Source Code for the Generation of Review Topics' Weight

Fangzhou Xie

May 13, 2020

Here we attach the codes for preprocessing our review scores to topic-weights.

```
import json
import os
import pdb
import pickle
import random
import sqlite3 as lite
from itertools import product
from operator import itemgetter

import numpy as np
import pandas as pd
import pyLDAvis
import pyLDAvis.gensim
import spacy
from gensim.corpora import Dictionary
from gensim.models import LdaModel, LdaMulticore
from gensim.models.coherencemodel import CoherenceModel
from IPython.display import Image

spacy.prefer_gpu()
random.seed(1)

data_file = 'data/reviews.sqlite'
ckpt_path = './ckpt/'
if not os.path.exists(ckpt_path):
    os.makedirs(ckpt_path)


class LDAReview():
    """wrap reviews and process by LDA"""

    def __init__(self, id_reviews, test_ratio):
        # self.data = id_reviews
        self.sentences, self.docid2oldid, self.docid2review, self.senid2docid
            preprocess(id_reviews)
        print('Random␣select␣documents:␣{}'.format(
            random.choice(self.sentences)))
        self.dictionary = dictionary = Dictionary(self.sentences)
        # pdb.set_trace()
        dictionary.filter_extremes(no_below=3, no_above=0.5, keep_n=100000)
        self.bow_corpus = bow_corpus = [
            dictionary.doc2bow(doc) for doc in self.sentences]
        doc_len = len(self.bow_corpus)  # should equal to length of sentences
        ids = list(range(doc_len))
        random.shuffle(ids)
        self.shuffled2senid = {v: k for k, v in enumerate(ids)}
        self.shuffledbow = [bow_corpus[i] for i in ids]

        test_ids = ids[:round(doc_len * test_ratio)]
        train_ids = ids[round(doc_len * test_ratio):]

        self.train_bows = [bow_corpus[i] for i in train_ids]
        self.test_bows = [bow_corpus[i] for i in test_ids]

    def cv(self):
        'cross␣validate␣the␣hyper-parameters.␣log_perplexity␣smaller␣the␣bett
        # best_ppl = 1e6
        # best_para = {'topic': 30,
        #              'ps': 2}
        params = [range(2, 20, 2), range(2, 5, 2)]
        df = pd.DataFrame(list(product(*params)),
                          columns=['topics', 'passes'])
        df['ppl'] = np.nan
        df['coh'] = np.nan
        # pdb.set_trace()
        for topic, ps in product(*params):
            model = run(self.train_bows, self.dictionary,
                        num_topics=topic, passes=ps)
            # alpha='auto', eta='auto')
            log_ppl = model.log_perplexity(self.test_bows)
            cm = CoherenceModel(model=model,
                                corpus=self.bow_corpus,
                                coherence='u_mass')
                coherence = cm.get_coherence()  # get coherence value
                df.loc[(df['topics'] == topic) & (
                    df['passes'] == ps), 'ppl'] = log_ppl
                df.loc[(df['topics'] == topic) & (
                    df['passes'] == ps), 'coh'] = coherence
                # pdb.set_trace()

            df.to_csv('parameters.csv', index=False, header=True)

    def run(self, loss='coh'):
        if os.path.exists('parameters.csv'):
            df = pd.read_csv('parameters.csv', header=0)
            # pdb.set_trace()
            if loss == 'ppl':
                topic, ps = df.loc[df['ppl'].idxmin()]['topics':'passes']
            elif loss == 'coh':
                topic, ps = df.loc[df['coh'].idxmin()]['topics':'passes']
            model = run(self.bow_corpus, self.dictionary,
                        num_topics=int(topic), passes=int(ps))
            with open('bow_dict.pk', 'wb') as f:
                pickle.dump([self.shuffledbow, self.dictionary], f)
            model.save('lda.model')
            total_topics = {k: v for k,
                            v in model.print_topics(-1, num_words=20)}
            topic_weight = {self.senid2docid[self.shuffled2senid[i]]: model[j
                            for i, j in enumerate(self.bow_corpus)}
            # save the topics, weights for docs, and document id to the orgin
            with open('js_topics.pk', 'wb') as f:
                pickle.dump(total_topics, f)
            with open('js_weight.pk', 'wb') as f:
                pickle.dump(topic_weight, f)
            with open('js_docid2oldid.pk', 'wb') as f:
                pickle.dump(self.docid2oldid, f)
        else:
            self.cv()


def run(bow_corpus, dictionary,
        num_topics=10, passes=2, n_workers=20, **kwargs):
    # best_ppl = 1e8
    # error in multicore version
    model = LdaMulticore(bow_corpus,
                         id2word=dictionary,
                         num_topics=num_topics,
                         passes=passes,
                         workers=n_workers,
                         random_state=0,
                         minimum_probability=0,
                         **kwargs)
    # for idx, topic in model.print_topics(-1):
    #     print('Topic: {} \nWords: {}'.format(idx, topic))
    return model


def preprocess(id_reviews):
    nlp = spacy.load('en_core_web_sm', disable=["tagger"])
    merge_ents = nlp.create_pipe("merge_entities")
    nlp.add_pipe(merge_ents)
    ids, reviews = zip(*id_reviews)
    reviewnew = []
    [reviewnew.append('␣'.join(i.split())) for i in reviews]

    docid2oldid, docid2review, senid2docid = {}, {}, {}
    sentences = []
    parsed_docs = [[[t.lemma_.lower() for t in sen if not
                        (t.is_stop or t.is_punct or len(t.lemma_.strip()) == 0)]
                       for sen in doc.sents]
                      for doc in nlp.pipe(reviews, disable=['tagger'])]
    sen_id = 0
    for i, sens in enumerate(parsed_docs):
        docid2oldid[i] = ids[i]
        docid2review[i] = reviews[i]
        sentences += sens
        for sen in sens:
            senid2docid[sen_id] = i
            sen_id += 1
    return sentences, docid2oldid, docid2review, senid2docid


def loaddata():
    conn = lite.connect(data_file)
    c = conn.cursor()
    result = c.execute("select id, comments from reviews;").fetchall()
    return result


def cvandsave():
    id_reviews = loaddata()
    lda = LDAReview(id_reviews, 0.3)
    lda.run()
    # pdb.set_trace()
    pass


def interpret():
    with open('js_topics.pk', 'rb') as f:
        total_topics = pickle.load(f)
    with open('js_weight.pk', 'rb') as f:
        topic_weight = pickle.load(f)
    with open('js_docid2oldid.pk', 'rb') as f:
        docid2oldid = pickle.load(f)
    # pdb.set_trace()
    # print('end')
    # write topics.csv
    with open('topics.csv', 'w') as f:
        f.write('num,' + ','.join(['word{}'.format(i)
                                   for i in range(10)]) + '\n')
    for k, v in total_topics.items():
        tpweight = [i.strip() for i in v.split('+')]
        tpweight.insert(0, str(k))
        with open('topics.csv', 'a') as f:
            f.write(','.join(tpweight) + '\n')

    # write doc weights
    dt = np.dtype('int,float')
    combined_dict = {}
    for k, v in topic_weight.items():
        oldid = docid2oldid[k]
        valuepair = np.array(v, dtype=dt)
        valuepair.dtype.names = ['ind', 'weight']
        if oldid in combined_dict:
            combined_dict[oldid] += valuepair['weight']
        else:
            combined_dict[oldid] = valuepair['weight']
    pd.DataFrame.from_dict(combined_dict, orient='index').to_csv('weights.csv
                                                                    index=True)


def ldaplot():
    model = LdaMulticore.load('lda.model')
    with open('bow_dict.pk', 'rb') as f:
        bow, dict = pickle.load(f)
    # pdb.set_trace()
    vis = pyLDAvis.gensim.prepare(model, bow, dict)
    # pyLDAvis.display(vis)
    # pdb.set_trace()
    pyLDAvis.save_html(vis, 'lda.html')


def main():
    cvandsave()
    interpret()
    ldaplot()


if __name__ == "__main__":
    main()
```