Séminaire TALEP

Alexis Nasr

mai 2016

Table of contents

Exemple

stack	buffer	depset
X	les petits ruisseaux font les grandes rivières	
X les	petits ruisseaux font les grandes rivières	
X les petits	ruisseaux font les grandes rivières	(ruisseaux, mod, petits)
X les	ruisseaux font les grandes rivières	(ruisseaux, mod, les)
X	ruisseaux font les grandes rivières	
X ruisseaux	font les grandes rivières	(font, suj ,ruisseaux)
X	font les grandes rivières	
X font	les grandes rivières	
X font les	grandes rivières	
X font les grandes	rivières	(rivières, mod, grandes)
X font les	rivières	(rivières, mod, les)
X font	rivières	(font, obj, rivières)
X	font	(X, root, font)
	X	
X		

Principe

- L'analyse d'une phrase $S = w_1 \dots w_n$ est vue comme une séquence d'opérations (appelées transitions).
- L'exécution de ces opérations construit l'analyse de S
- Une opération s'applique sur une configuration pour produire une nouvelle configuration.
- Une configuration $C = (\sigma, \beta, \Delta)$ où
  - β est une file qui contient initialement tous les mots de S. β_0 est le premier mot dans la file
  - σ est une pile contenant des mots de S au fur et à mesure de leur lecture. σ_0 est le mot au sommet de la pile
  - Δ est l'ensemble des dépendances construites (l'arbre de dépendances)
- Configuration initiale : $C_0 = (X, S, \emptyset)$

Les transitions

Arc-gauche$_r$ construit une dépendance étiquetée r ayant pour gouverneur le premier mot de la file (β_0) et pour dépendant le mot en sommet de pile (σ_0)

$$(\sigma|w_i, w_j|\beta, \Delta) \Rightarrow (\sigma, w_j|\beta, \Delta \cup \{(w_j, r, w_i)\})$$

Arc-droit$_r$ construit une dépendance étiquetée r ayant pour gouverneur σ_0 et pour dépendant β_0

$$(\sigma|w_i, w_j|\beta, \Delta) \Rightarrow (\sigma, w_i|\beta, \Delta \cup \{(w_i, r, w_j)\})$$

Shift enlève β_0 de la file pour le mettre dans la pile

$$(\sigma, w_i|\beta, \Delta) \Rightarrow (\sigma|w_i, \beta, \Delta)$$

Prédiction des transitions

- Un classifieur prédit pour toute configuration la transition la plus vraisemblable
- Une configuration est représentée sous la forme d'un vecteur de features
- Modèle simple : les décisions sont (presque) indépendantes les unes des autres
- La structure syntaxique produite (Δ) constitue une image des transitions effectuées jusque là

Quatre types de modèles de features

- **Features Lexicales**
  composition de deux fonctions : une fonction d'adresse (σ_0, σ_1, β_0 ...) une fonction d'attribut (forme, partie de discours, lemme ...)
- **Features Syntaxiques** portent sur la structure déjà construite
  ex: nombre de dépendants de σ_0
- **Features Linéaires** ex: distance de σ_0 et β_0
- **Features Configurationnelles** ex: taille de la pile, transition précédente ...
- Toutes les features peuvent être combinées entre elles pour constituer des features complexes
- 280 features atomiques définies (2^{280} features complexes possibles !)

Un exemple de système de features

1	s0l	
2	s0p	
3	s1p	
4	b0l	
5	b0p	
6	b1l	
7	b1p	
8	b2p	
9	b3p	
10	l_s0r	
11	r_s0r	
12	l_b0r	
13	r_b0r	
14	s0l	b0l
15	s0p	b0p
16	b0p	b0l
17	b0p	l_b0r
18	s1p	b1p

19	b1p	b2p			
20	s0p	b0p	b0l		
21	s0p	l_s0r	r_s0r		
22	s0p	s0l	b0p		
23	s0p	b0p	d_s0_b0		
24	s1p	s0p	b0p		
25	b0p	b1p	b2p		
26	b1p	b2p	b3p		
27	s0p	b0p	b1p		
28	b1p	b1l	b2p	b3p	
29	b1p	b1l	b2p	b2l	b3p
30	t1				
31	t2				
32	t3				
33	t1	t2			
34	t2	t3			
35	t1	t2	t3		

Performances sur le FTB

- LAS : 85.97 UAS : 87.83
- 417 187 features
- classifieur : perceptron moyenné
- 14 000 mots par seconde
- apprentissage 36 sec
- taille du modèle : 78 Mo
- on peut faire mieux, en jouant sur :
  - le système de features
  - le classifieur (SVM)
  - l'exploration de l'espace de recherche (beam search)
- Résultats de la thèse de Assaf Urieli : LAS : 89.35 UAS : 91.55

Analyse en flux

- Pas d'optimisation globale
- L'analyseur n'a pas besoin de connaître la frontière droite de la phrase pour faire l'analyse
- On n'a pas besoin de mettre toute la phrase dans la file (β), il suffit d'y mettre les k prochains mots à analyser
- la valeur de k dépend des features définies (β_k)

Analyse en flux

- Modification de la définition de la configuration d'acceptation
- Détection de la fin de phrase : établissement d'une dépendance (X, root, Y)
- Modification du treebank
  - Modification de la partie de discours de la ponctuation de fin de phrase (pour la distinguer des autres marques de ponctuation)
  - Ajout d'une dépendance `fin`

Transformation du treebank

1	Pourtant	ADV	10	mod	1	Pourtant	ADV	10	mod
2	,	PONCT	10	ponct	2	,	PONCT	10	ponct
3	ce	DET	5	det	3	ce	DET	5	det
4	"	PONCT	5	ponct	4	"	PONCT	5	ponct
5	jet	NC	10	suj	5	jet	NC	10	suj
6	de	P	5	dep	6	de	P	5	dep
7	l’	DET	8	det	7	l’	DET	8	det
8	éponge	NC	6	obj	8	éponge	NC	6	obj
9	"	PONCT	5	ponct	9	"	PONCT	5	ponct
10	est	V	0	root	10	est	V	0	root
11	doublement	ADV	12	mod	11	doublement	ADV	12	mod
12	paradoxal	ADJ	10	ats	12	paradoxal	ADJ	10	ats
13	.	PONCT	10	ponct	13	.	POINT	10	fin

Performances

seg rec : 97.65 seg prec : 94.80

LAS : 85.17 UAS : 87.13

sous segmentation : 28

sur segmentation : 64

Quelques erreurs

- Sur segmentation
  - C'est le cas des fédérations CGT (" // un coup médiatique autour d' un coup de force ") ....
  - M. Michel Charasse , ministre du budget , a ainsi déclaré au micro de RMC , lundi 30 décembre : "// C' est une affaire privée , et je ...
  - concocter une nouvelle loi (les précédentes sont de 1982 , 1986 , 1990 ... //) et la gauche s' abrite derrière l' indépendance ...
- Sous segmentation
  - Continuation ou liquidation (//) Le dépôt de bilan ne signifie pas forcément dans l' immédiat l' arrêt des émissions .
  - ... il faudrait en ajouter au moins deux autres : (//) La politique de rattrapage rapide du SMIC ...
  - " chantage au rabais " (//) " En Belgique et en Allemagne , nous achetons ...

Peut on se passer d'un segmenteur en phrases ?

- La décision de segmenter en phrase est prise très tôt dans la chaîne de traitement par le segmenteur en phrase
- Classifieur : est ce que la marque de ponctuation (principalement le point) marque la fin de la phrase ?
- Mr. Charasse a décidé de claquer la porte.
- Le segmenteur peut-il tirer profit d'informations syntaxiques ?

Transformation du treebank

1	Le	D	2	det
2	reste	N	6	suj
3	des	P+D	2	dep
4	actions	N	3	obj
5	étaient	V	6	aux_pass
6	réparties	V	0	root
7	entre	P	6	mod
8	M.	N	7	obj
9	Pierre	N	8	mod
10	Belfond	N	8	mod
15	et	C	8	coord
16	le	D	18	det
17	second	A	18	mod
18	marché	N	15	dep_coord
19	.	PONCT	6	ponct

1	Le	D	2	det
2	reste	N	6	suj
3	des	P+D	2	dep
4	actions	N	3	obj
5	étaient	V	6	aux_pass
6	réparties	V	0	root
7	entre	P	6	mod
8	M	ABBR	7	obj
9	.	POINT	8	abbrev
10	Pierre	N	8	mod
11	Belfond	N	8	mod
16	et	C	8	coord
17	le	D	19	det
18	second	A	19	mod
19	marché	N	16	dep_coor
20	.	POINT	6	fin

- Performances
  seg rec : 98.07 (97.65) seg prec : 94.73 (94.80)
  LAS : 85.26 (85.17) UAS : 87.24 (87.13)
  sur segmentation 23 (28)
  sous segmentation 65 (64)
- A faire : comparaison avec les performances d'un segmenteur

Segmentation sans ponctuation

- On enlève toute la ponctuation du corpus d'apprentissage
- On essaye de retrouver les frontières de phrases
- Tâche artificielle dans le cas d'un corpus journalistique
- Intéressant pour de l'écrit spontané (tchat) ou de l'oral

Problème de la ponctuation “syntaxique”

- Dans certains cas, les marques de ponctuation ont des dépendants
- Elles ne peuvent tout simplement être éliminées
- Transformation du FTB

4	MM.	NC	21	suj
15	Lacombe	NPP	14	mod
16	,	PONCT	15	coord
17	Koehler	NPP	16	dep_coord
18	et	CC	15	coord
19	Laroze	NPP	18	dep_coord
20	ne	ADV	21	mod
21	sont	V	0	root
22	pas	ADV	21	mod
23	membres	NC	21	ats
24	du	P+D	23	dep
25	PCF	NPP	24	obj
26	.	PONCT	21	ponct

14	MM.	NC	21	suj
15	Lacombe	NPP	14	mod
16	,	PONCT	17	coord
17	Koehler	NPP	15	dep_coord
18	et	CC	19	coord
19	Laroze	NPP	15	dep_coord
20	ne	ADV	21	mod
21	sont	V	0	root
22	pas	ADV	21	mod
23	membres	NC	21	ats
24	du	P+D	23	dep
25	PCF	NPP	24	obj
26	.	PONCT	21	ponct

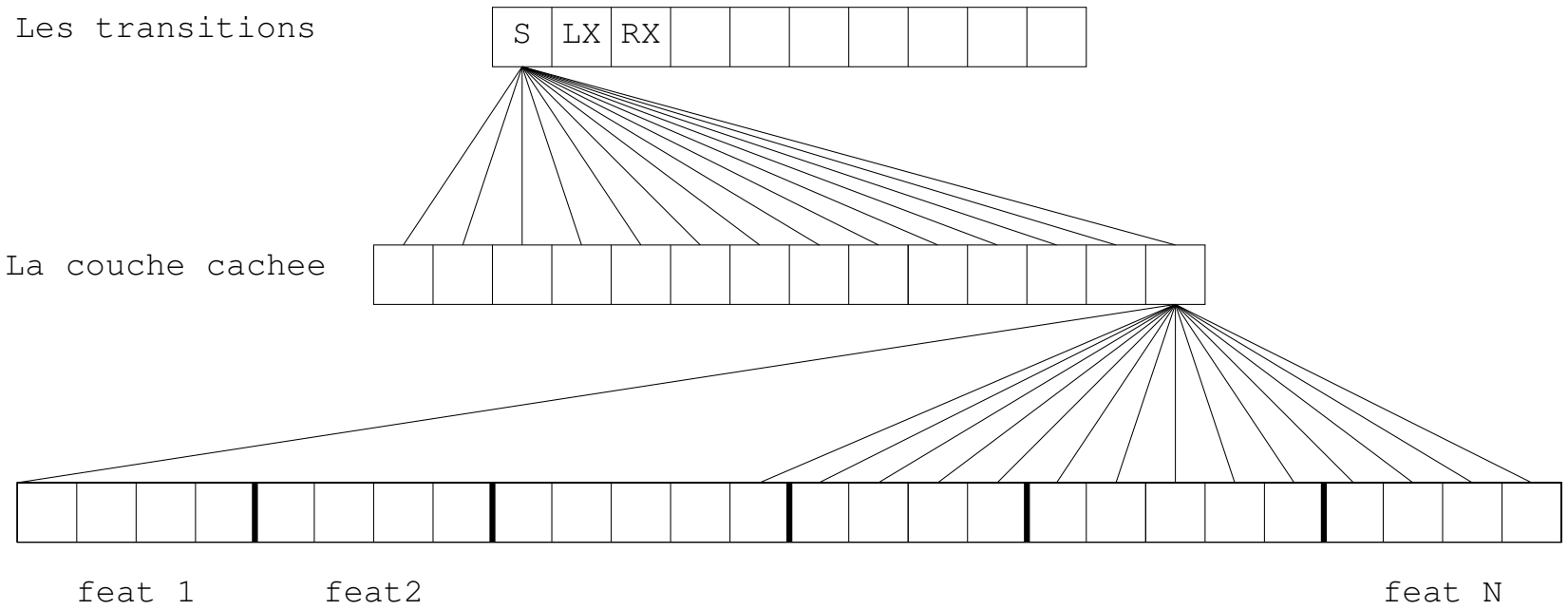
Premiers résultats

- seg rec : 42.40 seg prec : 33.83
- LAS : 80.45 UAS : 83.43
- bonne marge de progression !
- données plus réalistes : datcha et decoda
- prise en compte d'informations suplémentaires (ex: durée des pauses)

Prédiction à l'aide d'un réseau de neurones

- Les performances de l'analyseur reposent en grande partie sur la combinaison de features qui permettent de capter des dépendances importantes dans les données
- Exemple `s0p b0p d_s0_b0`
- L'espace de toutes les combinaisons de features élémentaires est gigantesque (2^N) pour N features
- Prédiction de la transition à effectuer par un réseau de neurone possédant une couche cachée
- Le réseau de neurones explore des combinaisons pondérées de features

Classifieur



- chaque feature élémentaire est représentée sous la forme d'un vecteur de scalaires
- pour les features non lexicales : vecteur binaire (one hot)
- pour les features lexicales : word embedding

Expérience 1 : combinaison de features

- uniquement des features élementaires non lexicales
- one hot encoding des features :
  `s0p, s0r, s1p, s1r, b0p, b0r, b1p, b1r, gs0p, ldep_s0r, rdep_s0r, ldep_b0r, rdep_b0r, ndep_b0, ndep_s0, dist_s0_b0`
- nb d'exemples 278 083
- taille entree 368
- taille sortie 49

Expérience 1 : combinaison de features

- Résultats

	100		150		200	
iter	LAS	UAS	LAS	UAS	LAS	UAS
25	69.74	75.57	67.35	72.78		
50	77.90	82.77	77.31	82.36		
75	79.10	84.02	78.82	83.68		
100	79.63	84.39	79.58	84.29	79.40	84.20
200	79.93	84.75	80.00	84.67	79.90	84.72
400	80.10	84.88	80.11	84.89	**80.24**	**84.95**
600	**80.43**	**85.25**	80.20	85.00	80.11	84.90
800	80.31	84.99	**80.19**	**85.04**		
1000	80.40	85.14	80.18	84.90		

- perceptron
  feat. indep. LAS : 73.05 UAS : 77.52
  combinaisons LAS : 79.21 UAS : 83.58
- le RN arrive à trouver de bonnes combinaisons de features

Expérience 2 : utilisation des word embeddings

- dimension des word embeddings : 50
- entrainés sur le web as a corpus avec word2vec.

	100		150		200		250		300	
iter	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS
100	80.42	82.78	80.38	82.69	80.55	82.94	80.11	82.48	79.85	82.26
200	80.64	83.07	**81.08**	**83.38**	80.91	83.25	80.56	82.94	80.88	83.29
300	80.39	82.89	80.79	83.28	80.61	83.03	80.88	83.27	80.09	82.63

- résultats décevants
- perceptron
  feat. indep. 81.19 83.53
  combinaison 85.97 87.83
- bons résultats obtenus par Chen&Manning EMNLP 2014
- Problème de paramétrage du RN ?

Supertagging

- supertag = partie de discours enrichie
  - valence passive
  - valence active
  - diathèse
  - ...
- nombre de supertags différents : 4725
- performances avec les supertags correct :
  UAS:97.02 LAS:96.00
- *supertagging is almost parsing*
- performances du supertagging : 87.88
- performances du parser avec supertags prédits :
  UAS: 89.83 LAS:87.75
- comment trouver une meilleure répartition de l'effort entre le supertagger et le parser ?

Structure d'un supertag

Un supertag n'est pas un objet atomique.
Il peut être vu comme :

- un arbre élémentaire TAG
- un vecteur à 22 dimensions

stag	cat	dsubcat	ssubcat	voice	comp	datshift	root	lfront
t3488	V	NP0	NP0	act	n	NA	S	AdvP#s#X_NP#s#0
t3489	A	NP0	NP0	NA	l	NA	S	IN#s#X_IN#s#X_NP#s#0
t3958	Ad	nil	nil	NA	n	NA	AdvP	nil

Expérience 1 : décomposition des supertags

- un supertag est décomposé en 22 features *indépendantes*
- ajout dans l'analyseur de nouvelles features élémentaires : `s0A` ... `s0Z`
- les résultats se dégradent : LAS:94.88 UAS:95.84
- on perd des dépendances importantes entre les 22 dimensions
- légère amélioration lorsque les nouvelles features sont ajoutées au supertags :
  supertags correct : UAS:97.46 (97.02) LAS:96.51 (96.00)
  supertags prédits : UAS:89.96 (89.83) LAS:87.86 (87.75)
- Comment découvrir les dépendances importantes ?
- Travail en cours : utiliser un RN

Expérience 2 : Réduction du nombre de supertags

- Toutes les dimensions sont elles porteuses d'information ?
- Sélection d'un sous-ensemble de dimensions (2^{22} sous-ensembles)
- Obtention d'un stagset réduit
- Exploration partielle de l'espace de recherche

LAS	UAS	nb stags	red.	red x LAS
91.97	93.38	80	0.98	90.40
91.54	93.09	255	0.95	86.57
91.46	93.02	236	0.95	86.86
93.40	94.72	401	0.91	85.43
85.66	87.26	32	0.99	85.08
95.58	96.43	4697	0.00	0.00

Expérience 2 : Réduction du nombre de supertags

- Résultats décevants et intéressants
- stag accuracy : 92.45 (87.88 avec 4725 stags)
- résultats de l'analyse avec les stags prédits :
  LAS : 82.13 UAS : 84.40
- tagset performant pour chacune des deux tâches prises séparément, mais très mauvais pour la tâche jointe

Pré-traitements déterministes

- On peut traiter la segmentation en phrase et les mots composés ambigus lors de l'analyse syntaxique
- Pourrait on étendre cela à l'étiquetage morpho-syntaxique et faire en sorte que l'analyseur prenne toutes ces décisions ?

Première expérience

- On crée des catégories ambiguës
- On passe de 31 à 99 parties de discours
- Le tagging est remplacé par un accès au lexique !
- LAS : 80.46 (85.65) UAS : 82.96 (87.61)
- L'ambiguïté n'est jamais levée
- Autres solutions
  - nouvelles features `s0N`, `s0V` le mot en sommet de pile peut-il avoir la catégorie N, V ?
  - embedding de pos tags et combinaison (somme ? moyenne ?)
  - Nouvelles transitions dans l'analyseur :$S_N, S_V \ldots$

Travail en cours et à venir

- Intégration des words embeddings
- Segmentation en phrases
- Intégration du tagging
- Quelle stratégie de contrôle pour l'analyseur ?
- Extension à la construction de la structure discursive/dialogique.