Natural Language Processing

December 22, 2017

1 Natural Language Processing

```
In [2]: from __future__ import print_function, division
        from builtins import range
        import nltk
        import numpy as np
        import matplotlib.pyplot as plt
        from nltk.stem import WordNetLemmatizer
        from sklearn.decomposition import TruncatedSVD
```

```
In [3]: wordnet_lemmatizer = WordNetLemmatizer()

        titles = [line.rstrip() for line in open('all_book_titles.txt')]

        # copy tokenizer from sentiment example
        stopwords = set(w.rstrip() for w in open('stopwords.txt'))
        # add more stopwords specific to this problem
        stopwords = stopwords.union({
            'introduction', 'edition', 'series', 'application',
            'approach', 'card', 'access', 'package', 'plus', 'etext',
            'brief', 'vol', 'fundamental', 'guide', 'essential', 'printed',
            'third', 'second', 'fourth', })
```

```
In [4]: def my_tokenizer(s):
            s = s.lower() # downcase
            tokens = nltk.tokenize.word_tokenize(s) # split string into words (tokens)
            tokens = [t for t in tokens if len(t) > 2] # remove short words, they're probably no
            tokens = [wordnet_lemmatizer.lemmatize(t) for t in tokens] # put words into base for
            tokens = [t for t in tokens if t not in stopwords] # remove stopwords
            tokens = [t for t in tokens if not any(c.isdigit() for c in t)] # remove any digits,
            return tokens
```

```
In [5]: # create a word-to-index map so that we can create our word-frequency vectors later
        # let's also save the tokenized versions so we don't have to tokenize again later
        word_index_map = {}
        current_index = 0
        all_tokens = []
        all_titles = []
        index_word_map = []
        error_count = 0
        for title in titles:
            try:
                title = title.encode('ascii', 'ignore').decode('utf-8') # this will throw except
                all_titles.append(title)
                tokens = my_tokenizer(title)
                all_tokens.append(tokens)
                for token in tokens:
                    if token not in word_index_map:
                        word_index_map[token] = current_index
                        current_index += 1
                        index_word_map.append(token)
            except Exception as e:
                print(e)
                print(title)
                error_count += 1
```

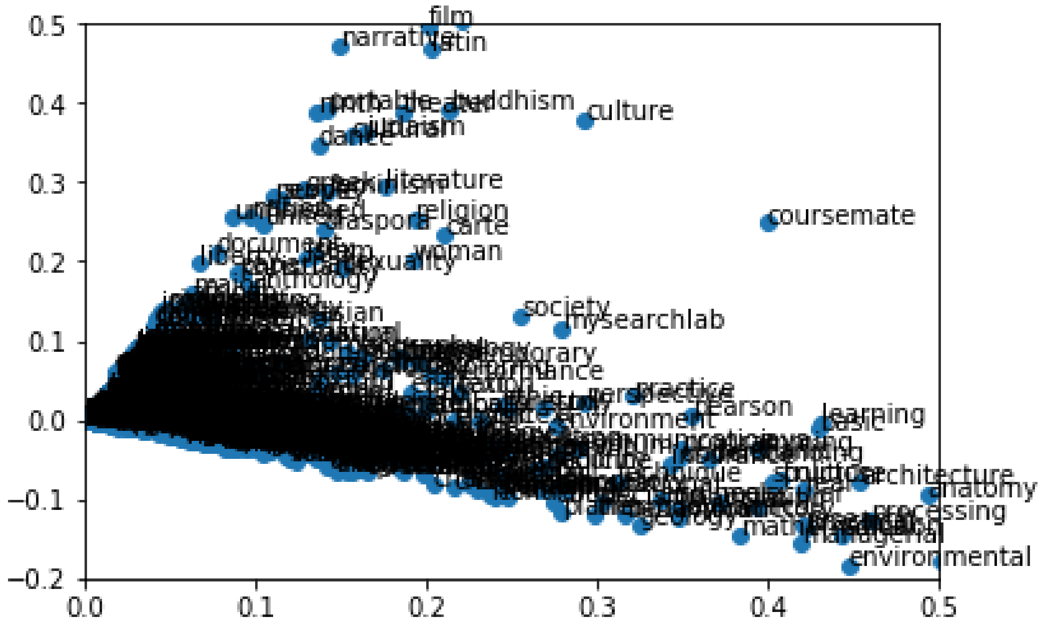
```
In [6]: print("Number of errors parsing file:", error_count, "number of lines in file:", len(tit
        if error_count == len(titles):
            print("There is no data to do anything with! Quitting...")
            exit()
```

```
Number of errors parsing file: 0 number of lines in file: 2373
```

```
In [7]: # now let's create our input matrices - just indicator variables for this example - work
        def tokens_to_vector(tokens):
            x = np.zeros(len(word_index_map))
            for t in tokens:
                i = word_index_map[t]
                x[i] = 1
            return x
```

```
In [8]: N = len(all_tokens)
        D = len(word_index_map)
        X = np.zeros((D, N)) # terms will go along rows, documents along columns
        i = 0
        for tokens in all_tokens:
            X[:,i] = tokens_to_vector(tokens)
            i += 1
```

```
In [9]: svd = TruncatedSVD()
        Z = svd.fit_transform(X)
        plt.scatter(Z[:,0], Z[:,1])
        for i in range(D):
            plt.annotate(s=index_word_map[i], xy=(Z[i,0], Z[i,1]))
        plt.xlim(0,.5)
plt.ylim(-.2,.5)
plt.show()
```


film
narrative
latin
buddhism
culture
dance
literature
coursemate
religion
diaspora
carte
woman
sexuality
anthology
society
mysearchlab
contemporary
performance
practice
perspective
environment
pearson
learning
basic
architecture
anatomy
processing
managerial
environmental