ANR COMPO

Génération conditionnée par la syntaxe

Idée générale

- Introduire dans le processus de génération de texte un analyseur incrémental
- Mesurer l'influence sur le processus de génération
- Mesurer l'influence sur le texte généré
- Est ce que cela offre un avantage en terme de quantité de données d'apprentissage ?

Quelques choix

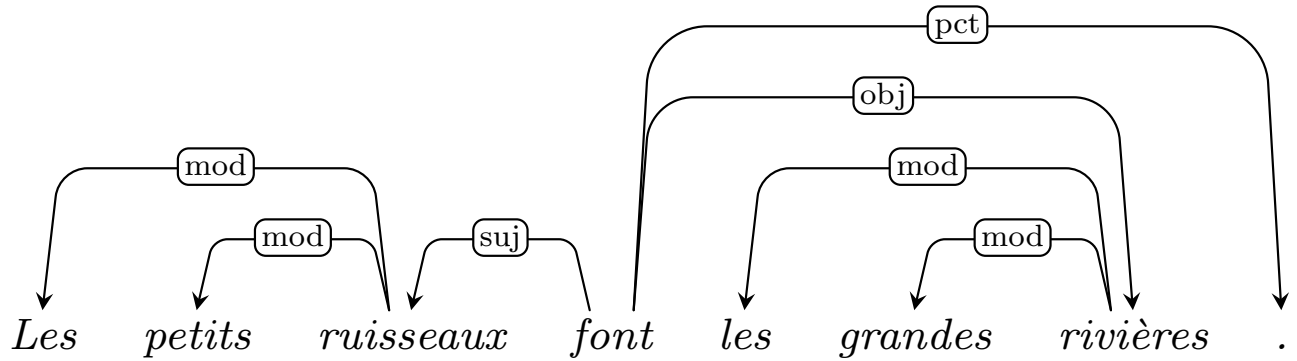
- Modèle de langage : Génération auto-régressive, Transformers
- Analyseur syntaxique : analyse en transition
- Apprentissage sur une quantité limitée de données

Analyse en transitions

- L'analyse est vue comme une séquence d'actions permettant de construire un arbre syntaxique à partir des mots de la phrase.
- Le coeur de l'analyseur est un classifieur prenant en entrée une *Configuration* et prédisant une action
- La configuration est composée d'un buffer (la phrase à analyser) une pile et l'arbre créé jusque là.
- Actions principales :
  - LEFT-X (création d'une dépendance gauche de type X)
  - RIGHT-X (création d'une dépendance droite de type X)
  - SHIFT (transfert d'un mot du buffer dans la pile)
  - REDUCE (élimination d'un mot de la pile)

Exemple

stack	buffer	operation	depset
	les petits ruisseaux font les grandes rivières .	SHIFT	
les	petits ruisseaux font les grandes rivières .	SHIFT	
les petits	ruisseaux font les grandes rivières .	LEFT-*mod*	(ruisseaux, mod, petits)
les	ruisseaux font les grandes rivières .	LEFT-*mod*	(ruisseaux, mod, les)
	ruisseaux font les grandes rivières .	SHIFT	
ruisseaux	font les grandes rivières .	LEFT-*suj*	(font, suj ,ruisseaux)
	font les grandes rivières .	SHIFT	
font	les grandes rivières .	SHIFT	
font les	grandes rivières .	SHIFT	
font les grandes	rivières .	LEFT-*mod*	(rivières, mod, grandes)
font les	rivières .	LEFT-*mod*	(rivières, mod, les)
font	rivières .	RIGHT-*obj*	(font, obj, rivières)
font rivières	.	REDUCE	
font	.	RIGHT-*pct*	(font, pct, .)
font .			



Pourquoi l'analyse en transition ?

- Processus glouton (comme le processus de génération)
- Peut être adapté facilement à une analyse incrémentale (l'analyseur ne voit rien de la partie de la phrase au delà du mot courant)
- Peut facilement être enrichi pour prédire les parties de discours (nouvelle action `POS(X)`).
- Bonne maîtrise du processus (Thèse Franck Dary)

Intégration des deux processus

- Analyse préalable du corpus d'apprentissage ! ! !
- Tout mot est associé à une action dans les données d'apprentissage
- Les séquences générées sont de la forme $m_1 p_1 a_1 \ldots m_n p_n a_n$ où m_i est un mot p_i une partie de dicours et a_i une action
- Deux modes de génération :
  - Les actions (a_i) et les parties de discours (p_i) sont générées par l'analyseur. Les mots (m_i) sont générés par le modèle de langage (les deux processus tournent en parallèle et chacun voit les prédictions de l'autre)
  - Un seul processus (transformer) prédit les deux types d'éléments, le processus d'analyse est implicite (plus de buffer, pile, configuration) tout est simulé (peut être) par le transformer

Les problèmes commencent !

- Les deux processus n'opèrent pas sur les mêmes unités linguistiques : tokens sous lexicaux pour le ML et mots pour l'analyseur
- Quelles solutions ?
  - Deux processus : construction des mots à partir des tokens (trivial) puis analyse
  - Intégrer la segmentation dans la structure syntaxique

Deux processus

- Facile à réaliser, mais on perd la régularité de la séquence générée $m_1 p_1 a_1 \ldots m_n p_n a_n$
- Au lieu d'avoir
  `la POS(DET) SHIFT constitution POS(NOM) LEFT-det`
- on aurait
  `la POS(DET) SHIFT cons -titu -tion POS(NOM) LEFT-det`
- on peut aussi introduire des actions bidon
  `la POS(DET) SHIFT cons POS(SUB) CONCAT titu POS(SUB) CONCAT tion POS(NOM) LEFT-det`

Intégration de la segmentation dans la structure syntaxique

- Décomposer les mots du corpus d'apprentissage de l'analyseur en tokens
- Relier les tokens par une dépendance gauche particulière
- La valence active et passive du mot décomposé sont reportées sur le dernier token
- Au lieu d'avoir
  `la shift constitution LEFT-det`
- on aurait
  `la shift cons shift -titu LEFT-concat shift -tion LEFT-concat LEFT-det`

Evaluation

- Quel effet sur le processus de génération ?
- on compare un ML standard M au modèle M_S appris comme décrit ci-dessus.
- Etant donné un texte $m_1 \dots m_i$ et une séquence de triplets $(m_1, p_1, a_1) \dots (m_i, p_i, a_i)$, on mesure la différence entre la distribution de probabilité du mot m_{i+1} calculée par M et M_s.
- Est ce que la présence du contexte syntaxique influe sur la prédiction lexicale ?

Evaluation 2

- Est-ce que le modèle M_S génère du texte syntaxiquement plus proche des données d'apprentissage ?
- Est-ce que M_S obtient de meilleurs résultats que M dans des contextes syntaxiquement contraints ?

Données

- Dans le scénario décrit ci-dessus l'apprentissage est réalisé à partir de zéro.
- Pas d'apprentissage sur des quantités gigantesques.
- Est-ce que le modèle M_S appris sur une quantité modeste de texte obtient de meilleures performances que le modèle M appris sur la même quantité de données ?
- Cas d'usage : des textes littéraires.
- Quantités de données limitées
- Peut on mieux reproduire le style d'un auteur à l'aide d'un modèle guidé par la syntaxe ?

Apprentissage non supervisé de la syntaxe

- Peut on se passer de la phase d'analyse syntaxique du corpus d'apprentissage et laisser le modèle apprendre ses propres structures syntaxiques ?
- On garde les actions de l'analyseur (`LEFT`, `RIGHT`, `SHIFT`, `REDUCE`)
- On impose des contraintes sur la structure syntaxique d'une phrase (dans le cas le plus simple : former un arbre)
- Apprentissage par renforcement, fonction de reward :
  - 1 si la structure générée est un arbre
  - 0 sinon
- probablement un peu simpliste