Text Mining Independent Study Report

Advised by Professor David Banks
Fall 2016 · Tianlin Duan

Introduction

Background

The goal of this project is to automatically generate synthesized documents from Wikipedia with pre-defined model, and use the corpus with ground truth as benchmark for testing and quantifying the uncertainty in identifying topics using the Latent Dirichlet Allocation model. The larger context of this project is a chapter on quantifying uncertainties in topic modelling in the dissertation of Duke Statistics phD candidate Christine Chai, also advised by Prof. David Banks.

Deliverables

The deliverables for this project consists of

1) Reproducible R code
2) A sample dataset of 100 synthesized documents generated with the aforesaid code

Each synthesized document is constructed such that it has:

- 500 words
- ½, ¼, ⅛ of content about topic X, the rest about topic Y (a similar topic to X) or topic A (a topic not directly related to X)
  - The sample dataset chose X=Statistics, Y=Calculus, A=Dance, meaning that a document from this sample dataset could be 50% Statistics + 50% Calculus, or 25% Statistics + 75% Dance etc.
- Content randomly generated
  - More details in the next section, The Code

The Code

The coding for this project has two parts, autoGetText.R that generates synthesized documents with any two specified topics, and cleanText.R that prepares the generated documents as a corpus for further LDA testing.

autoGetText.R

The code for this part consists of four functions and a main part to implement those functions with desired parameters (topics, document length etc.):

`script_urls(String topic)`

This function takes a topic and returns a list of links for subpages under that topic.

For example, `script_urls("Statistics")` returns links such as

```
[1] "/wiki/Standardized_testing_(statistics)"
[2] "/wiki/Standard_deviations"
[3] "/wiki/Statistical_survey"
```

...

It may also return links that are not directly related to the topic:

```
[24] "/wiki/Western_Electric_Company"
```
[1]

```
[73] "/wiki/Political_Science"
```

...

Such result represent the internal structure and relationships among topics (and subtopics), and thus should still be accepted.

`topicText(String[] tlist)`

This function takes in the list of links returned by `script_urls(topic)` and returns a list of lists of text (i.e. each element of the output is a list of text on a subpage) under that topic.

For example, `topicText(sList)` where `sList=script_urls("Statistics")` returns a list of 94 lists of text, and a list of text may be:

```
[[1]]
[1] "A Z-test is any statistical test for which the distribution of the test statistic under
the null hypothesis can be approximated by a normal distribution. Because of the central limit
theorem, many test statistics are approximately normally distributed for large samples. For
each significance level, the Z-test has a single critical value (for example, 1.96 for 5% two
tailed) which makes it more convenient than the Student's t-test which has separate critical
values for each sample size. Therefore, many statistical tests can be conveniently performed
as approximate Z-tests if the sample size is large or the population variance known. If the
population variance is unknown (and therefore has to be estimated from the sample itself) and
the sample size is not large (n < 30), the Student's t-test may be more appropriate."
 [2] "If T is a statistic that is approximately normally distributed under the null
hypothesis, the next step in performing a Z-test is to estimate the expected value θ of T
under the null hypothesis, and then obtain an estimate s of the standard deviation of T. After
that the standard score Z = (T − θ) / s is calculated, from which one-tailed and two-tailed
p-values can be calculated as Φ(−Z) (for upper-tailed tests), Φ(Z) (for lower-tailed tests)
and 2Φ(−|Z|) (for two-tailed tests) where Φ is the standard normal cumulative distribution
function."
...
```

[1] This link actually directs to https://en.wikipedia.org/wiki/Western_Electric_rules, a page on Western Electric rules, decision rules in Statistical Process Control for detecting "out-of-control" or non-random conditions on control charts.

`getNwords(int n,String[][] tTxt)`

This function takes a wordcount limit, `n`, and the list of lists of text returned by `topicText(tlist)` , and returns a randomly selected segment of text that has exactly `n` words from the middle part of a subpage in `tTxt`.

For example, `getNwords(125,sTxt)` where `sTxt=topicText( script_urls("Statistics"))` may return

```
[1] "Another drawback of systematic sampling is that even in scenarios where it is more
accurate than SRS its theoretical properties make it difficult to quantify that accuracy In
the two examples of systematic sampling that are given above much of the potential sampling
error is due to variation between neighbouring houses  but because this method never selects
two neighbouring houses the sample will not give us any information on that variation As
described above systematic sampling is an EPS method because all elements have the same
probability of selection in the example given one in ten It is not simple random sampling
because different subsets of the same size have different selection probabilities  eg the set
has zero probability of selection Systematic sampling can also"
```

Running the same code again will return a completely different paragraph with same length on a different subtopic under Statistics:

```
[1] "The history of fractals traces a path from chiefly theoretical studies to modern
applications in computer graphics with several notable people contributing canonical fractal
forms along the way According to Pickover the mathematics behind fractals began to take shape
in the th century when the mathematician and philosopher Gottfried Leibniz pondered recursive
selfsimilarity although he made the mistake of thinking that only the straight line was
selfsimilar in this sense In his writings Leibniz used the term fractional exponents but
lamented that Geometry did not yet know of them Indeed according to various historical
accounts after that point few mathematicians tackled the issues and the work of those who did
remained obscured largely because of resistance to such unfamiliar emerging concepts which
were sometimes referred"
```

This is because this function first randomly samples a subpage from the list of lists of text on the chosen topic, and then randomly samples a starting paragraph from the middle of the subpage, and also makes sure that the rest of the page is long enough for generating `N` words (If not, go back and sample again) to avoid generating NA.

Notice that the returned text only have alphabetic characters, and this is to get rid of citation brackets and special formatting before word counting, and also prepares for further cleaning.

`genDocN(String[][] t1Txt, String[][] t2Txt)`

This function takes in two list of lists of text returned by `topicText(tlist)` on two different topics, and generates a number of (50 for the sample dataset) synthesized documents that may be ½, ¼, ⅛ about topic 1 and the rest about topic 2. It writes out each document as .txt file and stores all documents in a list for further cleaning and testing.

This function uses `getNwords(n,tTxt)` internally, and given the pre-decided model for (topic1, topic2): (½, ½), (¼, ¾), or (⅛, ⅞), the `n` parameter is decided by the total length of the synthesized document, N, which is set in the body of the function but can also be specified as any other reasonable length. In this case, `n` can be 438, 375, 250, 125, or 62 words.

For example, `genDocN(sTxt, dTxt)` where
`sTxt=topicText(script_urls("Statistics")),`
`dTxt=topicText(script_urls("Dance"))` may generate a document with ⅛ (62 words) Statistical content and ⅞ (438 words) Dance content:

```
"In the context of statistics econometrics quantitative finance seismology meteorology and
geophysics the primary goal of time series analysis is forecasting In the context of signal
processing control engineering and communication engineering it is used for signal detection
and estimation while in the context of data mining pattern recognition and machine learning
time series analysis can be used for clustering classification query   Traditionally
distinctions are made between Ballroom Swing and Jazz Dance Swing styles East Coast Swing is a
standardized dance in American Style Ballroom dancing while Jive is a standardized dance in
International Style however both of these fall under the Ballroom Swing umbrella Jazz Dance
forms… (all dance content afterwards; entire document 500 words long)…"
```

The synthesized document will be written out with informative naming such as `"WikiDoc/Statistics_Dance/3_62_438.txt"` : the 3rd document with 62 words on Statistics and 438 words on Dance in the folder of `Statistics_Dance` under the master folder `WikiDoc` with all synthesized documents generated from Wikipedia.

Main function

This part implements the aforesaid functions with specified parameters (topics, length of each document). For example, the main code may be:
`farList=genDocN(`**`unique`**`(topicText(`**`unique`**`(script_urls("Statistics")))),` **`unique`**`(topicText(`**`unique`**`(script_urls("Dance")))))` , where the synthesized documents will be stored in text form in a list, and each document will be written out into the `Statistics_Dance` folder.

Notice that **`unique()`** is nested around `script_urls(topic)` and `topicText(tlist)` to remove duplicated content. Duplication may be seen in:

A. Duplicate links
B. Duplicate content returned by different link names.
   ```
   [6] "/wiki/Type_I_error"
   [7] "/wiki/Type_II_error"
   ```
   These two links in fact direct to the same Wikipedia page on Type I and type II errors, but since the embedded links were scraped from the source code for the Wikipedia page on Statistics, their names did not necessarily represent the name of the web page they link to.

The **`unique()`** function can also be applied within `script_urls(topic)` and `topicText(tlist)` , and the three functions implemented in the main function (again, `getNwords(n,tTxt)` is implemented internally in `genDocN(t1Txt, t2Txt)`) can also be implemented and stored separately for easier inspection and potential debugging.

cleanText.R

The code for this part uses all documents generated (100 for the sample dataset) as one corpus, and prepares it for further LDA testing.

Cleaning the corpus

The standard text data cleaning procedure included the following steps:

1. Keeping only alphabetic characters
   This step is already done when generating the synthesized documents.
2. Removing stop words
   The default list of stopwords from the R package `quanteda` is used for this step. A preparation step would be changing all characters in corpus to lowercase.
3. Stemming
   The `wordStem` function from the R package `SnowBallC` is used for this step. One caution for using this function is that the input needs to be a vector of words, and thus we can not directly apply the function to the entire corpus at once.
4. Final clean-up
   Trimming and stripping unnecessary whitespace from the corpus, and telling R to treat the post-cleaning documents in corpus as PlainTextDocument to comply with requirements for further processing.

Ngramming

The code for Ngramming in this project is modified based on the NGramming Processing Script (c) by Teague Henry. Two methods for n-gramming can be implement in the original script: p-value cutoff and count cutoff.

```
textToBigrams <- function(textList, cutoff = 100, p.value.test = F,
p.value.cut = .05, dfm.return = F, return.word.list = F, concat =
"_", BF.corr = F)
```

P-value cutoff

When using p-value cutoff, the output, `bigramTable,` is a n*4 matrix where n is the total number of possible bigrams, for example:

V1 (word 1)	V2 (word 2)	bigramCounts	pvalue
unbias	estim	11	1.768833e-23
estim	varianc	2	6.611956e-03
...	...	...	...

Notice that the p-value is much smaller than the range of value we usually see when approximating distribution of test statistics by normal distribution under Central Limit Theorem. This is because the code is calculating the exact probability of getting a specific bigram, $Pr(getting\ word\ 2 \mid getting\ word\ 1)$, under the binomial distribution, instead of approximation:

```
if(dim(bigramTable)[[1]] > 0){
      print(dim(bigramTable))
      for(i in 1:dim(bigramTable)[[1]]){
        prop1 <- unigramTable[.(bigramTable[i,V1]), unigramCounts]
        prop2 <- unigramTable[.(bigramTable[i,V2]),
unigramCounts]/(totalUnigramCount-prop1)
        pval <- pbinom(bigramTable[i,bigramCounts]-1, size = prop1,
prob = prop2, lower.tail = F)
        bigramTable[i, pvalue := pval]
      }
```

Here, `prop1` is the number of total appearances of word 1 in the corpus; `prop2` is the proportion of word 2 appearances in all non-word1 words in the corpus, i.e. the probability of getting word 2 assuming independence between word 1 and word 2 appearances; and thus `pval` is $Pr(number\ of\ times\ getting\ word\ 2 \geq bigram\ appearances)$ in `prop1` number of trials, where $Pr(getting\ word\ 2) = prop2$ for each trial.

The p-value calculated with Teague's code will be more precise than under CLT approximation, for which the derivation for p-value would be (modified based on Teague's original script):

```
      pi <- unigramTable[.(bigramTable[i,V1]),
unigramCounts]/totalUnigramCount
      pj <- unigramTable[.(bigramTable[i,V2]),
unigramCounts]/totalUnigramCount
      pij <- bigramTable[i,bigramCounts]/totalUnigramCount
      zscore <- (pij-pi*pj)/sqrt((pij*(1-pij)/totalBigramCounts))
      pval <- pnorm(-abs(zscore))
```

When using approximate instead of exact probability, the p-values are much smaller:

V1 (word 1)	V2 (word 2)	bigramCounts	pvalue
unbias	estim	11	0.000495883
estim	varianc	2	0.092212162
...	...	...	...

However, since the exact probability of getting a specific bigram can be calculated, approximation is not necessary and do not have good precision, and thus Teague Henry's original script canl be used to derive p-values, and the conventional 0.01 cutoff may be applied.

Count cutoff

Using count cutoff alone for deciding meaningful n-grams is not appropriate since:

1) One static cutoff for corpus of different sizes (eg. 100 vs. 10000 documents) will be quite biased and problematic
2) Output n-grams will largely depend on the topic proportions of the corpus. For example, in the sample dataset, every document in the 500 documents has Statistical content, and thus Statistical n-grams will unsurprisingly dominate the pool of identified n-grams if using same cutoff regardless of corpus composition.
3) Given results from using each of the ngramming method alone (count vs. p-value), p-value alone performed slightly better compared to count cutoff alone, though either method alone was flawed and the results for both were not optimal

Hybrid cutoff

Given the comparatively better and more stable performance of p-value cutoff and the problems with using count cutoff alone, p-value cutoff was selected as the main method for identifying n-grams in the corpus. However, a potential problem with using p-value cutoff alone is that it keeps all bigrams with p-values below the cutoff regardless of their total appearances in the corpus. However, if a "bigram" only appeared twice in a corpus of 500*100=50000 words, it will likely not provide much essential information and should not be automatically identified as a bigram that should be processed together.

As a result, count cutoff was incorporated into the n-gramming method as a second filter for identifying meaningful n-grams. For the sample dataset, "unbiased estimate" ("unbias estim" after stemming) was chosen as the benchmark bigram and its total of 11 appearances was used to filter for meaningful bigrams and other n-grams.

With the hybrid cutoff, 52 meaningful bigrams were identified from the corpus.[2]

Beyond bigram

To identify trigrams, quargrams, or even longer n-grams, the same function for identifying bigrams will be applied repeatedly with different concatenation format (eg. "_" for bigrams, "." for trigrams) until no n-gram is identified with the hybrid cutoff. Teague's function for identifying both bigrams and trigrams, `textToBigramsTrigrams` , is an implementation of such logic.

For the sample synthesized corpus, no n-grams were identified beyond the 52 bigrams.

[2] See Appendix for the full table of the 52 bigrams identified from the sample dataset with counts and p-values.

Discussion

The process of generating synthesized corpus also suggested some interesting next steps in optimizing web-scraping and more. One such next step would be identifying inline equation as a whole when processing. For example, the z score formula $Z = (x - \mu) / \sigma$ will become `"zx"` after all punctuations and non-alphabetic characters are removed, and embedded equations in LaTex-similar .SVG format such as $\tan S = \sqrt{\left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2}$ will become `"tanSzxzy"` when scraped from the Wikipedia page. Although the equations do not carry essential information for the interest of this project, identifying and preserving equations may be of interest for text mining / analysis projects in the field of education, transportation etc.

A more challenging and complicated next step would be automatically choosing the most appropriate count cutoff in the hybrid cutoff method for identifying meaningful n-grams for corpus of different size. The determination of the cutoff would depend not only on the size of the corpus, but also on the topic proportions of the corpus. For example, if the corpus is consists of abstracts for the JSM conference, then the cutoff for Statistical n-grams should be much higher than if the corpus is on a variety of topics including Statistics.

While the project can still be optimized in many ways, it has met the objectives for the scope of this independent study. Since the code is highly reproducible and can generate synthesized text documents with given ground truth from any topic on Wikipedia, it has the potential of generating larger datasets that may serve as the benchmark for other topic modelling projects. The synthesized corpus of 100 documents constructed with topics “Statistics”, “Calculus”, and “Dance” has served as the benchmark dataset for a chapter in the dissertation of phD candidate Christine Chai on quantifying uncertainties in identifying topics with the LDA model, and has also been used as benchmark to assess the performance of a trained model in a machine learning project of Master candidate Sophie Guo.

Acknowledgement

I would like to thank Professor David Banks for his guidance and encouragement throughout the semester. I would also like to thank all other students in the independent study group, phD candidates Christine Chai and Dereck Owen-Oas, and Master candidates Andrea Gao, Sophie Guo, Mengrun Li, Xiao Lin, Jun Yoon, for providing great support and insights for the project. I am immensely grateful for being in the Text Mining/ Topic Modeling independent study group this inspiring and enjoyable semester.

Appendix

Bigrams identified from synthesized Wikipedia corpus[3]

V1	V2	bigramCounts	pvalue[4]
standard	deviat	40	4.18E-91
random	variabl	40	4.05E-74
analyt	function	31	5.88E-51
unit	state	21	1.97E-46
normal	distribut	23	7.26E-45
null	hypothesi	16	6.86E-44
th	centuri	14	4.19E-40
lindi	hop	11	1.88E-37
time	seri	18	2.41E-37
inflect	point	19	2.55E-36
perform	art	20	1.34E-35
row	vector	13	1.14E-31
intellectu	rigour	11	1.53E-31
sampl	mean	28	3.21E-31
finit	differ	18	1.22E-30
academ	disciplin	12	1.22E-30
linear	map	15	2.85E-30
converg	probabl	17	1.87E-27
point	inflect	16	4.02E-27
control	group	13	1.02E-25
function	f	22	1.51E-25
symplect	manifold	10	2.67E-25
complex	analyt	14	3.60E-25

[3] All bigrams are in post-stemming form.
[4] Bigrams sorted by p-value in ascending order.

class	c	13	4.04E-25
fit	data	15	4.65E-24
unbias	estim	11	1.77E-23
column	vector	11	2.24E-23
meanunbias	estim	10	2.63E-23
sampl	size	15	6.00E-23
vector	space	13	6.66E-23
jazz	danc	10	8.79E-23
popul	mean	18	2.05E-22
x	y	13	1.37E-21
partial	deriv	10	4.18E-21
simpl	random	11	5.30E-21
hypothesi	test	11	7.52E-21
real	number	14	2.14E-20
differenti	function	17	1.72E-18
exist	continu	11	3.04E-18
probabl	distribut	14	9.50E-18
data	set	12	6.05E-15
danc	form	12	2.61E-14
random	sampl	13	3.06E-14
converg	distribut	11	6.24E-14
can	also	14	1.42E-12
can	made	10	1.44E-12
can	us	18	3.56E-11
function	differenti	11	4.49E-10
deriv	function	11	4.59E-10
variabl	x	10	5.71E-10
point	x	10	6.22E-08
on	can	10	9.20E-07