Learn About Latent Dirichlet Allocation in Python With Data From the News Articles Dataset (2016)

Learn About Latent Dirichlet Allocation in Python With Data From the News Articles Dataset (2016)

How-to Guide for Python

Introduction

In this guide, you will learn how to fit a Latent Dirichlet Allocation (LDA) model to a corpus of documents using the programming software Python with a practical example to illustrate the process. You are provided with links to the example dataset, and you are encouraged to replicate this example. An additional practice example is suggested at the end of this guide. This example assumes that you have the data file stored in the working directory being used by Python.

Contents

1 Latent Dirichlet Allocation

LDA is a generative model for the words appearing in the documents from a given corpus. It is a computational approach to the identification of the topic structure

underlying the documents. It assumes that each document contains a mixture of topics, and each topic is a distribution of words. Accordingly, each word appearing in the document is a draw from the topics comprising the document.

---

2 An Example in Python: Topics in the News Articles

This example demonstrates how to identify topics computationally from a large corpus of news articles. The found topics can help researchers understand what has been discussed in those articles without reading every one of them; they can be used as summary statistics in further quantitative analysis, and they can also be employed to better organize the documents for retrieval and recommendation applications.

This example uses a subset of data from the 2016 News Articles dataset (https://www.kaggle.com/asad1m9a9h6mood/news-articles/home). The news articles are collected from www.thenews.com.pk between January 2015, and March 2017, and they are related to business and sports. There are 2,692 rows in the dataset with each row corresponding to a news article. The dataset contains three columns corresponding to the heading, the content, and the publication date for each article, respectively. The column we examine is

- Article: the text content of each article.

2.1 The Python Procedure

Python is an open-source programming language. Python does not operate with pull-down menus. Rather, you must submit lines of code that execute functions and operations built into Python. It is best to save your code in a simple text file that Python users generally refer to as a script file. We provide a script file with this example that executes all of the operations described here. If you are not familiar with Python, we suggest you start with the introduction manual located at https://wiki.python.org/moin/BeginnersGuide. While most computer systems come

with a vanilla Python, we recommend installing the distribution made by Anaconda (https://www.anaconda.com/download/) as it contains many packages that are commonly used. This software guide uses this distribution and will prompt to install any package used here but not included in the Anaconda distribution.

For this example, we need the package “sklearn” for constructing the document-term matrix and fitting the LDA model. For the installation of this package, please see their official website (https://scikit-learn.org/stable/install.html). The Anaconda distribution of Python should have this package installed already. The particular functions needed from this package are “CountVectorizer” and “LatentDirichletAllocation”; they can be loaded as:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
```

We also need the “nltk” package for preprocessing the articles (see SAGE Research Methods Dataset on Pro-Processing for details.) For the installation of this package, please visit their official website (https://www.nltk.org/install.html). The Anaconda distribution of Python should have this package installed already. After the package is installed, we still need to download the dictionary of stop words for this package:

```
nltk.download('stopwords')
```

This dictionary only needs to be downloaded once. After it is downloaded, it can be loaded as the following every time you need it:

```
stopwords = set(nltk.corpus.stopwords.words('english'))
```

We also need the package “pandas” and “numpy” for general purposes. The installation instructions for those packages can be found at their websites (https://pandas.pydata.org/pandas-docs/stable/install.html and

https://www.scipy.org/install.html). If you are using the Anaconda distribution of Python, then the packages should be installed already. With the packages installed, we can load them as:

```
import pandas as pd
import numpy as np
```

To begin with the analysis, we must first load the data into Python. This can be done with the following code (assuming the data file is already saved in your working directory):

```
dataset=pd.read_csv('dataset-news-2016-subset1.csv')
```

The dataframe loaded above is a table where each row corresponds to a news article, and the column “Article” is the content of the news.

Before fitting the LDA model to the news articles, we perform common pre-processing procedures on the documents. See SAGE Research Methods Dataset on Pre-Processing for details. First, we convert all letters to lower case:

```
dataset['Article ']=dataset['Article'].str.lower()
```

Then, we replace punctuation marks with whitespaces:

```
dataset['Article']=dataset['Article'].str.replace('[^\w\s]',' ')
```

After that, we remove stop words (this process will also remove excessive white spaces between words):

```
dataset['Article']=dataset['Article'].apply(lambda x: “ “.join(i for i in x.split()
if i not in stopwords))
```

Then, we construct a document-term matrix with term frequencies as entries from

the processed documents (see SAGE Research Methods Dataset on Basics in Text Analysis for details):

```
vectorizer = CountVectorizer()
dtm = vectorizer.fit_transform(dataset.Article)
```

To obtain the list of words identified by the functions above, we run the following code which also prints out the number of words identified:

```
words=np.array(vectorizer.get_feature_names())
print(len(words))
```

Finally, we estimate the LDA topic model on the corpus of news articles, and we pick the number of topics to be 10:

```
lda = LatentDirichletAllocation(n_components=10, random_state=0)
lda.fit(dtm)
```

The first line of code above constructs an LDA model using the function "LatentDirichletAllocation." The first input to the function is the number of topics which is set to "n_components=10." The second input sets the seed for random number generators so that the function will yield the same results every time it is executed. Then, we fit the model to the data by calling the "fit" function of the model with the document-term matrix as input.

With the estimated model, we check the top 10 most likely words in each topic as the following:

```
words_in_topic = lda.components_
for i in range(words_in_topic.shape[0]):
print(words[np.argsort(words_in_topic[i])[-10:]])
```

The “components_” attribute of the estimated model is a topic by word matrix with each entry denoting the weight of the corresponding word in the corresponding topic. We then go through this topic-by-word matrix row by row and print out the 10 most likely words for each row. Note that the words are printed out in increasing order of their weights.

2.2 Exploring the Python Output

For each command, Python will return its output immediately. Here, we focus on the main results (Table 1).

Table 1: The Top 10 Words in Each Estimated Topic.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
com	goal	vehicle	england	said
https	manchester	driving	first	pakistan
front	minutes	maasdorp	test	strong
www	half	ndb	match	year
pk	header	company	two	would
thenews	leicester	vehicles	captain	also
src	belgium	tax	three	new
href	spain	uber	one	world
assets	chelsea	aiib	runs	team
css	minute	bank	second	country
Topic 6	**Topic 7**	**Topic 8**	**Topic 9**	**Topic 10**
open	france	million	croatia	percent

champion	cup	billion	mirza	said
title	final	iran	ioc	oil
world	messi	saudi	torch	year
grand	time	said	season	prices
final	club	oil	sania	us
tennis	champions	deal	hingis	market
number	world	arabia	crrc	crude
slam	tournament	states	flame	week
williams	argentina	trump	tourists	last

We can see that some topics (or distributions of words) indeed reveal latent themes or topics. For example, topics 2 and 7 appear to talk about football or soccer; topic 6 is probably a topic about tennis; topic 10 is about oil markets with the top words “oil, market, prices”; and topic 8 is similar but focuses more on the Middle East. There are topics that make sense but are not very informative. For example, topic 1 is probably about all the hyperlinks in the articles, and topic 3 is related to vehicles. There are also topics that do not make much sense. For example, topic 4 contains several ordinal words such as “one, two, first, second” and “england.” Therefore, we need to be cautious when interpreting the results from the LDA topic models: Use LDA as an exploratory tool to identify patterns and trends in the text data, but do not treat the results as “true meanings” of the text without further investigations in a more granular manner. Do not over-interpret the results!

---

3 Your Turn

You can download this sample dataset and see whether you can reproduce the

results presented here. The number of topics (T = 10) used in this example might not be optimal. Try out a few numbers from 5 to 20, for example, and pick the one that achieves the highest likelihood (or any metric of your choice). See how the results change with this optimal choice of T. Hint: the log-likelihood of a model can be obtained by the score() function of the fitted model with the document-term matrix as its input.