xperiment-createartificiallanguage

November 29, 2023

1 Preprocess Data For Artificial Language Experiments

1.1 0. Tokenize and Recombine all used files before using them

[ ]:
```
def tokenize(text):
  import re
  import string

  ret = []
  for token in text.split(' '):
    result_list = re.findall(r'\w+|[^\w\s]', token)
    flag = 0
    for token in result_list:
      if token in ['(', '[', '{', '}', ']', ')', '"', "'"]:
        flag = 1

    if flag:
      ret.append(''.join(result_list))
    elif len(result_list) > 2:
      ret.append(''.join(result_list))
    else:
      ret.extend(result_list)
  return ret

def recombine(list_of_str):
  import string
  ret = ""
  prev = ""
  for token in list_of_str:
    if prev == "-":
      ret = ret[:-1] + token + ' '
    elif token in string.punctuation:
      ret = ret[:-1] + token + ' '
    else:
      ret += token + ' '
    prev = token
  return ret[:-1]
```

```
def normalize_file(f_name):
  import re
  import string

  normalized = ""
  with open(f_name, 'r') as f:
    for line in f.readlines():
      normalized += recombine(tokenize(line)) + '\n'

  with open(f_name, 'w') as f:
    f.write(normalized)
  return normalized
```

[ ]:
```
corpora = [
    'flores_dev_english_sov', 'flores_dev_english_vos',␣
 ↪'flores_dev_english_vso', 'flores_dev_english_svo',
    'flores_english_sov', 'flores_english_vos', 'flores_english_vso',␣
 ↪'flores_english_svo'
]

for corpus in corpora:
  normalize_file(corpus)
```

1.2 A. Noising

Use: 1. A_noising –> function 2. A_noising_map –> pickle

[ ]:
```
def A_noising(corpus, noising_map):
  res = ''
  for sentence in corpus:
    tokens = tokenize(sentence)
    for i in range(len(tokens)):
      tokens[i] = noising_map.get(tokens[i], tokens[i])

    corrupted_sentence = recombine(tokens)
    res += corrupted_sentence +'\n'
  return res

import pickle
noising_map = {}
with open('A_noising_map.pickle', 'rb') as f:
  noising_map = pickle.load(f)
```

[ ]:
```
A_corpora = {}

for corpus in corpora:
  with open(corpus, 'r') as f:
```

```
    A_corpora[f'A_{corpus}'] = A_noising(f.readlines(), noising_map)

for filename in A_corpora.keys():
  with open(filename, 'w') as f:
    f.write(A_corpora[filename])
```

1.3 B. Cognates

Use: 1. B_compound_map –> pickle 2. B_compounding –> function

[ ]:
```
import pickle
compound_map = {}
with open('B_compound_map.pickle', 'rb') as f:
  compound_map = pickle.load(f)

def B_compounding(corpus, compound_map):
  res = ''
  for sentence in corpus:
    tokens = sentence.split(' ')
    if len(tokens) == 1:
      return str(tokens[0])

    bigrams = [(tokens[i], tokens[i+1]) for i in range(len(tokens) - 1)]

    for bigram_idx in range(len(bigrams)):
      if bigrams[bigram_idx] in compound_map:
        bigrams[bigram_idx] = (compound_map[bigrams[bigram_idx]], '')

        if bigram_idx == 0:
          bigrams[1] = ('', bigrams[1][1])
        elif bigram_idx == len(bigrams) - 1:
          bigrams[len(bigrams) - 2] = (bigrams[len(bigrams) - 2][0], '')
        else:
          bigrams[bigram_idx-1] = (bigrams[bigram_idx-1][0], '')
          bigrams[bigram_idx+1] = ('', bigrams[bigram_idx+1][1])

    reconstruct = []
    for i in range(len(bigrams)):
      if i != len(bigrams) - 1:
        if bigrams[i][0] != '':
          reconstruct.append(bigrams[i][0])
      else:
        if bigrams[i][0] != '':
          reconstruct.append(bigrams[i][0])
          reconstruct.append(bigrams[i][1])
        else:
          reconstruct.append(bigrams[i][1])
```

```
    reconstruct = ' '.join(reconstruct)
    res += reconstruct

  return res
```

[ ]:
```
B_corpora = {}

for corpus in corpora:
  with open(corpus, 'r') as f:
    B_corpora[f'B_{corpus}'] = B_compounding(f.readlines(), compound_map)

for filename in B_corpora.keys():
  with open(f'{filename}', 'w') as f:
    f.write(B_corpora[filename])
```

1.4 AB. A + B

Use: 1. B files –> file 2. AB_noising_map –> pickle 3. A_noising –> function

[ ]:
```
import pickle

AB_noising_map = {}
with open('AB_noising_map.pickle', 'rb') as f:
  AB_noising_map = pickle.load(f)
```

[ ]:
```
AB_corpora = {}
for corpus in corpora:
  with open(f'B_{corpus}', 'r') as f:
    AB_corpora[f'AB_{corpus}'] = A_noising(f.readlines(), AB_noising_map)

for filename in AB_corpora.keys():
  with open(filename, 'w') as f:
    f.write(AB_corpora[filename])
```

1.5 C/Dx + A

Use: 1. C1/C2/D1/D2_mapping.pickle –> 4 pickle files 2. translate_corpora –> function 3. —> Results in 8 files for each mapping 4. A_noising –> function 5. C1/C2/D1/D2_noising_map.pickle –> 4 pickle files

1.5.1 Translate corpus

[ ]:
```
corpora = [
    'flores_dev_english_sov', 'flores_dev_english_vos',␣
 ↪'flores_dev_english_vso', 'flores_dev_english_svo',
    'flores_english_sov', 'flores_english_vos', 'flores_english_vso',␣
 ↪'flores_english_svo'
```

```
]

import pickle
with open(f'C1_mapping.pickle', 'rb') as f:
  C1_map = pickle.load(f)

with open(f'C2_mapping.pickle', 'rb') as f:
  C2_map = pickle.load(f)

with open(f'D1_mapping.pickle', 'rb') as f:
  D1_map = pickle.load(f)

with open(f'D2_mapping.pickle', 'rb') as f:
  D2_map = pickle.load(f)

def translate_corpora(code, files, mapping):
  for file_ in files:
    res = ''
    with open(file_, 'r') as f:
      for sentence in f:
        tokens = tokenize(sentence)
        for i in range(len(tokens)):
          tokens[i] = mapping.get(tokens[i], tokens[i])

        corrupted_sentence = recombine(tokens)
        res += corrupted_sentence +'\n'

    with open(f"{code}_{file_}", 'w') as f:
      f.write(res)

translate_corpora('C1', corpora, C1_map)
translate_corpora('C2', corpora, C2_map)
translate_corpora('D1', corpora, D1_map)
translate_corpora('D2', corpora, D2_map)
```

1.5.2 C1

Use: 1. A_noising –> function 2. C1A_noising_map –> pickle

[ ]:
```
import pickle
with open('C1A_noising_map.pickle', 'rb') as f:
  C1A_noising_map = pickle.load(f)

C1A_corpora = {}
for corpus in corpora:
  with open(f'C1_{corpus}', 'r') as f:
    C1A_corpora[f'C1A_{corpus}'] = A_noising(f.readlines(), C1A_noising_map)
```

```
for filename in C1A_corpora.keys():
  with open(filename, 'w') as f:
    f.write(C1A_corpora[filename])
```

1.5.3 C2

1. A_noising –> function
2. C2A_noising_map –> pickle

[ ]:
```
import pickle
with open('C2A_noising_map.pickle', 'rb') as f:
  C2A_noising_map = pickle.load(f)

C2A_corpora = {}
for corpus in corpora:
  with open(f'C2_{corpus}', 'r') as f:
    C2A_corpora[f'C2A_{corpus}'] = A_noising(f.readlines(), C2A_noising_map)

for filename in C2A_corpora.keys():
  with open(filename, 'w') as f:
    f.write(C2A_corpora[filename])
```

1.5.4 D1

1. A_noising –> function
2. D1A_noising_map –> pickle

[ ]:
```
import pickle
with open('D1A_noising_map.pickle', 'rb') as f:
  D1A_noising_map = pickle.load(f)

D1A_corpora = {}
for corpus in corpora:
  with open(f'D1_{corpus}', 'r') as f:
    D1A_corpora[f'D1A_{corpus}'] = A_noising(f.readlines(), D1A_noising_map)

for filename in D1A_corpora.keys():
  with open(filename, 'w') as f:
    f.write(D1A_corpora[filename])
```

1.5.5 D2

1. A_noising –> function
2. D2A_noising_map –> pickle

[ ]:
```
import pickle
with open('D2A_noising_map.pickle', 'rb') as f:
  D2A_noising_map = pickle.load(f)

D2A_corpora = {}
for corpus in corpora:
  with open(f'D2_{corpus}', 'r') as f:
    D2A_corpora[f'D2A_{corpus}'] = A_noising(f.readlines(), D2A_noising_map)

for filename in D2A_corpora.keys():
  with open(filename, 'w') as f:
    f.write(D2A_corpora[filename])
```

1.6 C/Dx + B

Uses: 1. B_compounding –> function 2. C/Dx_compound_map.pickle –> pickle

1.6.1 C1

[ ]:
```
import pickle

C1_compound_map = {}
with open('C1B_compound_map.pickle', 'rb') as f:
  C1_compound_map = pickle.load(f)

for corpus in corpora:
  with open(f'C1_{corpus}', 'r') as f, open(f'C1B_{corpus}', 'w') as g:
    g.write(B_compounding(f.readlines(), C1_compound_map))
```

1.6.2 C2

[ ]:
```
import pickle

C2_compound_map = {}
with open('C2B_compound_map.pickle', 'rb') as f:
  C2_compound_map = pickle.load(f)

for corpus in corpora:
  with open(f'C2_{corpus}', 'r') as f, open(f'C2B_{corpus}', 'w') as g:
    g.write(B_compounding(f.readlines(), C2_compound_map))
```

1.6.3 D1

[ ]:
```
import pickle

D1_compound_map = {}
with open('D1B_compound_map.pickle', 'rb') as f:
```

```
  D1_compound_map = pickle.load(f)

for corpus in corpora:
  with open(f'D1_{corpus}', 'r') as f, open(f'D1B_{corpus}', 'w') as g:
    g.write(B_compounding(f.readlines(), D1_compound_map))
```

1.6.4 D2

[ ]:
```
import pickle

D2_compound_map = {}
with open('D2B_compound_map.pickle', 'rb') as f:
  D2_compound_map = pickle.load(f)

for corpus in corpora:
  with open(f'D2_{corpus}', 'r') as f, open(f'D2B_{corpus}', 'w') as g:
    g.write(B_compounding(f.readlines(), D2_compound_map))
```

1.7 C/Dx + B + A

Uses: 1. C/Dx + B files –> files 2. A_noising –> function 3. C/DxBA_noising_map –> pickle

1.7.1 C1

[ ]:
```
import pickle

C1BA_noising_map = {}
with open('C1BA_noising_map.pickle', 'rb') as f:
  C1BA_noising_map = pickle.load(f)

C1BA_corpora = {}
for corpus in corpora:
  with open(f'C1B_{corpus}', 'r') as f:
    C1BA_corpora[f'C1BA_{corpus}'] = A_noising(f.readlines(), C1BA_noising_map)

for filename in C1BA_corpora.keys():
  with open(filename, 'w') as f:
    f.write(C1BA_corpora[filename])
```

1.7.2 C2

[ ]:
```
import pickle

C2BA_noising_map = {}
with open('C2BA_noising_map.pickle', 'rb') as f:
  C2BA_noising_map = pickle.load(f)
```

```
C2BA_corpora = {}
for corpus in corpora:
  with open(f'C2B_{corpus}', 'r') as f:
    C2BA_corpora[f'C2BA_{corpus}'] = A_noising(f.readlines(), C2BA_noising_map)

for filename in C2BA_corpora.keys():
  with open(filename, 'w') as f:
    f.write(C2BA_corpora[filename])
```

1.7.3 D1

[ ]:
```
import pickle

D1BA_noising_map = {}
with open('D1BA_noising_map.pickle', 'rb') as f:
  D1BA_noising_map = pickle.load(f)

D1BA_corpora = {}
for corpus in corpora:
  with open(f'D1B_{corpus}', 'r') as f:
    D1BA_corpora[f'D1BA_{corpus}'] = A_noising(f.readlines(), D1BA_noising_map)

for filename in D1BA_corpora.keys():
  with open(filename, 'w') as f:
    f.write(D1BA_corpora[filename])
```

1.7.4 D2

[ ]:
```
import pickle

D2BA_noising_map = {}
with open('D2BA_noising_map.pickle', 'rb') as f:
  D2BA_noising_map = pickle.load(f)

D2BA_corpora = {}
for corpus in corpora:
  with open(f'D2B_{corpus}', 'r') as f:
    D2BA_corpora[f'D2BA_{corpus}'] = A_noising(f.readlines(), D2BA_noising_map)

for filename in D2BA_corpora.keys():
  with open(filename, 'w') as f:
    f.write(D2BA_corpora[filename])
```