Entraînement d'un parseur en dépendances pour le chinois avec Spacy

Scripts disponibles sur la branche origin/LF_chinois_conllu

Liza Fretel
M2 TAL Inalco (IM)
2 rue de Lille 75007 Paris
fretelliza@gmail.com

Abstract

Dans ce rapport, nous évaluons Spacy sur la tâche de d'annotation en dépendances du mandarin. Nous explorons divers paramétrages de Spacy pour comprendre plus en détail ce qui impacte les performances de l'entraînement. Nous allons notamment tester deux aspects de l'entraînement: la puissance du modèle et la quantité de données d'entraînement.

Keywords: Spacy, annotation en dépendances

1. Description de Spacy

Spacy permet d'utiliser des modèles de TAL pour traîter de nombreuses langues, dont le chinois. Spacy propose des modèles pré-entrainés pour le chinois, dont zh_core_web_sm, utilisant des CNN, ainsi que zh_core_web_trf, utilisant des transformers, plus performants mais requérant davantage de ressources computationnelles. De plus, cette bibliothèque permet de mettre en place sa propre pipeline de traitement. Notre intérêt se porte particulièrement sur l'annotation en dépendances. Dans la configuration de notre pipeline, nous avons un modèle (tok2vec2 ou transformer selon le type d'apprentissage voulu), suivi de "tagger" et de "parser". Spacy propose d'entraîner de façon supervisée ses modèles à partir de corpus pré-annotés. Pour la tâche d'annotation en dépendances, nous nécessitons de corpus au format conllu.

L'objectif est de tenter de comparer différents entraînements avec différents paramétrages et jeux de données de Spacy, et si possible de comparer nos résultats avec l'état de l'art et un autre outil, tel que Stanza ou Hanlp.

2. Description des données

Nous utilisons le corpus Lancaster Corpus of Mandarin Chinese (LCMC)[1] pour l'entraînement et la plupart des tests. Le corpus est pré-découpé en train(16081 phrases, 437676 tokens), dev (803 phrases, 20454 tokens) et test (1910 phrases, 50319 tokens). Le format d'annotation est celui du Penn Chinese[2], qui diffère d'Universal Dependency. Nous avons converti le corpus en UD. Dans le corpus d'origine, une des phrases du sous-corpus train était mal annotée, il y avait un cycle dans l'arbre de dépendance. Nous avons retiré cet exemple du jeu de données. Voir la partie 6. Annexe format données pour plus de détails sur les modifications apportées.

3. Expériences

Pour pouvoir réaliser nos expériences sur Spacy, nous avons modifié eval_basique.py afin qu'il ne mesure plus la précision sur les étiquettes POS, mais qu'il mesure les scores OLS, UAS et LAS, en comparant les deprel et head prédits par les modèles à ceux du corpus LCMC.

3.1. Entraînement efficacité

Dans un premier temps, nous entraînons un modèle Spacy sur l'ensemble du jeu de données (train et dev pour la validation) avec la pipeline "tok2vec2", "tagger","parser". Les performances obtenues sur le corpus de test sont présentées Table 1. Nous évaluons nos modèles de parsing à l'aide de trois mesures: OLS[3], UAS[4], et LAS[5], tout en regardant l'impact des tokens out-of-vocab absents du corpus de train. Pour les tokens OOV, nous remarquons des performances presque di-

[1] Lancaster Corpus of Mandarin Chinese

[2] Poiret_Dependencies_Mandarin_Chinese.pdf

[3] Orthogonal Label Unattached Score, tokens attachés avec le bon deprel, peu importe la tête.

[4] Unlabeled Attachment Score, tokens attachés à la bonne tête, peu importe le deprel.

[5] Labeled Attachment Score, tokens attachés à la bonne tête avec le bon deprel.

mesure	IV	OOV
OLS	0.412389	0.212192
UAS	0.290208	0.207195
LAS	0.232218	0.140906

Table 1: Performances sur un apprentissage CNN sur tout le corpus de train avec l'option "efficiency".

mesure	IV	OOV
OLS	0.531986	0.279874
UAS	0.433971	0.290356
LAS	0.364037	0.188679

Table 2: Performances sur un apprentissage CNN sur tout le corpus de train avec l'option "accuracy".

visées par deux pour l'OLS, et assez inférieures pour les scores UAS et LAS.

3.2. Entraînement précision

Ensuite, nous réalisons un entraînement "performance" en CPU, avec la pipeline "tok2vec2", "tagger","parser". Cet entraînement utilise un modèle pré-entraîné de Spacy, zh_core_web_lg. La progression est nettement plus rapide. A noter que si l'on supprime l'étape de "tagger" dans la pipeline, nous obtenons des résultats médiocres. L'annotation en partie du discours est une étape essentielle à l'annotation en dépendances. Les résultats sont présentés Table 2. Nous observons effectivement une progression de tous les scores d'un entraînement "accuracy" par rapport à l'entraînement précédent, "efficiency".

3.3. Entraînement transformer

Cette fois-ci, nous utilisons la pipeline "transformer", "tagger", "parser", toujours en "accuracy". Les modèles "transformer" mettent plus de temps à atteindre de bonnes performances car ils prennent en compte des informations contextuelles plus riches. Ce modèle est le format le plus puissant proposé par Spacy. Les résultats sont présentés Table 3. Nous remarquons un déclin au niveau des scores OLS, UAS et LAS, mais une amélioration sur les mots OOV, sans doûte grâce à l'utilisation d'embeddings. Ce modèle n'est donc pas conseillé dans le cas où on travaille sur des données avec peu de mots OOV, car il prend plus longtemps à apprendre et ses performances sont globalement moins bonnes qu'un entraînement sans embeddings.

mesure	IV	OOV
OLS	0.407639	0.300133
UAS	0.326815	0.286475
LAS	0.306624	0.257495

Table 3: Performances sur un apprentissage transformer sur tout le corpus de train avec l'option "accuracy".

mesure	IV	OOV
OLS	0.391621	0.169054
UAS	0.304140	0.220630
LAS	0.238826	0.136580

Table 4: Performances sur un apprentissage précision sur 100k tokens du corpus de train

4. Impact des données d'apprentissage

Nous allons refaire l'expérience numéro 2, entraînement précision, mais cette fois-ci sur moins de données d'apprentissage afin de mesurer l'impact de la quantité des données d'apprentissage.

4.1. 100k tokens

Avec environ 1/4 des données d'apprentissage utilisées, nous obtenons des scores environ 70 % aussi élevés qu'en utilisant la totalité du jeu de données. Voir Table 4 pour les résultats. Les scores n'ont pas chuté drastiquement par rapport à l'entraînement réalisé en section 3.2, voir Table 2. Bien que seulement un quart des données d'entraînements aient été utilisées, les scores ne chutent que de 30 %. En conclusion, l'apprentissage est réalisable sur un jeu de données plus petit que le corpus LCMC.

4.2. 250k tokens

Nous allons répéter l'expérience avec 250k tokens afin de déterminer si la progression est proportionnelle à la quantité de données utilisées à l'entraînement. Les résultats sont présentés Table 5. Cette fois-ci, les performances atteignent presque 90 % de celle de l'entraînement sur le corpus complet Table 2. En conclusion, la quantité de données a de moins en moins d'impact au fur et à mesure que les données augmentent.

mesure	IV	OOV
OLS	0.485185	0.238289
UAS	0.390568	0.268839
LAS	0.318905	0.155804

Table 5: Performances sur un apprentissage précision sur 250k tokens du corpus de train

mesure	IV	OOV
OLS	0.480157	0.406995
UAS	0.669032	0.683999
LAS	0.397961	0.360879

Table 6: Performances du modèle zh_core_web_trf sur le corpus de test du LCMC (pseudo état de l'art).

5. Comparaisons inter-modèles

Nous avons voulu comparer nos entraînements Spacy à un modèle de Stanza. Les performances de Stanza sont annoncées à 73.41% UAS et 70.65 % LAS[6] pour le chinois simplifié, un score faible par rapport aux modèles des autres langues. Cependant, l'utilisation de Stanza n'a pas fonctionné car le modèle pour le chinois n'est pas disponible sur Hugging Face, ou du moins je n'ai pas réussi à le télécharger. Afin d'entraîner un modèle Stanza soi-même, il faut télécharger un corpus UD; ce que je fais, mais en lançant la commande de téléchargement, une fois terminé, la préparation du corpus pour un entraînement de type "de-prel" plante, disant que le corpus n'a pas été trouvé.

Nous avons aussi voulu tester Hanlp, outil spécifique pour le chinois. Mais en raison de la mise à jour de tensorflow et de keras, nous n'avons pas réussi à exécuter Hanlp. J'ai essayé plusieurs versions de tensorflow, un environnement virtuel, mais rien n'a fonctionné.

N'existant pas beaucoup d'outils, et n'ayant pas réussi à faire fonctionner les deux outils cités précédemment, nous allons utiliser un modèle de Spacy pour le traitement du mandarin, zh_core_web_trf, comme référence. Sur le corpus de test, il obtient des scores très bons, alors qu'il n'a pas été entraîné sur le corpus LCMC. Il obtient les meilleurs scores sur UAS (Unlabeled Attachment Score), avec 67 % sur les IV et plus de 68 % sur les OOV. Ce qui veut dire qu'il arrive à bien reconstituer la structure de l'arbre de dépendances. Etonemment, il est meilleur sur les OOV que sur les IV sur la construction de la structure de l'arbre de dépendance. Son score OLS, atteignant 48 %, dépasse aussi les performances de tous nos entraînements sur le corpus de train. De plus, les OOV n'ont pas beaucoup d'impact sur la performance, excepté pour le score OLS qui passe de 48 % à 40 % sur les OOV.

[6] stanfordnlp.github.io/stanza/performance.html

UD	CTB
ADJ	JJ/VA/OD
ADP	P/BA/LB/LC
ADV	AD
AUX	AS/SB/VV
CCONJ	CC
DET	DT
INTJ	IJ
NOUN	M/NN/NT
NUM	CD
PART	DEC/DEG/DER/DEV/ETC/SP/MSP
PRON	PN
PROPN	NR
PUNCT	PU/VP
SCONJ	CS
SYM	PU
VERB	VC/VE/VV
X	FW/X

Table 7: Conversion des étiquettes POS UD/Chinese TreeBank

6. Annexe format données

Les différences entre les étiquettes du corpus d'origine et les étiquettes UD, présentés Table 7. Il y a une correspondance multiple entre CTB et UD, ce qui ne pose pas de problème de conversion de CTB vers UD.

Afin de supprimer la phrase mal annotée, qui générait une erreur à l'entraînement de Spacy, il est possible de procéder par recherche dichotomique dans le jeu d'entraînement, en relançant un entraînement avec la moitié des données, puis en diminuant ou augmentant la taille en fonction de si l'erreur se trouvait dans ce sous-corpus ou non. Puis, une fois localisée, nous l'avons éliminée du corpus. Elle se trouvait entre les lignes 291495 et 291552. Il est aussi possible de trouver cette erreur en faisant une recherche de regex dans une console vim: `.*36.*\n.*12.*\n.*12.*\n\n`.

7. Conclusion

L'utilisation de Spacy pour un entraînement en parsing n'atteint pas des scores extraordinaires comme ceux annoncés par les outils de TAL tels que Stanza. Mais cet outil reste fiable et adapté à l'entraînement de modèles sur un corpus spécifique. Nous aurions aimé pouvoir tester les modèles de Stanza ou effectuer un entraînement avec Stanza, ou utiliser Hanlp, mais en raison de l'évolution des technologies et des modèles publiés par les auteurs, nous nous sommes confrontées à des obstacles quant à la disponibilité de ces outils. A explorer dans un futur devoir !