selon l'ordre de la plus cohérente :

- **CONTINUE** : le locuteur parle d'une entité et il a l'intention d'en parler en futur
- **RETAIN** : le locuteur parle d'une entité et il a l'intention d'en changer en futur
- **SHIFT** : le locuteur a changé l'entité centrale
  - **Smooth-SHIFT** : après le changement, il a l'intention d'en parler en futur
  - **Rough-SHIFT** : après le changement, il a l'intention d'en changer en futur

La théorie de centralité (Centering theory) se base sur deux règles. La première règle annonce que si un élément de $C_f(U_n)$ est réalisé par un pronom dans l'énoncé suivant (U_{n+1}) alors $C_b(U_{n+1})$ doit être réalisée par un pronom. L'intuition, ici, est que la pronominalisation est un moyen courant pour marquer la saillance du discours. S'il y a plusieurs pronoms dans un énoncé réalisant des entités de l'énoncé précédent, l'un de ces pronoms doit réaliser le centre en arrière C_b. La deuxième règle concerne les transitions mentionnées précédemment, où "CONTINU" est plus cohérente que "RETAIN" que "Smooth-SHIFT" que "Rough-SHIFT". Ces transitions sont calculée selon le tableau 9.4. L'intuition de cette règle est que les discours qui continuent à se center sur la même entité sont plus cohérents que ceux qui basculent vers d'autres centres.

	$\mathbf{C_b(U_n) = C_b(U_{n-1})}$ **OU** $\mathbf{C_b(U_n) = NULL}$	$\mathbf{C_b(U_n) \neq C_b(U_{n-1})}$
$\mathbf{C_b(U_n) = C_p(U_n)}$	CONTINUE	Smooth-SHIFT
$\mathbf{C_b(U_n) \neq C_p(U_n)}$	RETAIN	Rough-SHIFT

Tableau 9.4 : *Transitions de la théorie de centralité*

Reprenons deux phrases de l'exemple précédent, et essayons de calculer la centralité

- John went to his favorite music store to buy a piano. U_1
  - $C_b(U_1)$ = NULL
  - $C_f(U_1)$ = {John, music store, piano}
  - $C_p(U_1)$ = John (le sujet)

- It was a store John had frequented for many years. U_2
  - $C_b(U_2)$ = John
  - $C_f(U_2)$ = {(music) store, John, years}
  - $C_p(U_2)$ = music store (sujet)
  - $C_b(U_2) \neq C_p(U_2) \wedge C_b(U_2) \neq C_b(U_1) \Rightarrow$ Rough-SHIFT

- He was excited that he could finally buy a piano. U_3

— $C_b(U_3)$ = music store
— $C_f(U_3)$ = {John, piano}
— $C_p(U_3)$ = John (sujet)
— $C_b(U_3) \neq C_p(U_3) \wedge C_b(U_3) \neq C_b(U_2) \Rightarrow$ Rough-SHIFT

3.2 Entity Grid model

Dans le modèle "entity grid", un document est représenté par une matrice où les phrases représentent les lignes et les entités représentent les colonnes. Chaque cellule de cette matrice contient la fonction grammaticale de l'entité dans la phrase : sujet (S), Objet (O), Autre (X) ou l'entité n'existe pas (_). La figure 9.7 représente un texte et sa représentation phrases/entités selon le modèle "entity grid".

1 [The Justice Department]$_S$ is conducting an [anti-trust trial]$_O$ against [Microsoft Corp.]$_X$ with [evidence]$_X$ that [the company]$_S$ is increasingly attempting to crush [competitors]$_O$.
2 [Microsoft]$_O$ is accused of trying to forcefully buy into [markets]$_X$ where [its own products]$_S$ are not competitive enough to unseat [established brands]$_O$.
3 [The case]$_S$ revolves around [evidence]$_O$ of [Microsoft]$_S$ aggressively pressuring [Netscape]$_O$ into merging [browser software]$_O$.
4 [Microsoft]$_S$ claims [its tactics]$_S$ are commonplace and good economically.
5 [The government]$_S$ may file [a civil suit]$_O$ ruling that [conspiracy]$_S$ to curb [competition]$_O$ through [collusion]$_X$ is [a violation of the Sherman Act]$_O$.
6 [Microsoft]$_S$ continues to show [increased earnings]$_O$ despite [the trial]$_X$.

	Department	Trial	Microsoft	Evidence	Competitors	Markets	Products	Brands	Case	Netscape	Software	Tactics	Government	Suit	Earnings	
1	S	O	S	X	O	–	–	–	–	–	–	–	–	–	–	1
2	–	–	O	–	–	X	S	O	–	–	–	–	–	–	–	2
3	–	–	S	O	–	–	–	–	S	O	O	–	–	–	–	3
4	–	–	S	–	–	–	–	–	–	–	–	S	–	–	–	4
5	–	–	–	–	–	–	–	–	–	–	–	–	S	O	–	5
6	–	X	S	–	–	–	–	–	–	–	–	–	–	–	O	6

Figure 9.7 : *Exemple de la représentation d'un document par entités (Barzilay et Lapata, 2008)*

Un document est considéré comme cohérent s'il suit une certaine forme de pattern. Ce pattern peut-être représenté par les fréquences de transitions grammaticales. Ex. "SS, SO, SX, S_, OS, OO, OX, O_, ...". Dans l'exemple précédent, la transition "S_" a une fréquence de 6. Un document peut être représenté par les probabilités des transitions grammaticales. Une probabilité est le ratio entre le nombre d'une transition et le nombre de toutes les transitions. Dans l'exemple précédent : P(S_) = 6/75. La figure 9.8 représente les probabilités des transitions de longueur 2 du document précédent.

	SS	SO	SX	S–	OS	OO	OX	O–	XS	XO	XX	X–	–S	–O	–X	––
d_1	.01	.01	0	.08	.01	0	0	.09	0	0	0	.03	.05	.07	.03	.59
d_2	.02	.01	.01	.02	0	.07	0	.02	.14	.14	.06	.04	.03	.07	0.1	.36
d_3	.02	0	0	.03	.09	0	.09	.06	0	0	0	.05	.03	.07	.17	.39

Figure 9.8 : *Exemple de la représentation des documents par transitions grammaticales (Barzilay et Lapata, 2008)*

Afin de juger qu'un texte soit cohérent, on utilise un algorithme d'apprentissage automatique où les probabilités des transitions grammaticales sont utilisées comme caractéristiques. Le modèle d'apprentissage peut être entraîné à attribuer des scores selon le niveau de cohérence. Dans ce cas, un texte non cohérent doit avoir un score inférieur à un autre cohérent. Le problème qui se pose : il est difficile d'avoir beaucoup

de textes annotés pour entraîner le système. Une solution est d'utiliser une méthode auto-supervisée ; créer un texte non cohérent à partir d'un texte cohérent. Ceci peut être accompli en appliquant un ordre aléatoire des phrases d'un texte cohérent.

Cette idée peut être utilisée pour tester les systèmes d'analyse de la cohérence : un texte cohérent doit avoir un score plus grand qu'un autre non cohérent. Donc, on peut prendre un texte cohérent (les romans, etc.) et créer un autre non cohérent selon une de ces techniques :

— Discrimination de l'ordre des phrases : ordre aléatoire des phrases et comparaison du score de cohérence avec celui de l'ordre original.
— Insertion de la phrase : modifier l'ordre d'une seule phrase et comparaison du score de cohérence avec celui de l'ordre original.
— Reconstruction de l'ordre des phrases : apprendre à ordonner les phrases

Discussion

Une phrase doit être bien formée ; ceci est garanti par la grammaire. Elle doit, aussi, avoir un sens ; ceci est garanti par la sémantique. Lorsqu'on fusionne plusieurs phrases qui ont un sens, ce n'est pas une garantie que ce texte soit compréhensible ou au moins naturel. Un texte doit être cohérent. Mais, c'est quoi la cohérence ? Comment elle peut être mesurée ? Il existe plusieurs théories pour représenter la cohérence qui peuvent être classées en deux approches : basées sur la structure ou basées sur l'entité. Les phrases (ou les clauses) d'un texte forment une structure qui se base sur des relations de cohérence. Aussi, elles doivent discuter la même entité. Les deux approches ont deux visions différentes ; qui sont complémentaires à mon avis. Afin de juger la cohérence d'un texte, il faut mieux utiliser les deux afin de vérifier deux aspects différents.

CHAPITRE 10

QUELQUES APPLICATIONS

LES LANGUES

Selon l'interaction avec l'utilisateur, un système peut être interactif ou non. En se basant sur la sortie, un système peut générer un ensemble de classes ou un autre texte. Un système de traitement du langage naturel peut traiter de la parole ou du texte. En utilisant ces critères, on peut classifier les applications du TALN en quatre catégories : Transformation, Interaction, Classification et Parole. La transformation sert à prendre un texte en entrée et générer un autre en sortie; comme la traduction automatique et le résumé automatique. L'interaction comporte les applications qui interagissent avec l'utilisateur; comme les systèmes de questions/réponses et de dialogue. La classification prend un texte en entrée et infère un classe; comme l'analyse des sentiments et la lisibilité. Enfin la parole consiste de la reconnaissance et la synthèse de la parole. Dans ce chapitre, on va présenter deux applications par catégorie.

Chaque jour beaucoup d'informations sont générées; en général, sous forme de textes non structurées. L'anglais est la langue la plus répondue; mais, on peut remarquer une augmentation du contenu des autres langues. Traiter ces textes sera bénéfique dans la résolution de nous besoins. Reprenons quelques motivations mentionnées dans le premier chapitre :

- Commerce : publicité, service clientèle, intelligence de marché, recrutement, etc.
- E-Gouvernance : communication gouvernement/citoyens, fouille d'opinions, etc.
- Santé : rechercher, analyser, interpréter et structurer les documents médicaux, prédire les maladies, utiliser les assistants virtuels, etc.
- Éducation : évaluation de la langue, correction des erreurs, apprentissage en ligne, etc.

1 Traduction automatique

La traduction automatique consiste à transformer un texte écrit en une langue origine vers un texte écrit en une langue destinataire. La motivation de cette tâche est claire : casser le barrière entre les être humains en facilitant la communication. Plusieurs méthodes ont été proposées pour cette tâche en utilisant des différentes techniques. En se basant sur le type de ces dernières, les méthodes de traduction automatique peuvent être classifiées comme indiqué dans la figure 10.1. L'approche directe est la plus simple; elle consiste à une traduction mot par mot. Donc, elle est utile seulement pour les langues qui sont proches syntaxiquement. On peut citer deux approches à base de règles : par transfert et en utilisant une interlingua. Les méthodes par transfert se basent sur des règles pour transformer un arbre syntaxique de la langue origine vers un arbre syntaxique de la langue destinataire. L'autre idée est de proposer tout un langage qui doit être universel. Ensuite, on peut concevoir des encodeurs à partir des langages naturels vers ce langage et des décodeurs de cet langage vers des langages naturels. On peut utiliser des corpus pour entraîner un système d'apprentissage automatique à traduire d'une langue vers une autre. Donc, on doit

se baser sur des statistiques. La terminologie “statistique” est souvent utilisée pour indiquer qu'il s'agit de l'apprentissage bayésien. Ces méthodes ont besoins de beaucoup de données pour entraîner les probabilités de chaque mot. En plus, des parties de textes sont toujours utilisées telles qu'elles sont. Pour améliorer la classification, on peut utiliser ces parties comme unités ; d'où les méthodes à base d'exemples. Finalement, les méthodes les plus répondues ces jours-là sont à base des réseaux de neurones. Le problème de traduction automatique peut être représenté comme un encodeur/décodeur.

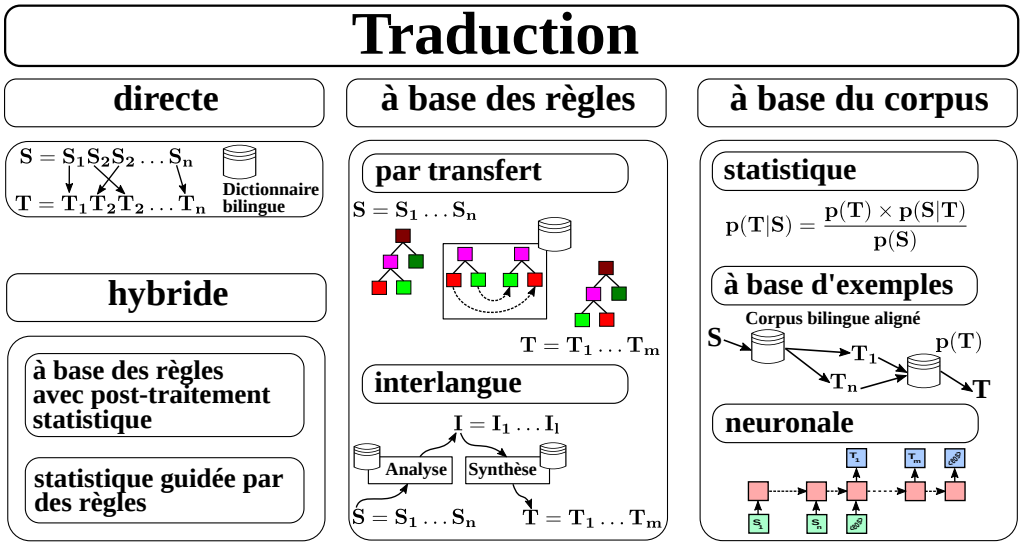


Figure 10.1 : *Approches de la traduction automatique*

1.1 Approche directe

Dans l'approche directe, on essaye de traduire le texte mot par mot. Le texte source S est traité comme une série de mots. Chaque mot S_i est remplacé par un mot T_i dans le texte destinataire T en utilisant un dictionnaire bilingue. Donc, pour implémenter une telle méthode, il nous faut un outil d'analyse morphologique de la langue source et un dictionnaire bilingue bien conçu. Les langues (source et destinataire) doivent être proches syntaxiquement (structures grammaticales proches) puisque cette approche ne supporte que le niveau morphologique. La traduction mot à mot est suivie par une étape de post-traitement pour organiser l'ordre des mots. Exemple, “SVO →VSO”, “adj + N →N + Adj”. Les systèmes qui suivent cette approche sont ceux développés avant 1967 : Météo, Weidner, CULT et Systran (premières versions).

1.2 Approche par transfert

Traduire les textes mot à mot limite le nombre des tuples de langues (source, destinataire) qu'on puisse traiter. L'idée de cette approche est de trouver l'arbre syntaxique de la phrase source S en utilisant l'analyse syntaxique. Ensuite, on cherche la traduction des mots S_i de la langue source vers des mots T_i de la langue destinataire en utilisant un dictionnaire bilingue. Après, on applique des règles pour transformer l'arbre syntaxique de la langue source vers un arbre syntaxique de la langue destinataire (comme l'exemple de la figure 10.2). Ces règles sont définies manuellement ou inférées en utilisant l'apprentissage automatique. Finalement, on génère le texte traduit T à partir de l'arbre syntaxique final.

Un des systèmes qui se basent sur cette approche est Apertium [1] (Forcada et al., 2011). La figure 10.3 représente l'architecture de ce système qui ce compose des modules suivants :

1. Apertium : https://www.apertium.org/ [visité le 2021-09-16]

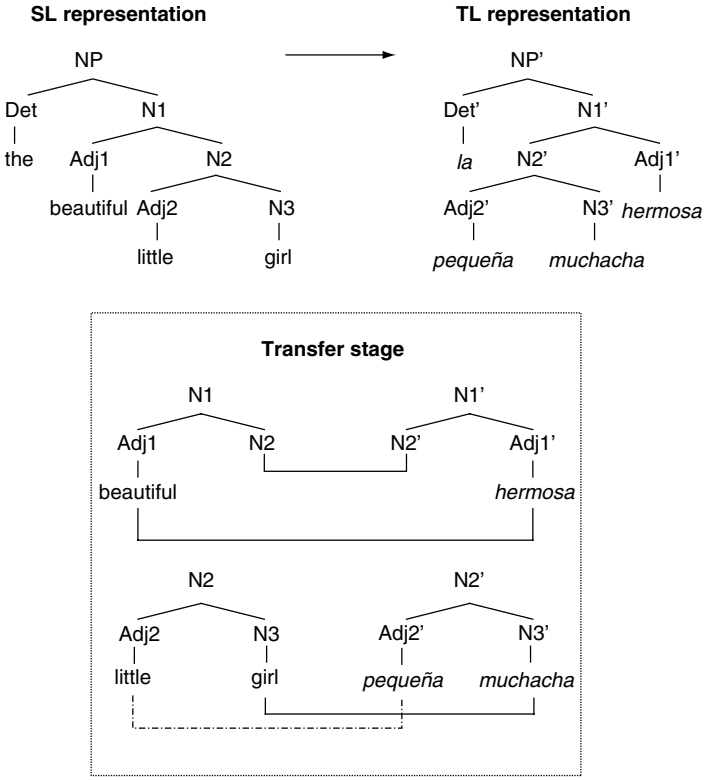


Figure 10.2 : *Exemple de règles de transfert syntaxique (Quah, 2006)*

— *deformatter* : éliminer les informations de format et garder seulement le texte. Il est utiliser pour lire le texte en utilisant plusieurs formats comme XML.
— *morphological analyser* : segmentation du texte et attribution à chaque mot une liste des lemmes avec leurs catégories grammaticales possibles.
— *PoS tagger* : un outil d'étiquetage morpho-syntaxique pour avoir les catégories grammaticales des mots.
— *lexical transfer* : traduire les mots (ou multi-mots) de la langue source vers la langue destinataire en utilisant un dictionnaire bilingue.
— *structural transfer* : c'est un module qui fournit le transfert syntaxique. Il contient un sous-module *chunker* qui applique une analyse syntaxique de surface (Chunking) sur le texte de source. Le sous-module *interchunk* applique des règles de transfert de la structure source vers la structure destinataire. Le sous-module *postchunk* applique des opérations de post-traitement sur la structure destinataire.
— *morphological generator* : appliquer des transformations morphologiques comme la conjugaison sur les mots.
— *post-generator* : appliquer des opérations d'orthographe. Par exemple, en anglais a + institute = an institute.
— *reformatter* : formater la réponse en utilisant un format spécifique comme XML, JSON, etc.

1.3 Approche interlingue

Dans la traduction automatique, on doit implémenter un système pour chaque pair de langues (source et destinataire). Par exemple, si on possède un système de traduction du français vers l'anglais et un autre de l'anglais vers l'arabe, on peut utiliser les deux systèmes afin de traduire du français vers l'arabe. Dans ce cas, l'anglais a été utilisée comme langue intermédiaire. L'idée de l'approche interlingue est d'utiliser un langage intermédiaire comme indiqué dans la figure 10.4. Pour ajouter une langue source, on doit seulement implémenter un analyseur de cette langue vers l'interlingua. Comme ça, cette langue peut être

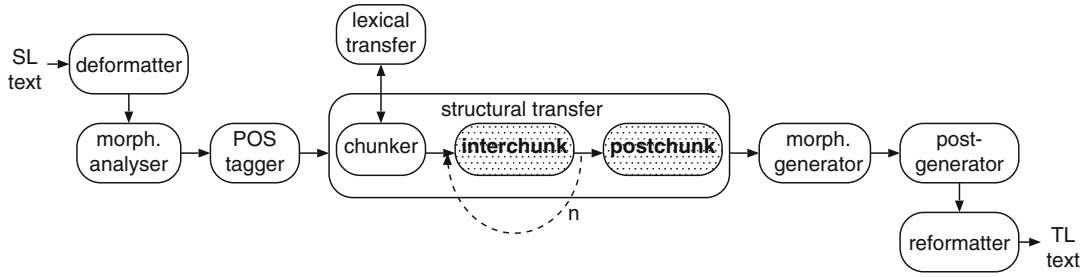


Figure 10.3 : *Architecture du système Apertium (Forcada et al., 2011)*

traduite vers toutes les langues disponibles comme langues destinataires du système. Afin d'ajouter une nouvelle langue destinataire, il suffit d'implémenter un générateur de cette langue à partir de l'interlingua. Donc, la traduction passe par les étapes suivantes :

— Analyser le texte source S pour avoir un arbre syntaxique
— Utiliser un dictionnaire entre la langue source et les concepts de l'interlingua
— Transformer l'arbre syntaxique (langue source) vers l'interlingue
— Transformer l'interlingue vers une arbre syntaxique (langue destinataire)
— Utiliser un dictionnaire entre la langue destinataire et les concepts de l'interlingua
— Générer le texte destinataire T

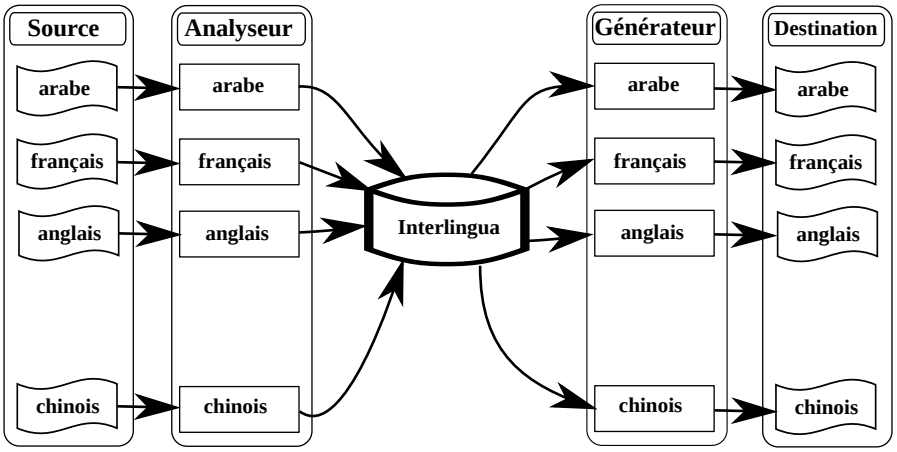


Figure 10.4 : *Motivation de l'approche interlingue*

L'interlingua doit être une représentation abstraite indépendante des langues ; elle doit être un langage universel. KANT (Czuba et al., 1998) est un exemple d'une interlingua (voir la figure 10.5).

KANTOO (Nyberg et Mitamura, 2000) est un système de traduction automatique qui utilise KANT comme interlingua. Son architecture est illustrée dans la figure 10.6. Le système garde une base de connaissance contenant des règles manuelles pour la transformation entre les langages naturels et KANT. L'analyseur applique une analyse syntaxique pour ensuite transférer la représentation sémantiquement vers KANT. Le générateur applique l'opération inverse : générer un texte à partir de la représentation KANT.

```
(*A-REMAIN  ; action rep for 'remain'
    (FORM FINITE)
    (TENSE PAST)
    (MOOD DECLARATIVE)
    (PUNCTUATION PERIOD)
    (IMPERSONAL -) ; passive + expletive subject
    (ARGUMENT-CLASS THEME+PREDICATE) ; predicate argument structure
    (Q-MODIFIER ; PP semrole (generic)
        (*K-DURING ; PP interlingua
            (POSITION FINAL) ; clue for translation
      (OBJECT ; PP object semrole
          (*O-TIME ; object rep for 'time'
              (UNIT -)
              (NUMBER SINGULAR)
              (REFERENCE DEFINITE)
              (DISTANCE NEAR)
              (PERSON THIRD)))))
(THEME ; object semrole
    (*O-DEFAULT-RATE ; object rep for ' default rate'
        (PERSON THIRD)
        (UNIT -)
        (NUMBER SINGULAR)
        (REFERENCE DEFINITE)))
(PREDICATE ; adjective phrase semrole
    (*P-CLOSE ; property rep for ' closer'
        (DEGREE POSITIVE)
        (Q-MODIFIER
            (*K-TO
                (OBJECT
                    (*O-ZERO
                    (UNIT -)
                    (NUMBER SINGULAR)
                    (REFERENCE NO-REFERENCE)
                    (PERSON THIRD)))))))
```

Figure 10.5 : *Représentation de la phrase "The default rate remained close to zero during this time." avec KANT interlingua (Czuba et al., 1998)*

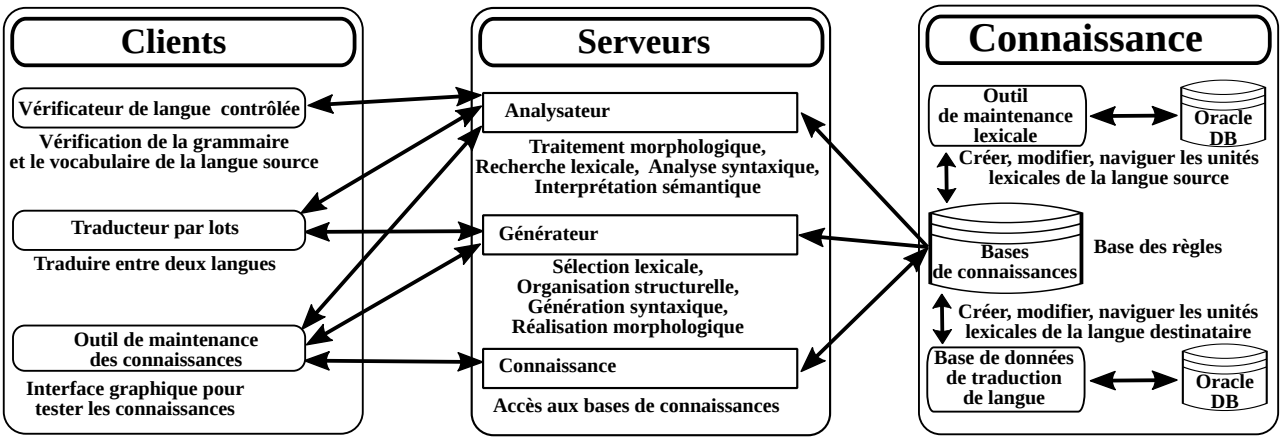


Figure 10.6 : *Architecture du système KANTOO, traduit à partir de (Nyberg et Mitamura, 2000)*

1.4 Approche statistique

Étant donné un texte source S et un autre destinataire T, la probabilité que T soit généré à partir de S est estimée en utilisant la théorie de Bayes comme indiqué par l'équation 10.1.

$$p(T|S) = \frac{p(T)p(S|T)}{p(S)} \propto \underbrace{p(T)}_{\text{Cohérence}} \underbrace{p(S|T)}_{\text{Fidélité}} \tag{10.1}$$

Le problème de traduction automatique revient à trouver le texte destinataire $\hat{T}$ qui maximise cette probabilité comme indiqué par l'équation 10.2

$$\hat{T} = \arg\max_{T} p(T)p(S|T) \tag{10.2}$$

La probabilité $p(T)$ peut être calculée en utilisant un modèle de langage entrainé sur la langue destinataire. Supposant que le modèle de langage est un modèle **N-gramme**, cette probabilité peut être calculée par l'équation 10.3

$$p(T) = \prod_{j=1}^{m} p(t_j|t_{j-N+1} \ldots t_{j-1}) \tag{10.3}$$

Nous avons vu comment entraîner un modèle de langage. Mais, afin d'entraîner le modèle de traduction $p(S|T)$, il faut avoir un corpus aligné : les mots de la langue source doivent être liés avec ceux correspondants de la langue destinataire. Il faut mentionner que la taille pose un problème lors de l'alignement. On

peut considérer cette probabilité comme la somme des probabilités de tous les alignements A possibles, comme indiqué par l'équation 10.4.

$$p(S|T) = \sum_{A} p(S, A|T) \tag{10.4}$$

Il faut mentionner qu'on doit avoir $(m+1)^n$ alignements possibles pour un texte source S de taille n et un texte traduit T de taille m. La probabilité qu'un texte source S soit généré avec un alignement A sachant un texte destinataire T peut être calculée en se basant sur les probabilités que chaque mot de S soit aligné avec A_i sachant tous les mots passés de S $(S_1^{i-1} = S_1 \dots S_{i-1})$, les alignements passés de A et tous les mots du texte traduit. Cette probabilité est formulée par l'équation 10.5.

$$p(S, A|T) = \prod_{i=1}^{n} p(S_i, A_i|S_1^{i-1}, A_1^{i-1}, T_1^m) \tag{10.5}$$

La probabilité élémentaire de S_i et A_i peut être décomposée selon l'équation 10.6.

$$p(S_i, A_i|S_1^{i-1}, A_1^{i-1}, T_1^m) = p(A_i|S_1^{i-1}, A_1^{i-1}, T_1^m) p(S_i|S_1^{i-1}, A_1^{i}, T_1^m) \tag{10.6}$$

La probabilité $p(S_i|S_1^{i-1}, A_1^i, T_1^m)$ peut être estimée en utilisant un corpus d'entrainement aligné. Quant à la probabilité $p(A_i|S_1^{i-1}, A_1^{i-1}, T_1^m)$, il existe plusieurs modèles pour l'estimer comme les modèles d'IBM (Brown et al., 1993). Le premier modèle d'IBM suppose une distribution uniforme entre les m mots de T. Dans ce cas, cette probabilité sera calculée en utilisant l'équation 10.7.

$$p(A_i|S_1^{i-1}, A_1^{i-1}, T_1^m) = \frac{1}{m+1} \tag{10.7}$$

Donc, l'équation 10.5 peut être reformulée comme l'équation 10.8.

$$p(S|T) = \frac{1}{(m+1)^n} \sum_{A} \prod_{i=1}^{n} p(S_i|S_1^{i-1}, A_1^{i}, T_1^m) \tag{10.8}$$

Dans le modèle IBM2, la probabilité des alignements se base seulement sur la position actuelle i, le mot actuel A_i, la taille du texte source n et la taille du texte destinataire m. Cette probabilité est entrainée en comparant l'alignement juste avec le reste des alignements. Elle peut être formulée comme l'équation 10.9.

$$p(A_i|S_1^{i-1}, A_1^{i-1}, T_1^m) = p(A_i|i, n, m) \tag{10.9}$$

Donc, l'équation 10.5 peut être reformulée comme l'équation 10.10.

$$p(S|T) = \sum_{A} \prod_{i=1}^{n} p(A_i|i, n, m) p(S_i|S_1^{i-1}, A_1^{i}, T_1^m) \tag{10.10}$$

Dans le modèle HMM (Vogel et al., 1996), la probabilité de l'alignement d'un mot A_i se base sur l'alignement du mot précédent A_{i-1}. Cela peut être représenté par l'équation 10.11.

$$p(A_i|S_1^{i-1}, A_1^{i-1}, T_1^m) = p(A_i|A_{i-1}, m) \tag{10.11}$$

Donc, l'équation 10.5 peut être reformulée comme l'équation 10.12.

$$p(S|T) = \sum_{A} \prod_{i=1}^{n} p(A_i|A_{i-1}, m) p(S_i|S_1^{i-1}, A_1^{i}, T_1^m) \tag{10.12}$$

1.5 Approche par exemples

Des fois, on trouve des segments de la langue source qui sont toujours alignés avec d'autres en langue destinataire. Donc, on peut calculer la probabilité d'un segment (ensemble de mots consécutifs) par rapport un autre. Moses [2] (Koehn et al., 2007) est un système de traduction par exemples. Il entraîne quatre modèles statistiques :

— $\phi(S|T)$: une table de traduction des segments composée des segments S, segments T équivalents et les probabilités.
— LM : modèle de langage de langue destinataire
— $D(T,S)$: modèle de distorsion qui attribue un coût à chaque réorganisation des segments d'une phrase
— Pénalité de mots $W(T)$: pour qu'une traduction ne soit pas longue ou courte

Ces modèles sont utilisés pour estimer la probabilité d'une traduction T sachant un texte source S avec des poids en suivant l'équation 10.13. Pour estimer $\hat{T}$, **Beam search** est utilisé.

$$p(T|S) = \phi(S|T)^{poids_\phi} \times LM^{poids_{LM}} \times D(T,S)^{poids_D} \times W(T)^{poids_W} \tag{10.13}$$

1.6 Approche neuronale

La traduction automatique est un problème d'encodage-décodage ; on encode le texte source vers une représentation partagée qui sera décodée vers un texte destinataire. Donc, on peut utiliser un encodeur-décodeur basé sur un réseau de neurones récurrent ; il est appelé modèle sequence-to-sequence (seq2seq). L'encodeur a comme but d'encoder une phrase de la langue source S. Le résultat est une représentation du contexte sous forme d'un vecteur. Ce vecteur du contexte est décodé vers une phrase de la langue destinataire T. Formellement, la probabilité de génération du texte T sachant un texte S est décomposée comme indiqué par l'équation 10.14.

$$p(T|S) = p(t_1|S)p(t_2|S,t_1)p(t_3|S,t_1,t_2)\dots p(t_m|S,t_1\dots t_{m-1}) \tag{10.14}$$

Donc, la solution du problème revient à maximiser cette probabilité comme indiqué par l'équation 10.15.

$$\hat{T} = \arg\max_T \prod_{i=1}^{m} p(t_i|S,t_1\dots t_{i-1}) \tag{10.15}$$

Un système de traduction automatique bien connu, qui utilise l'approche neuronale, est Google Translate [3] (Wu et al., 2016). La figure 10.7 représente l'architecture du système de traduction automatique neuronale de Google. L'encodeur se compose de huit couches **LSTM** dont la première est un **Bi-LSTM** afin de capturer le contexte future. Le texte d'entrée $X = x_1 \dots x_n$ sera encodé comme une séquence $\bar{x}_1 \dots \bar{x}_n$. Le décodeur, lui aussi, se compose de 8 **LSTMs**. Soit y_{i-1} la sortie passée du décodeur, la prochaine entrée est calculée en utilisant un mécanisme d'attention. On passe chaque vecteur $\bar{x}_t$ combiné avec y_{i-1} par un réseau de neurone à propagation avant avec une seule couche cachée, appelé $AttentionFunction$ pour avoir un nouveau vecteur $s_t = AttentionFunction(y_{i-1}, \bar{x}_t)$. Une fois tous les n vecteurs sont encodés, on applique une fonction Softmax sur eux. Ensuite, le résultat p_t est utilisé comme pondération du vecteur $\bar{x}_t$ dans une somme pondérée de tous les n vecteurs comme indiqué par l'équation 10.16

$$a_i = \sum_{t=1}^{n} p_t \cdot \bar{x}_t \quad \text{où} \quad p_t = \frac{e^{s_t}}{\sum_{j=1}^{n} e^{s_j}} \tag{10.16}$$

2. Moses : http://statmt.org/moses/ [visité le 2021-09-16]
3. Google Translate : https://translate.google.com/ [visité le 2021-09-16]

Le vecteur résultat est utilisé comme entrée du décodeur pour générer le mot suivant y_i. Afin de décoder la sortie, les auteurs utilisent **Beam search**.

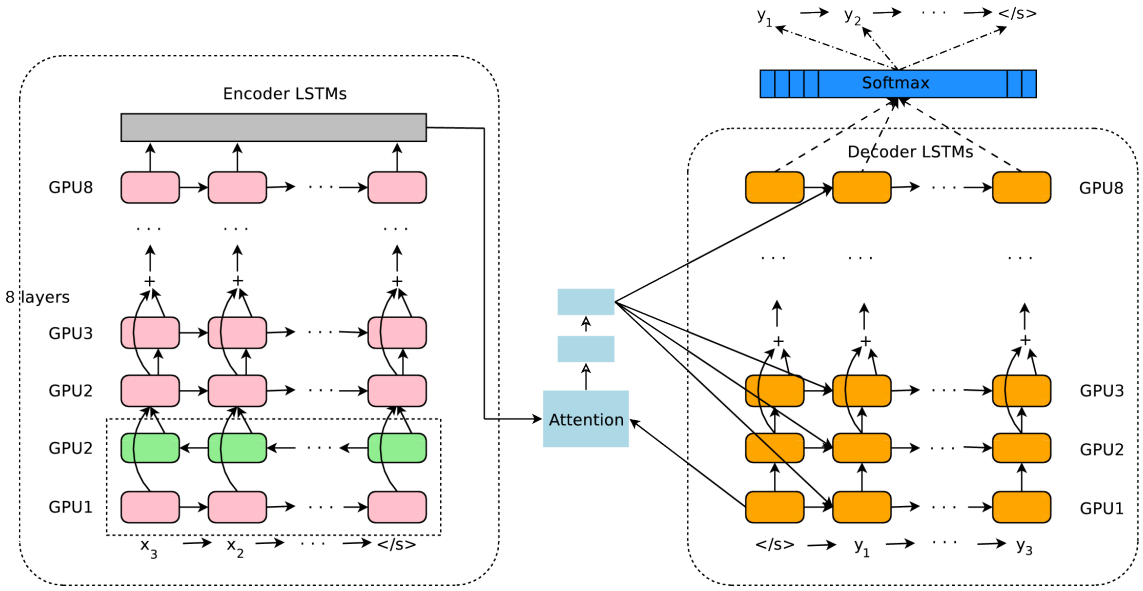


Figure 10.7 : *Architecture du système de traduction automatique neuronale de Google (Wu et al., 2016)*

Un autre système qui suit l'approche neuronale est OpenNMT [4] (Klein et al., 2017) dont l'architecture est indiquée dans la figure 10.8. Il utilise un modèle **seq2seq** avec le mécanisme d'attention.

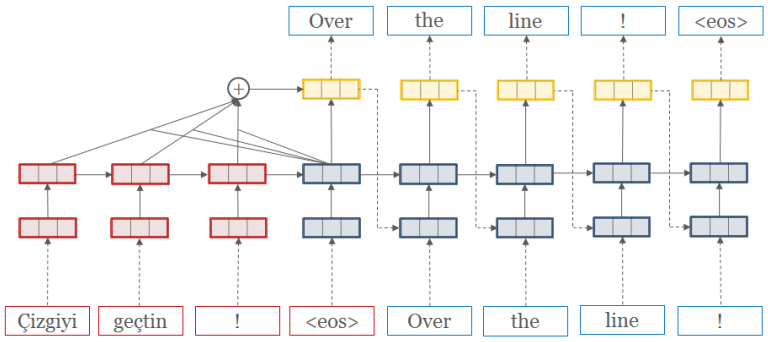


Figure 10.8 : *Architecture du système de traduction automatique neuronale OpenNMT (Klein et al., 2017)*

2 Résumé automatique

Le résumé automatique consiste à transformer un texte (images, vidéos ou un son) d'une forme longue vers une forme réduite (plus concise). Cette tâche est motivée par le besoin de gagner du temps de lecture et de traitement des grandes quantités d'informations. Un système de résumé automatique peut être classifié en utilisant plusieurs critères ; il peut appartenir à plusieurs classes au même temps. Ces critères sont regroupées en trois catégories : document d'entrée, but et document de sortie (Hovy et Lin, 1998 ; Sparck Jones, 1999). Cette classification est représentée par la figure 10.9.

Selon de document d'entrée, une méthode peut être classifiés en utilisant trois critères : l'unité, la spécialité et la forme. Un système de résumé automatique peut être mono-document (un document en entrée) ou multi-documents (plusieurs documents en entrée). Il peut être spécialisé à un domaine (Ex. médecine)

4. OpenNMT : https://opennmt.net/ [visité le 2021-09-19]

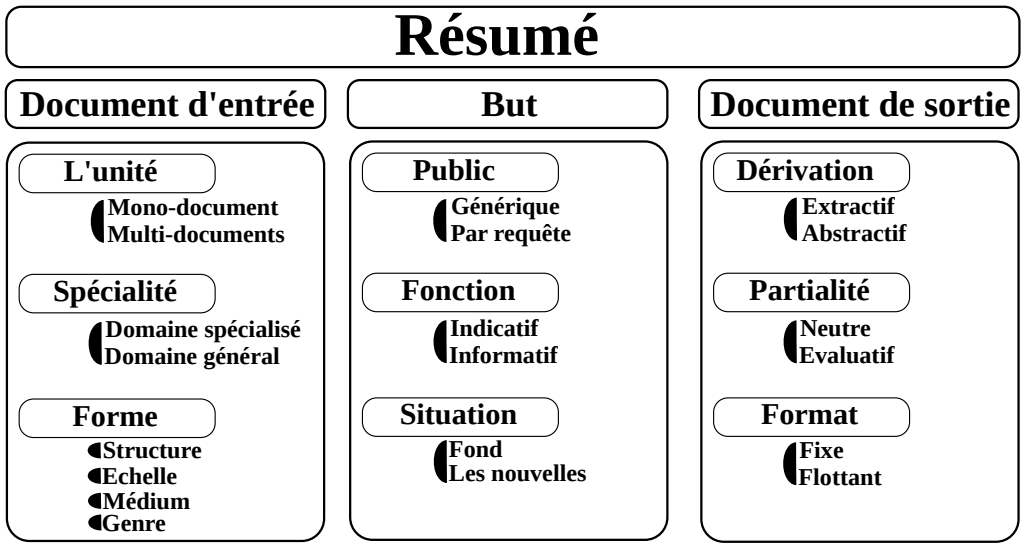


Figure 10.9 : *Classification des méthodes de résumé automatique selon Hovy et Lin (1998) ; Sparck Jones (1999)*

ou général (n'importe quel domaine). La forme d'un document d'entrée comporte plusieurs critères : la structure (document structuré ou non), l'échelle (taille du document : un tweet, un livre, etc.), le médium (texte, audio, vidéo, image) ou le genre (nouvelles, interviews, romans, etc.).

Selon le but, une méthode peut être classifiés en utilisant trois critères : le public, la fonction et la situation. Un système de résumé automatique peut être par requête (utiliser une requête utilisateur afin de générer le résumé) ou générique (utiliser le sujet du document afin de générer le résumé). Il peut être indicatif (une description globale du document) ou informatif (l'information essentielle du document). Il peut générer le fond (tout ce qu'est important dans le document) ou juste les nouvelles (tout ce qui est nouveau).

Selon le document de sortie, une méthode peut être classifiées en utilisant trois critères : la dérivation, la partialité et le format. Un système de résumé automatique peut être extractif (extraire des unités comme les phrases pour avoir un résumé) ou abstractif (générer un nouveau texte avec des nouveaux mots). Il peut être partiel (il n'ajoute aucune opinion au résumé) ou évaluatif (il ajoute des opinions au résumé). Le format du résumé peut être fixe (la même structure des résumés) ou flottant (des structures différentes selon des paramètres utilisateur).

Selon le système de résumé voulu, on peut décider une approche. Il existe plusieurs classifications des approches, parmi ces classifications :

— Classification de Nenkova et McKeown (2012) : les méthodes sont classifiées selon leur représentation en deux classes. Les méthodes qui se basent sur le sujet : mots du sujet, fréquences, analyse sémantique latente, modèles de sujets bayésiens, clustering. Les méthodes qui utilisent des indicateurs : par graphes, apprentissage automatique.
— Classification de Lloret et Palomar (2012) : les méthodes sont classifiées selon les techniques utilisées. Elles peuvent être : statistique, par graphes, basée discours ou par apprentissage automatique.
— Classification de Aries et al. (2019) : elle est similaire à la classification passée, mais elle regroupe les méthodes selon l'utilisation des ressources (puissance de calcul et données). Elles peuvent être : statistique, par graphes, linguistique ou par apprentissage automatique.

2.1 Approche statistique

Dans cette approche, les unités du texte (généralement, des phrases) sont attribuées un score de pertinence selon des critères statistiques. Parmi ces critères, on peut utiliser la fréquence des mots. L'équation 10.17 représente un exemple d'un score d'une phrase s_i basé sur TF-IDF.

$$Score_{\text{TF-IDF}}(s_i) = \sqrt{\sum_{w_{ik} \in s_i} (\text{TF-IDF}(w_{ik}))^2} \tag{10.17}$$

La position de la phrase est un bon critère de sa pertinence ; les phrases au début et à la fin parient plus importantes. L'équation 10.18 représente un score attribué à la phrase s_i d'un document D en utilisant sa position.

$$Score_{\text{pos}}(s_i) = \max(\frac{1}{i}, \frac{1}{|D| - i + 1}) \tag{10.18}$$

La taille de la phrase peut être utilisée comme critère ; on peut fixer une taille maximale ou minimale pour les phrases acceptées dans le résumé. L'équation 10.19 représente un score attribué à la phrase s_i en utilisant une taille minimale L_{min}.

$$Score_{\text{taille}}(s_i) = \begin{cases} 0 & si \quad (L_i \geq L_{min}) \\ \frac{L_i - L_{min}}{L_{min}} & sinon \end{cases} \tag{10.19}$$

Les mots des titres et des sous-titres sont des indicateurs de la pertinence d'une phrase. Soit T un titre, le score d'une phrase s_i peut être calculé en utilisant la fréquence des mots tf dans le document selon l'équation 10.20.

$$Score_{titre}(s_i) = \frac{\sum_{e \in T \cap s_i} \frac{tf(e)}{tf(e)+1}}{\sum_{e \in T} \frac{tf(e)}{tf(e)+1}} \tag{10.20}$$

Il y a plusieurs formulations de ces critères ; pas seulement celles mentionnées ici. Aussi, il existe d'autres critères comme : Centroid, Frequent itemsets, Analyse sémantique latente, etc.

Parmi les méthodes statistiques, on peut mentionner la méthode TCC (*Topic Clustering and Classification*) proposée par Aries et al. (2013) et implémentée dans un système appelé AllSummarizer [5]. L'architecture de cette méthode est illustrée dans la figure 10.10. L'idée est de regrouper les phrases comme sujets en utilisant une méthode de regroupement ; une phrase peut appartenir à plusieurs sujets. Pour ce faire, la similarité cosinus et un seuil de regroupement (Th) sont utilisés. Chaque phrase est attribuée un score qui indique combien elle peut représenter tous les sujets du document. Naive Bayes est utilisé pour apprendre les propriétés de chaque sujet et noter les phrases. On utilise un ensemble des caractéristiques f sur la phrase : TF (Uni-gramme, Bi-gramme), la position et la taille avant et après le pré-traitement. Une phrase s_i est jugée comme représentative du cluster c_j en utilisant une caractéristique f_k selon le score indiqué par l'équation 10.21.

$$Score(s_i, c_j, f_k) = 1 + \sum_{\phi \in s_i} P(f_k = \phi | s_i \in c_j) \tag{10.21}$$

Une phrase s_i est jugée pertinente si elle peut représenter tous les sujets (clusters) c selon toutes les caractéristiques f. Le score d'une phrase en se basant sur cette intuition peut être formulé par l'équation 10.22.

$$Score(s_i, \bigcap_j c_j, F) = \prod_j \prod_k Score(s_i, c_j, f_k) \tag{10.22}$$

Une autre méthode statistique est celle proposée par Oufaida et al. (2015). Les mots sont représentés en vecteurs en utilisant un **embedding** pré-entraîné (Polyglot). Premièrement, on commence par clustering

5. AllSummarizer : https://github.com/kariminf/allsummarizer [visité le 2021-09-16]

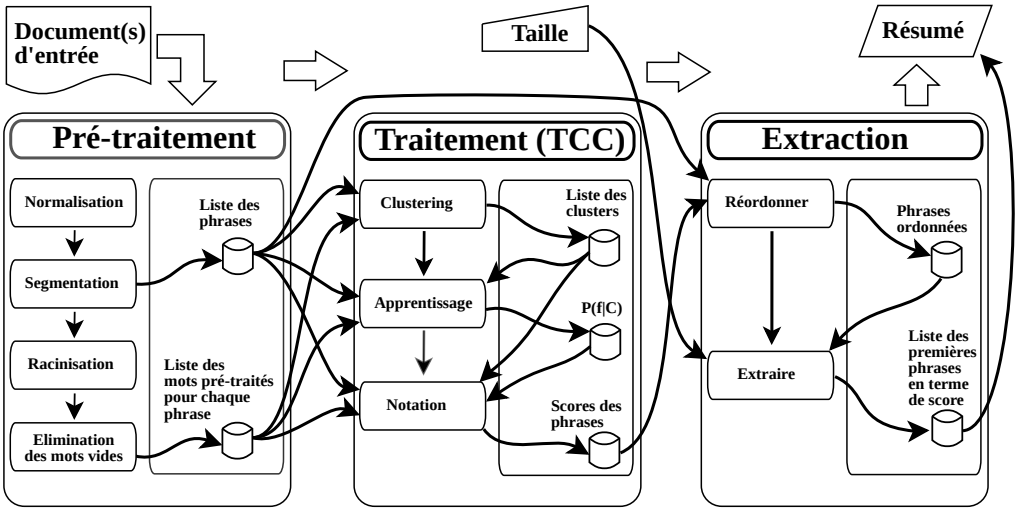


Figure 10.10 : *Architecture de la méthode TCC pour le résumé automatique statistique (Aries et al., 2013)*

des phrases pour extraire les sous-sujets du texte. Pour chercher le mot le plus similaire à un mot w_i dans une phrase S_2, on utilise une similarité entre les vecteurs (cosinus par exemple) comme indiqué par l'équation 10.23.

$$Match(w_i|S_2) = \arg\max_{w_j \in S_2} sim(Rep(w_i), Rep(w_j)) \tag{10.23}$$

Pour calculer la similarité entre deux phrases S_1 et S_2, on peut utiliser la fonction de matching comme précisé dans l'équation 10.24.

$$Sim(S_1, S_2) = \frac{\sum_{w_i \in S_1} Match(w_i|S_2) + \sum_{w_j \in S_2} Match(w_j|S_1)}{|S_1| + |S_2|} \tag{10.24}$$

Cette similarité est utilisée pour regrouper les phrases en sujets. Afin d'attribuer un score à une phrase, on utilise les scores des termes qui lui composent. Le score des termes est calculé selon une méthode appelée mRMR (Minimum Redundancy Maximum Relevance). On commence par la création d'une représentation termes/phrases indiquant la fréquence des termes dans chaque phrase (voir chapitre 6). Dans ce cas, un terme est représenté par un vecteur des phrases. Étant donné deux termes X et Y, on calcule l'information mutuelle comme indiqué par l'équation 10.25.

$$I(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \tag{10.25}$$

Les probabilités sont calculées par rapport aux clusters des phrases. La pertinence d'un terme T_i peut être calculée en se basant sur son information mutuelle avec la représentation cluster/phrases H par l'équation 10.26.

$$Pertinence(T_i) = I(T_i, H) \tag{10.26}$$

Sa redondance est calculée par rapport au résumé R par l'équation 10.27.

$$Redondance(T_i) = \frac{1}{|R|} \sum_{T_j \in R} I(T_i, T_j) \tag{10.27}$$

Le score final d'un terme peut être calculé en utilisant deux méthodes : MID ou MIQ (voir l'équation 10.28).

$$MID \equiv \max_{t \in T} Pertinence(t) - Redondance(t), \quad MIQ \equiv \max_{t \in T} Pertinence(t)/Redondance(t) \tag{10.28}$$

Le vecteur contenant tous les scores des termes est référencé par V_{mRMR}. Le score d'une phrase est sa similarité avec ce vecteur ; la plus similaire est ajoutée au résumé. Lors de l'ajout, on décrémente les scores des termes qui composent cette phrase dans le vecteur V_{mRMR}.

2.2 Approche par graphes

Dans cette approche, les phrases sont représentés comme nœuds d'un graphe $G(V, A)$; les arcs représentent les similarités entre ces phrases. Une méthode pour sélectionner les phrases pertinentes est d'utiliser les **propriétés du graphe**. Parmi les propriétés, on peut citer "Bushy paths" qui note une phrase s_i en se basant sur le nombre des arcs qui la relient avec les autres phrases. Ce score est exprimé par l'équation 10.29.

$$Score_{\#arcs}(s_i) = |\{s_j : a(s_i, s_j) \in A / s_j \in S, s_i \neq s_j\}| \tag{10.29}$$

Une autre propriété s'appelle "Aggregate Similarity" qui note une phrase s_i en se basant sur la somme des poids de ses arcs comme indiqué dans l'équation 10.30.

$$Score_{aggregate}(s_i) = \sum_{(s_i,s_j)\in E} sim(s_i, s_j) \tag{10.30}$$

Une autre approche est d'utiliser des **méthodes itératives** en mettant à jours les scores des nœuds par rapport aux voisins jusqu'à arriver à un état d'équilibre. Parmi les méthodes qui se basent sur cette technique, on peut citer TextRank (Mihalcea et Tarau, 2004). Le poids WS d'un nœud V_i est calculé selon l'équation 10.31.

$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j) \tag{10.31}$$

Le facteur d est fixé en général à 0.85. Le poids w_{ij} d'un arc entre les phrases S_i et S_j est calculé selon l'équation 10.32.

$$w_{ij} = \frac{|\{w_k \,/\, w_k \in S_i \text{ and } w_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)} \tag{10.32}$$

Les graphes ne sont pas utilisés seulement pour attribuer des scores aux phrases, mais aussi on peut les utiliser pour faire une pré-sélection. Dans la méthode SSF-GC (*Sentence statistical features - graph cumulative score*)(Aries et al., 2021), on commence par noter les phrases selon des caractéristiques statistiques (voir la figure 10.11). Ensuite, on crée un graphe en utilisant les similarités entre ces phrases. Le graphe est simplifié afin de garder les phrases les plus probables à être sélectionnées dans le résumé. Ensuite, on améliore le score des phrases en utilisant les propriétés du graphe.

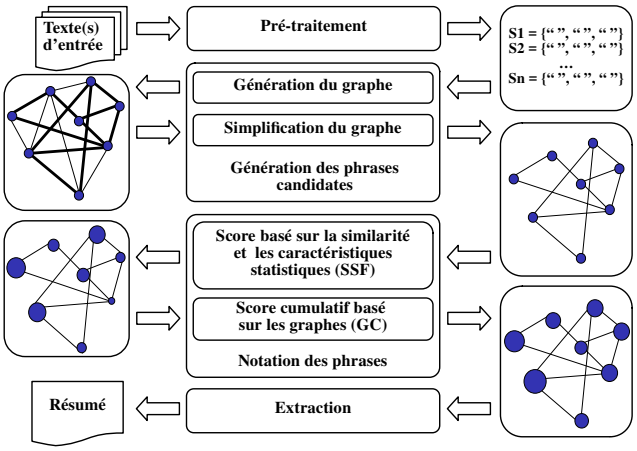


Figure 10.11 : *Architecture de la méthode SSF-GC pour le résumé automatique statistique (Aries et al., 2021)*

Un graphe $G(V, E)$ est construit en utilisant les phrases et leurs similarités cosinus sur les fréquences des mots. Le graphe est simplifié par l'élimination des nœuds faibles. Un nœuds est jugé faible s'il satisfait la

propriété exprimée par l'équation 10.33. $MImpN(v_i)$ est le nombre des voisins les plus importants ; ceux qui ont une similarité avec v_i supérieure à un seuil donné $Threshold$.

$$noeud\_faible(v_i) = (\sum_{(v_i,v_j)\in E} w_{ij} < \frac{1}{MImpN(v_i)}) \tag{10.33}$$

Les arcs faibles sont, aussi, éliminés selon l'équation 10.34

$$arc\_faible(v_i, v_j) = (w_{ij} < \frac{Threshold}{MImpN(v_i)}) \tag{10.34}$$

Une fois le graphe est simplifié, on score chaque phrase selon des caractéristiques statistiques $f_i \in F$ (vues précédemment). Ces scores sont considérés comme des probabilités, d'où la probabilité totale est la multiplication entre leurs probabilités comme indiqué par l'équation 10.35.

$$SSF(s_i/F) = \prod_{f_i\in F} score(s_i/f_i) \tag{10.35}$$

Une des méthodes pour améliorer le score SSF d'une phrase s_i est d'utiliser la somme des scores SSF de ses voisins pondérés par les similarités, comme indiqué par l'équation 10.36.

$$GC1(s_i) = SSF(s_i) + \sum_{(s_i,s_j)\in E} sim(s_i, s_j) * SSF(s_j) \tag{10.36}$$

Le graphe peut être aussi utilisé lors de l'ajout d'une phrase au résumé. Une des méthodes d'extraction proposées, il y a une qui minimise l'ordre *ord* de la similarité de la phrase s_i à ajouter avec la dernière phrase ajouté $dernier_{e4}$ afin de réduire la redondance. Elle vise à minimiser l'ordre inverse *iord* du score GC de la phrase, et donc maximiser son score de pertinence. L'équation 10.37 représente la méthode du calcul de la phrase à ajouter au résumé.

$$suiv_{e4} = \arg\min_i(iord\ gc(s_i) + ord\ sim(dernier_{e4}, s_i)) \text{ où } (dernier_{e4}, s_i) \in E \tag{10.37}$$

2.3 Approche linguistique

On peut utiliser une liste des mots qui sont pertinents au sujet, comme "significant", "impossible", etc. Ces mots sont appelés **Mots de sujet** et peuvent être utilisés pour noter les phrases. Edmundson (1969) a défini deux listes : Bonus (mots positivement pertinents) et Stigma (mots négativement pertinents). Le score d'une phrase s_i en se basant sur ces deux listes est indiqué par l'équation 10.38.

$$Score_{cue}(s_i) = \sum_{w\in s_i} cue(w) \text{ où } cue(w) = \begin{cases} b > 0 & \text{si } (w \in Bonus) \\ \delta < 0 & \text{si } (w \in Stigma) \\ 0 & sinon \end{cases} \tag{10.38}$$

Une autre méthode est d'utiliser des **indicateurs** qui sont des structures qui impliquent que la phrase les contenant a une chose importante à propos du sujet. Par exemple, "the principal aim of this paper is to investigate ...". La figure 10.12 représente un patron défini par Paice (1981) afin de noter les phrases. **[x]** veut dire qu'il existe x mots entre ce mot et le mot précédent. Le score d'une phrase est incrémenté avec la valeur **+y**. Les mots optionnels sont annotés par **?**.

D'autres méthodes utilisent des anaphores ou des représentations sémantiques (Ex. Wordnet) (**Co-référence**) afin d'améliorer les scores des phrases. La **structure rhétorique** peut être utilisée pour noter des phrases ou des syntagmes.

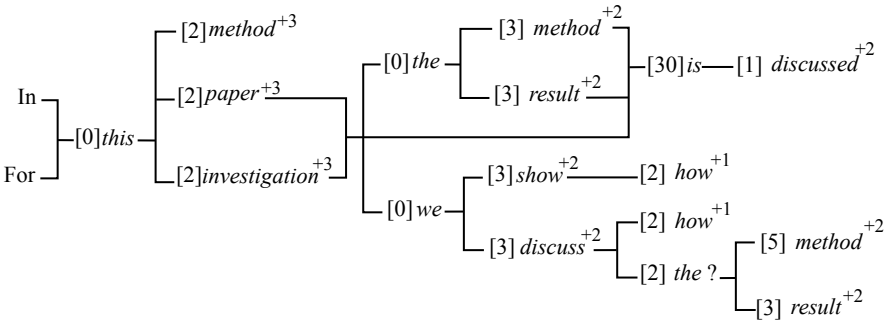


Figure 10.12 : *Exemple d'un patron simplifié (Paice, 1981).*

2.4 Approche par apprentissage automatique (ML)

En utilisant des **caractéristiques** sur les phrases, le document, etc., on peut apprendre à résumer un document en utilisant un algorithme de Machine Learning (ML). On peut soit régler des hyper-paramètres comme les poids des caractéristiques pour le score des phrases. Aussi, on peut utiliser l'apprentissage pour décider si une unité (phrase) appartient au résumé ou non (problème de classement). ML2ExtraSum [6] (Aries, 2020) est un système qui se base sur des caractéristiques définies manuellement afin d'estimer le score ROUGE-1 (une métrique pour l'évaluation des résumés). La figure 10.13 représente l'architecture de ce système. En entrée, on a plusieurs caractéristiques comme la liste des TF des mots de la phrase (*sent_tf_seq*), la liste des TF des mots du document (*doc_tf_seq*), la taille de la phrase (*sent_size*), etc. Ces caractéristiques sont transformées en utilisant un module de transformation configurable. Par exemple, on peut transformer les listes en scalaires. Un bloc de réseau de neurones à propagation avant est utilisé pour détecter la langue (en réalité, il va attribuer à chaque document un vecteur : une forme de clustering). Autres blocs sont utilisés pour calculer le score de la phrase en utilisant des critères : TF, Similarité, Taille et Position. Un bloc final est utilisé pour inférer le score ROUGE-1 en utilisant les scores intermédiaires.

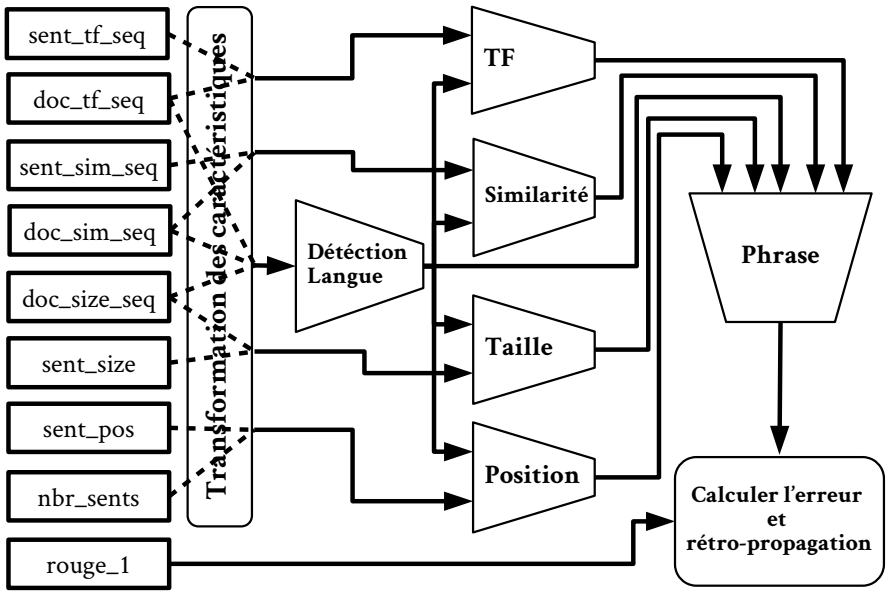


Figure 10.13 : *Architecture du système ML2ExtraSum qui utilise des caractéristiques avec ML (Aries, 2020)*

Une autre approche est d'identifier les concepts principaux à partir des documents et les liens entre eux pour avoir une hiérarchie. On appelle ça "**Bayesian topic models**". Daumé III et Marcu (2006) essayent

6. ML2ExtraSum https://github.com/kariminf/ML2ExtraSum [visité le 2021-09-16]

de générer des résumés en utilisant des requêtes utilisateurs (voir la figure 10.14). Pour ce faire, on essaye d'apprendre des modèles bayésiens à partir d'un ensemble D de K documents et un autre ensemble Q de J requêtes sur ces documents. Un modèle de langage général de l'anglais P^G est entraîné sur des documents génériques. Un autre modèle de langage des requêtes P^Q est entraîné sur Q. Le troisième modèle de langage P^D est entraîné sur les documents D ; c'est un modèle d'arrière-plan. $r[K, J]$ est une matrice booléenne, qui a la valeur 1 si le document $d \in D$ est pertinent à la requête $q \in Q$. Chaque mot w_{dsn} d'un document d, d'une phrase s et ayant la position n a une variable cachée z_{dsn} qui est un vecteur de taille $K + J + 1$. Ce vecteur contient un seul 1 et le reste de ses éléments est un 0. Il indique de quel document ce mot a été généré : d'un des K documents de D, d'un des J requêtes de Q ou d'un document générique. Du même, pour chaque phrase s d'un document d, on attribue un vecteur π_{ds} de taille $K + J + 1$. Ce vecteur contient le degré de croyance que la phrase a été générée d'un des documents D, Q ou l'anglais général. Il peut être utilisé pour décider la phrase qui est générée par D et Q et pas par un document quelque ou par seulement l'un des deux.

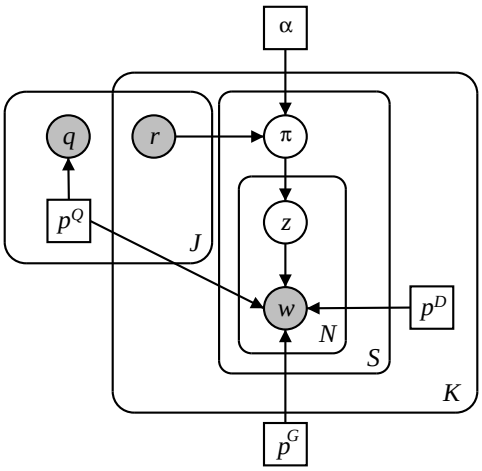


Figure 10.14 : *Représentation du réseau bayésien utilisé dans le résumé automatique par Daumé III et Marcu (2006)*

Une autre approche est d'utiliser des techniques de **deep learning**. Le système NAMAS [7] (Rush et al., 2015) utilise un réseau de neurone récurrent afin de générer un résumé à partir d'un petit texte. La figure 10.15 représente l'architecture du système qui sert à générer un résumé abstractif. Elle contient aussi un exemple d'un résumé généré à partir d'une phrase avec le degré d'attention pour chaque mot. La partie (a) représente un décodeur qui cherche le mot prochain du résumé y_{i+1} sachant la phrase en entrée x et les mots déjà générés pour le résumé $y_c \equiv [y_{i-c+1}, \ldots, y_i]$. La partie (b) représente un encodeur avec attention. Le mécanisme est similaire à celui présenté dans la traduction automatique.

Une autre méthode pour entraîner un système de résumé automatique est d'utiliser **reinforcement learning**. Il s'agit de l'utilisation des actions et des récompenses pour entrainer un système à générer des résumés. La figure 10.16 représente l'architecture du système de résumé proposé par Narayan et al. (2018) basé sur l'apprentissage par renforcement. Étant donné un document D avec n phrases, on veut extraire $m < n$ phrases pour construire le résumé. La sélection des phrases peut être formulée comme un problème de classement : classer une phrase comme appartenant au résumé (1) ou non (0).

On commence par décrire les deux encodeurs (phrases et document) et l'extracteur (décodeur). La phrase est encodée en utilisant des CNN sur la matrice contenant les mots (dans l'exemple, 7 mots encodés avec des vecteurs de 4 éléments). On utilise des filtres de déférentes tailles h (ici, 4 en bleu et 2 en rouge) plusieurs fois (ici, on a utilisé 3 différents filtres de même taille). Chaque filtre crée un vecteur $f \in \mathbb{R}^{k-h+1}$ où k est le nombre des mots dans la phrase (ici, le premier vecteur est de taille $7 - 4 + 1 = 4$, le deuxième

7. NAMAS : https://github.com/facebookarchive/NAMAS [visité le 2021-09-16]

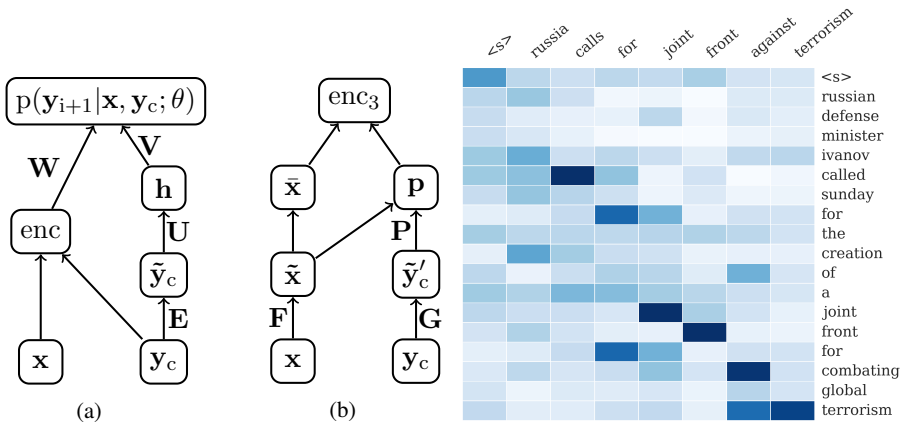


Figure 10.15 : *Architecture du système NAMAS utilisant deep learning pour le résumé automatique abstractif (Rush et al., 2015)*

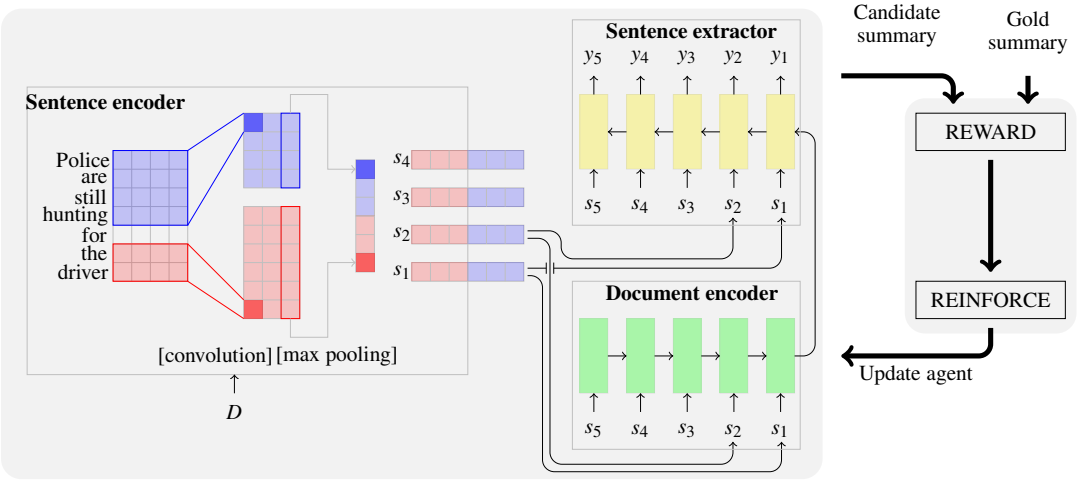


Figure 10.16 : *Architecture du système de Narayan et al. (2018) utilisant l'apprentissage par renforcement*

$7-2+1=6$). Le vecteur résultat est passé par un Max-Pool pour avoir une seule valeur. Dans l'exemple, on a utilisé trois filtres de taille 4 et trois de taille 2, donc on aura deux vecteurs de taille 3 qui sont concaténés à un seul vecteur représentant la phrase. Afin d'encoder le document, on passe les phrases par un réseaux récurrent **LSTM** en commençant par la représentation de la première phrase. L'extracteur est un **LSTM** qui prend une phrase s_i et estime une prédiction y_i en prenant en compte la représentation du document : $p(y_i|x_i, D)$.

Lors de l'entrainement, on peut sélectionner les m phrases d'un document D en utilisant la métrique ROUGE-1 avec un résumé manuelle comme phrases de références. On va annoter les phrases du document comme $y = y_1 \dots y_n$ où $y_i \in \{0, 1\}$; les phrases de références ont un label de 1. Le système peut être vu comme un agent qui utilise une politique $p(y_i|s_i, D, \theta)$ où θ représentent les paramètres du modèle. Il génère des labels $\hat{y} = \hat{y}_1 \dots \hat{y}_n$ où le résumé automatique est généré à partir des phrases les plus probables. L'agent est attribué une récompense (reward) r indiquant comment le résumé automatique est similaire à celui manuelle. Cette récompense est formulée comme la moyenne des F1-scores des métriques ROUGE-1, ROUGE-2 et ROUGE-L. L'agent est mis à jours en essayant de minimiser l'espérance négative de la récompense comme indiqué par l'équation 10.39 où p_θ veut dire $p(y|D, \theta)$.

$$L(\theta) = -\mathbb{E}_{\hat{y} \sim p_\theta}[r(\hat{y})] \tag{10.39}$$

Le gradient peut être calculé comme indiqué par l'équation 10.40.

$$\begin{aligned}\nabla L(\theta) &= -\mathbb{E}_{\hat{y}\sim p_\theta}[r(\hat{y}) \nabla \log p(\hat{y}|D,\theta)] \\ &\approx -r(\hat{y}) \nabla \log p(\hat{y}|D,\theta) \\ &\approx -r(\hat{y}) \sum_{i=1}^{n} \nabla \log p(\hat{y}_i|s_i,D,\theta)\end{aligned} \tag{10.40}$$

3 Questions-Réponses

Un système de questions-réponses est un système interactif qui génère des réponses aux questions des utilisateurs. La motivation derrière un tel système est claire : aider les utilisateurs à trouver des réponses. La figure 10.17 représente une classification des systèmes de questions-réponses. Un tel système peut être générique (répondre à n'importe quelle question) ou spécifique à un domaine (la médecine est parmi les domaines les plus ciblés). Il existe plusieurs approches pour implémenter un tel système : par Recherche d'Information (RI), à base de connaissance ou à base des modèles de langage. Ces trois approches vont être présentées dans cette section avec quelques méthodes comme exemples.

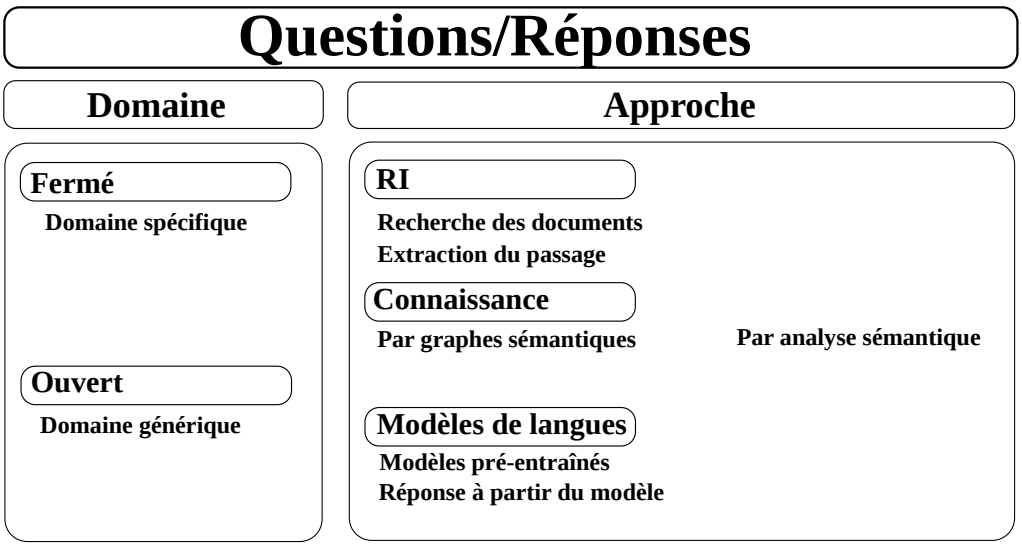


Figure 10.17 : *Classification des systèmes de Questions/Réponses*

3.1 Approche par RI

Dans l'approche par RI, on utilise un système de recherche d'information afin de récupérer les documents pertinents à la requête, ensuite on extrait le passage contenant la réponse. La figure 10.18 représente une architecture d'un système de questions/réponses par RI. Le système contient trois modules principaux : traitement de la requête, recherche des documents/passages et extraction de la réponse.

Dans le traitement de la requête, on applique deux tâches : formulation de la requête et détection du type de réponse. La formulation de la requête concerne les tâches vues dans le chapitre 2 : séparation des mots, suppression des mots vides et radicalisation. Le résultat est un ensemble des mots clés qui sont utilisés dans la recherche des documents. La réponse peut être : une personne, une place, une organisation, une cause, etc. Détecter le type de la réponse attendue va guider le système vers la bonne réponse (à extraire). Cela peut être accompli en utilisant une taxonomie comme Wordnet.

La recherche des documents se fait en utilisant les mots clés et l'index inversé des documents. Une fois les

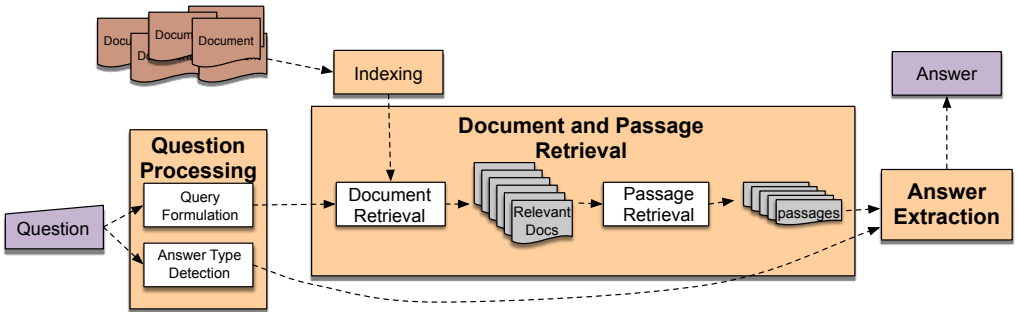


Figure 10.18 : *Architecture d'un système de questions/ réponses par RI (Jurafsky et Martin, 2019)*

documents les plus pertinents (en se basant sur un score) sont retournés, on les divisent en passages (paragraphes ou phrases). Afin de rechercher les passages, on peut appliquer la détection des entités nommées et filtrer ceux qui ne contiennent pas le type de la réponse. Dans ce cas, la recherche d'une cause plutôt d'une entité nommées sera plus difficile. Une autre solution est d'utiliser un algorithme d'apprentissage automatique afin de noter les passages. Parmi les caractéristiques qu'on puisse utiliser : nombre des entités nommées du type recherché, nombre des termes de la question, la séquence la plus longue similaire à la question, le rang du document, etc.

Une fois les passages pertinents sont identifiés, on passe à l'extraction de la réponse qui est une partie de ces passages. Une approche est d'utiliser un algorithme d'apprentissage automatique afin de détecter la réponse. Parmi les caractéristiques qui peuvent être utilisées : type de réponse et celui du syntagme, les mots clés de la question, la nouveauté (est ce qu'il y a un mot qui n'existe pas dans la question), la ponctuation (si la réponse est suivie par un virgule, point, etc.). Aussi, on peut utiliser des patrons comme "<REP>comme <QES> ; " qui permettent de détecter des réponses comme "des désordres développementaux comme l'autisme." à la question "C'est quoi l'autisme ?".

En utilisant les réseaux de neurones, on peut implémenter un système de questions-réponses dans le contexte de la tâche de lecture/compréhension. Pour chaque mot du passage, on calcule la probabilité d'être le début de la réponse et la fin de la réponse en se basant sur la question. C'est une tâche d'annotation des séquences qui peut être résolue en utilisant la technique **IOB**. Une méthode est celle de Chen et al. (2017), illustrée dans la figure 10.19. On commence par encoder la question en passant les représentations **GloVe** de ces mots par un réseaux récurrent Bi-LSTM, les vecteurs sont combinés en utilisant une somme pondérée. Afin d'encoder le passage, on utilise la représentation **GloVe** des mots, un mécanisme d'attention basé sur les mots de la question, un indicateur si le mot a apparu dans la question (0 ou 1) et des indicateurs de catégorie grammaticale et/ou le type de l'entité nommée. Ces vecteurs sont passés par un réseaux **Bi-LSTM** afin d'extraire les **embeddings** des mots par rapport au passage. Chaque vecteur d'un mot du passage est comparé avec le vecteur de la requête en utilisant une similarité afin de décider si le mot appartient à la réponse.

Nous avons vu dans le chapitre 6 que **BERT** peut être utilisé dans des tâche d'annotation des séquences. Devlin et al. (2018) passent la question et le passage au modèle pré-entraîné **BERT** (voir la figure 10.20). On utilise deux représentations, de début (S) et de fin (E), qui sont multipliés par la représentation de chaque mot et passées par une fonction softmax sur l'ensemble de tous les mots afin d'avoir la probabilité de début et celle de fin.

3.2 Approche par connaissance

Lorsqu'on possède une base de données structurée, cette approche est la plus adéquate. Dans le cas d'une base de connaissance sous forme de graphe (BabelNet), on peut utiliser l'annotation sémantique (Entity

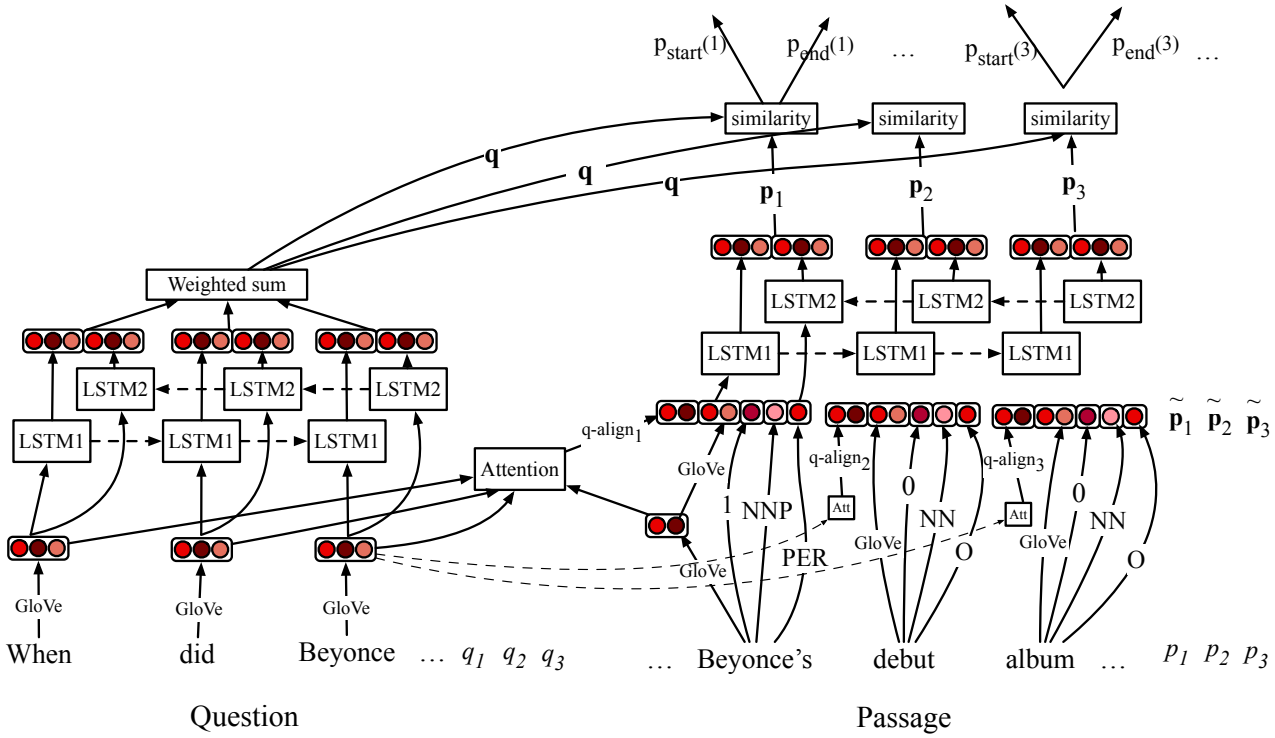


Figure 10.19 : *Extraction de la réponse par Bi-LSTM (Jurafsky et Martin, 2019)*

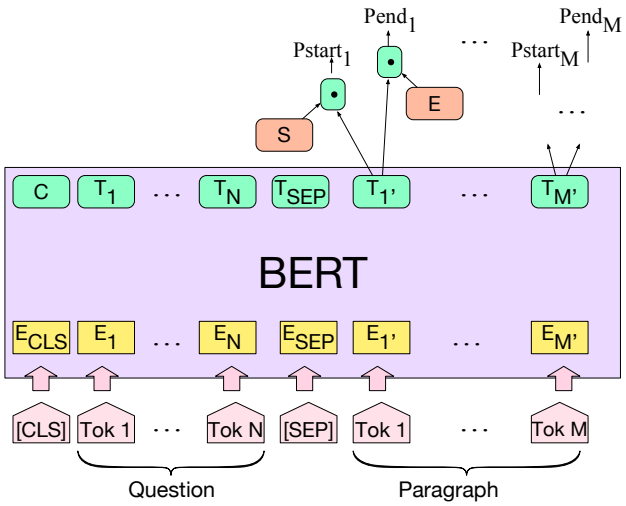


Figure 10.20 : *Extraction de la réponse par BERT (Jurafsky et Martin, 2019)*

linking). Ensuite, on doit déterminer le type de la relation recherchée (Ex. Place_naissance) afin de filtrer les entités. S'il y a plusieurs relations comme réponses, on peut calculer la similarité entre la question et la réponse (par exemple, en utilisant les **embeddings**). Une autre approche est en appliquant l'analyse sémantique sur la question afin de générer une forme structurée comme : lambda calcul, SQL, SPARQL, etc. Cette forme est utilisée afin d'interroger la base de données et récupérer la réponse. Un exemple des requêtes et leurs formes logiques est donnée dans la figure 10.21.

3.3 Approche par modèles de langage

Dans cette approche, on utilise un modèle de langage pré-entrainé. Ensuite, on le règle pour répondre aux questions à partir du modèle (pas besoin de passages). Dans (Roberts et al., 2020), on a entrainé un

Question	Logical form
What states border Texas?	$\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{texas})$
What is the largest state?	$\text{argmax}(\lambda x.\text{state}(x), \lambda x.\text{size}(x))$
I'd like to book a flight from San Diego to Toronto	SELECT DISTINCT f1.flight_id FROM flight f1, airport_service a1, city c1, airport_service a2, city c2 WHERE f1.from_airport=a1.airport_code AND a1.city_code=c1.city_code AND c1.city_name= 'san diego' AND f1.to_airport=a2.airport_code AND a2.city_code=c2.city_code AND c2.city_name= 'toronto'
How many people survived the sinking of the Titanic?	`(count (!fb:event.disaster.survivors fb:en.sinking_of_the_titanic))`
How many yards longer was Johnson's longest touchdown compared to his shortest touchdown of the first quarter?	ARITHMETIC diff(SELECT num(ARGMAX(SELECT)) SELECT num(ARGMIN(FILTER(SELECT))))

Figure 10.21 : *Exemple des formes logiques des questions (Jurafsky et Martin, 2020)*

modèle T5 sur la tâche de remplissage des textes absents (voir la figure 10.22). Ensuite, le modèle est réglé pour répondre aux questions sans saisir d'informations ou de contextes supplémentaires.

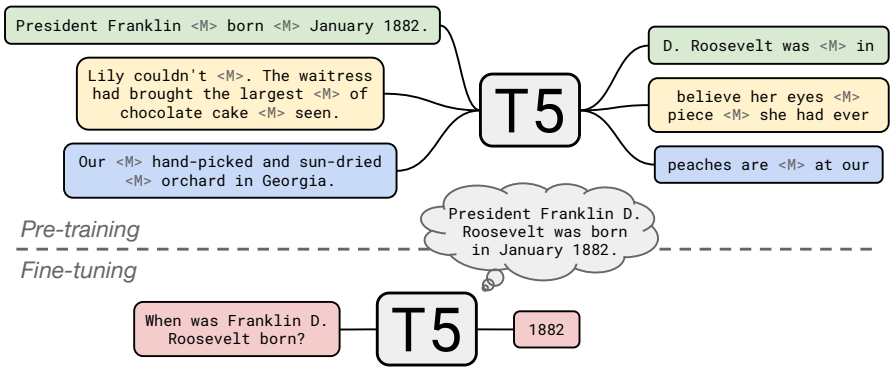


Figure 10.22 : *T5 comme modèle de réponse aux questions (Roberts et al., 2020)*

4 Systèmes de dialogue

Un système de dialogue est un système interactif qui communique avec les utilisateurs. Il peut être un agent orienté tâche ou un **chatbot** comme indiqué dans la figure 10.23. Les systèmes de dialogue orientés tâche visent à aider l'utilisateur à accomplir une tâche, comme la réservation de l'avion. Il existe deux types de ce système : frame-based et dialogue-state. Les **chatbots** ont comme but d'imiter la conversation humaine. Ils peuvent être des systèmes à base des règles, de la RI ou de la génération des réponses.

4.1 Orienté tâche : Frame-based

Un système de dialogue orienté tâche peut utiliser des cadres (frames) afin d'accomplir une tâche précise. Un cadre (frame) est une structure contenant des slots à remplir et des questions prédéfinies pour chaque slot. Par exemple, la figure 10.24 représente un exemple d'un cadre pour programmer un vol. Le système pose les questions correspondantes à un slot vide afin de le remplir. Lorsqu'on rempli un slot, on peut remplir les slots d'autres cadres en parallèle. Par exemple, le slot "RESERVATION_DATE" du cadre "HOTEL_RESERVATION" peut être rempli à partir du slot "ARRIVAL_DATE", ayant le même type, du cadre "FLIGHT_RESERVATION".

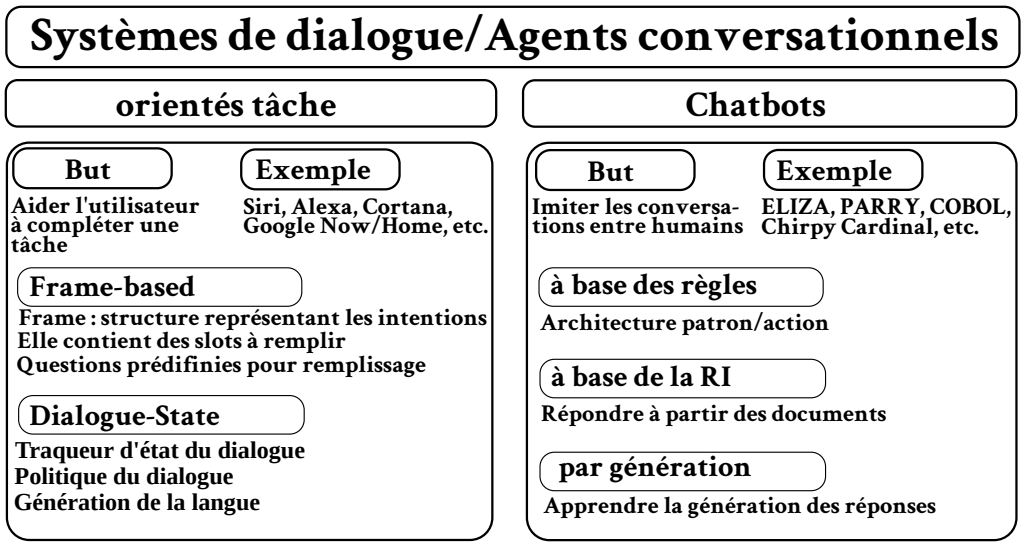


Figure 10.23 : *Classification des systèmes de dialogue*

Slot	Type	Question Template
ORIGIN CITY	city	"From what city are you leaving?"
DESTINATION CITY	city	"Where are you going?"
DEPARTURE TIME	time	"When would you like to leave?"
DEPARTURE DATE	date	"What day would you like to leave?"
ARRIVAL TIME	time	"When do you want to arrive?"
ARRIVAL DATE	date	"What day would you like to arrive?"

Figure 10.24 : *Exemple d'un cadre pour programmer un vol (Jurafsky et Martin, 2020)*

Lorsqu'il y a plusieurs domaines, on doit appliquer la **détection d'intention** pour savoir quel cadre à utiliser. Celle-ci est une tâche de classification qui vise à sélectionner le domaine (Ex. HOTEL vs. FLIGHT). Afin de remplir les slots, on peut utiliser une grammaire simplifiée. La figure 10.25 représente une grammaire sémantique et un arbre sémantique de la phrase "Show me flights from Boston to San Fransisco on Tuesday morning". Puisque le vocabulaire est limité et les phrases sont toujours les mêmes, une telle grammaire est facile à concevoir. En fait, on peut l'exprimer en utilisant des expressions régulières.

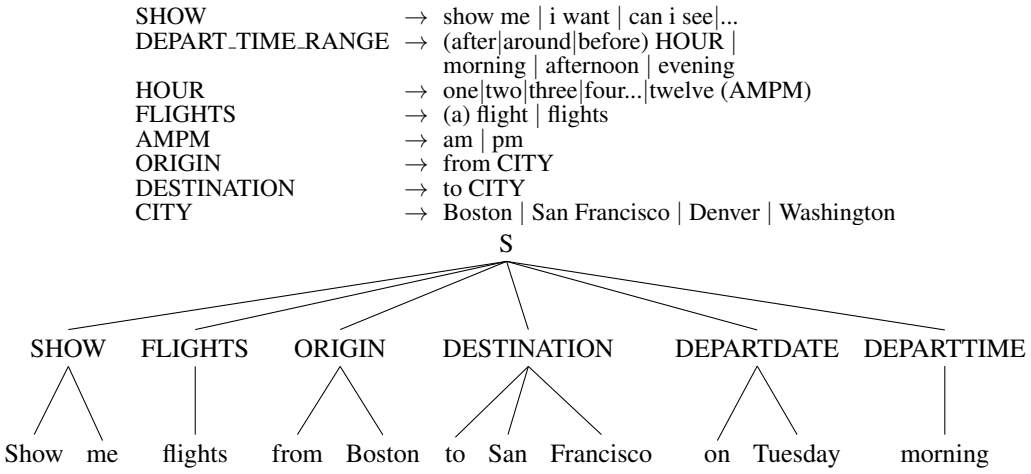


Figure 10.25 : *Exemple d'une grammaire sémantique et un arbre sémantique d'une phrase (Jurafsky et Martin, 2020)*

4.2 Orienté tâche : Dialogue-State

Un système de dialogue orienté tâche peut être conçu en utilisant les états de dialogue (Dialogue-state). Dans la figure 10.26, un système orienté tâche communique en utilisant la parole afin d'accomplir une tâche. Le système utilise des actes de dialogue afin de définir la conversation entre le système et l'utilisateur. Il garde les actes passés afin de décider quel est l'acte suivant. Comme les systèmes à base des cadres, il utilise des cadres avec des slots à remplir. En plus, il utilise une politique de dialogue pour l'aider à décider ce qu'il va faire où les questions qu'il peut poser. Cette politique est entaînée en utilisant un algorithme d'apprentissage afin de détecter l'action suivante $\hat{A}_i$ en se basant sur l'action précédente $\hat{A}_{i-1}$, le cadre précédent $Frame_{i-1}$ et le dernier tours système/utilisateur U_{i-1} (voir l'équation 10.41).

$$\hat{A}_i = \arg\max_{A_i \in A} P(A_i | Frame_{i-1}, A_{i-1}, U_{i-1}) \tag{10.41}$$

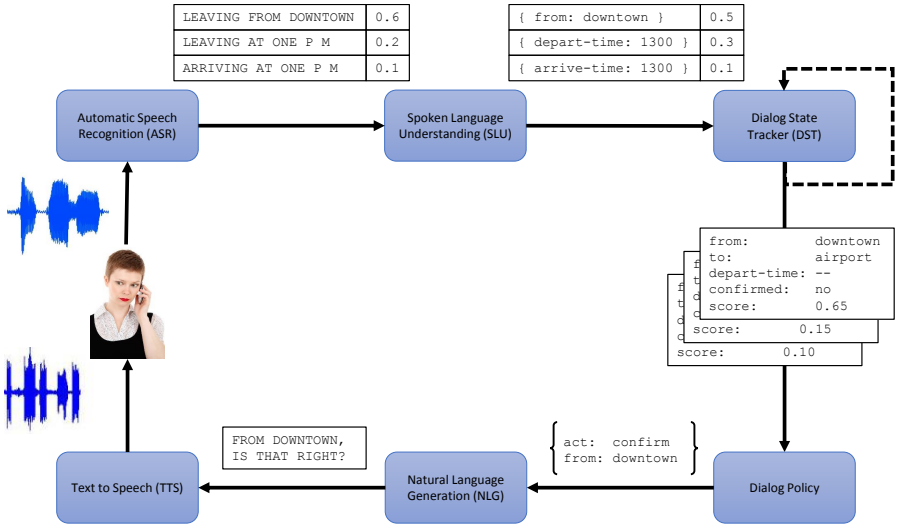


Figure 10.26 : *Architecture d'un système utilisant dialogue-state (Williams et al., 2016)*

Un acte de dialogue représente la fonction interactive de la phrase. La figure 10.27 représente un exemple des actes de dialogue du système de recommandation des restaurants HIS (Young et al., 2010). L'exemple indique quels sont les actes valides comme sortie du système et ceux valides comme entrée de l'utilisateur.

Tag	Sys	User	Description
HELLO($a=x,b=y,...$)	✓	✓	Open a dialogue and give info $a=x,b=y,...$
INFORM($a=x,b=y,...$)	✓	✓	Give info $a=x,b=y,...$
REQUEST($a,b=x,...$)	✓	✓	Request value for a given $b=x,...$
REQALTS($a=x,...$)	χ	✓	Request alternative with $a=x,...$
CONFIRM($a=x,b=y,...$)	✓	✓	Explicitly confirm $a=x,b=y,...$
CONFREQ($a=x,...,d$)	✓	χ	Implicitly confirm $a=x,...$ and request value of d
SELECT($a=x,a=y$)	✓	χ	Implicitly confirm $a=x,...$ and request value of d
AFFIRM($a=x,b=y,...$)	✓	✓	Affirm and give further info $a=x,b=y,...$
NEGATE($a=x$)	χ	✓	Negate and give corrected value $a=x$
DENY($a=x$)	χ	✓	Deny that $a=x$
BYE()	✓	✓	Close a dialogue

Figure 10.27 : *Actes de dialogue du système HIS (Young et al., 2010), figure prise de (Jurafsky et Martin, 2020)*

Un exemple d'un dialogue qui utilise les actes précédemment présentés est donné dans la figure 10.28. Le système commence par recevoir un acte "HELLO" et détecter qu'il s'agit d'une tâche de recherche et que le type recherché est une restaurant. Il lance un acte de confirmation de la requête en demandant quel est le plat préféré de l'utilisateur afin de recommander un restaurant.

	Utterance	Dialogue act
U:	Hi, I am looking for somewhere to eat.	`hello(task = find,type=restaurant)`
S:	You are looking for a restaurant. What type of food do you like?	`confreq(type = restaurant, food)`
U:	I'd like an Italian somewhere near the museum.	`inform(food = Italian, near=museum)`
S:	Roma is a nice Italian restaurant near the museum.	`inform(name = "Roma", type = restaurant, food = Italian, near = museum)`
U:	Is it reasonably priced?	`confirm(pricerange = moderate)`
S:	Yes, Roma is in the moderate price range.	`affirm(name = "Roma", pricerange = moderate)`
U:	What is the phone number?	`request(phone)`
S:	The number of Roma is 385456.	`inform(name = "Roma", phone = "385456")`
U:	Ok, thank you goodbye.	bye()

Figure 10.28 : *Exemple d'un dialogue du système HIS (Young et al., 2010), figure prise de (Jurafsky et Martin, 2020)*

Afin de remplir les slots, on doit déterminer le domaine et le slot visés par la phrase de l'utilisateur. Pour ce faire, on peut utiliser un algorithme d'apprentissage automatique afin de classer la phrase par intention, domaine et slot. Afin d'extraire les informations et remplir les slots, on peut utiliser l'étiquetage des séquences avec la technique **IOB**. Un exemple d'une telle méthode qui utilise **BERT** est illustré dans la figure 10.29.

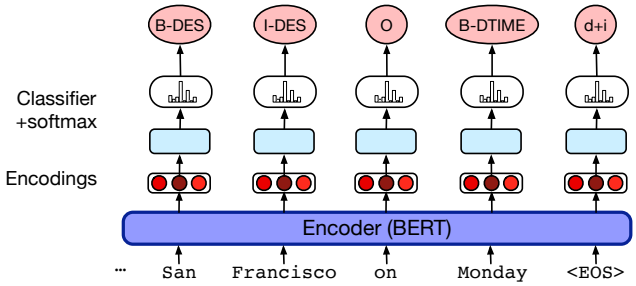


Figure 10.29 : *Exemple de remplissage des slots en utilisant BERT (Jurafsky et Martin, 2020)*

4.3 Chatbot à base des règles

Les **chatbots** ont comme but d'émuler une conversation humaine. Les premiers chatbots étaient à base de règles, comme le système ELIZA (Weizenbaum, 1966) qui émule une psychologue. Il se base sur des règles de type patrons/transformations. La règle [(.*) YOU (.*) ME]$_{[\text{Patron}]}$ → [WHAT MAKES YOU THINK I $2 YOU?]$_{[\text{Transformation}]}$ peut être utilisée pour répondre à You hate me → WHAT MAKES YOU THINK I HATE YOU?. Les patrons sont liés à une liste des mots. Le mot qui score le plus dans la phrase va déclencher plusieurs patrons. Parmi les, le patron le plus similaire à la phrase de l'utilisateur est utilisé.

4.4 Chatbot à base de la RI

Les **chatbots** à base de RI se basent sur des corpus de conversations C entre humains. Étant donnée une requête utilisateur q, on cherche la réponse $r \in C$ qui est plus similaire à la requête q comme indiqué par l'équation 10.42.

$$\text{Réponse}(q, C) = \arg\max_{r \in C} \frac{q.r}{|q||r|} \tag{10.42}$$

Les requêtes et les réponses peuvent être encodées en utilisant TF-IDF ou en utilisant les **embeddings**. Par exemple, en utilisant **BERT**, on peut encoder la requête comme un vecteur h_q et la réponse comme un vecteur h_r (la sortie "[CLS]") comme indiqué par l'équation 10.43.

$$h_q = BERT_Q(q)[CLS], \quad h_r = BERT_R(r)[CLS] \tag{10.43}$$

La réponse est celle qui maximise la similarité entre les deux vecteurs, comme indiqué par l'équation 10.44.

$$\text{Réponse}(q, C) = \arg\max_{r \in C} q.r \tag{10.44}$$

4.5 Chatbot par génération du texte

Un **chatbot** peut être modélisé comme un problème de génération du texte. On peut utiliser un encodeur/décodeur comme celui présenté dans la traduction automatique (voir la figure 10.30). Un mot de la réponse $\hat{r}_i$ est un mot du vocabulaire V qui est estimé en se basant sur la requête q et les mots précédemment générés comme dans l'équation 10.45

$$\hat{r}_i = \arg\max_{w \in V} p(w|q, r_1, \ldots, r_{t-1}) \tag{10.45}$$

On peut aussi utiliser des modèles de langage (comme **GPT**) entraînés sur des conversations.

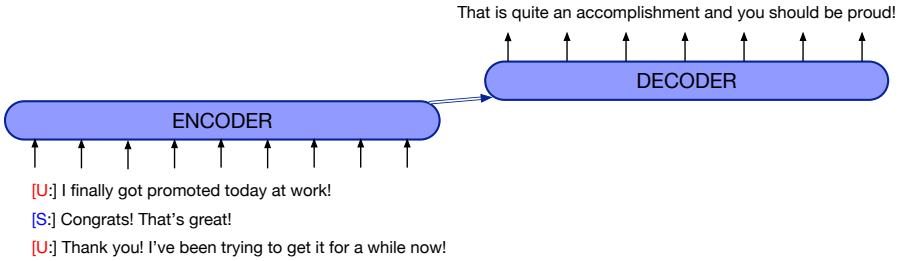


Figure 10.30 : *Exemple d'un chatbot à base d'un encodeur/décodeur (Jurafsky et Martin, 2020)*

5 Analyse des sentiments

L'analyse des sentiments consiste à identifier, extraire et quantifier l'état effectif et l'information subjective à partir d'un texte ou du parole. L'analyse des sentiments présente plusieurs tâches (voir la figure 10.31) : la subjectivité (est ce que le texte est subjectif ou objectif), la polarité (est ce que l'avis présent dans le texte est positif, négatif ou neutre), l'émotion (détecter l'émotion du texte : plaisir, colère, etc.). On peut, aussi, détecter l'opinion des utilisateurs sur plusieurs aspects. Par exemple, on peut détecter si les utilisateurs sont satisfaits par des aspects d'un téléphone comme : la taille, l'affichage, la batterie, etc. Il existe deux approches (qui peuvent être hybridées) pour analyser les sentiments : à base de connaissance et par apprentissage. Deux classifications plus détaillées peuvent être trouvées dans (Yue et al., 2019) et (Medhat et al., 2014).

5.1 Analyse des sentiments à base de connaissance

L'idée est d'attribuer un score de sentiment au texte en se basant sur des scores de ses mots. Seuls les mots indicateurs des sentiments qui peuvent être attribués des scores ; les reste sont neutres. Les adjectifs et les adverbes sont des bons indicateurs de sentiments. Par exemple, l'adjectif "bon" est un indicateur positif et donc l'adjectif "mauvais" est un indicateur négatif. Deux techniques sont utilisées pour identifier un mot : à base de dictionnaire et à base d'un corpus. En utilisant un dictionnaire comme **WordNet**, on peut enrichir une liste des indicateurs manuellement définie (synonymes, antonymes, etc.). En se basant sur un corpus, on peut enrichir une liste manuellement définie par la co-occurrence ; les mots qui sont souvent utilisés ensemble sont plus probables d'avoir la même polarité. Afin de calculer le score totale (phrase ou document), on fait la somme des scores des mots. Parmi les problèmes, la présence de la négation ; cela peut être réglé en utilisant la structure de la phrase comme **RST**.

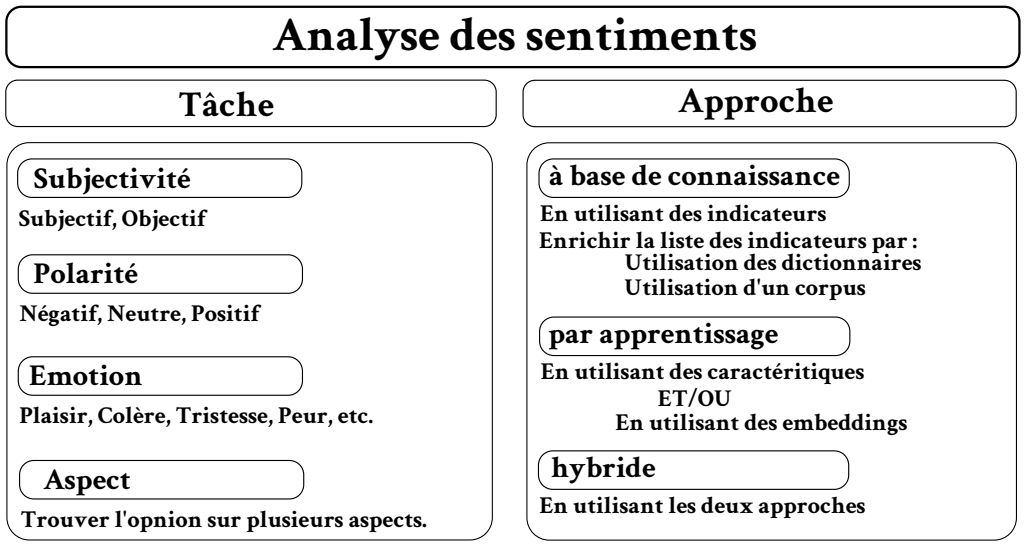


Figure 10.31 : *Classification des méthodes d'analyse de sentiment*

5.2 Analyse des sentiments par apprentissage automatique

On peut utiliser un algorithme d'apprentissage automatique avec des caractéristiques comme : la présence d'un mot et sa fréquence, les catégories grammaticales des mots, les mots et les syntagmes d'opinion, la négation, etc. Aussi, on peut utiliser les modèles de langage contextuelles afin d'apprendre la classe de sortie. La méthode la plus simple est d'utiliser un modèle **BERT** pré-entraîné. En entrée, on passe le texte et en sortie "[CLS]" on entraîne le modèle à estimer le sentiment. On peut enrichir le modèle en attachant un réseau à propagation avant (feed-forward) avec cette sortie. Aussi, on peut utiliser les derniers états cachés des mots avec des configurations comme CNN, RNN, etc.

5.3 Analyse des sentiments hybride

Dans cette approche, on essaye d'utiliser les deux approches précédentes. On va présenter deux méthodes proposées à l'ESI. On commence par la méthode de Bettiche et al. (2018) qui utilise la polarité des mots comme caractéristique d'un algorithme d'apprentissage. La figure 10.32 représente l'architecture de leur système qui vise à détecter la polarité des messages en dialecte algérienne sur les réseaux sociaux.

La méthode commence par l'enrichissement du vocabulaire vu que la dialecte écrite en lettres latins peut avoir plusieurs variations. Donc, pour regrouper les mots similaires, on utilise un ratio basé sur la distance de Lavenstein. Étant donné deux mots w_1 et w_2, ce ratio peut être calculé par l'équation 10.46.

$$Ratio = 1 - Levenstein(w_1, w_2)/(|w_1| + |w_2|) \tag{10.46}$$

Par exemple, Ratio(kolach, kollach) = 92%, Ratio(kolach, khlasse) = 38%.. Supposons que nous ayons un vocabulaire V_p positif et un autre V_n négatif qui sont définis manuellement. Pour chaque mot w du vocabulaire qui n'appartient pas à ces deux vocabulaires, on calcule son orientation sémantique comme indiqué par l'équation 10.47

$$SO(w) = \sum_{w_p \in V_p} PMI(w, w_p) - \sum_{w_n \in V_n} PMI(w, w_n) \tag{10.47}$$

Où

$$PMI(w_1, w_2) = \log \frac{p(w_1, w_2)}{p(w_1) * p(w_2)}$$

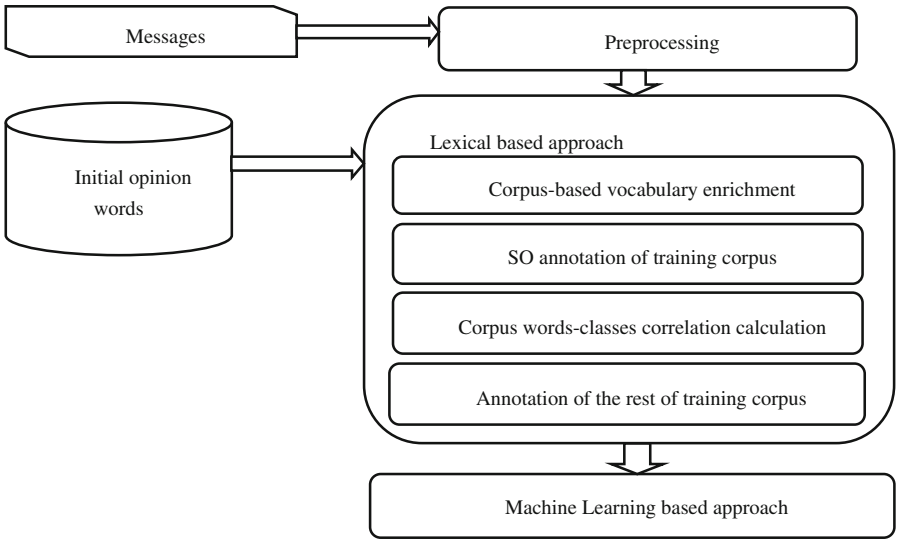


Figure 10.32 : *Architecture hybride proposée par Bettiche et al. (2018) pour détecter la polarité des messages en dialecte algérienne sur les réseaux sociaux*

La polarité du mot peut être calculée selon son orientation sémantique en utilisant l'équation 10.48

$$polarite(w) = \begin{cases} +1 & \text{SI } SO(w) > 0 \\ -1 & \text{SI } SO(w) < 0 \\ 0 & \text{SINON} \end{cases} \tag{10.48}$$

Les polarités des mots peuvent être utilisées comme caractéristiques d'un algorithme d'apprentissage automatique.

Une autre méthode hybride est celle proposée par Guellil et al. (2018) (voir la figure 10.33). En se basant sur un lexicon annoté en anglais (mots avec des scores de polarité), on traduit les mots vers l'arabe. Il faut prendre en considération que certains mots vont perdre leurs sens vu que l'utilisation des mots en anglais n'est pas garantie d'être équivalente au mots traduit dans les textes en arabe. En utilisant des messages de facebook, on essaye de les annoter selon les scores de polarités des mots ; chaque message est affecté à la classe positive ou négative. Ensuite, le corpus annoté automatiquement est utilisé afin d'entraîner un modèle de classification en utilisant des critères comme les **embeddings** des mots. J'ai une autre remarque concernant ça : le modèle entraîné va apprendre exactement la même fonction de celui par règle seulement avec des caractéristiques différentes ; il faut mieux annoter les messages manuellement.

6 Lisibilité

La lisibilité d'un texte peut être classifiée selon le niveau de difficulté ou le niveau du lecteur (voir la figure 10.34). Il existe deux approches : par règles ou par apprentissage automatique.

6.1 Formule

Les formules attribuent des scores aux textes en se basant sur des caractéristiques comme le nombre des mots, le nombre des phrases, etc. Un des scores utilisés pour juger la lisibilité des textes en anglais, on peut mentionner **Flesch-Kincaid Grade Level**. Cette formule se base sur le nombre des mots, le nombre des

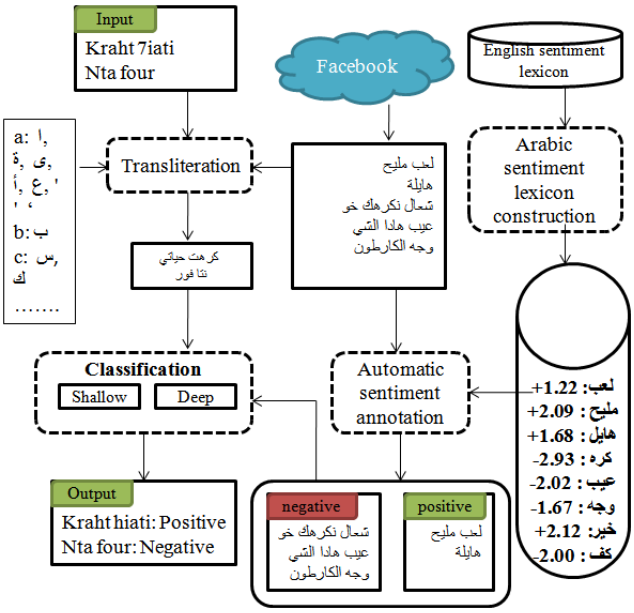


Figure 10.33 : *Architecture hybride proposée par Guellil et al. (2018) pour l'analyse des sentiments des messages en arabizi sur les réseaux sociaux*

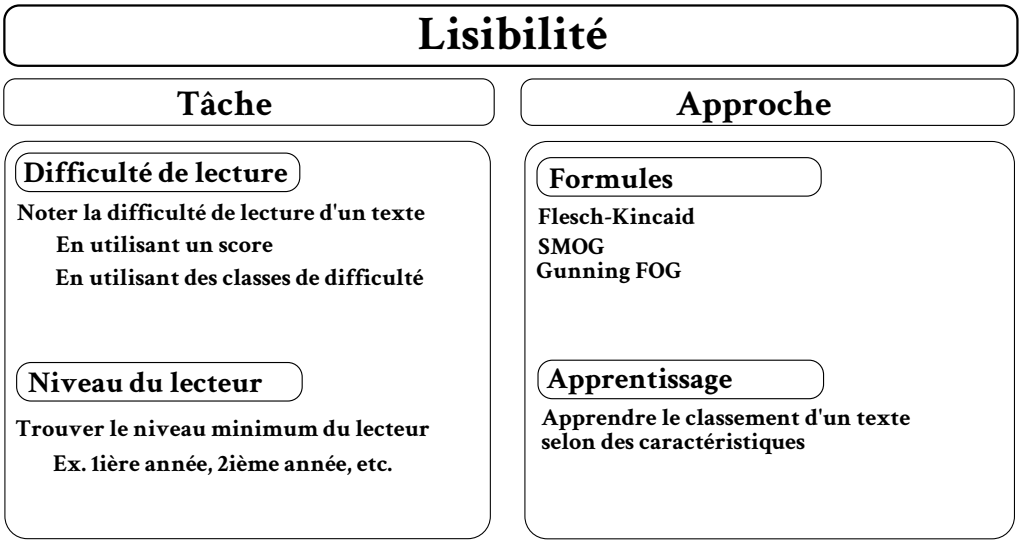


Figure 10.34 : *Classification des tests de lisibilité*

phrases et le nombre des syllabes comme indiqué par l'équation 10.49.

$$Score = 206.835 - 1.015(\frac{mots\ totaux}{phrases\ totales}) - 84.6(\frac{syllabes\ totales}{mots\ totaux}) \quad (10.49)$$

Selon ce score, on peut décider la difficulté d'un texte en se basant sur le tableau 10.1.

Un autre score pour la lisibilité de l'anglais est **Dale-Chall readability formula**. Ce score utilise une liste des mots difficiles, le nombre des mots et le nombre des phrases afin de calculer un score comme indiqué par l'équation 10.50.

$$Score = 0.1579(\frac{mots\ difficiles}{mots\ totaux}) + 0.0496(\frac{mots\ totaux}{phrases\ totales}) \quad (10.50)$$

La difficulté en se basant sur ce score est représentée par le niveau des étudiants qui peuvent lire le texte comme indiqué dans le tableau 10.2.

Score	Difficulté
90-100	Très facile à lire (Élève de 11 ans).
80-90	Facile à lire.
70-80	Plutôt facile à lire.
60-70	En clair (Élève de 13 ou 15 ans).
50-60	Plutôt difficile à lire.
30-50	Difficile à lire (Université).
0-30	Très difficile à lire (Diplôme universitaire).

Tableau 10.1 : *Niveau de difficulté en se basant sur le score Flesch-Kincaid*

Score	Difficulté
<= 4.9	Étudiant du 4ième.
5-5.9	Étudiant du 5ième et 6ième.
6-6.9	Étudiant du 7ième et 8ième.
7-7.9	Étudiant du 9ième et 10ième.
8-8.9	Étudiant du 11ième et 12ième.
9-9.9	Étudiant du 13ième et 15ième (collège).

Tableau 10.2 : *Niveaux de lisibilité en se basant sur le score de Dale-Chall*

Il existe plusieurs formules pour calculer la lisibilité d'un texte en anglais en utilisant plusieurs caractéristiques. Des formules pour tester la lisibilité des textes dans d'autres langues ont été proposées. Par exemple, la métrique **OSMAN** (El-Haj et Rayson, 2016) est utilisée afin de mesurer la lisibilité d'un texte en arabe suivant l'équation 10.51. Cette métrique se base sur le nombre des mots, le nombre des phrases, le nombre des mots difficiles (plus de 5 caractères sans diacritiques), le nombre des syllabes, le nombre des mots complexes (plus de 4 syllabes) et les mots Faseeh (les mots complexes avec les lettres ظ ، ذ ، وء ، يء ، ء ou qui se termine avec وَا ، ون).

$$Osman = 200.791 - 1.015 \times \left(\frac{mots\ totaux}{phrases\ totales}\right) - 24.181 \times \left(\frac{mots\ difficiles + syllabes + mots\ complexes + mots\ Faseeh}{mots\ totaux}\right) \quad (10.51)$$

6.2 Apprentissage automatique

Les formules utilisent des caractéristiques de surface comme le nombre des mots, le nombre des mots difficiles, etc. Elles n'utilisent pas des caractéristiques plus complexes des niveaux syntaxique, sémantique et pragmatique. Collins-Thompson (2014) propose l'utilisation de plusieurs caractéristiques allants du niveau morphologique jusqu'au niveau pragmatique comme entrée d'un algorithme d'apprentissage automatique (voir la figure 10.35). Les classes de difficulté peuvent être définies selon le besoin : niveau de l'étudiant, difficulté sur 12 degrés, etc.

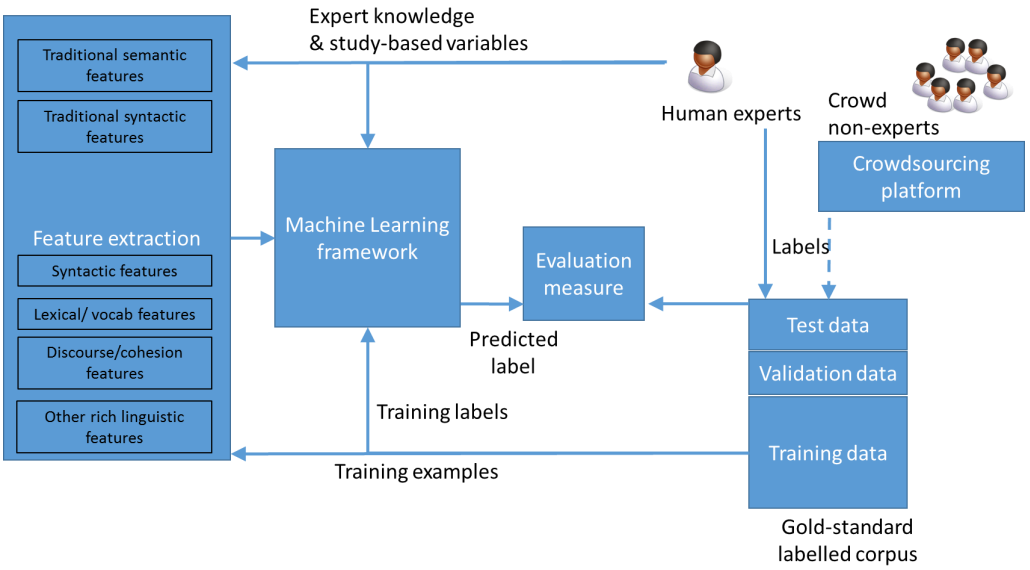


Figure 10.35 : *Pipeline d'estimation de difficulté de lecture avec apprentissage automatique (Collins-Thompson, 2014)*

7 Reconnaissance de la parole

La reconnaissance de la parole consiste à transformer des paroles vers un format textuel. Une telle tâche peut assister la rédaction sans utiliser les mains, communiquer avec des machines (comme le système de recommandation précédemment présenté), etc. La figure 10.36 représente une classification des systèmes de reconnaissance des paroles basée sur (Malik et al., 2020). Dans cette section, je vais présenter les étapes de la reconnaissance à la place des différentes approches. La classification sera présentée avec plus de détaille ici. Un système de reconnaissance de la parole peut être conçu seulement pour détecter les paroles d'un seul locuteur, n'importe quel locuteur ou il peut s'adapter aux nouveaux locuteurs. Selon le canal d'entrée, le système peut permettre les sons d'arrière-plan ou il exige un son clair en entrée. On peut aussi utiliser la taille du vocabulaire comme critère de classification. Un système avec un vocabulaire réduit comme les commandes est plus facile à implémenter qu'un système avec un grand vocabulaire comme le vocabulaire d'une langue. Selon l'approche de parole, le locuteur doit parler avec des pauses afin de faciliter la détection ou il peut parler d'une façon normale. Le style de la parole peut, aussi, affecter un système de reconnaissance : est-ce que le langage doit être standard ou on peut parler avec des sons qui n'ont pas de sens, comme "um".

7.1 Extraction des caractéristiques

L'extraction des caractéristiques consiste à transformer le signal sonore vers des vecteurs de caractéristiques acoustiques. Chaque vecteur représente l'information du signal encodée dans une petite fenêtre de temps. Premièrement, on commence par transformer le signal analogue à une représentation numérique. On doit appliquer un **échantillonnage** (sampling) afin d'extraire les valeurs du signal dans une durée définie. Afin de capturer les parties positives et négatives du signal, il faut prendre 2 échantillons par cycle. Les fréquences des paroles sont inférieurs à 10 KHz, d'où l'utilisation d'un taux d'échantillonnage : sampling rate = 20 KHz. Dans la téléphonie, la fréquence est 4KHz d'où : sampling rate = 8 KHz (Jurafsky et Martin, 2019). Une fois le signal est échantillonné, on passe à la quantification ; stocker les amplitudes sous formes des entiers. En général, les valeurs utilisées sont soit 8 bits (-128 à 127) ou 16 bits (-32768 à 32767). La valeur d'un échantillon dans un temps n est représentée comme $x[n]$.

Maintenant, nous avons un signal numérisé qui doit être divisé sous forme des fenêtres ; cela est appelé **fenêtrage**. Cette opération utilise une fenêtre pour capturer une partie d'un phonème appelée cadre

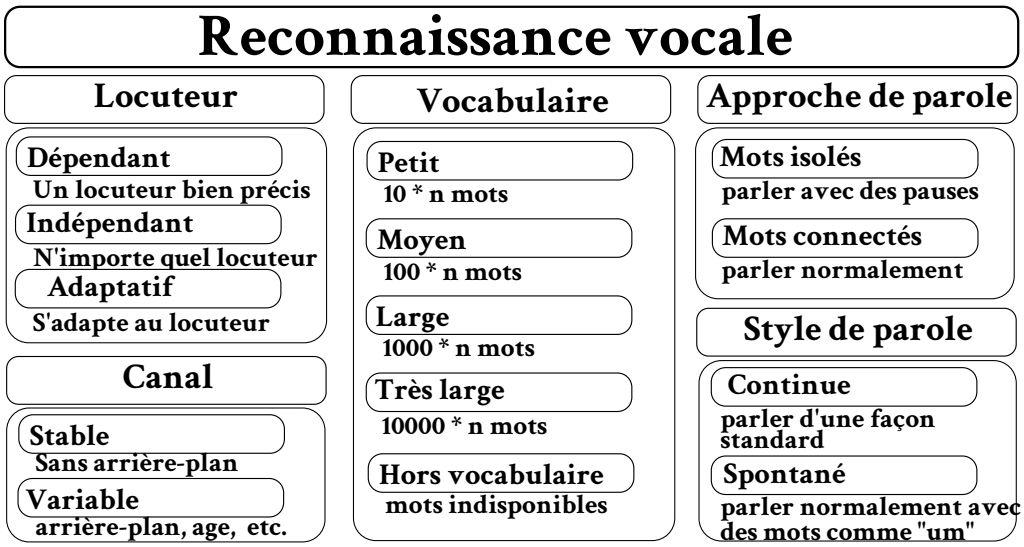


Figure 10.36 : *Classification des systèmes de reconnaissance de la parole*

(frame). Elle a deux paramètres : taille de la fenêtre (Window size) et décalage (Frame stride, shift, offset). Un exemple du fenêtrage est donné dans la figure 10.37.

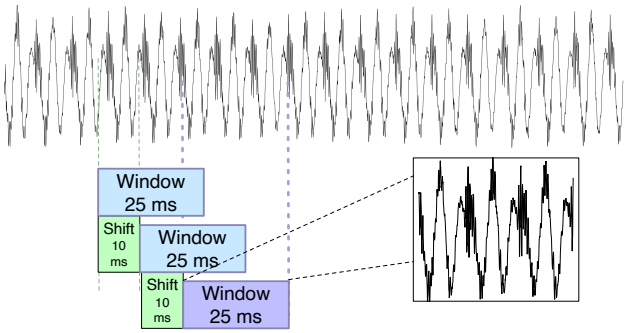


Figure 10.37 : *Exemple de l'opération du fenêtrage (Jurafsky et Martin, 2020)*

Pour extraire un cadre $y[n]$, on multiplie un signal de la fenêtre $w[n]$ par le signal initial $s[n]$ dans un temps n : $y[n] = w[n]s[n]$. La figure 10.38 représente un exemple de deux types de fenêtrage : rectangulaire et de Hamming. La méthode la plus simple est la fenêtre rectangulaire ; la fenêtre est représentée par l'équation 10.52

$$w[n] = \begin{cases} 1 & \text{si } 0 \leq n \leq L-1 \\ 0 & \text{sinon} \end{cases} \tag{10.52}$$

L'échantillonnage rectangulaire peut créer des problèmes avec l'analyse de Fourier. La solution est l'utilisation de la fenêtre de Hamming représentée par l'équation 10.53

$$w[n] = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi n}{L}) & \text{si } 0 \leq n \leq L-1 \\ 0 & \text{sinon} \end{cases} \tag{10.53}$$

Afin d'extraire les caractéristiques, il existe plusieurs méthodes : Mel Frequency Cepstral Coefficients (MFCC), Linear Prediction Coefficients (LPC), Linear Prediction Cepstral Coefficients (LPCC), Line Spectral Frequencies (LSF), Discrete Wavelet Transform (DWT), Perceptual Linear Prediction (PLP), etc. Ici, on va prendre MFCC comme exemple. On commence par l'extraction de la quantité d'énergie

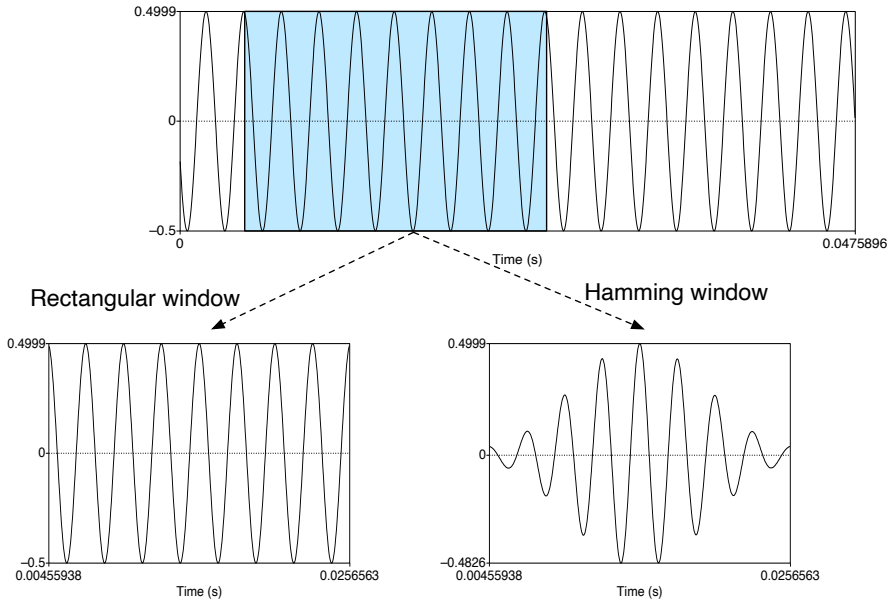


Figure 10.38 : *Fenêtrage rectangulaire vs. Hamming (Jurafsky et Martin, 2020)*

un signal contient dans les différentes bandes de fréquence. Ceci peut être accompli en utilisant : Discrete Fourier Transform (DFT) (Voir un exemple dans la figure 10.39). En se basant sur un signal fenêtré $x[n] \dots x[m]$, chacune des N bandes discrètes des fréquences peut être représentée sous forme d'un nombre complexe en utilisant l'équation 10.54.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \tag{10.54}$$

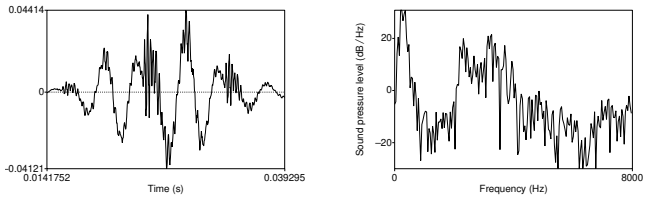


Figure 10.39 : *Exemple de la transformation DFT (a) une portion du signal du voyelle [iy] (b) son DFT (Jurafsky et Martin, 2020)*

L'audition humaine n'est pas également sensible à toutes les bandes de fréquences ; elle est moins sensitive dans des grandes fréquences. Modéliser le signal en se basant sur la perspective humaine peut aider la reconnaissance de la parole. Ceci peut être implémenté par la collecte des énergies d'une façon non égale à chaque bande de fréquence en se basant sur l'échelle de Mel. La figure 10.40 représente un exemple du filtre de Mel qui peut être calculé selon l'équation 10.55.

$$mel(f) = 1127 \ln(1 + \frac{f}{700}) \tag{10.55}$$

7.2 Reconnaissance

La parole est une série temporelle ; sa reconnaissance est une classification en utilisant des méthodes comme les **HMM** et les réseaux de neurones encodeurs-décodeurs. La figure 10.41 représente l'architec-

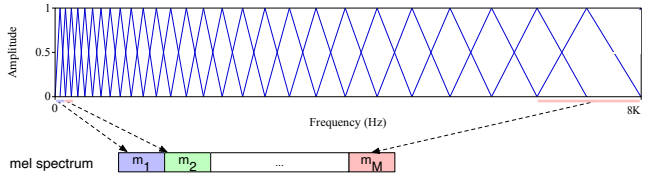


Figure 10.40 : *Exemple du filtre de Mel (Jurafsky et Martin, 2020)*

ture d'un système de reconnaissance de la parole en utilisation un encodeur-décodeur. L'encodeur commence à encoder les séquences du signal sous forme des représentations internes (contexte). Ce dernier est utilisé avec le mécanisme d'attention pour générer le texte caractère par caractère.

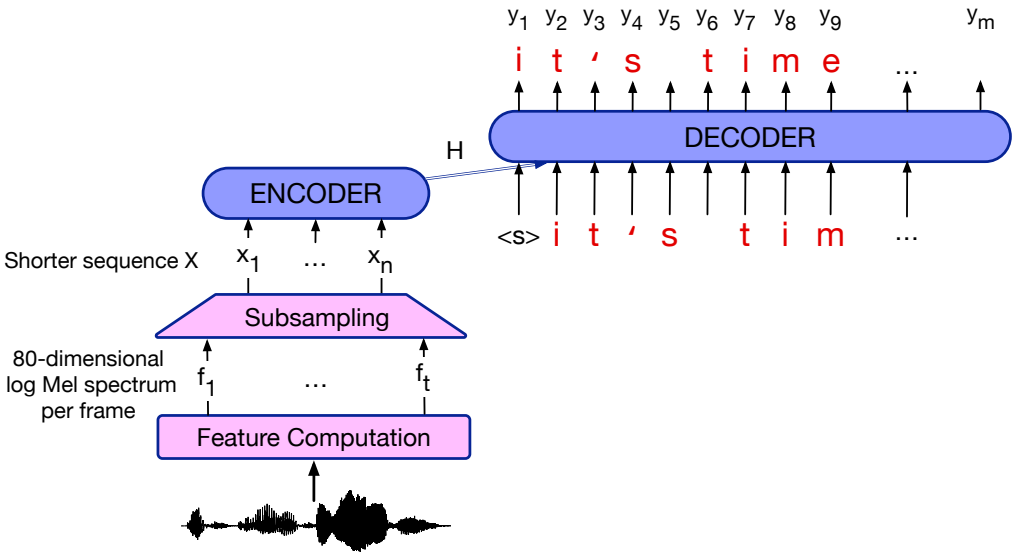


Figure 10.41 : *Exemple de reconnaissance en utilisant un encodeur-décodeur (Jurafsky et Martin, 2020)*

Formellement, la probabilité de générer un texte $y_1 \dots y_n$ peut être représentée en utilisant les probabilités de génération de chaque caractère y_i sachant les caractères précédemment générés et la représentation du signal X. Cela est indiqué par l'équation 10.56.

$$p(y_1, \dots, y_n) = \prod_{i=1}^{n} p(y_i | y_1, \dots, y_{i-1}, X) \tag{10.56}$$

Une méthode naïve pour maximiser cette probabilité est de maximiser la probabilité individuelle de chaque caractère lors de la génération comme indiqué par l'équation 10.57.

$$\hat{y}_i = \arg\max_{c \in V} p(c | y_1, \dots, y_{i-1}, X) \tag{10.57}$$

Ceci est un modèle de langage sur les caractères avec l'information de la parole. Afin d'entraîner ce modèle à bien générer un caractère sachant les caractères passés et le contexte, on doit avoir suffisamment de données. Les mots générés doivent appartenir au vocabulaire de la langue. Une solution est d'utiliser un modèle de langage séparé LM avec le modèle de reconnaissance afin d'améliorer la qualité du texte généré. La probabilité de générer une série de caractères Y sachant l'information de la parole X peut être représentée comme un score qui ajoute l'information du modèle de langage avec un taux λ, comme indiqué par l'équation 10.58.

$$score(Y|X) = \frac{1}{|Y|_{car}} \log p(Y|X) + \lambda \log p_{LM}(Y) \tag{10.58}$$

Ce score doit être maximisé ; une solution est l'utilisant des méthodes d'optimisation comme **Beam Search**. Il existe un autre problème qui est la génération du même caractère plusieurs fois consécutives. Ceci est dû au fait que la lettre a été prononcée sur plusieurs échantillons. Une solution est d'utiliser une technique appelée **Connectionist Temporal Classification (CTC)**.

8 Synthèse de la parole

Le but de synthèse de la parole est de générer des paroles à partir d'un texte. La figure 10.42 représente la classification des différentes méthodes de synthèse. On peut générer la parole d'un manière synthétique (à base de règles) ; on utilise des filtres phonétiques. Dans ce cas, le son généré n'est pas naturel. Une autre approche est de sauvegarder des phrases, des mots et des phonèmes afin de les concaténer pour avoir le son final. L'approche statistique est similaire à celle de la reconnaissance de paroles, seulement on inverse les entrées et les sorties.

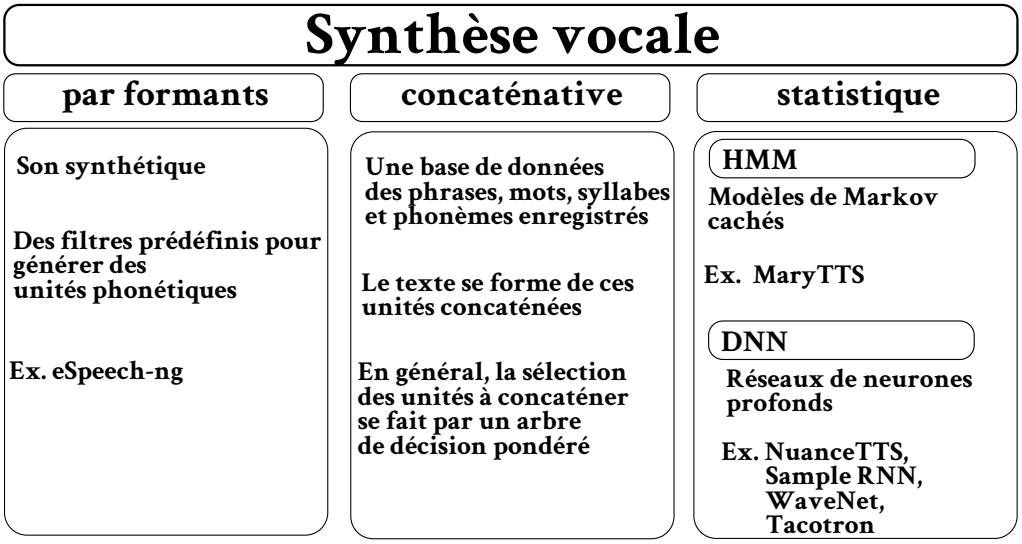


Figure 10.42 : *Classification des systèmes de synthèse vocale*

Selon Hinterleitner (2017), une architecture générale d'un système de synthèse vocale peut être représentée comme indiqué dans la figure 10.43. Le système peut être divisé sur quatre parties : TALN, génération de la prosodie, concaténation et génération des paramètres, et génération de la parole. Dans la première partie, on doit normaliser le texte par la conversion des abréviations, conversion des valeurs numériques, etc. Aussi, on doit détecter le stresse d'un mot en utilisant les suffixes et les préfixes. L'unité de génération de prosodie utilise des informations sur les accents de mots et de phrases ainsi que des informations sur des phrases pour créer la prosodie correspondante du texte orthographique, c'est-à-dire la durée, l'intensité et le ton. A partir de la représentation phonétique et les informations prosodiques, l'unité de concaténation crée une séquence continue de paramètres de signaux et/ou de gestes d'articulation. La dernière unité sert à générer la parole en utilisant ces paramètres suivant une des approches précédemment expliquées.

Discussion

Quelle est l'intérêt d'étudier un domaine sans applications ? Le traitement automatique du langage naturel est un domaine qui vise à émuler la capacité des être humains à communiquer soit par parole ou par texte. Dans le passé, ce domaine a été motivé seulement par la tâche de traduction automatique (surtout entre russe et anglais). Avec l'arrivé de l'internet et puis les réseaux sociaux, ce domaine serait d'un grand intérêt vu la nécessité de traiter toutes les informations qui circulent chaque jours. Comme chaque domaine,

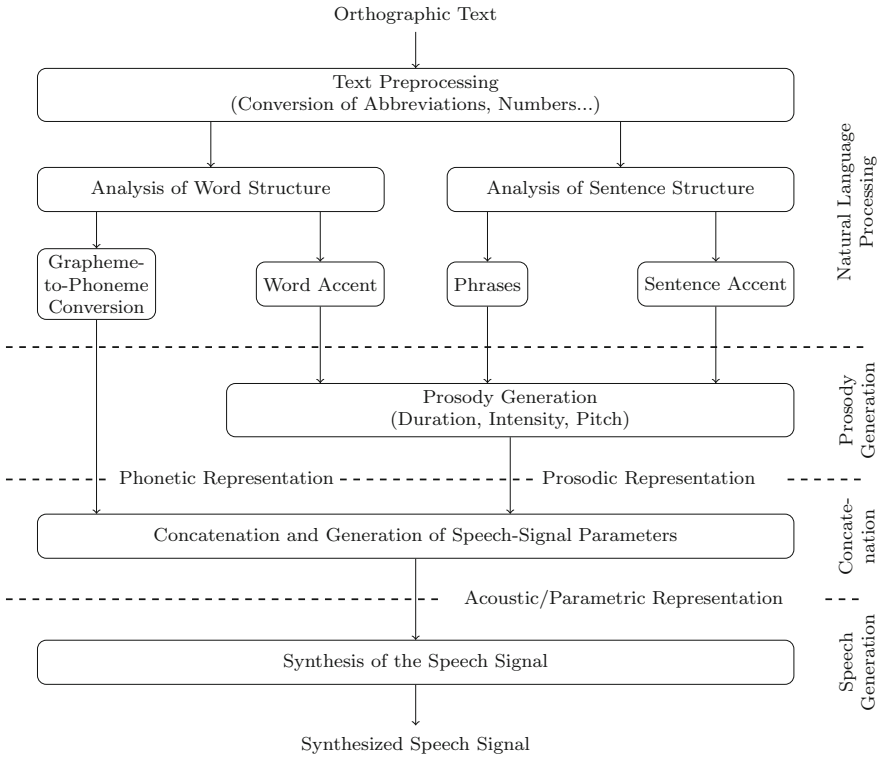


Figure 10.43 : *Architecture d'un système de synthèse vocale (Hinterleitner, 2017)*

TALN est un arme à double tranchant : il peut aider les peuples à communiquer, il peut automatiser des tâches sur les textes, etc. ; mais aussi, il peut être utilisé dans l'espionnage, la manipulation de l'opinion public, etc. Comme le proverbe de Spider-man le dit : "With great power comes great responsibility" (Avec un grand pouvoir vient une grande responsabilité).

Dans ce chapitre, on a présenté huit applications dont chaque deux représentent un type de tâches. Le premier type est la transformation du texte ; un texte en entrée doit être transformé à un autre différent en sortie. Les deux applications qui ont été présentées sont : la traduction automatique (un texte d'une langue vers une autre) et le résumé automatique (un texte long vers un autre plus petit). Le deuxième type est l'interaction avec l'utilisateur ; en utilisant des requêtes de l'utilisateur, on génère des réponses. Deux applications ont été présentées : les systèmes de questions/réponses et les systèmes de dialogue. Ces deux sont différents malgré que le dernier peut utiliser le premier afin de générer les réponses. Mais, les systèmes de dialogue ne répondent pas seulement aux questions ; ils peuvent poser des questions à l'utilisateur. Ils peuvent même mener une conversation comme celle des êtres humains. Le troisième type est la classification des textes ; un texte en entrée aura une classe en sortie. L'analyse des sentiments est la tâche la plus connue dans ce sens. La lisibilité est une autre tâche moins connue dans TALN ; la lisibilité dans le sens : le texte est-il difficile à lire ou non ? et pas dans le sens : le texte est-il bien formé ou non ? Enfin, la parole est prise comme un type à part. Elle contient deux applications : la reconnaissance de la parole et la synthèse de la parole. La première application prend une parole en entrée et la transforme à un texte, et la deuxième fait l'opération inverse. Il existe plusieurs applications qui sont implémentées en utilisant plusieurs approches et méthodes. Ce chapitre ne peut pas présenter le tout ; c'est juste une goutte dans l'océan.