Compte-rendu pour documents structurés

Projet final :

Réentraînement de modules de langue pour le Vietnamien

Tifanny NGUYEN
En collaboration avec Fanny BACHEY
INALCO
M2 TAL parcours IM

Abstract

Cet article présente les résultats obtenus à l'issu du dernier projet demandé par le cours Documents structurés. Ce projet, en collaboration avec Fanny BACHEY, présente l'étude et le réentraînement des outils disponibles en Vietnamien. Dans un premier temps nous ferons l'état de l'art des toolkits disponibles, puis, le réentraînement de ces outils sera présenté afin de déterminer celui qui est le plus performant.

Keywords: NLP, TAL, Toolkit, Outil, Vietnamien

1. Etat de l'art

Malgré une population assez nombreuse et des diasporas installées dans le monde, le Vietnamien est une langue qui reste peu explorée dans le domaine du TAL. Les toolkits disponibles qui permettent de traiter des textes vietnamiens dans différentes tâches sont peu nombreux, surtout lorsque nous en cherchons un récent et à jour. Le premier devoir donné durant ce cours nous a permit de nous concentrer sur deux toolkits : UnderTheSea (UTS)et pyvi.

Les types de traitements disponibles :

- **Tokenization** : le vietnamien étant une langue monosyllabique, certains mots peuvent être composés de deux syllabes qui n'ont pas la même signification lorsqu'ils sont ensembles et séparés. Il est nécessaire de tokenizer les textes afin de ne pas compter un mot unique comme deux différents.
- **POS tagging** : tâche primordiale dans le NLP, le pos tagging permet d'attribuer à chaque mot son tag grammatical.
- pyvi fait également de l'accent adding/removing et UTS permet de faire beaucoup plus de choses avec le chunking, le dependency parsing, du NER[1], de l'analyse de sentiments, etc.

Ces deux modules peuvent être considérés comme "récents", avec pyvi dont la dernière release a été en 2020 et UTS en mars 2023. Un des contributeurs de pyvi est le créateur d'UTS. Ce sont de bons outils qui obtiennent de bons résultats lors de tests.

Pour le projet final, nous avons donc décidé de nous concentrer sur ces deux BaO et de comparer leurs résultats avec le modèle multilingue de spacy.

[1] Name Entity Recognition : permet de placer un tag tel que "Lieu", "Personne", etc.

2. Plan de travail idéal

Voici les étapes que nous comptions suivre pour travailler :

1. Réentraîner UTS ou pyvi et comparer l'accuracy, etc. avec ceux donnés sur le tokenizer et le pos tagging.
2. Réentraîner le modèle multilingue de SPACY sur le même corpus que pyvi ou UTS et vérifier les résultats.
3. Faire un prolongement avec gensim.
4. Réentraîner le modèle SPACY avec les vecteurs issus de gensim.
5. Comparer les résultats.

Ces étapes, bien que claires et directes, se sont révélées très difficiles à suivre.

3. Données

Pour réentraîner des modèles, nous avons besoin de corpus mais la recherche de corpus annotés et assez grands était difficile. Ayant décidé de travailler sur les pos et le tokenizer, il fallait des corpus qui étaient annotés en conséquence.

3.1. VLSP

En observant les deux git, nous pouvions voir que les corpus de 2013, 2016 et 2018 étaient simultanément demandées pour exécuter les scripts d'entraînement mais seulement disponibles après une demande auprès de la VLSP[2] signée par un professeur (si acceptée, bien entendu). Par chance, le corpus de 2016 est disponible sur le git d'UTS et annoté avec des POS et des tags IOB, ce qui nous a permit de faire nos entraînement pour le pos tagging dessus après l'avoir nettoyé pour ne retenir que les mots et les POS.

Lors de l'entraînement d'UTS, que nous détaillerons plus bas, le dataset train était de 14861 et le test de 2831.

3.2. UTS WTK

En explorant les scripts qui ont été entraînés pour le tokenizer par mots, nous avons vu que UTS utilisait un autre corpus : UTS WTK. Il est disponible sur leur hugging face[3] et est annoté exprès pour. Il y a trois choix, mais le large ayant fait planter mon pc et le base étant celui qu'ils ont utilisé, c'est donc celui-ci que nous avons pris pour réentraîner leur modèle.

Lors de l'entraînement de UTS, que nous verrons plus en détail en dessous, nous pouvons voir que le dataset de train est de 8000 et le dataset de test de 1000.

3.3. VTB

Le git de pyvi nous donne un dataset nommé vtb.txt, nous ne savons pas réellement d'où il provient mais il est annoté pour les pos ainsi que pour le tokenizer. Ces données ont été nettoyées.

4. (Ré)Entraînement

Il était question de réentraîner des outils sur les mêmes corpus afin de pouvoir comparer les résultats que les créateurs ont obtenus. Une tâche qui paraît simple mais qui ne l'était pas :

4.1. pyvi

Sur leur page web[4], pyvi vante un score f1 de 98% pour le tokenizer et de 92% pour le pos tagging. Nous avons retrouvé deux scripts d'entraînements sur leur git : train_tokenizer.ipynb et train_tokenizer.py mais il nous a été impossible de les exécuter pour comparer les scores :

- Les corpus demandés n'étaient pas disponibles
- Une ligne dans le .ipynb appelait une fonction qui n'était disponible que pour python2.0 alors que ce dernier ne fonctionne plus.

4.2. UTS

Pour UTS, aucun score n'est indiqué. Cela rend difficile la comparaison, mais nous avons quand même exécuté les scripts d'entraînement pour le pos tagging et le word tokenizer : nous les retrouvons sous le git[5] dans les dossiers /examples/pos_tag ou encore /examples/word_tokenize sous le nom de **train_local.py** et **train_local_dataset.py** respectivement.

Le pos tagging a été entraîné sur le corpus vlsp2016 (au lieu de celui de 2013 que nous n'avons pas) et le word tokenizer sur le corpus UTS WTK. En plus des corpus, nous avons changé le nombre d'itérations et au bout de plusieurs tests, nous avons conclus que le meilleur nombre d'itération est de 100 itérations pour les deux.

La ligne de commande à exécuter pour les deux scripts était celle-ci :

```
HYDRA\_FULL\_ERROR=1 python3 nom\_script.py
++'dataset.include\_test=True'
```

5. Résultats

Traitement	Precision	Recall	F1-Score
POS micro_avg	0.964	0.964	0.964
POS macro_avg	0.927	0.906	0.915
Tokenizer	0.927	0.937	0.932

Table 1: Résumé des scores obtenus

Le script d'UTS nous a permis, avec seqeval, d'avoir plusieurs mesures : la precision, le recall, le f1-score et nous avons ajouté l'accuracy dans le fichier crf_trainer.py qui se trouve dans là où le module est installé.

Pour les pos, l'accuracy était de : **96,5%**.

Pour le word tokenizer, l'accuracy était de : **96%**.

6. Spacy

6.1. xx_sent_ud_sm

Ce modèle multilingue est entraîné pour le vietnamien, parmis plusieurs autres langues. Il ne fait pas de POS tagging, seulement de la tokenisation avec un score d'accuracy à 99% (pour toutes les langues).

[2] https://vlsp.org.vn/About

[3] https://huggingface.co/undertheseanlp

[4] https://pypi.org/project/pyvi/

[5] https://github.com/undertheseanlp/underthesea

6.2. Réentraînement de spacy

Il nous est impossible de réentraîner le tokenizer, tout simplement à cause de la façon dont spacy est fait.

Cependant, nous avons d'entraîner spacy sur les pos : pour cela, nous avons repris le corpus VLSP2016 pour pouvoir comparer les résultats et suivi les deux TPs donnés par le professeur au début du semestre.

Nous avons eu des difficultés à transformer les corpus en fichier .conll car celui-ci demande 3 colonnes : mots, pos, IOB tag mais les mots en vietnamien, étant monosyllabiques, ont parfois besoin de plusieurs syllabes pour en former un seul.

J'ai fait un script **separation.py** qui permet de séparer ces mots(?) sur plusieurs lignes et d'adapter les tags IOB : par exemple, nuoc mat[6] N B-NP est devenu nuoc N B-NP et mat N I-NP. Pour les mots qui ont été taggé comme O, j'ai fait en sorte de les transformer en B-E (pour entity) et I-E afin de les regrouper : par exemple, mai mai R O s'est transformé en mai R B-E et mai R I-E.

Ceci a permit de transformer les corpus en .spacy et donc l'output d'un modèle.

6.3. Evaluation du modèle

Avec plusieurs tests et le temps qui passait malgré l'aide d'un GPU, nous avons conclus qu'avec 2000 max_steps et un batch_size de 128, nous avions de bons résultats : une accuracy de **88%**.

Avec le modèle, nous avons utilisé le script eval_basique.py que nous avons un peu modifié pour l'évaluer. Le score d'accuracy pour les POS est ressorti à : 88% pour le corpus gold, 71% pour le corpus test. Les scores étaient, sans surprise, plus bas que pour le corpus UTS, mais nous n'avons pas le temps de l'améliorer.

6.4. Vecteurs

Si notre compréhension des vecteurs était la bonne, il s'agirait d'un moyen pour améliorer le tokenizer. Pour cela, nous avons pris le corpus vtb.txt de pyvi que nous avons clean pour ne garder que les mots tokenisés et transformer ce fichier nettoyé en format .bin, parce que le init vector de spacy ne prend plus de fichier binaire.

Avec le init vector, nous avons essayé d'entraîner le modèle sur le corpus UTS WTK, mais cela a échoué...pour des raisons que nous ne comprenons pas.

7. Conclusion

Ce projet était au dessus de nos moyens, et non pas de nos compétences, dans le sens où trouver une BaO open-source et récente qui partage corpus et scripts pour refaire les entraînements n'est pas une mince affaire. De plus, j'ai personnellement de la chance d'avoir un GPU et d'avoir pu faire tourner les entraînements parce que Fanny, ou de nombreux autres camarades, n'ont pas pu à cause d'un PC qui plante.

Il est difficile de faire des comparaisons avec le peu de temps que nous avons eu, à cause des autres projets, mais aussi parce que les BaO ne nous donnent pas leurs résultats.

Ceci dit, il était très interessant de modifier et de faire tourner les scripts pour voir les différents résultats que nous pouvions obtenir : nous pouvons conclure que UTS a fait un meilleur travail avec les résultats de pos tagging à 96% comparé aux 88% de spacy sur le corpus gold.

Nous avons mis les scripts et corpus à disposition sur la branche FB_TN_rendufinal du git du cours.

7.1. PS :

PS : Sachez que le manque de temps m'a fait résumer nos aventures, mais nos galères ont été très, très nombreuses.

PPS : J'ai du retard car je ne sais pas exporter ce fichier en pdf.

PPPS : Encore plus de retard car les images ne s'affichaient pas aux bonnes places.

8. Annexes

```
[2024-01-17 18:48:59,000][underthesea.trainers.crf_trainer][INFO] - Finish train
[2024-01-17 18:48:59,127][underthesea.trainers.crf_trainer][INFO] - Start tagger
Classification report:

              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: Résultats obtenus après train_local.py pour 50 itérations.

[6] Les accents ont été omis à cause de l'encodage

```
2024-01-17 18:58:29,156 Finish train
2024-01-17 18:58:29,280 Start tagger
Classification report:

              precision    recall  f1-score   support

           W      0.927     0.937     0.932     18262

   micro avg      0.927     0.937     0.932     18262
   macro avg      0.927     0.937     0.932     18262
weighted avg      0.927     0.937     0.932     18262

Accuracy score :
0.9605941459152468
```

Figure 2: Résultats obtenus après train_local_dataset.py.

	precision	recall	f1-score	support
A	0.74	0.77	0.75	5407
C	0.93	0.84	0.88	2667
CH	1.00	1.00	1.00	9404
E	0.94	0.89	0.91	4091
FW	0.50	0.60	0.55	181
I	0.05	0.50	0.09	4
L	0.94	0.91	0.92	1247
M	0.95	0.97	0.96	2538
N	0.86	0.89	0.87	19721
Nc	0.83	0.82	0.82	2222
Np	0.83	0.87	0.85	4781
Nu	0.92	0.91	0.91	349
Ny	0.84	0.67	0.74	254
P	0.93	0.89	0.90	3166
R	0.86	0.85	0.86	4516
T	0.52	0.42	0.46	326
V	0.87	0.84	0.85	16009
Vy	0.00	0.00	0.00	0
X	0.40	0.51	0.45	406
Z	0.08	0.50	0.13	2
accuracy			0.87	77291
macro avg	0.70	0.73	0.70	77291
weighted avg	0.88	0.87	0.88	77291

Figure 3: Résultats obtenus pour le pos tagging avec spacy.