CSE 519 -- Data Science (Fall 2017)
Project Progress Report
Rohan Vaish - 111447435
Kiranmayi Kasarapu - 111447596
Shruti Nair - 111481332

Project: Automatic Building of Book Indices

Goals Achieved in accordance with the comments suggested.

Set up evaluation environment	Executed as a part of Data Preprocessing and Evaluation model
Implement Baseline Algorithm	Developed a frequency based algorithm as baseline model
Place connections in the project	Process flow - Data Preprocessing, Baseline model, Evaluation model
Evaluate on that part	Evaluation model developed

DATA PREPROCESSING

1. **Latex File Parsing:**
   We identified 10 latex files from the data set we created earlier, which are compilable with a latex compiler and generated proper PDF's. This is needed because all latex files we collected are not properly showing up the PDF files. So for our purpose we identified 10 files for now to work on.
   For parsing these latex files, we used regular expression matching and extracted plain text from the tex files. The regular expression matches on Latex syntax and extract the text. For example: to extract any equation, the regular expression for it is as follows:

```
r = re.compile(r'\\begin{equation}(.*?)\\end{equation}')
text = re.sub(r, ' ', text)
```

```
# remove LaTeX tags
# - remove completely some LaTeX commands that take arguments
to_remove = [r'\\vspace', r'\\medskipamount', r'\\hspace', r'\\tableofcontents', r'\\noindent',
             r'\\maketitle',r'\\footnote', r'\\centering', r'\\IEEEpeerreviewmaketitle', r'\\Large',
             r'\\includegraphics', r'\\IEEEauthorrefmark', r'\\label', r'\\begin', r'\\maketitle',
             r'\\end', r'\\big', r'\\right', r'\\left', r'\\documentclass',
             r'\\usepackage', r'\\bibliographystyle', r'\\bibliography',
             r'\\cline', r'\\multicolumn', r'\\printindex', r'\\makeindex', r'\\small', r'\\title',
             r'\\author', r'\\thanks', r'\\cite', r'\\ref', r'\\today', r'\\date', r'\\inde',
             r'\\bdf', r'\\edf', r'\\tt', r'\\ece', r'\\bce', r'\\la', r'\\ete', r'\\od', r'\\par',
             r'\\ebla', r'\\dd', r'\\d', r'\\ls', r'\\ubf', r'\\ble', r'\\od\\', r'\\ri', r'\\ls_', r'\\ga',
             r'\\ti', r'\\vom', r'\\textwidth', r'\\itl', r'\\bf', r'\\vy', r'\\een', r'\\ri ', r'\\rit',
             r'\\snos', r'\\plo', r'\\bte', r'\\rod', r'\\ele', r'\\bpf', r'\\epf', r'\\bbla', r'\\en', r'\\vpi',
             r'\\ben', r'\\nenu', r'\\renu', r'\\bcor', r'\\text', r'\\it', r'\\fenu', r'\\atc', r'\\bpn',
             r'\\trn', r'\\bpro', r'\\epro', r'\\ep', r'\\ima', r'\\ecor', r'\\pagebreak', r'\\qed', r'\\t',
             r'\\qe', r'\\bco', r'\\evo', r'\\rm', r'\\ata', r'\\em', r'\\index' ]
```

Figure No 1

Numerous such regular expression are applied on the whole latex text and we removed certain unnecessary tags that were not required such as **\textbf{}** etc. After numerous iterative removal of tags and extracting text from regular expression, we were able to get plain text in a pretty readable format. The snippet of the plain text and code is as shown below.

```
 Riemannian geometry Richard L  Bishop   Preface  These lecture notes are based on the course in R
iemannian geometry at the University of Illinois over a period of many years  The material derives
from the course at MIT developed by Professors Warren Ambrose and I M Singer and then reformulated
in the book by Richard J  Crittenden and me    Geometry of Manifolds    Academic Press  1964  That
book was reprinted in 2000 in the AMS Chelsea series  These notes omit the parts on differentiable
manifolds  Lie groups  and isometric imbeddings  The notes are not intended to be for individual s
elf study  instead they depend heavily on an instructor s guidance and the use of numerous problem
s with a wide range of difficulty  The geometric structure in this treatment emphasizes the use of
the bundles of bases and frames and avoids the arbitrary coordinate expressions as much as possibl
e  However  I have added some material of historical interest  the Taylor expansion of the metric
in normal coordinates which goes back to Riemann  The curvature tensor was probably discovered by
its appearance in this expansion  There are some differences in names which I believe are a substa
ntial improvement over the fashionable ones  What is usually called a   distribution   is called a
   tangent subbundle    or   subbundle    subbundle  for short  The name   solder form    solder f
orm  never made much sense to me and is now labeled the descriptive term   universal cobasis    un
iversal cobasis   The terms   first Bianchi identity   and   second Bianchi  cobasis universal  id
```

Figure No 2

2. Stop Words Removal:

For removing the stopwords from the above generated plain text, we used an inbuilt python library called "nltk" which is a Natural Language Toolkit. It provides with corpus, which has stopwords for English language.

```
from nltk.corpus import stopwords
stop_words=stopwords.words('english')
```

Apart from the stopwords in the nltk library, there are other stopwords which are newly added. We included the newly added stop words to the above stop_words. The file after removing the stopwords looked like this.

```
cofinite set onto cofinite equal identity let defined finally let mio assume aio1 generi
j mqq condition compatible let generic set containing hence containing generic set conta
set tdense set dense topen set open mfn let name tfull set dense double-name tfull name
name regular particular case double-name tregular components define tregular hull regula
 intersects use lemma io prove first claim let consider generic set containing see follo
lent double-name equivalent double-names tequivalent equivalent resp  mrh assume arh1 re
 equivalence assume generic definition forces required establish regularity assume force
 rh3 immediately follows rh2 prove opposite direction suffices show names equivalent ass
idqa full regular double-name generic set tidentity name ri fdouble-name representation
s mqq assume generic sets full regular double-name suffices define therefore assume exis
 condition satisfies true immediately need somewhat modify names shrinking wlog assume r
ts pairs forces generic claim still mggle generic set similarly generic set construction
et generic set lemma therefore generic definition thus generic construction forced hand
 dq1 thus addition choice forces generic similarly fix regularity condition theorem let
ense construction claim indeed let generic set case 1 since incompatible original choice
us indeed set open dense arguments case 1 also imply moreover inherits regularity fexten
uble-name extends let consist pairs condition satisfying let defined way explained follo
rcing theorem definition set form simply genericity easily follows restriction md r let
 a0ext1 generic set generic b a0ext2 generic set generic b a0ext3 belongs 0ext1 generic
 however generic product forcing thus generic sets hence easily required 0ext3 check dq
ext1 hand hence generic required verification dq dq1 b also simple merh assume aerh1 aer
et set generic containing prove set generic lemma 0ext still hence required erh3 assume
lar fincreasing sequences sinc suppose set pairwise compatible define double-name zzqabi
e set qm closed sense a3ext3 strictly increasing sequence double-name belongs 3ext1 supp
ice let claim let verify dq dq1 assume generic set containing prove generic note first s
erefore follows set xi generic lemma io conclude least filter dq dq1 b k sets satisfy fo
a 0ext 0ext3 follows definition set xi generic continue dq dq1 prove genericity let dens
orollary qq condition compatible forces compatible condition forces opposite contradicti
oreover lemma 0ext sets xi generic however construction hence choice thus consider gener
```

Figure No 3

PROCESS FLOW

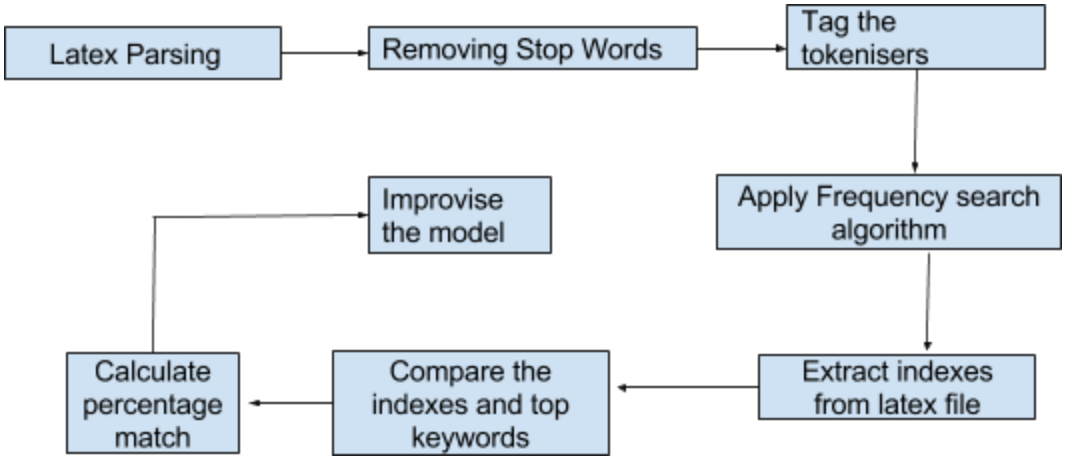

Figure No 4

BASELINE MODEL

For the baseline model, we implemented the frequency search algorithm shown above to extract the most frequently occurring words in the document. Based on the number of indices, the user inputs, say n, we extracted the top n frequently occurring words and listed them out as the indices for the document.

1. **Keyword extraction:**
   We used the same library NLTK for tokenizing the resulted document after the above step. Once tokenizing was done, we applied a counter algorithm, which returns the words in the document and their frequency of occurrence in the document. The algorithm is as follows:

   Algorithm:

a. Have a large corpus of text against which we can compare.
b. Tag each tokenizer according to its type after removing all the stop words.
c. Find the relative frequency of words in the corpus. For example, if your corpus is "the green way is very green way green". Relative frequency is.. [(the, 1/8), (green, 3/8), (way, 2/8), (is, 1/8), (very, 1/8),]
d. Get the relative frequency of words in search string.
e. Divide frequency of search string by corpus, and also take care of the cases, when the word is not included in the string.
f. Sort the answer by the result obtained in d and generate the top n words as given by the user.

The screenshot of the top 10 frequently occurring keywords obtained for a text file are as follows:

```
file1 = open("/Users/pradeepkumarnama/Desktop/CSE519-2017-Project-master/PlainText/1_stop.txt")
line = file1.read()
num = int(input("Enter number of keywords: "))
find_keyword(line,num)
```

```
Enter number of keywords: 10
```

```
[('od', 47736.00000000001),
 ('elements', 9092.571428571428),
 ('countable', 7955.999999999999),
 ('sets', 8160.0),
 ('solovay', 254592.00000000003),
 ('model', 8398.0),
 ('vladimir', 15912.000000000002),
 ('kanovei', 15912.000000000002),
 ('iitp', 15912.000000000002),
 ('ras', 15912.000000000002)]
```

Figure No 5

2**. Index Words extraction:**

We extracted all the indices from the original latex file. For this purpose, we used the concept of regular expression matching and string manipulation and focused on extracting string between \index{*} tags. This is not necessarily limited to finding index between braces and can be expanded in accordance with the different kind of latex files available.

```
if line.startswith('\index'):
        lines.append(line.strip())
if re.match("^[a-zA-Z0-9 !-]*$", element):
        updated_list.append(element)
```

The extracted string found out were as follows:

```
extract_key_words()
```

```
Solovay model
double-name
set!dense
set!open
name!full
double-name!full
name!regular
double-name!regular
regular hull
name!equivalent
double-name!equivalent
extends
```

Figure No 6

EVALUATION MODEL

We varied the size of the indices that needs to be generated and compared the generated indices with the actual indices from the latex file, that we extracted as part of task(4). We calculated the percentage of common indices we are able to extract using the algorithm and plotted on a bar chart against the percentage obtained from the index size. We have shown the results below for 3 latex files below.

For the first file named 1.txt, the result were as follows:

Enter number of indices: 10
Number of matches: 2
Percentage matches: 20.0
[0.0, 9.090909090909092, 18.181818181818183, 18.181818181818183, 18.181818181818183]
[1, 2, 3, 4, 5]

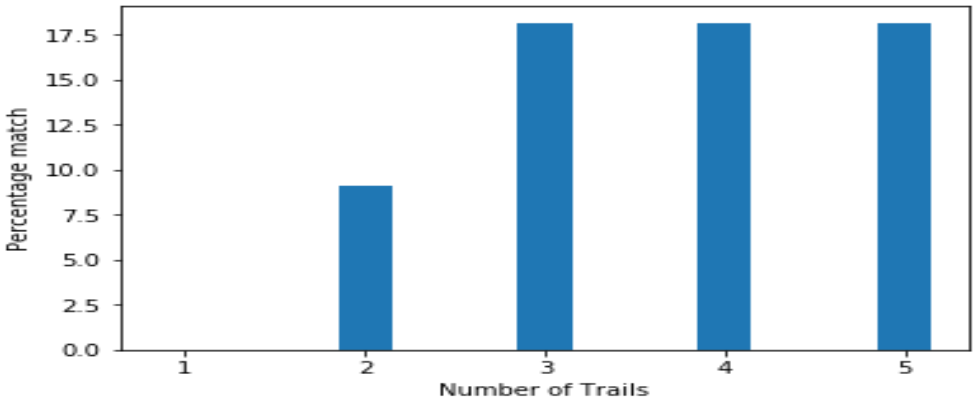

Figure No 7

The graph describes the result of the five trials as given by the user. The number of indices were varied in each trial and it can be seen that, the more the number of indices the better are the results. The maximum percentage of indices matched in the first file were 13. The less percentage can be attributed to the frequency based approach rather than the context based approach. Coupled with that, each author has a unique way of indexing which needs to be taken into consideration.

Similarly, in the case of file 7.txt, the percentage of matching increases as the number of indices is increased.

Enter number of indices: 32
Number of matches: 4
Percentage matches: 12.903225806451612
[3.125, 9.375, 12.5, 12.5, 12.5]

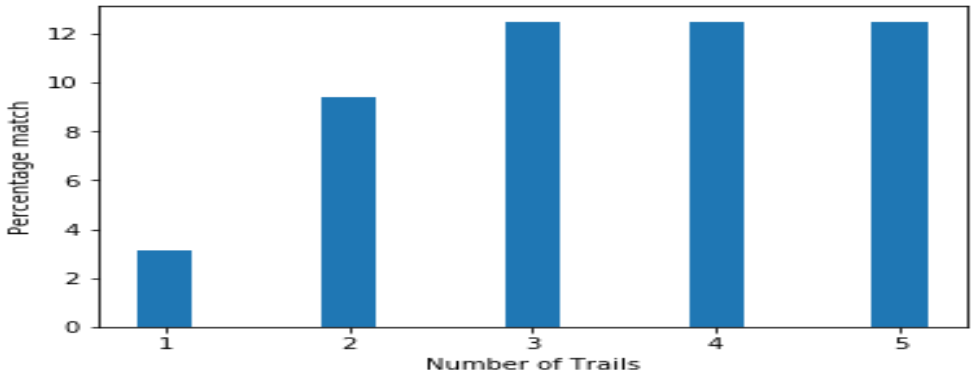

Figure No 8

EVALUATION ENVIRONMENT

Snippets of User Interface

User enters the number of top keywords, which he/she wants to be displayed. This interface is also then used to compare the indices with the top keywords generated.

Figure No 9

In this snippet, the user enters the number of trials he/she wants to try to find out the percentage matches between the original index from the latex file and top keywords generated according to the frequency based algorithm.

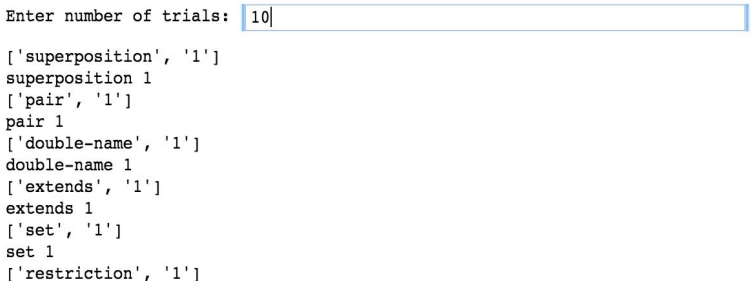

Figure No 10

FUTURE GOALS AND TASKS

1. To use various advanced Natural Language Processing Algorithms to build indices on a more context based rather than on frequency based.
2. Analyze the keywords that are extracted using NLP and evaluate how good or bad these keywords are with respect to the baseline model.
3. Find the percentage of indices that are matching with the NLP algorithms to the actual indices.
4. Verify how much they are similar to the baseline model, or how much they deviate from the baseline model.
5. Finally put the indices back into the latex file, i.e. marks the indices we generated in the latex file, so that when the file is compiled, the indices are generated in the PDF.
6. Handling bivariate and multivariate words for better processing of the baseline model.
7. Use other latex files without indices and see how the indices are generated in the PDF.

8. Sometimes the indices which are there in the latex files are not just between /index{*} tag, a command is defined and that is used to generate the index. Need to identify such latex files and figure a way to use them for the evaluation process.