Rendu Document Structuré

Compte-rendu du projet final :

Entraînement de modèles de langue pour le Vietnamien

Fanny Bachey
INALCO
Cours de document sctructuré
Travail en collaboration avec Tifanny Nguyen
Scripts sur le git du cours, branche : FB

Abstract

Ce document constitue le rendu pour l'évaluation de la matière, document structuré. L'étude et l'entraînement des modèles ont été faits en collaboration avec Tifanny Nguyen et nous avons choisi de travailler sur le Vietnamien. Dans une première partie, nous aborderons l'état de l'art et les modèles déjà existants. Puis, nous décrirons rapidement le corpus que nous avons utilisé pour enfin expliciter nos entraînements et les résultats que nous avons obtenus.

Keywords: NLP, modèle de langage, vietnamien

1. État de l'art

Dans un premier temps, nous avons cherché les outils déjà existants et qui permettaient un traitement du Vietnamien comme la tokenisation, le POS-tagging, ajout et suppression d'accents... Nous avons concentré notre étude sur deux outils, Pyvi et UndertheSea puisqu'en effet, Spacy ne prend pas en charge le vietnamien et très peu d'outils sont disponibles.

1.1. UnderTheSea

UnderTheSea est une bibliothèque en open source développée pour le traitement de la langue vietnamienne. Il permet de faire différentes tâches de TAL. Voici la liste des outils utilisables :

- Segmentation de mots
- Segmentation de phrases
- Normalisation de texte
- Chunking
- POS tagging
- Analyse de la syntaxe
- Entités nommées
- Classification de texte
- Analyse de sentiment
- Speech to text

Certains modules peuvent être réentrainés, mais il n'y a aucune métrique d'évaluation, difficile donc de faire la comparaison par la suite. Les outils ne sont pas tous entraînés sur les mêmes corpus, certains sont accessibles, d'autres non.

1.2. Pyvi

Pyvi est aussi une bibliothèque qui permet de faire du traitement sur le vietnamien, elle possède moins d'outils.

- Tokenization
- POS tagging
- Accents removal
- Accents adding

En revanche, les métriques d'évaluation sont communiquées, ce qui nous permet d'avoir un premier aperçu sur la qualité de nos résultats. Mais comme nous l'avons constaté au début, il y a très peu d'outils. Même si la pipeline multilingue de spacy : xx_sent_ud_sm permet de tokeniser et d'annoter avec les POS.

2. Les Corpus

Pour nos entraînements, nous avons dû utiliser des corpus différents en fonction des tâches et modifier les données pour qu'elles soient utilisables pour les entraînements de modèles. En gardant quand même des corpus identiques pour les tâches identiques.

2.1. Pour la TOKENISATION

Pour la tokenisation, nous avons utilisé le corpus UTS_WTK, disponible sur le lien suivant : Datasets UTS_VTK., nous n'avons pas trouvé d'informations complémentaires sur l'origine du corpus et nous avons utilisé "base" pour entraîner nos données car "large" faisait planter nos machines. Le corpus train fait 8000 phrases et le test 1000.

2.2. Pour le POS TAGGING

Pour le POS Tagging, nous avons utilisé le corpus VLCP2016. C'est un corpus de 14861 phrases pour train et un dataset de 2831 pour le test. Nous avons été contraints de modifier les données, car on ne pouvait pas réentrainer le module multilingue de Spacy avec les données sous cette forme. Nous avons donc modifié la structure des données pour avoir seulement une syllabe par ligne, en utilisant le format BOI, pour préciser les frontières de mots. Pour améliorer le POS tagging, nous utilisons aussi le corpus VTB pour entraîner les prolongements de mots.

3. ENTRAÎNEMENTS

Nous avons réussi à réentraîner plusieurs modèles, ainsi, nous pouvons établir quelques comparaisons. Pour la tokenisation, nous avons seulement réussi à réentraîner UTS, il semble qu'il ne soit pas possible de le faire pour Spacy. Néanmoins, pour le POS tagging, nous avons réussi l'entraînement des deux modèles, ce qui nous permet de faire une comparaison !

3.1. Tentative infructueuse : Pyvi

Le premier modèle que nous avons tenté de réentraîné est Pyvi, en particulier le tokenizer. Mais un des corpus n'était pas en accès libre et il fallait utiliser une librairie seulement utilisable avec python2 et même en essayant de downgrade les versions dans mon environnement, je n'ai pas réussi à le faire fonctionner. C'est dommage, car c'est le seul modèle pour lequel nous avions des évaluations pour faire des comparaisons.

3.2. TOKENISATION : UTS

Nous avons pu réentraîner le word tokenizer de la bibliothèque UnderTheSea. Pour ce faire, nous avons récupéré le script d'entraînement disponible sur le github. Voici les caractéristiques : Nous avons un corpus train de 8000 phrases et un corpus test de 1000 phrases. Avec 100 itérations, on obtient une accuracy de 96%. Comme l'accuracy n'est pas donnée par UnderTheSea, on peut maladroitement comparer notre résultat avec

	precision	recall	f1-score	support
A	0.957	0.893	0.924	3950
C	0.958	0.959	0.958	2297
CH	1.000	1.000	1.000	9765
E	0.936	0.980	0.958	3816
FW	0.956	0.795	0.868	219
I	0.875	0.718	0.789	39
L	0.989	0.988	0.988	1157
M	0.991	0.972	0.981	2623
N	0.959	0.971	0.965	15381
Nc	0.947	0.955	0.951	2196
Np	0.977	0.991	0.984	4123
Nu	0.979	0.953	0.966	341
Ny	0.938	0.906	0.922	202
P	0.973	0.970	0.971	2593
R	0.952	0.942	0.947	4260
T	0.685	0.717	0.701	258
V	0.958	0.958	0.958	12651
Vy	1.000	1.000	1.000	6
X	0.849	0.681	0.756	207
Z	0.667	0.769	0.714	13
micro avg	0.964	0.964	0.964	66097
macro avg	0.927	0.906	0.915	66097
weighted avg	0.964	0.964	0.964	66097

Accuracy score :
0.9643705463182898

Figure 1: Résultat du POS tagging avec modèle UTS

l'accuracy de pyvi qui est de 0.985. La nôtre est donc légèrement plus basse, mais avec un outil plus puissant, on pourrait essayer de faire plus d'itération pour obtenir de meilleurs résultats en veillant à ne pas surrentraîner le modèle.

Voici la ligne de commande que nous utilisons pour lancer l'entraînement et obrenir l'évaluation : HYDRA_FULL_ERROR=1 python3 train_local_dataset.py ++'dataset.include_test=True'

Il nous est impossible de comparer autrement ce résultat, car nous n'avons pas pu entraîner le tokeniser de spacy, et nous n'avons pas les résultats pour UTS.

3.3. POS TAGGING : UTS et SPACY

3.3.1. UnderTheSea

Pour le POS TAGGING, nous avons d'abord réentraîner le modèle d'UTS. Comme pour le word_tokenizer, nous avons utilisé le script disponible sur le Github. Nous avons un corpus de train de 14861 et un corpus de test de 2831. Nous obtenons une accuracy de 96,5% avec 100 itérations.

Voici la ligne de commande que nous avons utilisée : HYDRA_FULL_ERROR = 1 python3 train_local.py ++'dataset.include_test=True'

Nous sommes satisfaites de ses résultats, le modèle fonctionne assez bien, mais pour avoir une comparaison, nous avons entraîné le modèle multilingue de spacy. Il est capable de classer relativement bien tous les POS.

3.3.2. SPACY

Pour reentraîné le POS-tagging du modèle multilingue de Spacy, nous utilisons le corpus

POS	SPACY	UTS
I	0,09	0,789
FW	0,55	0,868
Z	0,13	0,714
Vy	0,00	1

Table 1: Comparaison de f1-score sur certains POS

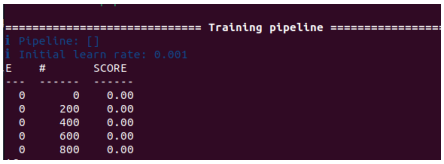

Figure 2: Résultat avec la config W2V

VLSP2016 avec : pour le train : 14861 phrases, pour le dev : 2000 phrases et pour le test : 2831 phrases. Nous avons dû dans un premier temps manipuler les données parce qu'elles n'étaient pas dans le bon format pour être prises en compte par Spacy. Nous avons utilisé un script python qui permet de mettre une syllabe par ligne (separation.py).

Une fois le corpus au bon format, nous avons pu réentrainer le modèle. Une fois qu'il est entraîné, nous utilisons le script d'évaluation que nous avons utilisé en cours pour évaluer notre modèle. Nous obtenons une accuracy de 0, 87 pour notre modèle avec 2OOO steps. Sachant que nous avons utilisé le même corpus que pour UTS, on voit que l'accuracy est moins bonne. On peut comparer les deux modèles et voir quels POS n'ont pas correctement été annotés.

On peut voir sur la Table 1 que certains POS ont très mal été pris en charge par notre modèle spacy ce qui explique notre accuracy plus faible avec ce modèle. Il faudrait trouver pourquoi nous obtenons des résultats plus faibles pour ces POS en particulier. Notamment en regardant au niveau des données par exemple.

article

4. Les vecteurs : Gensim

Nous avons essayé d'ajouter des prolongements de mots avec Gensim. Pour ce faire, nous avons dans un premier temps formaté les données. Nous avons récupéré le corpus vtb.txt qui à l'origine contient des annotations (POS-tagging), nous l'avons nettoyé de ses annotations pour ensuite le tokeniser. Une fois qu'il est tokenisé, nous utilisons gensim pour faire des vecteurs et récupérer le modèle ! Attention, pas en binaire, car Spacy ne prend pas en argument de sa commande init les fichiers encodés en binaire (cf le script gensim.py qui permet de récupérer le fichier à mettre en ligne de commande). Une fois que la commande est lancée, on récupère un dossier dans lequel se trouve un fichier config. J'ai essayé de le lancer, mais j'obtiens seulement des 0 à toutes les étapes. Nous avons essayé de le refaire avec le corpus annoté, mais nous obtenons le même résultat : nous ne savons pas d'où peut provenir l'erreur. Il serait intéressant de prendre plus de temps pour comprendre l'erreur et essayer de rajouter les vecteurs au modèle afin de voir si cela nous permet d'obtenir de meilleurs résultats.

5. Difficultés rencontrées

Plusieurs améliorations pourraient être faites. Tout d'abord, nous avons dû entraîner les modèles avec la machine de Tifanny, car la mienne n'était pas assez puissante et plantait avant d'avoir terminé les entraînements, ce qui forcément ralentit le travail. En outre, nous avons eu du mal à trouver des corpus faciles d'accès ; il est, par exemple pour certains, nécessaire de faire une demande écrite. Même si je vous l'accorde, nous n'avons pas eu besoin d'aller chercher un CD-ROM en Corée. Néanmoins, en entraînant les modèles, nous avons eu énormément de difficultés, entre les scripts manquants, les corpus indisponibles, les outils qui ne fonctionnent qu'avec python2... qui n'est plus pris en charge. Nous avons perdu beaucoup de temps, ce qui nous a empêché de vraiment prendre le temps d'améliorer l'entraînement des modèles avec un meilleur nettoyage des corpus ou la modification des hyper-paramètres. Le présent rendu est très succinct et liste surtout les réussites, mais ce fut un parcours semé d'embûches.

6. Amélioration

Il serait très utile d'avoir accès à d'autres corpus pour pouvoir entraîner avec de "meilleures" données, notamment des données pour améliorer le POS-tagging. Il pourrait être aussi très utile de comprendre le problème avec les vecteurs et de réussir à l'ajouter au modèle pour voir si cela nous permet d'améliorer notre accuracy même si au vu des résultats, il semblerait que le problème vienne de certains POS que notre modèle ne prend pas en compte.