Project Report

Y Sasi Kiran(CS13B055), K Sai Srinivas(CS13B039)

April 30, 2016

Abstract

This project aims at solving source code author attribution problem which is basically identifying author given a code file. We use Latent Dirichlet allocation (LDA) for modeling the source code files as a generative model. We then find the similarity between topic distributions of different authors to identify the most likely author of the test files.

1 Introduction

Author attribution problem is one where we identify the most likely author of a given document. It is one of the well researched topics in English Language and is used in identifying authors of novels, blogs, etc. Source code author attribution is a problem along similar lines but aims at identifying the author of source code. Source code author attribution has applications in many fields like plagarism detection, digital forensics, etc.

One of the approaches to solve this problem is to convert the given data into feature vectors and apply a SVM on this data identifying each author as a class. This approach looses its feasibility due to two reasons. When the number of authors to be identified becomes large, then we need to use one-many or one-one approach of SVM for classification both of which are are not so good as binary class SVM. Moreover, the identification of good features for converting data into feature vector may not be intuitive.

We use Latent Dirichlet allocation (LDA) which is a probabilistic topic modeling mechanism to address these issues. LDA works well for multiple authors and the features are all latent. We provide a brief introduction of Blei's LDA and the use of LDA in generic Author attribution. We then show the application of LDA to source code author attribution and the performance of this on certain data-sets.

2 Latent Dirichlet Allocation

LDA was first proposed by Blei et.al. It is a probabilistic model and it is obtained by assuming that the documents have been generated from an underlying probabilistic distribution. The basic idea of LDA is that documents are represented as random mixtures of latent topics where each topic is characterized by

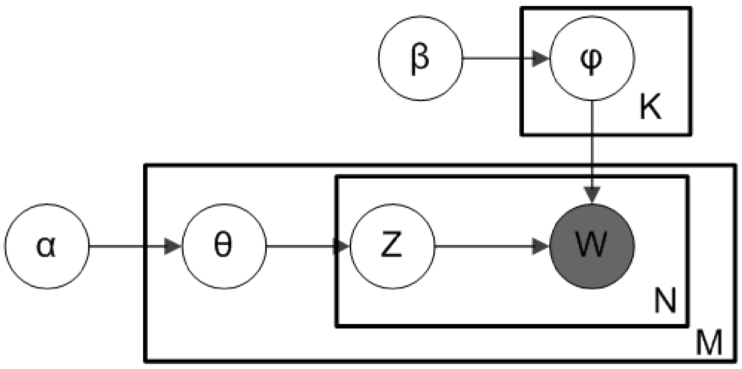

Figure 1: The figure shows the plate representation of LDA model.

a distribution of words. The Figure 1 shows the Probabilistic model for LDA

LDA assumes the following generate model for all the documents corpus.

1. Choose N the number of words in each document from Poisson(ξ)
2. Choose θ_i, the per document topic distribution from Dirichlet(α)
3. Choose φ_k, the per topic word distribution from Dirichlet(β).
4. Then for each of the words $w_{i,j}$, choose a topic $z_{i,j}$ from Multinomial(θ_i) and choose the word $w_{i,j}$ from Multinomial($\varphi_{z_i,j}$)

So,the Joint probability is given by

$P(\theta, z, w|\alpha, \beta) = P(\theta|\alpha)\Pi_{n=1}^{N} P(z_n|\theta)P(w_n|z_n, \beta)$ and hence

$P(W|\alpha, \beta) = \int P(\theta|\alpha)\Pi_{n=1}^{N} \Sigma_{z_n}(z_n|\theta)P(w_n|z_n, \beta)d\theta$ which is document generation probability.

Initially, the basic idea was to assume a generative model for each author and find the α and β for that author. Now, given the test document we find the probability of that document being generated given α and β for each author and report that author which has the highest probability. The problem with this approach is that the computation of probability(exact inference) is intractable. Hence, We use Gibbs Sampling.

We use Gibbs Sapling for the purpose of estimating theta and phi in this smoothed LDA which is described below.

2.1 Gibbs Sampling

Gibbs sampling is a Markov Chain Monte Carlo method for sampling. The basic idea of Gibbs sampling here is to generate samples(which are topic assignments) for a specific word given the samples for all other words. The probability of topic assignments for a given word is found as follows.

$$P(z_i = j|z_{-i}, W) = P(w_i|z_i = j, z_{-i}, W_{-i})P(z_i = j|z_{-i})$$

where z_i is the topic for the ith word given all other topic assignments(z_{-i}).i.e. the probability of the ith word being assigned topic j is the product of probability of the word given the topic j multiplied with the probability of the topic j given the document.

The terms respectively are

$$P(w_i|z_i = j, z_{-i}, W_{-i}) = \frac{n^{(w_i)}_{-i,j} + \beta}{n^{(\cdot)}_{-i,j} + W\beta} \text{ and } P(z_i = j|z_{-i}) = \frac{n^{(d_i)}_{-i,j} + \alpha}{n^{(d_i)}_{-i,.} + T\alpha}$$

where $n^{(w_i)}_{-i,j}$ is the number of instances of word w_i assigned to topic j, not including the current one.
$n^{(\cdot)}_{-i,j}$ is the total number of words assigned to topic j not including current one.
$n^{(d_i)}_{-i,j}$ is the number of words in document d_i assigned to topic j not including the current one.
$n^{(d_i)}_{-i,.}$ is the total number of words in document d_i
α and β are the smoothing hyper-parameters.

Thus the algorithm is to iteratively sample the topic for each word from the distribution of topics given all the other assignments and all the words. Gibbs sampling converges to the original posterior distribution of the topic-word probability and document-topic distribution after large number of iterations.

3 Author Attribution

Author attribution using LDA is addressed by Seroussi et.al. In his work, he employs the LDA in the following manner to find the author for normal data.

- An Approach of LDA + Hellinger is used for author attribution.
- We first assume a generative model for all the training documents and then generate the document-topic probability(θ) for each document.
- Now given a test document, we find the document topic probability using the same generative model as train.
- We then find the Hellinger distance between the test document's θ and each of the train document's θ and output the author that has the least Hellinger distance as the author of the test file.

The Hellinger distance between two document-topic distributions is given by

$$D(\theta_1, \theta_2) = \sqrt{\frac{1}{2}\Sigma^T_{t=1}(\sqrt{\theta_{1,t}} - \sqrt{\theta_{2,t}})^2}$$

We have two approaches we can adopt to represent the training data. One method is where we take all the training documents as themselves and so we have multiple training documents for each author.This approach is called LDAH-M. The other approach is where we combine all the training files of an author into a single file and this approach is called LDAH-S. We employ LDAH-S in all our experiments.

4 Source Code Author Attribution

We may apply LDA trivially on source code directly or may take training data to consist of only punctuations in the source code.

Source code author attribution is slightly different from general author attribution. The main difference comes from the fact that source code author attribution is more dependent on structure of source code rather than the actual words employed themselves. This is due to the following reasons.

- Two authors working across same project and using the same language will use similar vocabulary. This happens due to common methods, classes, variable names etc.
- Two authors using different languages may be classified based on the language constructs rather than their style of coding.
- Authors using specific constructs in their comments (eg. both use Italian in their comments) will be classified as same author due to the comments ignoring their style.

We propose that performing LDA on the punctuations of the source code is generally preferred. However, LDA on raw source code gives results when used across different projects and similar languages.

We choose LDAH-S over LDAH-M because considering all the documents of a single author as a single file gives all the possible variations of an author in a single document-topic distribution.

4.1 Raw source code

In this method, we consider the training documents to consist of raw source code files. This method is effective over the method where we remove punctuations from the source code and keep only words. We give an example for this case.

We consider the snippets of code as follows.

Listing 1: code 1

```
if
{

}
```

Listing 2: code 2

```
if{

}
```

If punctuations are not removed, there will be a difference between the two codes because "if{" is treated as a single word in the second case which will not be so, if the punctuations are removed. This makes the difference between two authors who follow the above style of writing.

4.2 Only Punctuations

We consider punctuations of a code as anything that is not an alphabet or a number. Performing LDA on source code data containing only punctuations can differentiate between authors because different authors use different formatting for their code which causes a difference in punctuations.

Examples include

- number of object methods(calling involves ".")
- number of functions(resulting in { })
- type of control structures used(switch-if)
- number of comments and also their commenting style(// versus /*)

However there are limitations as seen by the following two code samples

Listing 3: code 1

```
{
        int;
}
{
        int;
}
```

Listing 4: code 2

```
{
    {
        int;
        int;
    }
}
```

In the above two cases , the vocabulary is the same and the number of times each symbol occurs is the same. Since LDA considers documents as bag-of-words, it does not consider the structure. The above two cases are treated similarly by LDA.

To avoid this we can encode the level of nesting by using a symbol whose length varies by the nesting level. In the first case, we would add two words "n", "n" whereas in the second case we add "n", "nn" indicating the level of nesting. This is a suggestion for future work.

α	β	K	Accuracy (in %)
0.5	0.1	5	17/59 = 28.8
0.5	0.1	10	33/59 = 55
0.5	0.1	12	39/59 = 66.1
0.5	0.1	15	34/59 = 57.6
0.5	0.1	20	29/59 = 49.1
0.5	0.1	50	32/59 = 54.2
0.5	0.1	100	30/59 = 50.8

Table 1: The table shows the accuracy obtained by varying K keeping α and β constant.

5 Results

We have used the source code data from UCI-source code repository. We show results of LDA on punctuations performed on SDS-source-repo-18K dataset which was further refined specific author-wise.

5.1 Data-set Description

The training data-set consists of 7 authors, each author approximately having 10k-20k lines of code in the training file. However, note that it gives appreciable results when there are as few as 2000 lines per author.

The test-set consists of 59 files randomly selected from these seven authors.

We remove all the alphanumeric characters from both the training and test files and retain only punctuation.

5.2 Performance

The parameters of LDA are

- The number of documents in the corpus(M).This depends on the corpus.
- The size of vocabulary(V). This depends on corpus.
- Number of features(K) which denotes the number of latent features used by LDA. The default number of latent features are 10.
- Number of iterations for which the LDA sampling procedure is repeated. The default number of iterations are 2000
- α and β which are smoothing parameters. The default values of α is 0.5 and is varied depending on K, and the default value of β is

The parameters to be varied are K, α, β and number of iterations.

α	β	K	Accuracy (in %)
0.01	0.1	12	40/59 = 67.8
10	0.1	12	17/59 = 28.8
0.5	0.01	12	38/59 = 64.4
0.5	10	12	10/59 = 16.9

Table 2: The table shows the accuracy obtained by varying α and β keeping K constant.

5.3 Explanation

K indicates the number of latent topics in the corpus. Table 1 shows the variation of accuracy for different values of k.

Here, we observe that accuracy initially increases with increase in K, and then decreases as K further increases.

This is because a small value of K indicates that each word has a small choice of topics and hence the internal structure is not fully realized by the final document topic distribution. Mis-classifications will be common.

For large values of K, the per-document-topic distribution becomes uniform. Thus a test document becomes closer to every train document and hence less variation is observed in the Hellinger distance. This leads to larger number of misclassifications.

It is also not surprising to see that for small values of α and β, the accuracy is better. This is because α and β are smoothing parameters and smaller the smoothing parameters, the closer we get to the actual distribution. This is observed in Table2.

There is improvement on increasing the number of iterations to 3000, but after that there is not much variation.

6 References

http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

https://people.cs.umass.edu/ wallach/courses/s11/cmpsci791ss/readings/griffiths02gibbs.pdf

http://www.aclweb.org/anthology/W11-0321