```
"""TF-IDF"""

import numpy as np
from scipy.stats import rankdata
from colgibbs import col_gibbs


def tfidf(data, num_words):
    """Calculates term frequency-inverse document frequency and returns
       the indices of the 'num_words' words with the highest score."""
    D, W = np.max(data[:, 0]), np.max(data[:, 1])

    tf = np.zeros((D, W))
    for d, w, train_count, _ in data:
        tf[d-1][w-1] = train_count
    tf = 0.5 + 0.5 * (tf / tf.max(axis=1)[:, np.newaxis])

    return (-tf.max(axis=0)).argsort()[:num_words]


def pickrows(data, word_indices):
    """Given word_indices, reurns all the rows of matrix 'data' whose
       second element is in 'word_indices'."""
    return data[np.isin(data[:, 1], word_indices+1)]


if __name__ == '__main__':
    print("Loading in data...")
    orig_data = np.loadtxt('data/nips.data', dtype=np.int)

    with open("data/nips.vocab", "r") as f:
        vocab = np.array([w.strip() for w in f.readlines()])

    word_indices = tfidf(orig_data, 500)
    pruned_data = pickrows(orig_data, word_indices)
    chosen_words = vocab.take(word_indices)

    rank_to_word = {rank: word for rank, word in zip(rankdata(word_indices, method='dense'),
chosen_words)}

    # first 250 chosen words:
    #    ['feature', 'decision', 'schemas', 'wahba', 'application', 'kernel',
    #    'icx', 'stream', 'michael', 'sources', 'approaches', 'source',
    #    'optimal', 'field', 'momentum', 'sound', 'ica', 'approximation',
    #    'display', 'fig', 'synapse', 'margin', 'modules', 'game', 'release',
    #    'polynomial', 'leg', 'legal', 'pomdp', 'chip', 'grammar', 'space',
    #    'synaptic', 'owl', 'chess', 'synapses', 'decoding', 'dynamic',
    #    'probabilities', 'disparity', 'stress', 'vowel', 'road', 'string',
    #    'monotonicity', 'monte', 'mapping', 'chain', 'song', 'prior',
    #    'neuroscience', 'firing', 'fish', 'natural', 'map', 'fusion',
    #    'distribution', 'option', 'representation', 'pan', 'signal',
    #    'linsker', 'hyper', 'endmember', 'consonant', 'filter', 'character',
```

```
    #    'arbor', 'arc', 'ekf', 'graph', 'vor', 'problem', 'dyna', 'chaos',
    #    'channel', 'distance', 'optimization', 'magnetic', 'ppserver',
    #    'gain', 'cortical', 'scene', 'vmos', 'iii', 'cholinergic', 'noise',
    #    'implementation', 'node', 'stimuli', 'weighting', 'rod', 'weight',
    #    'member', 'circuit', 'stimulus', 'family', 'entropy', 'roc',
    #    'analog', 'synergy', 'fault', 'stochastic', 'onset', 'biases',
    #    'speaker', 'markov', 'recurrent', 'clause', 'model', 'delay',
    #    'impulse', 'classifier', 'classification', 'class', 'factor',
    #    'phonetic', 'planning', 'system', 'rules', 'search',
    #    'predisposition', 'tin', 'epoch', 'city', 'skill', 'bias', 'ltm',
    #    'sparse', 'parse', 'processes', 'stopping', 'fraud', 'features',
    #    'connection', 'ann', 'anna', 'storage', 'perturbation', 'lld',
    #    'annealing', 'operator', 'robot', 'module', 'ltp', 'wave',
    #    'regularization', 'image', 'unlearning', 'part', 'region', 'malcom',
    #    'analysis', 'gamma', 'science', 'modular', 'regression', 'iiii',
    #    'spatial', 'parsec', 'waves', 'gradient', 'wavelet', 'stock',
    #    'type', 'processing', 'images', 'policy', 'faces', 'artifact',
    #    'navigation', 'retina', 'method', 'batch', 'contour', 'similarity',
    #    'motif', 'motion', 'head', 'bat', 'cross', 'subject', 'basis',
    #    'label', 'convex', 'mse', 'converter', 'sweeping', 'ring',
    #    'harmony', 'average', 'trial', 'sat', 'manifold', 'tree', 'carlo',
    #    'trees', 'swimming', 'light', 'forgeries', 'mean', 'critic',
    #    'variance', 'car', 'variables', 'form', 'dot', 'convolution',
    #    'hebbian', 'capacity', 'lesion', 'harmonic', 'contrast',
    #    'controller', 'surround', 'csrf', 'potential', 'drf', 'level',
    #    'oscillator', 'drift', 'manager', 'simulation', 'subscriber',
    #    'user', 'mrf', 'convergence', 'letter', 'preposition', 'ridge',
    #    'ham', 'movement', 'calibration', 'call', 'harmonet', 'motor',
    #    'penalty', 'current', 'svr', 'neighbor', 'sample', 'forward',
    #    'softassign', 'posterior', 'handwriting', 'bag', 'saliency', 'hand',
    #    'cue', 'utility']

    pruned_data[:, 0] = rankdata(pruned_data[:, 0], method='dense')
    pruned_data[:, 1] = rankdata(pruned_data[:, 1], method='dense')

    print("Starting sampling...")
    logjoints, logpreds, _, _, _, _ = col_gibbs(pruned_data, 50, idx_to_word=rank_to_word,
nb_iters=10000)
```