Commentaires sur les travaux d'Alain et Romain

Kim Antunez

Table des matières

1 Créer un rapport automatique

1.1 Rendu de Rmarkdown avec du code python

Exemple à partir des premières lignes du code d'Alain.

```
import os
import string
import re
import math
from math import sqrt
import numpy as np
import random
import time
random.seed(1)


os.chdir('C:/Users/Kim Antunez/Documents/Projets_autres')
print(string.punctuation + "'")
```

```
## !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~'
```

```
def mise_en_forme_phrase (phrase):
    phrase = phrase.lower()
    # On elève la ponctuation mais ça peut se discuter (garder les @ et #?)
    phrase = re.sub('( @[^ ]*)|(^@[^ ]*)',"nickname", phrase) #Remplace @... par nickname
    #supprime toutes les ponctuations par défaut + les apostrophes bizarres
    phrase = phrase.translate(str.maketrans('', '', string.punctuation + "''"))
    # On enlève les passages à la ligne
    phrase = re.sub('\\n', ' ', phrase)
    # On enlève les espaces multiples et les espaces à la fin des phrases
    phrase = re.sub(' +', ' ', phrase)
    phrase = re.sub(' +$', '', phrase)
    return(phrase.split())
#f = open('data/sample_3.txt')
#raw = f.read()
#print(type(raw))
with open('data/sample_3.txt', encoding="utf-8") as myfile:
    phrases = [mise_en_forme_phrase(next(myfile)) for x in range(10000)]
print(phrases[0:1])
#raw = ''.join(['''.join(phrase) for phrase in phrases])
```

```
## [['il', 'mérite', 'd', 'être', 'bloquer', 'la', 'lettre', 'de', 'l', 'alphabet']]
```

2 Commentaires des codes d'A&R

2.1 Alain

2.1.1 autograd est depreciated

L'option suivante semble depreciated :

```
input = autograd.Variable(input, requires_grad=True)
#UserWarning: torch.autograd.variable(...) is deprecated, use torch.tensor(...) instead
#UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().d
```

J'ai remplacé par ça et le warning a disparu :

```
input = Variable(torch.Tensor(input), requires_grad=True) #KIM
```

Ou sûrement mieux en une étape : cf. ce qu'a fait Romain :

```
W1 = Variable(torch.randn(embedding_dims, voc_size).float(), requires_grad=True)
```

Faire la même chose avec "output"

2.2 Romain

RAS à première vue.

2.3 Croisement des deux codes

2.3.1 W2, faut-il la transposer ?

Pour W2 ou output vous avez la même chose à une transposée près : choisir ce qui est le mieux entre les deux pour cohérence avec explication du modèle

2.3.2 Etape de création des couples contexte / target

Différence d'approche pour la création des couples contexte target. Alors qu'Alain définit les couples dans une fonctions à part, Romain le fait à l'intérieur de l'algorithme de mise à jour de W1 et W2. Intuitivement je dirais que les 2 approches sont équivalentes.

2.3.3 Deux approches différentes pour le produit matriciel

Alain le fait en une étape mais avec une fonction plus etoffée de création des targets contexte (creer_echantillon)

```
data = torch.matmul(input[focus,], torch.t(output))
```

Romain décompose comme cela

```
x = Variable(get_input_layer(focus)).float()
y = Variable(torch.from_numpy(np.array([context])).long())
z1 = torch.matmul(W1, x)
z2 = torch.matmul(W2, z1)
```

Idem je sais pas ce qui est mieux.

2.3.4 Une fonction de loss en une ligne ou deux ?

Déjà évoqué alors que Romain utilise cela

```
log_softmax = F.log_softmax(z2, dim=0)
loss = F.nll_loss(log_softmax.view(1,-1), y)
```

Alain le fait en une étape

```
loss = F.cross_entropy(data.view(1,-1), torch.tensor([context]))
```

3 TODO-LIST pour février

— Kim : comprendre le modèle suite à cours particulier d'Alain et Romain puis replonger simplement dans la décomposition des étapes du modèle (multiplication des matrices, descente de gradients)
— Tous : faire UN modèle unique commun à tous. Pour cela créer des fonctions avec des options du type "negative_sampling = TRUE / FALSE" pour éviter la multiplication des fichiers. Voir intégrer le modèle dans un "package" pour ensuite avoir un fichier ou on le fait tourner sur les données tests et un fichier ou on le fait tourner sur les vrais tweets. De même créer des fonctions pour "évaluer" les sorties du modèle (fonctions ACP, ACP_interactive, TSNE, TSNE_interactive)
— Tous : une fois ces fonctions créées, estimer les étapes qui durent longtemps (avec des fonctionnalités python) et les optimiser si possible...
— Tous : faire tourner modèle + évaluation sur l'ensemble des tweets à notre dispo et voir combien de temps ça met
— Tous : implémenter l'évaluation avec nearest neighbor et human judgment agreement.