Etiquetage d'un Corpus Oral par Composition de CRF
Stage de IRAD M2 recherche

par
Samer TAALAB
Laboratoire Informatique Fondamental d'Orléans

Septembre 2009

Remerciement

Je tiens à remercier dans un premier temps, Christel VRAIN, Directrice du LIFO, Frédéric LOULERGUE, Responsable du MASTER IRAD et spécialement Isabelle TELLIER, Maître de ce stage, qui a suivi tous les détailles de mon travail et pour le temps qu'elle m'a consacré tout au long de ce stage.

Je remercie aussi Sylvie BILLOT, Denys DUCHIER, Jean-Philippe PROST, Guillaume CLEUZIOU pour l'expérience enrichissante et pleine d'intérêt qu'elles m'ont fait vivre durant ces cinq mois.

Je remercie l'équipe linguistique, Iris Eshkol et mes camarades Laetitia Guillot, Anne Laure MOREAU et Gautier LEVACHER pour leur coopération.

Résumé

L'objectif de ce stage est de mettre en œuvre des techniques d'apprentissage automatique (en particulier les CRFs, Conditional Random Fields ou Champs Aléatoires Conditionnels) pour apprendre à étiqueter les mots d'un corpus issu de transcriptions de la parole avec leur catégorie grammaticale ("nom", "adjectif", "verbe", etc.).
En effet, il existe des outils pour étiqueter automatiquement des textes écrits avec ces catégories, mais ils ne se comportent pas bien avec les données issues de transcriptions orales, du fait des incorrections syntaxiques (répétitions, hésitations, ruptures de construction, etc.) qui y figurent. Des corrections manuelles sont alors nécessaires. Or, le laboratoire LLL d'Orléans dispose de tels corpus corrigés manuellement, qui peuvent donc servir de base à des programmes d'apprentissage automatique, notamment ceux faisant appel aux CRFs.

L'objectif du stage sera donc d'une part, de comparer les performances des CRFs entraînés sur des corpus corrigés manuellement, avec ceux obtenus par les outils standards d'étiquetage fonctionnant sans apprentissage, mais pas spécialisés dans les corpus oraux, d'autre part de trouver des nouvelles méthodes pour l'apprentissage avec les CRF. Or deux nouvelles méthodes seront l'objet de développements et expériences.
La première est d'appliquer l'apprentissage niveau par niveau sur des étiquettes organisées hiérarchiquement.
La deuxième est d'appliquer l'apprentissage par composition de CRF après décomposition des étiquettes en groupes ou composants.

Table des matières

1 Introduction **3**

2 Description de Corpus de données **5**
2.1 Comment le corpus est obtenu 5
2.2 CORDIAL 6
2.3 Etiquetage 8
2.4 Prétraitement Linguistique 8

3 Théorie de CRF (Conditional Random Field) **10**
3.1 Apprentissage automatique 10
3.2 Le modèle CRF 11
3.2.1 Introduction 11
3.2.2 Les modèles graphiques dirigés 11
3.2.3 Les modèles graphiques non dirigés 13
3.2.4 Apprentissage 16
3.3 Travaux de recherche sur les CRF 16
3.4 Outils de traitement 18
3.4.1 Boîte à outils CRF++ 18
3.4.2 Boîte à outils CRFSuite 19
3.5 Fonctionnement Général de CRF++ 20
3.5.1 Théorie de l'algorithme de CRF++ 20
3.5.2 fonctionnement de CRF++ 21

4 Expérience et Résultats **23**
4.1 Prétraitement de données d'entrée de CRF++ 23
4.1.1 Prétraitement de format pour CRF++ 24
4.1.2 Prétraitement des features 24
4.1.3 Prétraitement des résultats 25
4.2 Evaluation de l'apprentissage 25
4.2.1 Validation croisée 25
4.2.2 Définition de F-mesure 26
4.3 Experiences et tests 27
4.3.1 Expérience avec Lemmes 27
4.3.2 Expérience sans Lemmes 30
4.3.3 Expérience combinaison de CRF 31
4.3.4 Expérience composition de CRF 33

5 Conclusion et perspective **40**

A Groupe des Etiquettes 'CORDIAL' **42**

B Groupe des Etiquettes Hiérarchiques **43**

C Remplacement des Etiquettes **45**

Chapitre 1

Introduction

Dans ce stage nous allons nous intéresser à l'application d'outils basés sur les techniques de l'apprentissage automatique pour étiqueter un corpus transcrit de l'oral.
Le but de cet étiquetage est d'essayer de retrouver les catégories syntaxiques, morphologiques et sémantiques d'un texte oral. Ce type de travail est très important pour les études linguistiques et la recherche sur l'évolution de la langue française. Pour chaque catégorie grammaticale (syntaxe, morphologie, sémantique) il y a un groupe d'étiquettes bien défini par des linguistes. Ces trois niveaux d'étiquettes sont formés par une hiérarchie arborescente dont le première niveau contient les étiquettes syntaxiques (16 étiquette) {ADJ (adjectif), DET (Déterminant), ADV (Adverbe), V (Verbe), N (Nom), P(Prénom),. . . }, le deuxième niveau contient les étiquettes morphologique (72 étiquettes) {ADJ, ADJFS, ADJFP, ADJMS(Adjectif masculin singulier), ADJMP, DET, DETFP(Déterminant féminin pluriel), DETFS,. . . } et le troisième niveau contient les étiquettes sémantiques (107 étiquettes){ADJFP, ADJFS, ADJI, ADJMP, ADJMS, ADV, CH, CONJCOO, CONJSUB, DETFPDEF, DETFPDEM, DETPPOSS, V1SVINDP, V2PCON, VMPPARP,. . . }.

On va voir dans la suite qu'on peut diviser les étiquettes selon des groupes non hiérarchiques qui possèdent des propriétés spécifiques. Ces groupes définissent la notion de composantes des étiquettes à partir desquelles on peut reconstruire la même hiérarchie d'étiquettes précédente. La notion de groupes ou de composantes peut être appliquée à d'autres domaines que l'étiquetage de textes oraux.

Le corpus dont nous disposons est corrigé manuellement mais il n'est pas compatible avec les trois niveaux d'étiquettes que nous avons. D'où la nécessité de prétraitements linguistiques pour le rendre compatible et obéissant à la hiérarchie d'étiquettes choisie. De plus les parties Mot et Lemme du texte ont besoin de prétraitements à la main pour corriger les fautes d'orthographes.
Les outils disponibles d'annotation des corpus textuels sont nombreux et divers. Ils sont basés sur différentes théories d'apprentissage automatique. Il y a beaucoup de techniques d'apprentissage automatique qui sont appliquées à l'étiquetage de texte. Nous allons nous intéresser à celles qui sont basées sur le modèle des HMM(Hidden Markov Models) comme les CRF (Conditional Random Fields) et les MEMM (Maximum Entropy Markov Models). Nous essayons ici de mettre en œuvre le modèle des CRF.
Dans la suite, nous allons décrire le corpus sur lequel nous avons travaillé en clarifiant les

problèmes d'étiquetage. Puis nous allons évoquer l'analyseur syntaxique CORDIAL qui a été utilisé pour réaliser l'étiquetage initial du corpus étudié. Ensuite nous présentons le modèle générale des CRF et son application à l'apprentissage de l'étiquetage de séquences de données (textes). Après nous allons détailler les expériences avec les résultats les plus significatifs.

Chapitre 2

Description de Corpus de données

2.1 Comment le corpus est obtenu

Les données utilisées pour le test viennent d'un grand corpus de l'ESLO (Les Enquêtes Sociolinguistiques à Orléans,). ESLO a consacré des travaux à la parole non littéraire ainsi que des études fondamentales dans le domaine de la linguistique descriptive. [3, Iris Eshkol,2009]
Le corpus original de l'ESLO est constitué de **317** heures de parole spontanée, évalué à **4500000** mots avec une grande variété de situation :

- entretiens, contacts informels : discussions entre amis, enregistrements au hasard en micro caché (rue, magasins, etc.).
- interviews de personnalités du monde syndical, politique, universitaire et de l'administration d'Orléans (maire, évêque,...).
- conférences-débats sur des thèmes variés. et d'autres...etc.

Le corpus (d'Orléans) a été reconstruit par CORAL/LLL (Centre Orléanais de Recherche en Anthropologie et Linguistique) en 2005 afin de mettre le corpus dans un formt compatible avec les méthodes et techniques actuelles. Il s'agit donc de 700 heures d'enregistrement avec 1000.000 mots.
Les 'étapes de la reconstruction sont les suivantes :

- Numérisation
- Gestion du corpus et méta données (normalisation au format XML)
- Transcription (synchronisée + XML).

Les problèmes propres aux corpus de transcription de l'oral après la reconstruction linguistique sont la segmentation et les ambiguïtés dans le texte. Ces types de problèmes sont appelés des disfluences.
Exemple d'un échantillon du corpus (question/réponse) :

RC :
est est-ce que vous voulez bien me décrire une journée de travail ?
GJ 131 :
dans l'arrivée du du courrier pour le le courrier qui est ins- insuffisamment affranchi qui est à taxer je prépare tout ça euh pour la distribution des préposés je mets les surtaxes et puis je les distribue à chaque euh chaque euh préposé et il doit encaisser la surtaxe dont le montant de la surtaxe euh chez les AG ensuite je passe à un autre genre d'opération vous savez [rire] ça fait beaucoup de détails aussi ça alors j'ai des des tas de de choses

quoi contrôle de contrôle de machines à affranchir installation de machines à affranchir euh euh distributeurs de euh automatiques...
(*107mots*)
Il voulais dire :
GJ 131 : dans l'arrivée du courrier pour le courrier qui est insuffisamment affranchi qui est à taxer, je prépare tout ça pour la distribution des préposés. je mets les surtaxes et puis je les distribue à chaque préposé, et il doit encaisser la surtaxe dont le montant de la surtaxe chez les AG. ensuite je passe à un autre genre d'opération, vous savez ça fait beaucoup de détails, j'ai des tas de contrôle d'installation de machines à affranchir distributeurs automatiques.. (*82mots*)
si on compte les disfluences on remarque qu'il y en a un taux important et qu'il faut en tenir compte (23% de mots inutiles).
De point de vu du traitement automatique, on remarque les problèmes suivante :

- Absence de ponctuation : plus de 130 mots sans ponctuation (points, virgules, trait, saut de ligne...) !
- Amorces des mots interrompus, amorces de morphèmes : *ins- insuffisamment.*
- Particules/marqueurs discursives, insertion : *euh, quoi, ben ....*
- Répétitions de mots ou de séquence de mots, auto correction : *des des tas de de choses quoi contrôle de contrôle de...*

Ceci va compliquer l'utilisation de techniques d'apprentissage automatique pour lesquelles la notion de phrase (début, fin,...) est un facteur important.

2.2 CORDIAL

CORDIAL est un analyseur syntaxique et grammatical spécialisé en langue française. Il reçoit en entrée un texte quelconque et fournit en sortie une version étiquetée de ce texte. À chacun des mots et expressions du texte, différentes informations sont associées : lemme (forme générique, ex. "devoir" pour "due", "mignon" pour "mignonnes", etc.), numéro du mot dans la phrase, numéro de proposition, type grammatical du mot, genre et nombre du mot, appartenance à un groupe et numéro du pivot du groupe, fonction grammaticale du mot dans la proposition.
CORDIAL utilise pour l'étiquetage :

- Des Dictionnaires de grande taille (plus de 118 000 noms communs, soit plus de 700 000 mots pour le français)
- Des règles de désambiguïsation lexicale s'appuyant sur des statistiques effectuées à partir d'un corpus de plus de 500 millions de mots
- Une analyse syntaxique s'appuyant sur plus de 14 000 règles basées sur plus de 800 000 informations lexicales.

CORDIAL est très Rapide en exécution : plus de 16000 mots analysés à la seconde sur Pentium $3,2GHz$.
La démarche choisie pour la constitution et l'étiquetage du corpus par CORDIAL est composée de plusieurs étapes successives :

- transcription avec Transcriber : ce qui donne un fichiers (XML) (fichier.trs)
- Les fichier.trs sont alignés avec le fichier sonore ou nettoyés sous forme fichier.txt .
- Les fichiers .txt nettoyés encore sont étiquetés par Cordial : cnr. Pui
- Les fichiers sont corrigés à la main.

Exemple : fichier sortie de transcriptions (008B.trs) :
Ce type des fichiers sont annotés par un expert humain à l'aide du logiciel Transcriber et sauvegardés dans un format XML. Ce format inclut des définitions de Thèmes, de Locuteurs, puis des sections thématiques et des tours de parole. L'annotation est synchronisée sur le flux audio en utilisant des attributs et des balises dédiés.
Etapes 2 et 3 :

```
<Turn speaker="spk1" startTime="97.254" endTime="213.606">
<Sync time="97.254"/>
dans l'arrivée du du courrier
<Sync time="98.96"/>
<Sync time="99.537"/>
pour le le courrier qui est ins- insuffisamment affranchi qui est à taxer
<Sync time="103.591"/>
...
```

Les deux fichiers suivants sont les sorties de Cordial sous la format (FORMES, LEMMES, Etiquettes)

Etape3 :Fichiers Cordial	Etape4 :Fichiers Cordial Corrigée à la main
1.est-ce que est-ce que ADV	1.est-ce que est-ce que ADV
2.vous vous PPER2P	vous vous PPER2P
3.pouvez pouvoir VINDP2P	pouvez pouvoir VINDP2P
4.m' me PPER1S	m' me PPER1S
5.expliquer expliquer VINF	expliquer expliquer VINF
6.comment comment SUB	comment comment SUB
7.vous vous PPER2P	vous vous PPER2P
8.faites faire VINDP2P	faites faire VINDP2P
9.vous vous PPER2P	vous vous PPER2P
10.une un DETIFS	une un DETIFS
11.omelette omelette NCFS	omelette omelette NCFS
12.?? PCTFORTE	?? PCTFORTE
13.\ r	

Problemes :
Premièrement à chaque étape de l'étiquetage il y a des erreurs à corriger. En général les problèmes relevés sont les suivants :

- problèmes propres à Cordial
- problèmes 'classiques' : des disfluences et des ambiguïtés
- étiquettes à modifier : mon = *DETPOSS* → *DETPOSSMS* (Détérminant possessif→Détérminant possessif masculin singulier)
- étiquettes à ajouter : je man = *NI*
- choix à prendre :
  - `au siège`(au= *DETMS* par Cordial)
  - `on est favorisé` (verbe ou adjectif)
  - `Et puis mon dieu euh nous avons` (un ou deux mots)

Deuxièmement, le dernier fichier corrigé sera traité avec des outils qui implémentent le modèle des CRF. Chaque CRF peut accepter un tel fichier s'il satisfait à des formats spécifiques définis par l'outil lui-même. L'exemple précédent risque de ne pas être accepté

avant le traitement (nombre de colonnes pas fixé, frontière de phrase. . .).

2.3 Etiquetage

L'étiquetage est une opération qui consiste à attribuer à chacune des unités lexicales d'un corpus une étiquette donnant des informations morphosyntaxiques sur cette unité comme sa catégorie grammaticale dans son contexte d'utilisation.
Les problèmes de l'étiquetage [10, Nguyen, 2003] viennent de la segmentation aveugle des textes sans information syntaxique ou sémantique. Un mot peut être divisé en plusieurs unités ou morphèmes (par exemple dans le cas de mots composés). Au contraire, plusieurs mots en séquence peuvent être groupés en une seule unité : des locutions, des noms propres composés, des numéros composés, des mots composés, etc. En fonction de la définition des unités lexicales et/ou de l'application, les descriptions des classes et des étiquettes morphosyntaxiques peuvent inclure un ou plusieurs traits comme la catégorie syntaxique, le lemme, le genre, le nombre, etc.
Le corpus qui vient de l'Oral porte des erreurs et des ambiguïtés sur des mots qui ne sont pas bien définis dans la langue française et qui peuvent avoir plusieurs utilisations selon la personnel qui parle (exemple : `euh, ben, ba oui,...`).

Un étiqueteur reçoit un fichier texte et renvoie un ficher texte dont les unités sont étiquetées. Le corpus qu'on va utiliser est étiqueté avec l'étiqueteur CORDIAL. Un tableau d'étiquettes de 'CORDIAL' sont détaillées dans l'annex(A).

2.4 Prétraitement Linguistique

Il y a deux types de traitement linguistique qui sont appliqués sur le texte utilisé pour tester l'apprentissage avec l'outil CRF++.
- **Traitements individuels :**
Ce type de traitement ne peut pas être exprimé en utilisant des expressions régulières. Les linguistes lisent le texte et corrigent les erreurs à la main. Les erreurs corrigées sont les fautes d'orthographe ou de grammaire. Aussi les fautes dues aux mots qui ont plusieurs sens et qui ne peuvent être déterminés que selon le contexte (ex : ***la porte, il porte***). Ce type de corrections ne peut pas être déterminé ni suivi pour être automatiser. Les corrections se font ici par opération d'édition : ajouter, effacer, remplacer un ou plusieurs mots par d'autres.
- **Traitement en Bloc :**
Ce type de traitements porte souvent sur des mots de l'ensemble des étiquettes utilisées. Le but est de changer une ou plusieurs catégories grammaticales pour attribuer des nouvelles étiquettes pour détailler un autre groupe d'étiquettes. Ce sont des changements répétitifs pour des mots avec des positions fixes dans les lignes du fichier manipulé. Il s'agit de remplacer des groupes d'étiquettes par des autres, ou de combiner plusieurs lignes en une seule ligne ou d'ajouter de nouvelles colonnes d'étiquettes au texte. Ces types de traitement peuvent être réalisés en utilisant les expressions réguliers et à l'aide de scripts " Shell, Sed et AWK" sous linux qui facilitent la définition et l'application

des expressions réguliers sur plusieurs fichiers textuel en même temps. Les traitements se divisent en plusieurs groupes de manipulations :
Nettoyage : éliminer les mots et lettres inutiles.
par exemple :
('JK', 'VS', '#', '=', ' :', '[', ']',',',' ?'), les chiffres (324, 12,...),...
Remplacement : pour changer les étiquettes qui viennent de CORDIAL et qui ne sont pas correctes de point de vu linguistique (syntaxique ou morphologique). Ou bien pour introduire plus de détail sur les étiquettes en introduisant une nouvelle colonne.
par exemple :
Rechercher 'ADJD' et la Remplacer par 'DET DETDEM DETDEM'

ADJP → *DET DETPOSS DETPOSS*
ADJMP → *ADJ ADJMP ADJMP*

Groupement de plusieurs lignes en un seul ligne
exemple :

c'est	*c'est*	*PRES*	*PRES*	*PRES*	⇒	*c'est_pourquoi c'est_pourquoi PRES PRES PRES*
pourquoi	*pourquoi*	*ADV*	*ADV*	*ADV*		

Le tableau suivant montre les étiquettes redéfinies par les linguistes. Elles sont différentes des étiquettes que le corpus initial a utilisé. Ceci nous ammène à utiliser les CRF pour les retrouver. Les étiquettes sont détaillées en trois niveau. Chaque niveau contient les parties gramaticalles ajoutées à ce niveau :
L_0 : les étiquettes de niveau syntaxique.
L_1 : les étiquettes de niveau morphologique.
L_2 : les étiquettes de niveau sémantique.

L_0	L_1	L_2
ADJ	*MS, MP, FS, FP, I*	*idem*
N	*NCMS NCMP NCFS NCFP NCI NP*	*idem*
V	*1SINDP 1SINDI 1SINDF 1SINDPS 1SSUB 1SCON*	*idem*
V	*2SINDP 2SINDI 2SINDF 2SINDPS 2SSUB 2SCON 2SIMP*	*idem*
V	*3SINDP 3SINDI 3SINDF 3SINDPS 3SSUB 3SCON*	*idem*
V	*1PINDP 1PINDI 1PINDF 1PINDPS 1PSUB 1PCON 1PIMP*	*idem*
V	*2PINDP 2PINDI 2PINDF 2PINDPS 2PSUB 2PCON 2PIMP*	*idem*
V	*3PNDP 3PINDI 3PINDF 3PINDPS 3PSUB*	*idem*
V	*VINF VPARPRES VMSPARP VMPPARP VFSPARP VFPPARP*	*idem*
CONJ	*idem*	*SUB, COO*
P	*MS MP FS FP*	*REL INT PER POSS DEM IND*
P	*I*	*REL INT IND PER DEM*
DET	*MS FS P I*	*DEM POSS DEF IND INT*
ADV PCT	*idem*	*idem*
PREP PRES	*idem*	*idem*
INT CONJ	*idem*	*idem*
CONJ CH MI	*idem*	*idem*

TABLE 2.1 – les trois niveaux des étiquettes utilisées

Les étiquettes sont détaillées dans l'annex(B).
et si un niveau n'a pas de étiquettes détaillées on utilise celles du niveaux precedent *'idem'*. comme ADJMS est utilisée dans niveau L_1 et L_2. les *ADV CH PREP PRES MI INT PCT CONJ* sont utilisées dans tous les niveaux. de même pour les verbes etc.

Chapitre 3

Théorie de CRF (Conditional Random Field)

3.1 Apprentissage automatique

L'apprentissage automatique (ou apprentissage artificiel) est l'étude des algorithmes qui permettent aux programmes de s'améliorer automatiquement par expérience [définition de Tom Mitchell dans son livre "Machine Learning"]. Du côté des linguistes, l'apprentissage automatique permet de constituer des ressources linguistiques robustes et à large couverture par rapport à la constitution manuelle de ressources qui est en général longue et coûteuse. De plus ces ressources ne dépendent pas d'une langue spécifique.
Les méthodes d'apprentissage mises en œuvre peuvent être symbolique (inférence grammaticale, PLI...) , à base de modèles probabilistes (génératifs ou discriminants) ou à base de similarités (voisinages, analogie, "memory-based learning"...). Les domaines d'applications peuvent être [ATALA, http ://www.atala.org/] :

- l'analyse de la parole
- l'annotation de corpus (étiquetage lexical, syntaxique, fonctionnel, thématique, sémantique...)
- le clustering et la classification de textes (suivant différents critères possibles : auteur, contenu, opinion...)
- la recherche d'information
- l'extraction d'information (y compris : extraction et typage des entités nommées)
- les systèmes questions/réponses
- l'acquisition ou l'amélioration de ressources linguistiques (y compris : automates, grammaires, cadres de sous catégorisations, ontologies de concepts...)
- le résumé automatique
- la traduction automatique

Les techniques d'apprentissage sur des textes sont des outils logiciels qui implémentent des algorithmes d'apprentissage (domaine d'intelligence artificielle) dont la base théorique s'appuie sur l'approche probabiliste d'un graphe dirigé ou non dirigé. Le modèle des chaînes de Markov HMM (Hidden Markov Models). Un des modèles qui a montré des bons résultats est le modèle CRF (Conditional Random Fields). Ce Modèle peut être appliqué sur les corpus de texte qui viennent de la parole.

3.2 Le modèle CRF

3.2.1 Introduction

Dans ce chapitre nous allons introduire la notion de probabilité jointe et la probabilité conditionnelle liée aux modèles d'apprentissages à base génératifs ou à base discriminants et comment sont représentées par un modèle graphique dirigé et non dirigé. Nous allons utiliser quelques définitions de probabilité qui nous va servir sans entrer dans les détaille, puis nous allons parler brièvement de modèle graphique dirigé avec la notion des 'nœuds parents' et le cas spéciale si le modèle est séquentiel. puis nous allons parler de modèle graphique non dirigé avec la notion de 'clique' et le cas spéciale si le modèle est séquentiel pour arrivé au modèle de CRF et la notion des features.
Définition : Soit $X = X_1, X_2, \ldots, X_n$ un champ de variable aléatoire discret (non continue) qui prend ses valeur pour chaque variable X_i sur un ensemble dénombrable Z. La probabilité pour que le champ X prend la valeur $\{x_1, x_2, \ldots, x_n\}$, donné par :

$$P(x_1, x_2, \ldots, x_n) = P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) \tag{3.1}$$

On définit la probabilité jointe des variable X_i par $P(x_1, x_2, \ldots, x_n)$ où l'ensemble des valeur de X sont définis sur Z^n .
Exemple : Deux dés composent un champ $X = \{X_1, X_2\}$ où X_i prend ses valeur sur $Z = \{1, 2, 3, 4, 5, 6\}$, et l'ensemble des valeur de $X \in Z^2$.
Si un dé représente un des X_i, La probabilité pour que la variable X_1 prend la valeur $x_1 = 2$ est $P(X_1 = x_1) = P(X_1 = 2) = 1/6$, et $P(X_1 = 3, X_2 = 6) = 1/36$.
Si on prend $n = 20$, l'ensemble des valeurs de champ X sera défini sur Z^{20}. Par calcule simple on aura $6^{20} = 3.65615844 * 10^{15}$ possibilité pour la distribution jointe des X_i du champ X, ce qui est impossible à modéliser par l'ordinateur.

3.2.2 Les modèles graphiques dirigés

Définitions : Deux variable X,Y sont indépendantes si leur probabilité jointe s'écrit sous la forme :

$$P(X, Y) = P(X).P(Y) \tag{3.2}$$

La représentation graphique de cette relation est donnée par la figure 3.1

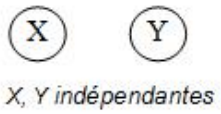

FIGURE 3.1 – X et Y sont independantes

Si deux variables sont non indépendantes, on note la probabilité de X sachant Y par $P(X|Y)$.
Si $\{X_i, X_j \in X\}$ et X_i ne dépendent que de X_j alors :

$$P(X_i|X_1, X_2, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n) = P(X_i|X_j) \tag{3.3}$$

La représentation graphique de cette relation est donnée par la figure 3.2 Exemple :
pour $X = X_1, X_2, X_3, X_4$ représenté par le graphe suivant 3.3 On a :

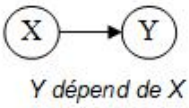

FIGURE 3.2 – Y dépend de X

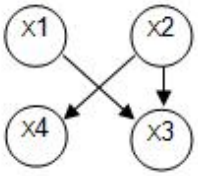

FIGURE 3.3 –

$$\begin{aligned} P(X) &= P(X_1).P(X_2).P(X_3|X_1,X_2).P(X_4|X_2) \\ P(X4|X_1,X_2,X_3) &= P(X_4|X_2) \end{aligned}$$

- Pour deux sous ensembles X_A, X_B des variables de champ X, si les deux sous ensembles X_A, X_B sont indépendants alors :

$$P(X_A, X_B) = P(X_A).P(X_B) \tag{3.4}$$

- De même si $X_C \in X$ La probabilité conditionnelle de X_A, X_B (considérés comme indépendants) sachant X_C, est donnée par :

$$P(X_A, X_B|X_C) = P(X_A|X_C).P(X_B|X_C) \tag{3.5}$$

et si X_A ne dépend que de X_C alors on a $P(X_A|X_B, X_C) = P(X_A|X_C)$
Exemple : Pour la séquence dans la Figure(3.4)

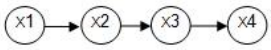

FIGURE 3.4 – Exemple

$$P(X_1, X_2, X_3, X_4) = P(X_1).P(X_2|X_1).P(X_3|X_2).P(X_4|X_3)$$

Définition : Les parents $\pi(X_1)$ d'un nœud X_1 dans un graph dirigé, sont définis par l'ensemble de toutes les variables $\{X_j \in X\}$ qui sont reliées avec X_i par des arêtes dirigées vers X_i. et on a : $P(X_i) = \pi(X_i|\pi(X_i))$ et en général on a :

$$P(X_A) = \prod_{i=1}^{n} P(X_i|\pi(X_i)) \tag{3.6}$$

Exemple : Pour la figure (Figure3.5) on a :

$$\begin{aligned} \pi(X_3) &= \{X_1, X_2\} \\ \pi(X_4) &= \{X_1\} \\ P(X_1, X_2, X_3) &= P(X_1|\pi(X_1)).P(X_2|\pi(X_2)).P(X_3|\pi(X_3)) \\ &= P(X_1).P(X_2|X_1).P(X_3|X_1, X_2) \\ P(X_3, X_4) &= P(X_3|\pi(X_3)).P(X_4|\pi(X_4)) \\ &= P(X_3|X_1, X_2).P(X_4|X_1) \end{aligned}$$

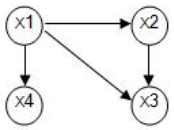

FIGURE 3.5 – Parents dans un graphe dirigé

Les formules (3.6) et (3.3) sont très importantes pour la modélisation de distribution joint du champ X et ils expriment la notion de factorisation (en décomposant la probabilité P en un produit des facteurs) et montrent que l'indépendance conditionnelle permet de réduire la dimension de l'espace sur le quel la distribution joint $P(x, y)$ prend ses valeurs. Alors l'espace de probabilité Z^n de $P(X_i|X_1, X_2, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n) = P(X_i|X_j)$ qu' on a vu se réduit à Z^2. D'où la réduction de complexité de modélisation et de calcule. Dans les modèle HMM est basé sur les modèle de graphe dirigé et les variable sont interdépendant et elles sont liées sous forme de chaîne séquentielle de variables figure (3.6). Il modélise une séquence d'observation $X = \{x\}_t^T$ en considérant qu'il y a des nœuds cachés $Y = \{y\}_t^T$ de nombre fini.et chaque x_t est l'observation d'une variable y_t à l'instant t est annoté par une étiquette comme Det ou ADJ ...

$$p(y, x) = \prod_{Yt=1}^{T} p(y_t|y_{t-1}).p(x_t|y_t) \tag{3.7}$$

On remarque que pour HMM que y_t ne dépend que de y_{t-1} le nœud précèdent et que x_t l'observation actuel ne dépend que de y_t

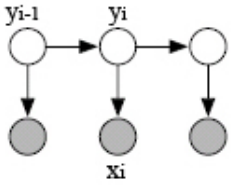

FIGURE 3.6 – Modèle de HMM

Dans ce qui suit on va retrouver la même notion de factorisation et de l'indépendance conditionnelle mais avec le modèle de graphe non dirigé sur le quelle le modèle CRF est basé.

3.2.3 Les modèles graphiques non dirigés

Un graphe non dirigé peut être représenté par $G = (V, E)$, où V est un ensemble de nœuds et E un ensemble des arêtes entre les nœuds. Chaque nœud représente une variable $Xi \in X = \{X_1, X_2, \ldots, X_n\}$. **Définition :** Soient les sous ensembles $A, B, C \subset V$ on dit que B sépare A de C, si il n'existe aucune arête qui relie un nœuds de A avec nœud de C (tous passent par B)⇒
si $X_a \in A, X_b \in B, X_c \in C, P(X_c|X_a, X_b) = P(X_c|X_b)$

Exemple : Dans le graphe non dirigé Figure (3.7), si $A = \{X_1\}, B = \{X_2\}, C = \{X_3, X_4\}$, alors A, C sont **séparés** par B et on a :

$$P(X_1|X_2, X_3, X_4) = P(X_1|X_2)$$

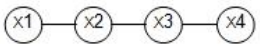

FIGURE 3.7 – Graph non-dirigé

Définition : Un clique C est l'ensemble des nœuds d'un graphe non dirigé qui sont reliés deux à deux entre eux. Un clique est appelé maximale s'il n'est pas contenu dans une autre clique.
La figure 3.8, $c_2 \subset c_1$. c_1 est une clique maximale, c_2 n'est pas maximal. Le théorème de

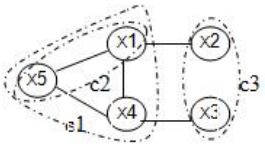

FIGURE 3.8 – Clique dans un graph non-dirigé

Hammersley-Clifford [5, Hammersley-Clifford] donne la distribution de x on fonction des fonctions appelées locales sur les cliques.
Théorème : Soit un graphe G dont l'ensemble des cliques est C. La distribution de probabilité P d'un champ aléatoire de Markov (un modèle graphique non dirigé) est décomposable en un produit des fonctions Ψ_c définies sur les cliques c de l'ensemble C des cliques de G si et seulement si cette distribution de probabilités est strictement positive.

$$P(x) = \frac{1}{Z} \prod_{c \in C} \Psi_c(X_c) \tag{3.8}$$

Où Z est un coefficient de normalisation assurant que $p(x)$ est bien une probabilité. Les fonctions de cette factorisation sont appelées fonctions de potentiel. Les fonctions de potentiel sont des fonctions positives ou nulles, qui prennent leurs paramètres une réalisation x_c de l'ensemble de variables aléatoires X_c. Le coeficient de normalisation Z s'exprime de la manière suivante :

$$Z = \sum_{x \in X^n} \prod_{c \in C} \Psi_c(X_c) \tag{3.9}$$

Exemple : Pour le graphe Figure (3.7) on a [4, Flurent Jousse]
$P(x_1, x_2, x_3, x_4) = \frac{1}{Z}\Psi_1(x_1)\Psi_2(x_2)\Psi_3(x_3)\Psi_4(x_4)\Psi_{1,2}(x_1, x_2)\Psi_{2,3}(x_2, x_3)\Psi_{3,4}(x_3, x_4)$
Dans le modèle CRF qui est un graphe non dirigé nous avons deux types de nœuds dans le graphe qui représentent deux types de variables :

1. Les variable qui dépendent de l'observations x.

2. Les variable qui dépendent de l'annotations y.

La distribution peut être exprimée pour un graphe dirigé et par un graphe non-dirigé Figure(3.9) de la manière suivant :

$$P(x,y) = p(y)\prod_{k=1}^{K} p(x_k|y) \tag{3.10}$$

$$P(x,y) = \frac{1}{Z}\prod_{A} \Psi_A(x_A, y_A) \tag{3.11}$$

Ces deux représentations sont semblables si on considère que :

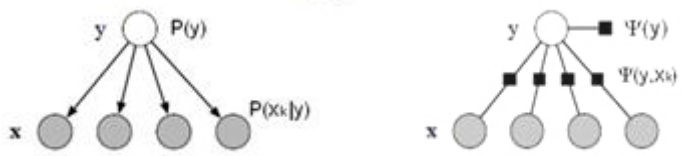

FIGURE 3.9 – graphe dirigé et graphe non-dirigé

$$P(y) = \Psi(y) \text{ et } P(x_k|y) = \Psi(y, x_k)$$

Pour un graphe séquentiel Figure(3.6)

$$P(y,x) = \prod_{t=1}^{n} P(y_t|y_{t-1}) \tag{3.12}$$

Définition : soient y et x deux champs aléatoires et soit un vecteur de paramètres ;

$$\Lambda = \{\lambda_k\} \in R^k$$

$f_k(y, y', x_t)^K$, $k = 1$ est une fonction à valeur réel. On définit CRF comme la distribution de $P(y|x)$ par :

$$P(y|x) = \frac{1}{Z(x)} exp\left\{\sum_{k=1}^{k} \lambda_k f_k(y_t, y_{t-1}, x_t)\right\} \tag{3.13}$$

avec $Z(x)$ comme un facteur de normalisation de la forme suivant :

$$Z(x) = \sum_{y} exp\left\{\lambda_k f_k(y_t, y_{t-1}, x_t)\right\}. \tag{3.14}$$

Les fonctions f_k sont appelées featuers et les λ_k sont les paramètres du modèle. Pour définir le modèle il faut connaitre les deux ensembles $\{f_k\}$ et $\{\lambda_k\}$.Le problème de trouver les paramètres et les features, revient à un problème de maximisation le vraisemblance pour la distribution

$$P(y|x,\lambda) = \frac{1}{Z(x)} exp(\sum_{i} \lambda_j F_j(y,x)) \tag{3.15}$$

Ce maximum existe puisque le problème est d'allure convexe.
Une méthode bien connue pour résoudre ce type de problème c'est la méthode de gradient descendant itérative.

3.2.4 Apprentissage

On considère les données d'apprentissage $\left\{x^{(k)}, y^{(k)}\right\}_{k=1}^{N}$, où $x^{(k)}$ une observation et $y^{(k)}$ l'annotations correspondant, la probabilité

$$P(\left\{y^{(k)}\right\} \mid \left\{x^{(k)}\right\}, \lambda)$$

de l'équation (3.15) précédente est appelé le vraisemblance. On cherche à maximiser le log de vraisemblance de P sur les données d'apprentissage :

$$L(\lambda) = \sum_{k} \left[log \frac{1}{Z(x^{(k)})} + \sum_{j} \lambda_j F_j(y^{(k)}, x^{(k)}) \right]$$

Exemple : dans l'annotation de texte issu de la parole on prend la question suivant : Comment vous faites vous une omelette considéré comme Observation x L'ensemble d'annotation possible :

$$y = \{adverbe, PPER2P, VINDP2P, DETIFS, Nom\}$$

Qui apprêtent dans l'ordre suivant selon le temps :

t	x_i	y_i
-2	comment	Adv
-1	vous	PPER
0	faites	VIND
+1	vous	PPER
+2.	une	DET
+3.	omelette	Nom

à l'instant t= 0 le mot x_i = 'faites' lemmatisé en 'faire' classifier comme $y_i = ' VIND'$ et les fonctions features se traduit pour l'exemple précédent comme le suivant : $t_j(y_{i-1}, y_i, x_t) = \begin{cases} 1 & \text{si } y_{i-1} = \text{'PPER' et } y_i = \text{'VIND' et } x_t = \text{'faire' à la position } i \\ 0 & \text{sinon} \end{cases}$

$s_k(y_i, x_t) = \begin{cases} 1 & \text{si } y_i = 'VIND' \text{ et } x_t = 'faire' \text{ à la position } i \\ 0 & \text{sinon} \end{cases}$

De cette manière le modèle apprend les feature et les paramètres.

3.3 Travaux de recherche sur les CRF

Le méthode des CRF est très intéressant et beaucoup de recherches ont essayé d'améliorer le performance et les résultats de l'apprentissage des CRF soit des améliorations sur les algorithme soit des améliorations sur le choix des features des CRF. Les plus récent entre les recherches qui ont introduit des nouvelles techniques, on a :

[6, Lafferty et al., 2001] ont introduis Les CRF et ils ont montré plusieurs avantages du modèle des CRF sur les modèles HMM et MEMM surtout en mesure de performance

sur des données de langues naturelles.

[13, Charles Sutton, Andrew McCallum, 2006] ont démontré l'efficacité des CRF sur les données implicitement dépendantes (relationnelles) et surtout qui enrichissent les caractéristiques (Features) des CRF. De plus ils ont étudié le modèle 'skip-chain CRF' pour modéliser les dépendances non voisines entre entités, et ils ont introdui de nouveaux types de features.

Une étude sur l'efficacité des features par [8, Andrew McCallum,2003] montre qu'il y a un algorithme itératif qui peut améliorer l'efficacité des features de %40 en comparaison avec celles qui sont construites à la main. Leur domaine d'application est l'extraction et la segmentation des entités nommées.

[4, jousse, 2006] a appliqué les CRF sur les arbres XML et il a implémenté un outil XCRF pour l'annotation de données arborées. Il a montré comment faire un bon choix sur les caractéristiques, appelées features, du modèle des CRF.

Un modèle très intéressant appelé "MoP-MEMM" (Mixture-of-Parents Maximum Entropy Markov Models) est défini par [**?**, David Rosenberg, Dan Klein, and Ben Taskar, 2007] Ils ont montré comment étendre le modèle "MEMM" en profitant des propriétés des CRF pour modéliser conditionnellement des nœuds qui ne sont pas voisins. Ils ont appliqué MoP-MEMM à la reconnaissance des entités nommées et à la classification de pages web.

[12, Liam Stewart, Xuming He, Richard S. Zemel,2008] ont introduit la notion de features paramétrisables pour réduire la complexité du modèle d'apprentissage : le Modèle est appelé "CFOE" (Conditional Field Of Expert). Selon l'étude qu'ils ont faite les features paramétrisables peuvent être n'importe quelle combinaison d'observations, d'étiquettes et des nœuds auxiliaires cachés. Le résultat expérimental sur des exemples d'extraction d'information montre que le 'CFOE' en précision et en F1-mesure donne un bon résultat en comparaison avec le modèle 'IOHMM' (Input Output Hidden Markov Model).

Le modèle des CRF donne les meilleurs résultats dans plusieurs domaines comme le traitement automatique de la parole, l'extraction de méta données (les disfluences de parole) et la reconnaissance de phrases qui viennent de la parole [7, Yang Liu, Andreas Stolcke, Elizabeth Shriberg, Mary P. Harper]. L'outil Mallet implémente CRF pour l'extraction de méta données [9, McCallum, Andrew Kachites,2002].

3.4 Outils de traitement

Le modèle CRF est implémenté avec plusieurs boites à outils et en plusieurs formes. Chaque forme satisfait à des besoins et des spécifications précisées par le développeur et selon l'application et les types de données pour lesquelles on a choisi le CRF.
Chaque outil a sa façon pour annoter les textes et utilise un format spécial de textes d'entrée ce qui nécessite des traitements pour reformater le corpus d'entrer pour qu'il soit compatible avec l'outil utilisé et pour effectuer des validations fiables des tests réalisés et pour analyser et illustrer les résultats obtenus.

L'étude des features des CRF (de point de vu utilisateur, les features sont des méta données déduites à partir du texte à annoter et qui aident les CRF à retrouver le meilleur étiquetage possible) est la partie la plus importante dans les expérience que nous allons réaliser. Un prétraitement est nécessaire pour construire et aligner ces features avec chaque texte à tester.
En général les opérations de prétraitement prennent plus de 75% de travaille expérimental. Dans la suite nous allons citer quelques caractéristique de boite à outils CRF++ et boite à outils CRFSuite.

Nous avons pris la décision d'utiliser CRF++ seulement et pas plusieurs outils pour de prétraitement lourd des données et pour se concentrer à trouver les meilleur features possibles d'une part, d'autre part à cause des raisons suivantes :

- CRF++ plus rapide que CRFSuite car il est parallélisé.
- CRF++ permet de préciser comment les fonctions features sont construites, CRFSuit ne donne pas cette possibilité.
- Taux de mémoire utilisé par CRF++ est petit.
- CRF++ [1, crfpp, 2009] est l'outil le plus récent entre les outils qui implémentent les CRF .

3.4.1 Boîte à outils CRF++

CRF++ est une implémentation des CRF qui possède les caractéristique suivantes :
Version : 2009-05-06 : CRF++ 0.53 [1, crf++]
Source : Open source en C++ / STL / version parallelisable
Langue : pas de limitation (Français, Anglais,. . .)
Fichier d'entré : elle est composé de phrases séparées par un ligne vide, une phrase est composé de suite de plusieurs tokens et chaque token est situé sur un ligne différent. Un token correspond à un ligne et il est composé de plusieurs colonnes (ex : Mot, Lemme, Etiquette) mais le nombre de colonne est fixé au niveau de même fichier. Un tabulation ou espace séparent les colonnes. La dernière colonne représente la colonne qui va être appris par CRF++. Par exemple le fichier d'enter (tableau3.1).
Fichier de sortie : La sortie est le modèle appris par CRF++ qui n'est pas accessible pour la lecture qu'à partir de CRF++.
Features : CRF++ permet aux utilisateurs de préciser comment construire les fonctions features selon un fichier appelé 'Template'. Un fichier Template dont chaque ligne est un

-2	comment	Adverb	Adv
-1	vous	PPER2P	PPER
0	faites	VINDP2P	VIND
+1	vous	PPER2P	PPER
+2.	une	DETIFS	DET
+3.	omelette	Nom	Nom

TABLE 3.1 – fichier de test format CRF++

Template	Feature développé
# Unigram U01 :%x[-1,0] U02 :%x[0,0] U03 :%x[1,0]	 comment vous faites
U16 :%x[-1,1]/%x[0,1] U17 :%x[0,1]/%x[1,1]	comment/vous vous/faites
U21 :%x[-1,1]/%x[0,1]/%x[1,1]	comment/vous/faites
# Bigram B	

TABLE 3.2 – fichier de test format CRF++

template est utilisé pour spécifier un token de données d'entré (tableau3.2). Il y a deux types de template u-gramm (uni-gramm) et b-gramm (bi-gramm) qui aident à changer la conception et la façon de génération des features.
Options d'utilisation : Il y a deux programmes CRF_learn pour apprentissage et CRF_test pour tester le modèle avec un fichier de test. On peut préciser plusieurs niveau de précision et ils génère des paramètre d'apprentissage (iter, terr, serr, diff...). Et de plus on peut préciser plusieurs niveaux de régularisation d'erreur CRF_L1 et CRF_L2.
Mesure de précision : il est compatible avec [CoNLL2000 shared task], il donne les N-best sortie et les mesures précision, rappel et 'F1-mesure'.
exemple de Template

3.4.2 Boîte à outils CRFSuite

CRFSuite est une Implémentation de CRF pour l'annotation de données séquentielles.
Resource : [CRFSuite]
Version : CRFsuite 0.8 (2009-03-17)
Source : Open source en C
Langue : pas de limitation
Fichier d'entré : même format de fichier que CRF++.
Fichier de sortie : un format pour le modèle construit est accessible à l'aide de logiciel 'Constant Quark Database (CQDB)', mais il est difficile à utiliser.
Features : la possibilité de définir des features sur les états. Mais les features de transition entre les états du modèle sont générés automatiquement. Ce qui est un point faible par rapport au CRF++.

Options d'utilisation : On peut préciser plusieurs niveaux de régularisation d'erreur L1 et L2.
Mesure de précision : il est compatible avec [CoNLL2000 shared task](Conference on Computational Natural Language Learning), il donne les mesures de précision, rappel et F1-mesure.

3.5 Fonctionnement Général de CRF++

3.5.1 Théorie de l'algorithme de CRF++

L'entraînement de CRF++ est basé sur l'algorithme de L-BFGS quasi-newton (Limited memory Broyden - Fletcher - Goldfarb - Shanno) pour résoudre des problèmes d'optimisation numériques de grand dimension. Cet algorithme est une amélioration pour l'approximation de matrice de Hessian . [www.wikipedia.org/wiki/L-BFGS]
La matrice de Hessian est une matrice carré de dérivé partielle de seconde ordre d'une fonction $f(X)$ par rapport a ses variable $X = (x_1, .., x_N)$, cette matrice décrit la courbure de la fonction. La matrice de Hessian est utilisée largement pour résoudre les problèmes d'optimisation avec les méthodes de Newton qui s'exprime sous forme local de Taylor pour une fonction. tel que

$$y = f(\mathbf{x} + \Delta\mathbf{x}) \approx f(\mathbf{x}) + J(\mathbf{x})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}} H(\mathbf{x})\Delta\mathbf{x}$$

où J(x) est la matrice de "Jacobian" (vecteur de gradient) de $f(x)$ et $H(x)$ Hessian de $f(x)$. $H(x)$ est difficile à calculer ,Or l'algorithme quasi-Newton est développé pour approximer la matrice Hessian $H(x)$. L'algorithme L-BFGS ne mémorise pas la matrice de Hessian le fait qui est très coûteux si la dimension n de la matrice devient grande.
Par contre L-BFGS maintient l'historique de dernière m mise à jour de la position de x et le gradient$\nabla f(x)$.
Généralement m n'est pas plus grand que 10. Ces mises à jour sont utilisées implicitement pour effectuer les opérations qui nécessitent l'utilisation de Hessian (ou sa inverse). Le méthode de Newton utilise la dérivé du seconde ordre pour l'approximation pour trouver le minimum de la fonction $f(x)$.

$$f(x_k + \Delta x) \approx f(x_k) + \nabla f(x_k)^T \Delta x + \frac{1}{2}\Delta x^T B \Delta x$$

où (∇f) est le gradient et B est une approximation de matrice de Hessian.

$$\nabla f(x_k + \Delta x) \approx \nabla f(x_k) + B\Delta x$$

et l'initialisation de ce gradient à 0 donne le pas de Newton :

$$\Delta x = -B^{-1}\nabla f(x_k)$$

L'approximation de la matrice Hessian par B est choisie pour satisfaire

$$\nabla f(x_k + \Delta x) = \nabla f(x_k) + B\Delta x$$

3.5.2 fonctionnement de CRF++

CRF++ est une logicielle libre de l'implémentation de Champs Aléatoire Conditionnelle (CRF) pour la segmentation et l'annotation de données séquentielles. CRF++ est très important dans de le domaine de Traitement Automatique de la langue TAL et ses applications. CRF++ est composé de deux modules principales "CRF_Learn" et "CRF_Test" qui sont paramétrables pour l'optimisation de résultats selon des critères qui dépendent de CRF++. La figure suivante montre le fonctionnement générale de la boite à outil CRF++.

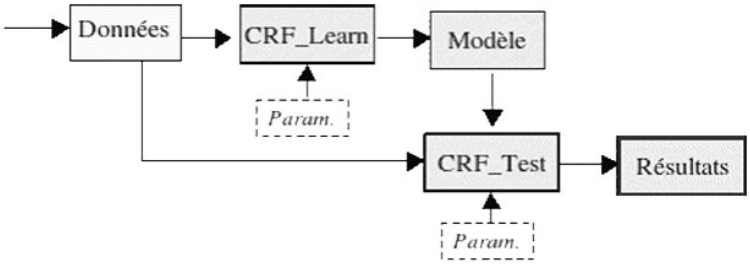

FIGURE 3.10 – Fonctionnement Général de CRF++

"CRF_Learn" : Module pour l'apprentissage qui prends comme entrée un fichier de données textuelles bien formaté et rends un fichier binaire qui est le modèle construit et qui va être utilisé par CRF_Test. Des paramètres sont utilisés pour optimiser le fonctionnement de CRF_Learn : Changement de paramètre $'c'$ qui influe le sur-apprentissage (overfitting) de corpus d'apprentissage. Changer l'algorithme de régularisation qui influe sur la performance de CRF++ en temps et en mémoire. Utilisation de deux processeurs en parallèle pour gagner de temps.....

"CRF_Test" : Le module de teste qui utilise le modèle résultat de CRF_Learn et génère le résultat sous forme d'un fichier textuelle qui est sous la même forme que le fichier d'entré après ajouter une nouvelle colonne de résultat après la dernière colonne. La colonne ajoutée représente les annotations estimées par CRF_Test en utilisant le Modèle généré et qui vont être le sujet de mesure et comparaison. Des paramètres sont utilisés pour optimiser le fonctionnement de CRF_Test(ex : N meilleur résultats,...)

Exemple :

texte d'entré (trois colonnes : Mot, Lemme et étiquette)

```
1) voilà    voilà     PREP
2)
3) ça       ça        P
4) marche   marcher   V
5) ou       ou        CONJ
6) quoi     quoi      ADJ
7) ?        ?         PCT
8)
9) euh      euh       INT
```

texte de sortie (quatre colonnes : Mot, Lemme, étiquette, étiquettes estimé)

```
1) voilà    voilà     PREP   PREP
2)
3) ça       ça        P      P
4) marche   marcher   V      V
5) ou       ou        CONJ   ADV
6) quoi     quoi      ADJ    DET
7) ?        ?         PCT    PCT
8)
9) euh      euh       INT    INT
```

Ici nous avons les colonnes représentent successivement col1 = 'Mots', col2 = 'Lemmes', col3 = 'étiquettes correctes' et col4='étiquettes estimées par CRF++'. Pour estimer l'occurrence de l'apprentissage il faut comparer les deux dernières colonnes (col3 et col4) et qui vont être le sujet de la mesure F-mesure.

Chapitre 4

Expérience et Résultats

Dans ce chapitre nous allons parler des expériences que nous avons réalisées et les taches de prétraitements qui précèdent chaque test pour préparer le corpus avec les features nécessaires. Le corpus utilisé est le corpus traité à la main et celui dont nous avons parler à la chapitre('Description de Corpus de données'2). Puis nous parlerons de la méthode de validation de test. A la fin nous allons citer parmi tous les tests que nous avons réalisés, ceux qui donnent des résultats significatifs.

4.1 Prétraitement de données d'entrée de CRF++

Nous avons vu dans le paragraphe 3.5.2 que CRF++ exige une format spécifique pour les fichiers d'entrées pour rendre le fichier d'entrer compatible avec CRF++. En effet, il y a plusieurs niveaux de prétraitement principalement pour le module d'apprentissage 'CRF_Learn ' et pour le module de test " CRF_Test " (figure4.1). En général nous avons les quatre niveaux de traitement suivants :

1. Traitement de formate.
2. Traitement pour la validation croisée.
3. Traitement pour créer des features.
4. Traitement pour illustration des résultats de tests.

Dans la suite nous allons détailler ces quatre types de traitement.

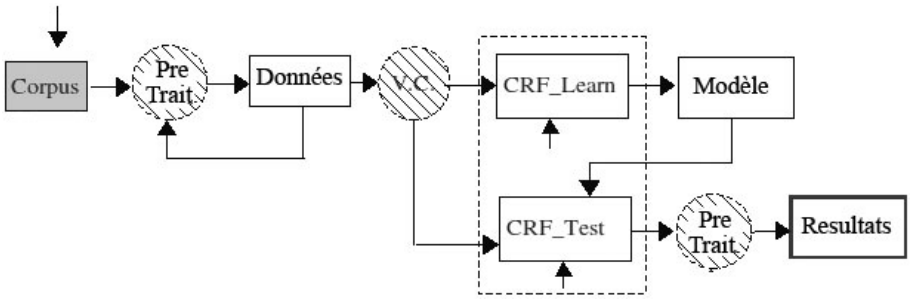

FIGURE 4.1 – Prétraitement de données d'entrées de CRF++ et traitement pour illustration de résultats

4.1.1 Prétraitement de format pour CRF++

Les données d'entrer doivent respecter un format fixe définit par CRF++. Comme nous avons vu dans le paragraphe 3.5.2, il faut vérifier :

- Un fichier est composé d'un nombre fixe de colonnes.
- Chaque mot doit être situé sur une ligne différente.
- Tous les mots dans un ligne sauf le dernier sont considérés comme features servent à prédire les étiquettes qu'existent dans la dernier colonne.
- Les mots et les étiquettes sont séparées par tabulations '\t'.
- Une suite de lignes qui terminent avec une ligne vide '\n', constitue une phrase.

Ce type de traitements est souvent fait à la main et automatisable pour la séparateurs entre colonnes et entre ligne et pour d'autres caractères spéciaux. De plus le CRF++ se bloque des qu'il trouve un erreur, ce qui pose un problème pour la correction des erreurs. C'est pourquoi nous avons changé un module de CRF++ 'features_index.ccp' pour que CRF++ génère tous les erreurs possibles de format avec leurs numéros du ligne dans le texte initiale.
Dans l'exemple suivant nous avons un texte vient de l'orale et il est étiqueté avec " CORDIAL ". Chaque ligne est composée de cinq colonnes qui représentent respectivement mots, lemmes, étiquettes (L0 : étiquettes niveau syntaxique), étiquettes (L1 : étiquettes niveau morphologique) et étiquettes (L2 : étiquettes niveau sémantique).
exemple :

```
1) voilà    voilà     PREP   PREP      PREP
2)
3) ça       ça        P      PI        PIDEM
4) marche   marcher   V      V3SINDP   V3SINDP
5) ou       ou        CONJ   CONJ      CONJCOO
6) quoi     quoi      ADJ    ADJMS     ADJMS
7) ?        ?         PCT    PCT       PCT
8)
9) euh      euh       INT    INT       INT
```

4.1.2 Prétraitement des features

Le but de ce type de traitement est de créer les features qui aident CRF++ à améliorer le résultat de l'apprentissage et pour la génération de nouvelles données de testes.
Il s'agit de créer les nouveaux features et les ajouter au texte à tester en construisant un ou plusieurs colonnes. Pour chacun des mots de texte on ajoute un feature ou plusieurs en respectant la compatibilité de format de texte avec CRF++.
Choix des features :
Il y a plusieurs types de features utilisés :

- features sur 'Les Lemmes' qui est fourni avec le texte initial qui est déduit en utilisant des ressources externes comme un dictionnaire et des règles syntaxiques.
- features qui sont construits à partir des mots de première colonne par extraction de dernières lettres de mot.
- features par combinaison des mot et lemme pour créer des nouveaux features.
- Utilisation des résultats d'un teste comme des features pour construire d'autres données de teste.

Pour construire les features on utilise des scriptes " Shell " sous Linux avec les outils " Sed et AWK ".

4.1.3 Prétraitement des résultats

L'outil CRF++ donnes comme résultats de teste des fichier textuels dont la dernière colonne contient les étiquettes retrouvées par CRF++. Il faut un prétraitement pour le mesure de similarité entre les étiquettes correctes (colonne avant le dernière) et les étiquettes résultats. Pour cela on calcule F-mesure pour chacune des étiquettes utilisées et pour la totalité des étiquettes. De cette manière on peut juger le choix des features et son influence sur la détection de l'annotation correcte et sa précision.
Pour certaine testes, il est possible que les données de fichier de sorite deviendrions le sujet d'un autre test. D'où la nécessiter de supprimer des colonnes et ajouter des autres en gardant les étiquettes retrouvées.
Pour Illustrer les résultats sous forme des diagrammes statistique on traite le résultat par un tableur pour bien visualiser les données et faire des comparaisons entre les résultats de plusieurs testes.

4.2 Evaluation de l'apprentissage

Le système d'apprentissage est basé sur des algorithmes qui donnent des résultats qui ne sont pas exactes et peuvent changer selon les échantillons d'apprentissage utilisés. Les résultats obtenus ne sont jamais parfait mais ils peuvent toujours dépasser une certaines limites connu distinguent le domaine d'application. On définie un méthode de test par des mesures et des critères bien adaptés à ce type de test et qui lui accordent un niveau de confiance et fiabilité. Pour chaque domaine d'application il faut préciser la façon dont le test se déroule. La méthode 'Validation Croisée' est un méthode de test bien adapté pour avoir des résultats d'évaluation fiables dans les domaines d'apprentissage et traitement automatique de la langue. Avec La 'Validation Croisé' on va utiliser le mesure de similarité 'F-mesure'.
Dans le cas d'annotation des corpus qui viennent de l'oral, Nous allons évaluer un système d'apprentissage supervisé qui a pour but de compare la similarité entre une ensemble d'étiquettes d'entrées considérés correctes contre l'ensemble d'étiquettes de sortie trouvé par le système.

4.2.1 Validation croisée

La validation croisée est une méthode d'estimation de fiabilité d'un modèle fondé sur une technique d'échantillonnage. La validation croisée est bien la répétition de deux opérations :
- Apprentissage sur une échantillon de données, et
- Application du modèle résultat de la première opération sur un autre échantillon.

La validation qu'on va adapter est appelée 'Validation Croisée 9/10' ; qui effectue l'opération d'apprentissage sur un échantillon égal à 9/10 de texte utilisé et l'opération de test sur le 1/10 qui reste, et d'une manière circulaire pour effectuer dix fois d'apprentissage et dix fois de test. Le résultat final est le résultat de mesure après cumuler tous les résultat de

dix tests.
Exemple :
si le texte à apprendre est composé de 1000 lignes (1000 mots) alors :

- Apprentissage (text(101..1000)) → modèle1 ; puis
  teste (modèle1, ligne(1..100)) ⇒ Res1
- Apprentissage (text(1..100, 201..1000)) → modèle2 ; puis
  test (modèle2, ligne(101..200)) ⇒ Res2
- Apprentissage (text(1..200, 301..1000)) → modèle3 ; puis
  test (modèle3, ligne(201..300)) ⇒ Res3
- ...
- Apprentissage (text(1..900)) → modèle10 ; puis
  test (modèle10, ligne(901..1000)) ⇒ Res10
- Concaténation de tous les résultats sur un fichier ;
  Cat(Res1, Res2, ...Res10) ⇒ Res
- Application de F-mesure ⇒ le fichier " Res-crf "

Un prétraitement ici est nécessaire pour générer les blocs de texte pour l'apprentissage et les blocs de texte pour les tests (figure :4.1), et puis pour récupérer les résultat de chacun de test et les rassembler en un seul fichier sur lequel on va appliquer un scripte pour calculer le 'F-mesure' totale et 'F-mesure' pour chaque étiquettes de l'ensemble d'étiquettes considéré.

4.2.2 Définition de F-mesure

F-mesure
Une mesure populaire qui combine la précision et le rappel est leur pondération, nommée F-mesure (soit F-measure en anglais) ou F-score :

$$F = \frac{2 * (precision.rappel)}{(precision + rappel)}$$

Ceci est connu comme mesure F1, car précision et rappel sont pondérés de façon égale. Il s'agit d'un cas particulier de la mesure générale F ? (pour des valeurs réelles positives de ?) :

$$F_\beta = \frac{(1 + \beta^2) * (precision.rappel)}{(\beta^2 * precision + rappel)}$$

Si on note
- NC : nœuds correctement annotés et
- NA : les nœuds qu´ il fallait annoter
- NE : nœuds ayant été annotés ou extraits

Le rappel et **La précision** sont donnés par :

$$Rappel = \frac{NC}{NA} \quad , \quad Precision = \frac{NC}{NE}$$

4.3 Experiences et tests

Nous allons détailler les expériences que nous avons réalisées avec CRF++ et les différents features que nous avons testés et les résultats obtenus avec chaque group de features. Pour cela nous allons utilisé les notations suivantes :

- On note L_0 le première niveau d'étiquettes à retrouver qui représente les étiquettes de niveau syntaxique.
- On note L_1 le deuxième niveau d'étiquettes à retrouver qui représente les étiquettes de niveau morphologique)
- On note L_2le deuxième niveau d'étiquettes à retrouver qui représente les étiquettes de niveau sémantique. Les groupes d'étiquettes sont détaillés dans l'annexe "groupes des étiquettes".
- La forme d'un fichier d'entrer est noté $\{mots, lemmes, L_0, L_1, L_2\}$.
- La forme d'un fichier de sortie est noté $\{mots, lemmes, L_0, L_1, L_2, Res\}$
- $CRF(L_0|feature(mots, lemmes)) \rightarrow Res_{L0}$; apprendre par CRF++ pour retrouver le groupe d'étiquettes de L_0 à partir (en connaissant) les features construits à partir des *mots* et *lemmes*. Les étiquettes résultants Res_{L0} composent la dernière colonne dans le fichier de sortie. D'où on obtient un nouveau fichier texte de la forme {mots, lemmes, L_0, Res_{L0}} où les deux dernières colonnes L_0 et Res_{L0} seront le sujet de mesure de similarité F-mesure puisque L_0 contient les étiquettes correctes et Res_{L0} contient les étiquettes trouvées ou extraites par CRF++.

Exemple : On peut utiliser le résultat d'un test pour construire les features de test suivant par combinaison des résultats du test précédent :
1- $CRF(L_0|$ feature(mots, lemmes))$\rightarrow Res_{L0} \Rightarrow$ {mots, lemmes, L_0, Res_{L0}}
2- $CRF(L_1|$ feature(mots, lemmes, L_0, Res_{L0}))$\rightarrow Res_{L01} \Rightarrow$ {mots, lemmes, L_0, Res_{L0} ,L_1, R_{L01}}
En général les features sont construits sur les informations qui existent dans les colonnes autres que les colonnes d'étiquettes L_0, L_1 et L_2.
Pour les features construits sur les *mots* et *lemmes*, nou allons utilisé les notations suivantes :
Soit mot_i un mot et $lemme_i$ le lemmes correspondant dans un fichier d'entrer, alors il existe un $Racine_i$ telque

$$\begin{aligned} mot_i &= Racine_i + R_{mot} \\ lemme_i &= Racine_i + R_{lemme} \end{aligned}$$

On note Dn_X une colonne qui contient les 'n' dernière lettres de la colonne 'X'.
Cette notation sera utilisée pour ajouter nouveaux features au texte initiale. Par exemple : pour $n = 2$ $D2_{lemme}$ la colonne composée de deux dernières lettres du colonne 'lemme' dans le texte d'entré.

4.3.1 Expérience avec Lemmes

les test suivants vont utiliser les lemmes dans le texte initial pour construire les features.

test A : Choix de features les $Mots$ et les $Lemmes$:
A partir de fichier $\{mots, lemmes, R_{mot}, R_{lemme}, L_0|L_1|L_2\}$ nous allons effectuer les test suivants :
$CRF(L_0|feature(mots, lemmes)) \rightarrow Res_{L0}$
$CRF(L_1|feature(mots, lemmes)) \rightarrow Res_{L1}$
$CRF(L_2|feature(mots, lemmes)) \rightarrow Res_{L2}$

Résultat([1]) :

etiq.	Nb.feat.	Mots	Extr.	Corr.	Précision	Rappel	F-mesure
L_0	$1.5 * 10^6$	16486	16486	15301	0.92812	0.92812	0.92811
L_1	$7 * 10^6$	16486	16486	14253	0.86470	0.86470	0.86469
L_2	$10 * 10^6$	16486	16486	14142	0.85781	0.85781	0.85780

test B : Choix de features le différence entre mot et $lemme$
Nous allons construire le fichier d'entré suivant :
$\{mots, lemmes, R_{mot}, R_{lemme}, L_0|L_1|L_2\}$ telque
si $mot_i = lemme_i$ alors $R_{mot} = R_{lemme} =' x'$ et nous allons effectuer les test suivants :
$CRF(L_0|feature(mots, lemmes, R_{mot}, R_{lemme})) \rightarrow Res_{L0}$
$CRF(L_1|feature(mots, lemmes, R_{mot}, R_{lemme})) \rightarrow Res_{L1}$
$CRF(L_2|feature(mots, lemmes, R_{mot}, R_{lemme})) \rightarrow Res_{L2}$

Résultat :

etiq.	Nb.feat.	Mots	Extr.	Corr.	Précision	Rappel	F-mesure
L_0	$1.5 * 10^6$	16486	16486	15384	0.93315	0.93315	0.93314
L_1	$7 * 10^6$	16486	16486	13245	0.89083	0.89083	0.89082
L_2	$11 * 10^6$	16486	16486	13137	0.88357	0.88357	0.88356

test C : Choix de features le différence entre mot, lemme, $D2_{mot}$ et $D3_{lemme}$
Nous allons construire le fichier d'entré suivant
$\{mots, lemmes, R_{mot}|D2_{mot}, R_{lemme}|D3_{lemme}, L_0|L_1|L_2\}$ telque
si $mot_i = lemme_i$ alors on ajoute $D2_{mot}$ et $D3_{lemme}$ et nous allons effectuer les test suivants :
$CRF(L_0|feature(mots, lemmes, R_{mot}|D2_{mot}, R_{lemme}|D3_{lemme})) \rightarrow Res_{L0}$
$CRF(L_1|feature(mots, lemmes, R_{mot}|D2_{mot}, R_{lemme}|D3_{lemme})) \rightarrow Res_{L1}$
$CRF(L_2|feature(mots, lemmes, R_{mot}|D2_{mot}, R_{lemme}|D3_{lemme})) \rightarrow Res_{L2}$

Résultat :

etiq.	Nb.feat.	Mots	Extr.	Corr.	Précision	Rappel	F-mesure
L_0	$3 * 10^6$	16486	16486	15554	0.94346	0.94346	0.94345
L_1	$14 * 10^6$	16486	16486	14838	0.90003	0.90003	0.90002
L_2	$20 * 10^6$	16486	16486	13576	0.82348	0.82348	0.82347

test D : Choix de features le différence entre mot, lemme, $D3_{mot}$ et $D3_{lemme}$
Nous allons construire le fichier d'entré suivant :
$\{mots, lemmes, R_{mot}|D2_{mot}, R_{lemme}|D3_{lemme}, L_0|L_1|L_2\}$ telque

1. etiq. :étiquettes, Nb.feat. :nombre de features, Extr. : nombre d'étiquettes extraites par CRF++, Corr. : nombre d'étiquettes classées correctement par CRF++

si $mot_i = lemme_i$ alors on ajoute $D3_{mot}$ et $D3_{lemme}$ et nous allons effectuer les test suivants :
$CRF(L_0|feature(mots, lemmes, R_{mot}|D3_{mot}, R_{lemme}|D3_{lemme})) \rightarrow Res_{L0}$
$CRF(L_1|feature(mots, lemmes, R_{mot}|D3_{mot}, R_{lemme}|D3_{lemme})) \rightarrow Res_{L1}$
$CRF(L_2|feature(mots, lemmes, R_{mot}|D3_{mot}, R_{lemme}|D3_{lemme})) \rightarrow Res_{L2}$

Résultat :

etiq.	Nb.feat.	Mots	Extr.	Corr.	Précision	Rappel	F-mesure
L_0	$3 * 10^6$	16486	16486	15537	0.94243	0.94243	0.94242
L_1	$14 * 10^6$	16486	16486	14808	0.89821	0.89821	0.89820
L_2	$20 * 10^6$	16486	16486	14698	0.89154	0.89154	0.89153

Conculsion de test avec lemme

Les résultats de quatre testes précédents A, B, C, D sont résumés par le tableau suivant :

etiq,	F-mesure A	F-mesure B	F-mesure C	F-mesure D
$L_0(16)$	0,92811	0,93314	0,94345	0,94242
$L_1(72)$	0,86469	0,89082	0,90002	0,89820
L_2 (107)	0,85780	0,88356	0,82347	0,89153

TABLE 4.1 – Résultats de tests A, B, C et D

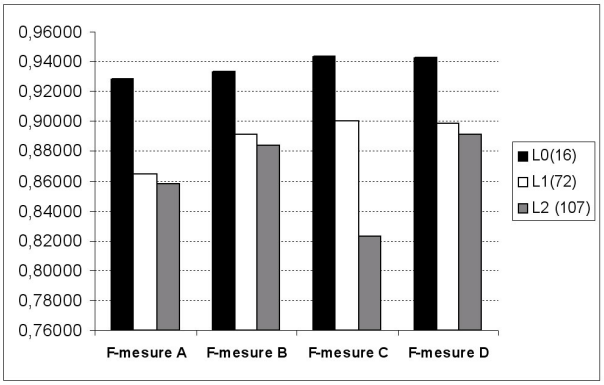

FIGURE 4.2 – comparaison de F-mesure pour les test A, B, C, D

On remarque sur la figure (4.2) que pour le meilleur apprentissage de niveau d'annotation avec les étiquettes de L_0 et L_1 il faut utiliser les features de **'Test C'** (différence entre mot et lemme et 2-3 dernière lettres du mot-lemme).
De même pour le meilleur apprentissage de niveau L_2 il faut utiliser les features construits en 'Test D' (différence entre mot et lemme et 3-3 dernière lettres du mot-lemme), ce qui nous paraît le plus stable pour le choix des features.
On remarque sur la figure (4.3) que en moyenne on a
$A \leq C \leq B \leq D$.
D'où on en peut construire un bon teste en choisissant des features qui satisfont les features de C et D et avoir le meilleur résultats des deux.

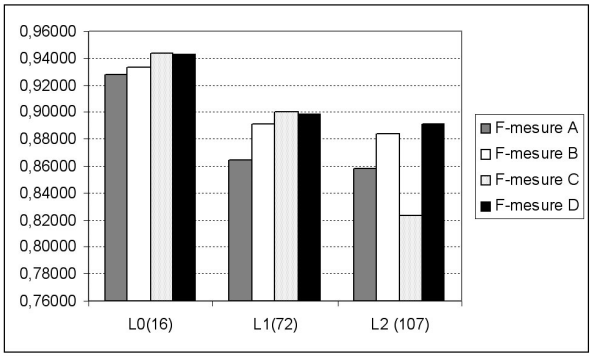

FIGURE 4.3 – F-mesure pour les test A, B, C, D groupement par niveaux d'étiquettes

4.3.2 Expérience sans Lemmes

Les testes suivants sont réalisés sans utilisation de lemme dans les fichiers d'entrés
test E :
Choix de features Sans Lemmes ,mot et 3-dernière-lettres du mot
Les features que nous allons construire sont F(A) où A est choisit de la façon suivant :
On définit l'ensemble de terminaisons
$Term$ ={**ismes, istes, aires, tions, asses, ables, iques, ances, isme, iste, aire, tion, asse, able, lles, enne, ment, ique, ance, ité, eur, ure, ien, ite, ise, lle, ons, ent, er, ir, re, es, és, ez, al, at, el**}, et une foction T qui test si une terminaison appartient à $Term$ est une terminaison de mot_i. Alors on défini :

$$\mathrm{F}(mot_i) \left| \begin{array}{ll} = Term(mot_i) & \text{si } Term(mot_i) \neq null \\ = D3_{mot_i} & \text{sinon} \end{array} \right.$$

Nous allons effectuer les test suivants :
$CRF(L_0|feature(mots, F(mot))) \rightarrow Res_{L0}$
$CRF(L_1|feature(mots, F(mot))) \rightarrow Res_{L1}$
$CRF(L_2|feature(mots, F(mot))) \rightarrow Res_{L2}$

Résultat :

etiq.	Nb.feat.	Mots	Extr.	Corr.	Précision	Rappel	F-mesure
L_0	$1.3 * 10^6$	16486	16486	15195	0.92169	0.92169	0.92168
L_1	$6.0 * 10^6$	16486	16486	14522	0.88086	0.88086	0.88085
L_2	$9.0 * 10^6$	16486	16486	14415	0.87437	0.87437	0.87436

test F :
Choix de features mots, 3-dernière-lettres du Lemme
Ici nous allons tester l'influence de trois dernière lettres de lemme. Nous allons construire des features sur $D3_{lemme}$
Nous allons effectuer les testes suivants :
$CRF(L_0|feature(mots, D3_{lemme})) \rightarrow R_{L0}$
$CRF(L_1|feature(mots, D3_{lemme})) \rightarrow R_{L1}$
$CRF(L_2|feature(mots, D3_{lemme})) \rightarrow R_{L2}$

Résultat :

etiq.	Nb.feat.	Mots	Extr.	Corr.	Précision	Rappel	F-mesure
L_0	$1.2 * 10^6$	16486	16486	15427	0. 93576	0.93576	0.93575
L_1	$5.6 * 10^6$	16486	16486	14337	0.86964	0.86964	0.86963
L_2	$8.0 * 10^6$	16486	16486	14246	0.86412	0.86412	0.86411

Conculsion de test sans lemme

Les résultats de quatre tests précédents D, E, F sont résumés par le tableau suivant : On remarque sur la figure (4.4) que sans utilisation de 'Lemme' on perd 2% de résultat

Etiq.	f-mesure D	**f-mesure E**	f-mesure F
L0	0,94242	**0,92168**	0,93575
L1	0,8982	**0,88085**	0,86963
L2	0,89153	**0,87436**	0,86411

TABLE 4.2 – Résultats de tests D, E et F

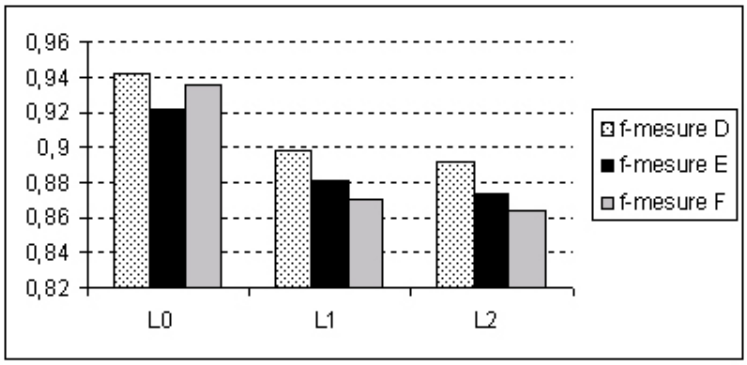

FIGURE 4.4 – comparaison de F-mesure pour les tests D, E et F

de F-mesure en comparaison avec le meilleur test 'TestD' et gagne du temps d'exécution de test. Cette diminution en F-mesure revient à utiliser moins de features dans le test 'Test E' que dans le test 'Test D', puisque le nombre de features générés est le moitié que le test 'Test D'en comparaison avec 'Test D'.
Mais si on compare 'Test E' avec 'Test D' qui ont presque le même nombre de features, on remarque que F-mesure de 'Test E' plus grand que F-mesure de 'Test F'.
On peut assurer le même résultat si on compare 'Test E' avec 'Test A'. A la fin on peut déduire que la**qualité** des features joue le rôle le plus important.

4.3.3 Expérience combinaison de CRF

Nous allons faire des tests en utilisant le résultat de teste d'un niveau d'étiquettes comme feature dans le test qui le suit.

test H : Choix de features mots, Lemmes et les resultats R_{L0}, R_{L01}

Nous allons utiliser les mots et lemmes comme features pour apprendre le niveau L_0. Puis le résultat R_{L0} va être utilisé avec les mots et lemmes comme features pour retrouver le niveau L_1 d'étiquettes.
Nous allons effectuer les tests suivants :
$CRF(L_0|feature(mots, lemme)) \rightarrow R_{L0}$
$CRF(L_1|feature(mots, lemme, R_{L0}) \rightarrow R_{L01}$
$CRF(L_2|feature(mots, lemme, R_{L01}) \rightarrow R_{L012}$

Résultat :

etiq.	Nb.feat.	Mots	Extr.	Corr.	Précision	Rappel	F-mesure
L_0	$1.5 * 10^6$	16486	16486	15301	0.92812	0.92812	0.92811
L_1	$7.0 * 10^6$	16486	16486	14255	0.86467	0.86467	0.86466
L_2	$11.0 * 10^6$	16486	16486	14091	0.85472	0.85472	0.85471

test I : Nous allons créer les features sur mot, $lemme$, $D3_{lemme}$, Res_{L0} et Res_{L01}
Ce test est dérivé du 'Test C'. Ici nous allons utiliser les mot, $lemme$ et $D3_{lemme}$ comme features pour apprendre le niveau L_0. Puis le résultat R_{L0} va être utilisé avec le même fichier précédent pour apprendre le niveau L_1 d'étiquettes etc.
Nous allons effectuer les test suivants :
$CRF(L_0|feature(mot, lemme, D3_{lemme})) \rightarrow R_{L0}$
$CRF(L_1|feature(mot, lemme, D3_{lemme}, Res_{L0}) \rightarrow R_{L01}$
$CRF(L_2|feature(mot, lemme, D3_{lemme}, Res_{L0}, Res_{L01}) \rightarrow R_{L012}$

Résultat :

etiq.	Nb.feat.	Mots	Extr.	Corr.	Précision	Rappel	F-mesure
L0 :	$3 * 10^6$	16486	16486	15554	0.94346	0.94346	0.94345
L1 :	$12 * 10^6$	16486	16486	14441	0.87595	0.87595	0.87594
L2 :	$19 * 10^6$	16486	16486	14296	0.86716	0.86716	0.86715

test J : Nous allons créer les features sur mot , R_{mot}, R_{lemme}, $D3_{mot}$, $D3_{lemme}$, Res_{L0}, Res_{L01}
Ce test est dérivé du 'Test D'. Nous allons effectuer les test suivants :
$CRF(L_0|feature(mot, R_{mot}, R_{lemme}, D3_{mot}, D3_{lemme})) \rightarrow R_{L0}$
$CRF(L_1|feature(mot, R_{mot}, R_{lemme}, D3_{mot}, D3_{lemme}, Res_{L0})) \rightarrow R_{L01}$
$CRF(L_2|feature(mot, R_{mot}, R_{lemme}, D3_{mot}, D3_{lemme}, Res_{L0}, Res_{L01})) \rightarrow R_{L012}$

Résultat :

etiq.	Nb.feat.	Mots	Extr.	Corr.	Précision	Rappel	F-mesure
L0 :	$3 * 10^6$	16486	16486	15537	0.94243	0.94243	0.94242
L01 :	$12 * 10^6$	16486	16486	14875	0.90228	0.90228	0.90227
L012 :	$18 * 10^6$	14868	14868	13232	0.88996	0.88996	0.88995

Conculsion de combinaison de CRF

Les résultats de quatre testes précédents I, J, C et D sont résumés par le tableau (4.3) et la figure (4.5). On remarque que le résultat d'apprentissage par combinaison des CRF du 'Test J' déduit du 'Test D' avec même features donne même résultats que le test 'Test

D'.
Mais de point de vu complexité de calcule et prétraitement Il est préférable d'utilisé les le test 'Test D'. D'autre plus on remarque le choix de features de test 'Test D' ou 'Test J' donne des résultats plus stables ce qui est préférable au choix des features de 'Test C' ou 'Test I'.

etiq,	F-mesure D	F-mesure J	F-mesure C	F-mesure I
L0	0,94242	0,94242	0,94345	0,94345
L1	0,8982	0,90227	0,90002	0,87594
L2	0,89153	0,88995	0,82347	0,86715

TABLE 4.3 – Résultats de tests A, B, C et D

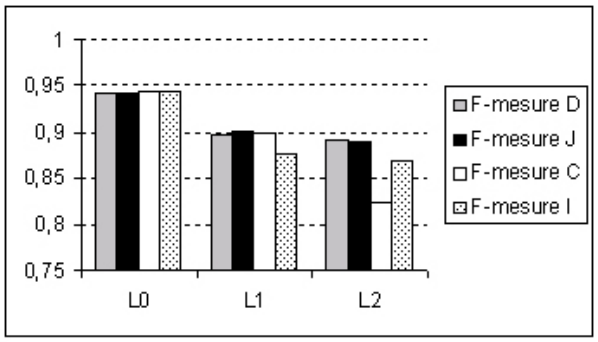

FIGURE 4.5 – comparaison de F-mesure pour les test I, J, C et D

4.3.4 Expérience composition de CRF

L'étiquetage que nous avons étudié a une structure hiérarchique comme le montre la figure 4.6 Première niveau représente les étiquettes de catégorie syntaxique, deuxième ni-

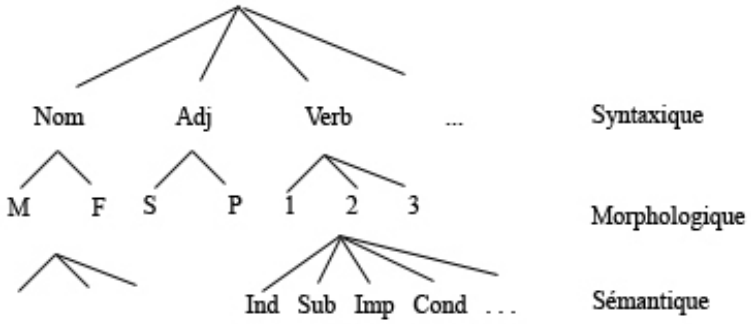

FIGURE 4.6 – Arbre Hiérarchique des étiquettes

veau représente les étiquettes de catégorie morphologique, et troisième niveau représente les étiquettes de catégorie sémantique. Cette structure est purement linguistique. De point de vue apprentissage (informatique) nous pouvons construire l'arbre des étiquettes d'une manière différente en permutant les niveaux syntaxiques, morphologiques et sémantiques

en gardant la même allure des étiquettes figure (4.7)
D'ou il y a beaucoup de permutations possibles pour reconstruire l'ensemble des étiquettes

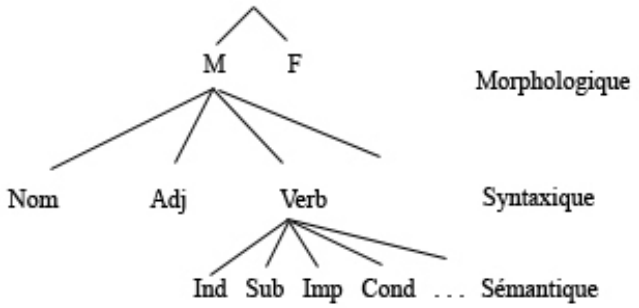

FIGURE 4.7 – Arbre Hiérarchique des étiquettes inversée

avec leurs catégories syntaxiques, morphologiques et sémantiques.

Définition : L'ensemble de symbole d'étiquettes noté $'S'_e$ est l'ensemble des symboles utilisées pour définir un groupe d'étiquettes conformément avec leur propriétés de compostions. par exemple l'étiquette $'DETFP' =' DET|F|P'$ est composé de trois symboles : 'DET' (Déterminant), 'F' (Féminin) et 'P' (Pluriel) est correct mais $'DETFM' =' DET|F|M'$ n'est pas correcte car un 'DET' ne peut pas être féminin et masculin au même temps. On définit le terme 'Compatible' pour deux symboles si ces deux symboles ne peuvent pas se trouver dans une étiquette au même temps.
Par exemple :
$\{M, F, I\}$,$\{S, P\}$,$\{1, 2, 3\}$ sont trois groupes de symboles compatibles
$\{M, P, 3\}, \{DET, M, P, DEM\}$ sont deux groupes de symboles non compatibles.
Définition : On appelle composant d'étiquettes un groupe de symboles compatibles.

Dans les exemples suivants on a chaque étiquette est composée de symboles des groupes de symboles compatibles

DETMPDEM $\leftrightarrow$ DET|M|P|DEM
DETerninant|**M**asculin|**P**luriele|**DEM**enstratif

V2PSUB $\leftrightarrow$ V|2|P|SUB
Verbe de 2^{ieme} groupe **P**luriel **SUB**jenctif

On remarque pour un déterminant ne peut pas être au même temps masculin et féminin, ni un verbe peut être au même temps deuxième groupe et troisième group ni être subjonctif et conditionnelle au même temps. Alors nous allons chercher une solution possible de composantes l'ensemble des étiquettes des trois niveaux grammaticales de l'arbre Hiérarchique. Pour ça nous allons suivre les réécriture suivants :

– L'ensemble des étiquettes qui représente tous les verbe :
$\{V\} \otimes \{1, 2, 3\} \otimes \{S, P\} \otimes \{CON, IMP, SUB, INDF, INDI, INDP\}$

– L'ensemble des étiquettes qui représente tous les Déterminants :
$\{DET\} \otimes \{M, F, I\} \otimes \{S, P\} \otimes \{IND, DEM, DEF, POSS, INT\}$
– L'ensemble des étiquettes qui représente tous les pronoms :
$\{Pr\} \otimes \{M, F, I\} \otimes \{S, P\} \otimes \{REL, IND, DEM, PER\}$
– L'ensemble des étiquettes qui représente tous les pronoms Invariant :
$\{PrI\} \otimes \{POSS, IND, DEM, PER, INT\}$
– L'ensemble des étiquettes qui représente tous les Nom :
$\{NC\} \otimes \{M, F, I\} \otimes \{S, P\}$
– L'ensemble des étiquettes qui représente tous les adjectifs :
$\{ADJ\} \otimes \{M, F, I\} \otimes \{S, P\}$
– Les restes ont le même ecriture {ADV}, {CONJ}, {CH},{PCT}...

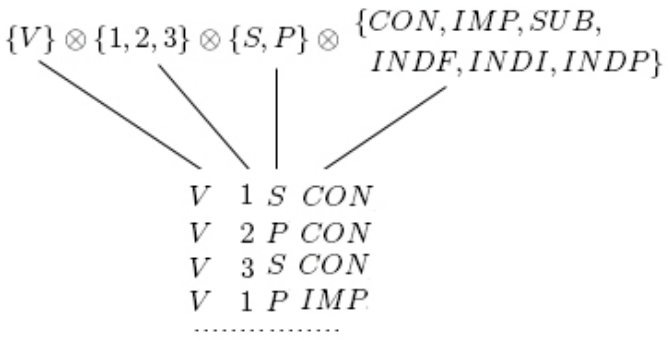

FIGURE 4.8 – Reecriture des étiquettes de 'V'

La figure (figure :4.8) comment on peut reproduire toutes les étiquettes qui se composent avec 'V'.
Si on note :
$C0 = \{V, DET, Pr, PrI, NC, ADJ, ADV, CONJ, PCT, CH, INT, MI\}$
$C1 = \{1, 2, 3\}$
$C2 = \{M, F\}$
$C3 = \{S, P\}$
$C4 = \{CON, IMP, SUB, INDF, INDI, INDP\}$
$C5 = \{IND, DEM\}$
$C6 = \{POSS, INT\}$
$C7 = \{DEF\}$
$C8 = \{PER\}$
$C9 = \{REL\}$
On aura le tableau de compatibilité le tableau4.4 montre que nous pouvons grouper des composants en restant compatible. Par exemple si en regroupe G1 = C1 + C2 car 1,2,3 ne figure jamais avec M,F et par contre on ne peut pas regrouper 1,2,3 et S, P. De même on ne peut pas regrouper M,F et S, P.
de même, on peut regrouper C4 avec C5 , C6, C7, C8 ou C9.

Utilisation des composantes :
Supposons que nous n'avons pas regroupé des composants Alors pour appliquer l'apprentissage sur le texte donné Il faut appliquer CRF sur le texte avec chaque composant.

$\otimes$	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	$C9$
$\{V\}$	x	-	x	x	-	-	-	-	-
$\{DET\}$	-	x	x	-	x	x	x	x	-
$\{Pr\}$	-	x	x	-	x	-	-	x	x
$\{PrI\}$	-	-	-	-	x	x	-	x	-
$\{NC\}$	-	x	x	-	-	-	-	-	-
$\{ADJ\}$	-	x	x	-	-	-	-	-	-
$\{ADV\}$	-	-	-	-	-	-	-	-	-
$\{CONJ\}$	-	-	-	-	-	-	-	-	-
$\{PCT\}$	-	-	-	-	-	-	-	-	-
$\{CH\}$	-	-	-	-	-	-	-	-	-
$\{INT\}$	-	-	-	-	-	-	-	-	-
$\{MI\}$	-	-	-	-	-	-	-	-	-

TABLE 4.4 – Compatibilité des composantes des étiquettes

C'est à dire :
$CRF(C_0|feature(mots, lemmes)) \rightarrow Res_{C0} \Rightarrow fichier\{mots, lemmes, C_0, Res_{C0}\}$
$CRF(C_1|feature(mots, lemmes)) \rightarrow Res_{C1} \Rightarrow fichier\{mots, lemmes, C_1, Res_{C1}\}$
...
$CRF(C_9|feature(mots, lemmes)) \rightarrow Res_{C9} \Rightarrow fichier\{mots, lemmes, C_9, Res_{C9}\}$
Or le résultat de teste suivant nous montre qu'il vaut mieux regrouper les composantes C1, C2,... C9 de la manière suivante :

$G0 = C0$
$G1 = C1 \oplus C2$
$G2 = C3$
$G3 = C4 \oplus C5 \oplus C6 \oplus C7 \oplus C8 \oplus C9$
Remarque : Il faut bien définir un ordre pour les composants pour la composition des étiquettes et pour rester compatible avec l'ensemble d'étiquettes à retrouver. par exemple : 'DETFS', 'DETMP' sont acceptable mais 'FSDET', 'DETPM' ne sont pas acceptables. Le tableau de compatibilité de ce groupement est donné par le tableau(4.6).
Nous avons étiqueté le corpus avec ces nouveaux composant (groupes) et nous avons choisi les meilleurs choix des features pour réaliser les tests. Les résultats de tests trouvés avec les étiquettes de chaque groupe G0, G1, G2 et G3 sont dans le tableau (4.5). Le fichier de sortie de test a la forme de composition de CRF est donné par le tableau(4.7). Pour la méthode d'apprentissage par composants définir les scriptes de prétraitements pour déduire les résultats de tests ce qui n'est pas encore fini. De plus il faut adapter un mesure qui reflète les propriétés de compatibilité des composant et les symboles définies par chaque composant. On réalité on peut juger plus précisément que dire qu'un mot est étiqueté 'vrai' ou 'faux' ! on peut donner un pourcentage pour le jugement. Par exemple : le première ligne du tableau(4.7) 'la' est 'DETFSDEM' si c'était 'DETMPDEM' ce qui est faux mais moitié 'vrai' car le mot 'la' au moins est trouvé comme 'DET'.
Test K : Nous avons réaliser un test par CRF pour apprendre L_2 avec les specification suivants des features :
$CRF(L_2|Mot, Lemme, D3_{mot}, G0, G1, G2, G3) \rightarrow Res_{L2}$

Résultat : le **F-mesure = 0,89**.
Ce résultat peut être amélioré par un autre chois de composition de groupes G_i.(Reste à démontrer).
Un modéle de décision que nous pensions à réaliser est le calcule de résultats à partie des CRF de chaque composante 'G0, G1, G2, G3' et les alignées ensemble (comme le montre le tableau4.7),puis déduire l'étiquette à chaque ligne à l'aide des régles de compatibilité des éléments des composantes. Un exemple de régule de compatibilités :
1) si element de G0 noté Elem(G0)='ADV' Alors Elem(G1)=Elem(G2)=Elem(G3)= {}
2) si element de G0 noté Elem(G0)='CH' Alors Elem(G1)=Elem(G2)=Elem(G3)= {}
3) si element de G0 noté Elem(G0)='CONJ' Alors Elem(G1)=Elem(G2)= {}, Elem(G3)={COO,SUB}
D'où pour la régle(1), il faut éliminer tous les symboles des colonne G2, G3, G4 (tableau 4.7) en face de ADV. Car un adverbe ne peut se combiné avec d'autres symboles.
Test L : Une résultat par calcule à la main, donne un **F-mesure = 0,93**
. Un très bon résultat peut être améliorer si on essaye de créer des composantes différentes de composantes simples que nous avons utilisées.
La figure(4.9) montre une comparaison entre tests 'TestD', 'Test K' et 'Test L'.

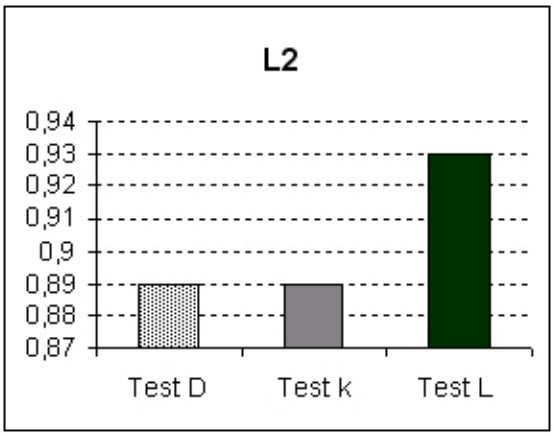

FIGURE 4.9 – comparaison de F-mesure pour les tests D, K et L

Conclusion de composition de CRF

Ce méthode a des avantages sur le méthode d'apprentissage avec les étiquettes d'arbre hiérarchique sont les suivants :

- Nombre d'étiquettes plus petit que 40 étiquettes en comparaison au nombre d'étiquettes de dérniére niveau hiérarchique L_2 qui est ≥ 80.
- Nombre de features générés par CRF++ varie de $0.3*10^6$ à $2.5*10^6$ tandis que avec les étiquettes classiques varie de $10*10^6$ à $20*10^6$. D'où le temps d'apprentissage est beaucoup plus rapide (1/10).
- Les règles de compatibilités peuvent jouer un rôle très important dans la décision de préférer une étiquette d'un mesure (dépend d'une choix de features) à une autre étiquette par autre mesure (autre choix de features). Nous pouvons toujour préférer le choix de meilleur résultat.
- Il est possible de combiner les meilleurs résultats de plusieurs mesures.

C0	F-mesure
ADJ	0,76091
ADV	0,94352
CH	0,80596
CONJ	0,95447
DET	0,96846
INT	0,98038
MI	0,45225
N	0,91274
Pr	0,95991
PCT	1,00000
PREP	0,96090
PRES	0,97993
PRO	0
V	0,96390
Tot.	**0,94345**

C1,C2	F.
1	0,95274
2	0,92938
3	0,94503
F	0,85922
I	0,93474
M	0,83895
x	0,94619
Tot.	**0,92120**

C3	F-mesure
I	0,92892
P	0,91750
S	0,91745
x	0,94790
Tot.	**0,92938**

C4-9	F-mesure
CON	0,4210
DEF	0,9324
DEM	0,9540
IMP	0,6363
IND	0,8024
INDF	0,0000
INDI	0,8622
INDP	0,9395
INF	0,8419
INT	0,4615
PARP	0,8758
PER	0,9707
PERS	0,0000
POSS	0,9316
REL	0,8678
SUB	0,0741
x	0,9716
Tot.	**0,94958**

TABLE 4.5 – La mesure pour retrouver les étiquettes des groupes G1, G2, G3 et G4

– Le méthode de composition est un méthode général applicable à d'autres types de problèmes d'apprentissage par les CRFs.

$G0$	$G1$	$G2$	$G3$
$\{V\}$	x	x	x
$\{DET\}$	x	x	x
$\{Pr\}$	x	x	x
$\{PrI\}$	-	-	x
$\{NC\}$	x	x	-
$\{ADJ\}$	x	x	-
$\{ADV\}$	-	-	-
$\{CONJ\}$	-	-	-
$\{PCT\}$	-	-	-
$\{CH\}$	-	-	-
$\{INT\}$	-	-	-
$\{MI\}$	-	-	-

TABLE 4.6 – Compatibilité des composantes des étiquettes

Mot	**Lemme**	$D3_{mot}$	$G0$	$G1$	$G2$	$G3$	L_2 à trouver
la	le	a	DET	F	S	DEF	DETFSDEF
situation	situation	ion	N	F	S	x	NCFS
de	de	de	PREP	x	x	x	PREP
mon	mon	mon	DET	I	I	POSS	DETIPOSS
fils	fils	ils	N	I	I	x	NCI
oui	oui	oui	ADV	F	S	CON	ADV
mais	mais	ais	CONJ	x	x	x	CONJCOO
c'est	c'est	est	PRES	x	x	x	PRES
à_cause_de	à_cause_de	_de	ADV	I	I	x	PREP
la	le	a	DET	F	S	DEF	DETFSDEF
situation	situation	ion	N	F	S	x	NCFS
géographique	géographique	que	CONJ	I	I	x	ADJI
de	de	de	PREP	x	x	x	PREP
la	le	a	DET	F	S	DEF	DETFSDEF
ville	ville	lle	N	F	S	x	NCFS
euh	euh	euh	INT	x	x	x	INT

TABLE 4.7 – retrouver les étiquettes par méthode de composants

Chapitre 5

Conclusion et perspective

Conclusion :
Nous avons montré dans ce stage que les techniques des CRF (Conditional Random Fields) sont des bonnes techniques pour apprendre à étiqueter les mots du corpus issu de transcriptions de la parole. Avec le groupe d'étiquettes classées hiérarchiquement que nous avons construit avec les linguistes nous avons trouvé des bons résultats pour les trois niveaux d'étiquettes, 94% pour le niveau syntaxique, 90% pour le niveau morphologique et 89% pour le niveau sémantique.
En comparaison au 'CORDIAL' et après correction à la main et prétraitement du corpus, le résultat de l'apprentissage du groupe d'étiquettes en F-mesure égale à 89% par rapport à 96% pour 'CORDIAL' qui fonctionne sans apprentissage, en tenant en compte que 'CORDIAL' s'appuie sur des dictionnaires et sur un analyseur syntaxique pour l'étiquetage.
De plus, la notion de phrase (début, fin, ponctuation ...) est très importante pour les CRF. Ce qui est absent dans le corpus oral qu'on a utilisé.
Par contre, avec l'outil CRF++ et sans les lemmes qui viennent avec le corpus oral (Tests sans lemmes) où les features ont été construits à partir des mots seulement, nous avons eu en F-mesure un pourcentage de 87% avec un différence de 2%.
A la fin On peut dire que la port est ouvert devant les CRF et les choix sont différentes pour améliorer leurs résultats sur tous pour la notion de décomposition des étiquettes et sa application à des autres domaines d'apprentissage par les CRF.

Perspective :
Les CRF appris fiables sur le corpus issu de transcriptions de la parole avec ses ambiguïtés et les phrases males structurés.

Au court terme :
Nous pensons dans la future tester des différents corpus écrits scientifique ou littéraire, ou ceux du issus de magazines.
Aussi nous penserions à segmenter les phrases du corpus oral en appliquant les CRF et essayer de tester l'apprentissage sur le nouveau corpus segmenté.

Au long terme :
Nous pensons dans la future approfondir la modélisation des composant pour trouver un

modèle de système d'apprentissage par composants des CRF et l'appliquer aux corpus oraux et non oraux et retrouver des autres domaines d'application des 'CRF par Composants'.
Au niveau de l'implémentation de CRF avec l'outil CRF++, nous penserions à changer les parties qui gèrent la création des features pour générer des règles d'apprentissage par exemples négatives ou pour générer des feature à partir des expressions réguliers.

Annexe A

Groupe des Etiquettes 'CORDIAL'

Les étiquettes utilisées

ADJFP	INT	PCTFAIB	SUB	VINDP1P
ADJFS	NCFIN	PCTFORTE	UEUPH	VINDP1S
ADJHMS	NCFP	PDP	VCONP1P	VINDP2P
ADJIND	NCFS	PDS	VCONP1S	VINDP2S
ADJINT	NCHMS	PIFS	VCONP2P	VINDP3P
ADJINV	NCI	PII	VCONP2S	VINDP3S
ADJMIN	NCMIN	PIMP	VCONP3P	VINDPS1S
ADJMP	NCMP	PIMS	VCONP3S	VINF
ADJMS	NCMS	PIPIG	VIMPP1P	VPARPFP
ADJNUM	NCPIG	PISIG	VIMPP2P	VPARPFS
ADJORD	NCSIG	PPER1P	VIMPP2S	VPARPMP
ADJPIG	NHMIN	PPER1S	VINDF1P	VPARPMS
ADJSIG	NPFS	PPER2P	VINDF1S	VPARPRES
ADV	NPHSIG	PPER2S	VINDF2P	VSUBP1S
COO	NPI	PPER3P	VINDF3S	VSUBP2S
DETDEM	NPMS	PPER3S	VINDI1P	VSUBP3P
DETDFS	NPSIG	PREP	VINDI1S	VSUBP3S
DETDMS	PRFS	VINDI2P		
DETDPIG	PRI	VINDI3P		
DETIFS	PRMS	VINDI3S		
DETIMS				
DETPOSS				

ADJFP = ADJ F P = **Adj**ectif **f**eminin **p**luriel
ADJFS = ADJ F S = **Adj**ectif **f**eminin **s**ingulier

Annexe B

Groupe des Etiquettes Hiérarchiques

Les trois niveaux d'étiquettes utilisées :

Etiquettes niveau syntaxique :

ADJ, N, V, CONJ, P, DET, ADV, CH, PREP, PRES, MI, INT, PCT, CONJ

Etiquettes niveau morphologique :

ADJMS, ADJMP, ADJFS, ADJFP, ADJI
NCMS, NCMP, NCFS, NCFP, NCI, NP
V1SINDP, V1SINDI, V1SINDF, V1SINDPS, V1SSUB, V1SCON
V2SINDP, V2SINDI, V2SINDF, V2SINDPS, V2SSUB, V2SCON, V2SIMP
V3SINDP, V3SINDI, V3SINDF, V3SINDPS, V3SSUB, V3SCON V1PINDP, V1PINDI, V1PINDF, V1PINDPS, V1PSUB, V1PCON, V1PIMP
V2PINDP, V2PINDI, V2PINDF, V2PINDPS, V2PSUB, V2PCON, V2PIMP
V3PNDP, V3PINDI, V3PINDF, V3PINDPS, V3PSUB
VINF, VPARPRES, VMSPARP, VMPPARP, VFSPARP, VFPPARP
CONJ
PMS, PMP, PFS, PFP, PI
DETMS, DETFS, DETP, DETI

Etiquettes niveau sémantique :

ADJMS, ADJMP, ADJFS, ADJFP, ADJI
NCMS, NCMP, NCFS, NCFP, NCI, NP
V1SINDP, V1SINDI, V1SINDF, V1SINDPS, V1SSUB, V1SCON
V2SINDP, V2SINDI, V2SINDF, V2SINDPS, V2SSUB, V2SCON, V2SIMP
V3SINDP, V3SINDI, V3SINDF, V3SINDPS, V3SSUB, V3SCON
V1PINDP, V1PINDI, V1PINDF, V1PINDPS, V1PSUB, V1PCON, V1PIMP
V2PINDP, V2PINDI, V2PINDF, V2PINDPS, V2PSUB, V2PCON, V2PIMP
V3PNDP, V3PINDI, V3PINDF, V3PINDPS, V3PSUB
VINF, VPARPRES, VMSPARP, VMPPARP, VFSPARP, VFPPARP
CONJSUB, CONJCOO

PMSREL, PMSINT, PMSPER, PMSPOSS, PMSDEM, PMSIND
PMPREL, PMPINT, PMPPER, PMPPOSS, PMPDEM, PMPIND
PFSREL, PFSINT, PFSPER, PFSPOSS, PFSDEM, PFSIND
PFPREL, PFPINT, PFPPER, PFPPOSS, PFPDEM, PFPIND
PIREL, PIINT, PIIND, PIPER, PIDEM
DETMSDEM, DETMSPOSS, DETMSDEF, DETMSIND, DETMSINT
DETFSDEM, DETFSPOSS, DETFSDEF, DETFSIND, DETFSINT
DETPDEM, DETPPOSS, DETPDEF, DETPIND, DETPINT
DETIDEM, DETIPOSS, DETIDEF, DETIIND, DETIINT

Annexe C

Remplacement des Etiquettes

Le scripte de replacement pour changer les étiquettes initial de CORDIAL par les étiquettes hiérarchiques :

```
#/bin/bash
sed "{
s/m\o047\tme\tP\tPI\tPIPER\ /m\o047\tme\tP\tP1I\tP1IPER/g
s/me\tme\tP\tPI\tPIPER\ /me\tme\tP\tP1I\tP1IPER/g
s/qui\tqui\tPRI\ /qui\tqui\tP\tPI\tPIREL/g
s/que\tque\tPRI\ /que\tque\tP\tPI\tPIREL/g
s/qu'\tque\tPRI\ /qu'\tque\tP\tPI\tPIREL/g
s/où\toù\tPRI\ /où\toù\tP\tPI\tPIREL/g
s/pron\tpron\tNCI\ /pron\tpron\tMI\tMI\tMI/g
s/pi\tpi\tN\tNCI\tNCI\ /pi\tpi\tINT\tINT\tINT/g
s/l\o047\tle\tP\tPI\tPIPER\ /l\o047\tle\tP\tP3I\tP3IPER/g
s/l\o047\tle\tPPER3S\ /l\o047\tle\tP\tP3I\tP3IPER/g
s/on\ton\tPPER3S\ /on\ton\tP\tP3I\tP3IPER/g
s/ma\tmon\tDET\tDET\tDETPOSS\ /ma\tma\tDET\tDETFS\tDETFSPOSS/g
s/son\tson\tDET\tDET\tDETPOSS\ /son\tson\tDET\tDETI\tDETIPOSS/g
s/son\tson\tDET\tDET\tDETIPOSS\ /son\tson\tDET\tDETI\tDETIPOSS/g
s/t\o047\tte\tP\tPI\tPIPER\ /t\o047\tte\tP\tP2\tP2PER/g
s/s\o047\tse\tPPER3S\ /s\o047\tse\tP\tP3I\tP3IPER/g
s/s\o047\tse\tP\tPI\tPIPER\ /s\o047\tse\tP\tP3I\tP3IPER/g
s/soi\tsoi\tP\tPI\tPIPER\ /soi\tsoi\tP\tP3I\tP3IPER/g
s/je\tje\tP\tPI\tPIPER\ /je\tje\tP\tP1I\tP1IPER/g
s/j'\tje\tP\tPI\tPIPER\ /j'\tje\tP\tP1I\tP1IPER/g
s/tu\ttu\tP\tPI\tPIPER\ /tu\ttu\tP\tP2I\tP2IPER/g
s/il\til\tP\tPMS\tPMSPER\ /il\til\tP\tP3MS\tP3MSPER/g
s/il\til\tPPER3S\ /il\til\tP\tP3MS\tP3MSPER/g
s/elle\telle\tPPER3S\ /elle\telle\tP\tP3FS\tP3FSPER/g
s/elle\telle\tP\tPFS\tPFSPER\ /elle\telle\tP\tP3\tP3FSPER/g
s/elle\telle\tP\tPI\tPIPER\ /elle\telle\tP\tP3\tP3FSPER/g
s/on\ton\tP\tPI\tPIPER\ /on\ton\tP\tP3I\tP3IPER/g
s/lui\tlui\tPPER3S\ /lui\tlui\tP\tP3I\tP3IPER/g
s/y\o047\ty\tPPER3S\ /y\o047\ty\tP\tP3I\tP3IPER/g
```

s/en\ten\tPPER3S\ /en\ten\tP\tP3I\tP3IPER/g
s/se\tse\tPPER3S\ /se\tse\tP\tP3I\tP3IPER/g
s/se\tse\tP\tPI\tPIPER\ /se\tse\tP\tP3I\tP3IPER/g
s/nous\tnous\tP\tPI\tPIPER\ /nous\tnous\tP\tP1I\tP1IPER/g
s/vous\tvous\tP\tPI\tPIPER\ /vous\tvous\tP\tP2I\tP2IPER/g
s/ils\til\tP\tPMP\tPMPPER\ /ils\til\tP\tP3MP\tP3MPPER/g
s/ils\til\tPPER3P\ /ils\til\tP\tP3MP\tP3MPPER/g
s/elles\telle\tP\tPFP\tPFPPER\ /elles\telle\tP\tP3FP\tP3FPPER/g
s/elles\telle\tPPER3P\ /elles\telle\tP\tP3FP\tP3FPPER/g
s/toi\ttoi\tP\tPI\tPIPER\ /toi\ttoi\tP\tP2I\tP2IPER/g
s/moi\tmoi\tP\tPI\tPIPER\ /moi\tmoi\tP\tP1I\tP1IPER/g
s/lui\tlui\tPPER3S\ /lui\tlui\tP\tP3MS\tP3MSPER/g
s/soi\tsoi\tPPER3S\ /soi\tsoi\tP\tP3I\tP3IPER/g
s/eux\tlui\tPPER3P\ /eux\tlui\tP\tP3MP\tP3MPPER/g
s/eux-mêmes\tlui-même\tPPER3P\ /eux-même\tlui-même\tP\tP3MP\tP3MPPER/g
s/lui-même\tlui-même\tPPER3S\ /lui-même\tlui-même\tP\tP3MS\tP3MSPER/g
s/elle-même\telle-même\tPPER3S\ /elle-même\telle-même\tP\tP3FS\tP3FSPER/g
s/soi-même\tsoi-même\tPPER3S\ /soi-même\tsoi-même\tP\tP3I\tP3IPER/g
s/elle-même\telle-même\tPPER3S\ /elle-même\telle-même\tP\tP3FS\tP3FSPER/g
s/cela\tcela\tP\tP\tPDEM\ /cela\tcela\tP\tPI\tPIDEM/g
s/le\tle\tPPER3S\ /le\tle\tP\tP3MS\tP3MSPER/g
s/le\tle\tP\tPI\tPIPER\ /le\tle\tP\tP3MS\tP3MSPER/g
s/la\tla\tPPER3S\ /la\tle\tP\tP3FS\tP3FSPER/g
s/la\tle\tPPER3S\ /la\tle\tP\tP3FS\tP3FSPER/g
s/la\tla\tP\tPI\tPIPER\ /la\tla\tP\tP3FS\tP3FSPER/g
s/les\tle\tPPER3P\ /les\tle\tP\tP3I\tP3IPER/g
s/les\tle\tP\tPI\tPIPER\ /les\tle\tP\tP3I\tP3IPER/g
s/leur\tleur\tPPER3P\ /leur\tleur\tP\tP3I\tP3IPER/g
s/leur\tleur\tP\tPI\tPIPER\ /leur\tleur\tP\tP3I\tP3IPER/g
s/ce\tce\tP\tP\tPDEM\ /ce\tce\tP\tPMS\tPMSDEM/g
s/ceux\tcelui\tP\tPP\tPPDEM\ /ceux\tcelui\tP\tPMP\tPMPDEM/g
s/ça\tça\tP\tP\tPDEM\ /ça\tça\tP\tPI\tPIDEM/g
s/cet\tce\tDET\tDET\tDETDEM\ /cet\tce\tDET\tDETMS\tDETMSDEM/g
s/ces\tce\tDET\tDET\tDETDEM\ /ces\tce\tDET\tDETI\tDETIDEM/g
s/ces\tce\tDET\tDETP\tDETPDEM\ /ces\tce\tDET\tDETI\tDETIDEM/g
s/ce\tce\tDET\tDET\tDETDEM\ /ce\tce\tDET\tDETMS\tDETMSDEM/g
s/cette\tce\tDET\tDET\tDETDEM\ /cette\tce\tDET\tDETFS\tDETFSDEM/g
s/leur\tleur\tDET\tDET\tDETPOSS\ /leur\tleur\tDET\tDETFS\tDETFSPOSS/g
s/leur\tleur\tDET\tDET\tDETPOSS\ /leur\tleur\tDET\tDETMS\tDETMSPOSS/g
s/votre\tvotre\tDET\tDET\tDETPOSS\ /votre\tvotre\tDET\tDETMS\tDETMSPOSS/g
s/leurs\tleur\tDET\tDET\tDETPOSS\ /leurs\tleur\tDET\tDETP\tDETPPOSS/g
s/vos\tvotre\tDET\tDET\tDETPOSS\ /vos\tvotre\tDET\tDETP\tDETPPOSS/g
s/votre\tvotre\tDET\tDETMS\tDETMSPOSS\ /votre\tvotre\tDET\tDETI\tDETIPOSS/g
s/mes\tmon\tDET\tDET\tDETPOSS\ /mes\tmon\tDET\tDETP\tDETPPOSS/g
s/mon\tmon\tDET\tDET\tDETPOSS\ /mon\tmon\tDET\tDETI\tDETIPOSS/g
s/pi\tpi\tN\tNC\tNCI\ /pi\tpi\tINT\tINT\tINT/g

s/hum\thum\tINT\ /hum\thum\tINT\tINT\tINT/g
s/voilà\tvoilà\tPREP\tPREP\tPREP\ /voilà\tvoilà\tPRES\tPRES\tPRES/g
s/voilà\tvoilà\tINT\tINT\tINT\ /voilà\tvoilà\tPRES\tPRES\tPRES/g
} ” < $1 ¿ $1.new

Annexe D

Exemple de traitement des features

Le scripte ‘trait5’ montre comment construire les features 3 dérnières lettres du ‘mot’ et ‘lemme’ dans le corpus à tester

```
#!/bin/bash
rm -f  $1.tr5

count=0
while read word0
do
a=`wc -l $1 | sed 's/\([0-9]*\).*/\1/'`
count=`expr $count + 1`
word1=`echo $word0 | sed 's/^\([0-9a-zA-Z\_\?\o047\-]*\).*/\1/'`
word2=`echo $word0 | sed 's/^'$word1'[ \t]//'| sed 's/^\([0-9a-zA-Z\_\?\o047\-]*\).*/\1/'`
word3=`echo $word0 | sed 's/^'$word1'[ \t]'$word2'[ \t]//'`

lw1=${#word1}
lw2=${#word2}

if [ "$word1" != "$word2" ]
then
        if [ $lw1 -gt $lw2 ]
        then
          lw=$lw2
        else
          lw=$lw1
        fi
        i=1
        while [ $i -le $lw ]&&[ "${word1:0:$i}" = "${word2:0:$i}" ]
        do
          i=`expr $i + 1`
        done
         i=`expr $i - 1`

        if [ "${word1:$i:$lw1}" != "" ]
        then
          rword1="${word1:$i:$lw1}"
        else
          rword1="x"
        fi
```

```
        if [ "${word2:$i:$lw2}" != "" ]
        then
        rword2="${word2:$i:$lw2}"
        else
        rword2="x"
        fi


else
        if [ $lw1 -ge 3 ]
        then
          rword1="${word1:`expr $lw1 - 3`}"
        else
          rword1=$word1
        fi
        if [ $lw2 -ge 3 ]
        then
          rword2="${word2:`expr $lw2 - 3`}"
        else
          rword2=$word2
        fi
        echo $word1 $lw1 $rword1   "|"  $word2 $lw2 $rword2
fi

if [ "$word0" != "" ]
then
        echo "$word1  $word2          $rword1         $rword2         $word3" | sed 's/ /\t/g'
>> $1.tr5
else
        echo "" >> $1.tr5
fi
done < $1
```

Bibliographie

[1] Crf++ : Yet another crf toolkit. 2009.

[2] Dan Klein David Rosenberg and Ben Taskar. Mixture-of-parents maximum entropy markov models. 2007.

[3] Iris Eshkol. http ://www.univ-orleans.fr/eslo/spip.php ?article45. 2009.

[4] Rémi Gilleron, Florent Jousse, Isabelle Tellier, and Marc Tommasi. Xml document transformation with conditional random fields. In *INEX*, pages 525–539, 2006.

[5] J. M. Hammersley and P. Clifford. Markov field on finite graphs and lattices. 1971.

[6] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[7] Yang Liu, Elizabeth Shriberg, Andreas Stolcke, Dustin Hillard, Mari Ostendorf, and Mary P. Harper. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on Audio, Speech & Language Processing*, 14(5) :1526–1540, 2006.

[8] Andrew McCallum. Efficiently inducing features of conditional random fields. In *UAI*, pages 403–410, 2003.

[9] Andrew McCallum and Kachites. Mallet : A machine learning for language toolkit. 2002.

[10] Thi Minh Huyen Nguyen, Laurent Romary, and Xuan Luong Vu. Une étude de cas pour l'étiquetage morpho-syntaxique de textes vietnamiens. In *Traitement Automatique des Langues Naturelles - TALN'2003*, Batz-sur-mer, France, 06 2003. ATALA (Association pour le Traitement Automatique des LAngues), none. Colloque avec actes sans comité de lecture. nationale.

[11] Naoaki Okazaki. Crfsuite : a fast implementation of conditional random fields crfs. 2007.

[12] L. Stewart, Xuming He, and R. S. Zemel. Learning flexible features for conditional random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8) :1415–1426, Aug 2008.

[13] C. Sutton and A. Mccallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.