Homework 3 – Steven Lin

Part 1 : Document similarity

0. Splitting, tokenization, normalization

Code:

- hw3.py

Inputs:

- classbios.txt

Outputs:

- Folder classbios_by_person
- Folder classbios_by_person_normalized

2. boolean

Code:

- hw3.py

Inputs:

- classbios.txt

Outputs:

- boolean.txt (cosine matrix)
- boolean_top.csv (top 3)
- boolean_bottom.csv (bottom 3)
- boolean_output.txt (top 3 and bottom 3)

Which three pairs of documents are the most similar? Which are least similar?

top			
rank	**indices**	**names**	**value**
1	(2, 15)	('bhargava_anuj', 'lin_luis')	0.303230853
2	(24, 25)	('sun_yuan', 'tu_ye')	0.281988479
3	(2, 3)	('bhargava_anuj', 'bozoklar_berk')	0.281642037

bottom			
rank	**indices**	**names**	**value**
1	(13, 17)	('leboeuf_mark', 'mays_jacob')	0.070459485
2	(9, 13)	('herrera_alejandro', 'leboeuf_mark')	0.086258195
3	(16, 19)	('maravilla_francisco', 'nekkanti_sivamadhav')	0.087626782

3. tf-idf similarity

Code:

- hw3.py

Inputs:

- classbios.txt

Outputs:

- tf_idf.txt (cosine matrix)
- tf_idf_top.csv (top 3)
- tf_idf_bottom.csv (bottom 3)
- tf_idf_output.txt (top 3 and bottom 3)

Which three pairs of documents are the most similar? Which are least similar?

top			
rank	**indices**	**names**	**value**
1	(2, 15)	('bhargava_anuj', 'lin_luis')	0.11897304
2	(4, 10)	('cambo_scott', 'heston_matthew')	0.103557395
3	(2, 3)	('bhargava_anuj', 'bozoklar_berk')	0.100004777

bottom			
rank	**indices**	**names**	**value**
1	(9, 13)	('herrera_alejandro', 'leboeuf_mark')	0.005919047
2	(13, 17)	('leboeuf_mark', 'mays_jacob')	0.007173167
3	(9, 10)	('herrera_alejandro', 'heston_matthew')	0.008669488

Does this more advanced method seem to be doing a better job than just using binary similarity? Why?

Yes, the more advanced method seems to be performing better. The new method represents each document as a weighted tf-idf vector, which increases with the number of occurrences within a document and increases with the rarity of the term in the collection. In other words, it makes use of the fact that rare terms are more informative than frequent terms. Thus, two documents that contain rare items in the collection are likely to be more related. This is clearly seen in the documents by cambo_scott and heston_mathew, which appears to be similar in the tf-idf method but not the boolean. These documents both contain rare words like “Technology and Social Behavior”, “sociotechnical systems” and “qualitative”. This makes sense, since most of the classbios are from the MSiA program, while these bios are from people of the Technology and Social Behavior programs.

Part 2 : Index and Retrieval

3. Indexing

Code:

- hw3_scraping.py
- Indexer.java
- MainCreateIndex.java

Inputs:

- webscraped_data.json (for Java)

Outputs:

- webscraped_data.json (from Python)

The python code webscrapes the wikipedia page and saves the results in Json file, which is used as input for the Lucene index in Java.

Regarding the special cases in which countries have more than one capital, the capital was chosen such that it is the capital that is either declared, official, legislative or constitutional. This selection will ensure that only one capital is chosen for each country that has multiple capitals. Therefore, 249 countries and 249 capitals were included in the data (one capital for each country).

The strip out of the html from the web pages before sending to the English analyzer was done in python during the web scraping. Note that the webscraping extracted the text from the paragraphs, as this is more relevant for the document and does not capture nuances in captions and other information in the Wikipedia page that might not be relevant to the document.

4. Retrieval

Code:

- SearchEngine.java
- MainSearchBoolean.java
- MainSearchPhrase.java
- MainSearchFuzzy.java
- MainSearch.java

Inputs:

- None

Outputs:

- query_boolean.txt
- query_phrase.txt
- query_fuzzy.txt
- query_interesting.txt

Queries using the field "city_text" (the city's pages):

- **'Greek' and 'Roman' but not 'Persian': Use a BooleanQuery**

  Query: +city_text:greek +city_text:roman -city_text:persian
  Results found: 30
  Tiraspol, Transnistria (0.16824351)
  Sukhumi, Abkhazia (0.15047915)
  Tunis, Tunisia (0.12993431)
  Tripoli, Libya (0.11545949)
  Lisbon, Portugal (0.11374697)
  Sofia, Bulgaria (0.093560524)
  Algiers, Algeria (0.090479545)
  Bucharest, Romania (0.08921147)
  Podgorica, Montenegro (0.08569528)
  Cairo, Egypt (0.07498337)
  Ljubljana, Slovenia (0.072851576)
  Zagreb, Croatia (0.07191507)
  Budapest, Hungary (0.066828944)
  Bern, Switzerland (0.06499827)
  Gibraltar, Gibraltar (0.06499827)
  Monaco, Monaco (0.064668946)
  Skopje, Republic of Macedonia (0.063978694)
  Bangui, Central African Republic (0.060595714)
  Montevideo, Uruguay (0.057751894)
  Bratislava, Slovakia (0.056873485)

Berlin, Germany (0.055430524)
Madrid, Spain (0.050893016)
Dublin, Republic of Ireland (0.045446783)
Kiev, Ukraine (0.045446783)
Amsterdam, Netherlands (0.04304812)
Wellington, New Zealand (0.042496752)
Havana, Cuba (0.041530106)
Warsaw, Poland (0.04062392)
Copenhagen, Denmark (0.037872322)
Buenos Aires, Argentina (0.034608424)

- **'Shakespeare', even if it's misspelled: Use a FuzzyQuery**

  Query: city_text:shakespeare~2
  Results found: 4
  London, United Kingdom (0.06778301)
  Cairo, Egypt (0.06710176)
  Prague, Czech Republic (0.06710176)
  Washington, D.C., United States (0.0575158)

- **the words 'located below sea level' near each other: Use a PhraseQuery with a slop factor of 10.**

  Query: city_text:"locat below sea level"~10
  Results found: 1
  Baku, Azerbaijan (0.09178771)

- **interesting query of your choice: "Food" but not "Drink"**

  Query: city_text:food -city_text:drink
  Results found: 80
  Adamstown, Pitcairn Island, Pitcairn Islands (0.17206141)
  Lilongwe, Malawi (0.10138816)
  Gustavia, Saint Barthélemy, Saint Barthélemy (0.100369155)
  Porto-Novo, Benin (0.08603071)
  Stepanakert, Nagorno-Karabakh Republic (0.08111052)
  Pyongyang, North Korea (0.07169225)
  Pago Pago, American Samoa (0.07169225)
  Suva, Fiji (0.07169225)
  Bujumbura, Burundi (0.07169225)
  Conakry, Guinea (0.07169225)
  Palikir, Federated States of Micronesia (0.07169225)
  Jakarta, Indonesia (0.070971705)
  Khartoum, Sudan (0.070971705)
  Brasília, Brazil (0.064123504)

Bangui, Central African Republic (0.064123504)
San José, Costa Rica, Costa Rica (0.06083289)
Basseterre, Saint Kitts and Nevis (0.057353802)
Singapore, Singapore (0.052682832)
Tórshavn, Faroe Islands (0.050184578)
Ouagadougou, Burkina Faso (0.050184578)
San Salvador, El Salvador (0.048092626)
Berlin, Germany (0.048092626)
Nairobi, Kenya (0.048092626)
Port of Spain, Trinidad and Tobago (0.043461114)
Rabat, Morocco (0.043015353)
Hargeisa, Somaliland (0.043015353)
Port Moresby, Papua New Guinea (0.043015353)
Windhoek, Namibia (0.043015353)
Kinshasa, Democratic Republic of the Congo (0.043015353)
Minsk, Belarus (0.04055526)
Phnom Penh, Cambodia (0.04055526)
Abu Dhabi, United Arab Emirates (0.04055526)
Baghdad, Iraq (0.04055526)
Tokyo, Japan (0.04055526)
Hong Kong, Hong Kong (0.04007719)
Bangkok, Thailand (0.037252385)
Port Louis, Mauritius (0.035846125)
Astana, Kazakhstan (0.035846125)
Djibouti (city), Djibouti (0.035846125)
Freetown, Sierra Leone (0.035846125)
Sana'a, Yemen (0.035846125)
Charlotte Amalie, United States Virgin Islands, United States Virgin Islands (0.035846125)
Tashkent, Uzbekistan (0.035846125)
Caracas, Venezuela (0.035846125)
Taipei, Taiwan (0.035485853)
Ulaanbaatar, Mongolia (0.035485853)
Cape Town, South Africa (0.035485853)
Dhaka, Bangladesh (0.035485853)
Stockholm, Sweden (0.035485853)
Prague, Czech Republic (0.035485853)
Havana, Cuba (0.030416446)
Dublin, Republic of Ireland (0.030416446)
Beijing, China (0.030416446)
Manila, Philippines (0.030416446)
Accra, Ghana (0.028676901)
Tripoli, Libya (0.028676901)
Nicosia, Cyprus (0.028676901)
Gaborone, Botswana (0.028676901)
Nicosia, Northern Cyprus (0.028676901)

Riga, Latvia (0.028676901)
Hanoi, Vietnam (0.028676901)
London, United Kingdom (0.02534704)
Warsaw, Poland (0.02534704)
Sofia, Bulgaria (0.025092289)
Tehran, Iran (0.025092289)
Monaco, Monaco (0.025092289)
Bucharest, Romania (0.025092289)
Ankara, Turkey (0.025092289)
Bratislava, Slovakia (0.025092289)
San Juan, Puerto Rico, Puerto Rico (0.025092289)
Yerevan, Armenia (0.025092289)
Pristina, Kosovo (0.025092289)
Beirut, Lebanon (0.025092289)
Thimphu, Bhutan (0.021507677)
Tegucigalpa, Honduras (0.021507677)
Canberra, Australia (0.021507677)
Kiev, Ukraine (0.021507677)
Ljubljana, Slovenia (0.021507677)
Moscow, Russia (0.017923063)
Buenos Aires, Argentina (0.017923063)

5. Latent Dirichlet Allocation

Code:

- JsonFreq.java
- hw3_lda_v2.py

Inputs:

- vectorFreq.json (for python)

Outputs:

- vectorFreq.json (from java)
- td_matrix_sample.csv
- top10_topics_words.csv
- extra python output
    - topic_word_distribution.png
    - doc_topic_distribution.png
    - top_doc_topics.csv

Extract the term-document matrix from your index for the set of country pages

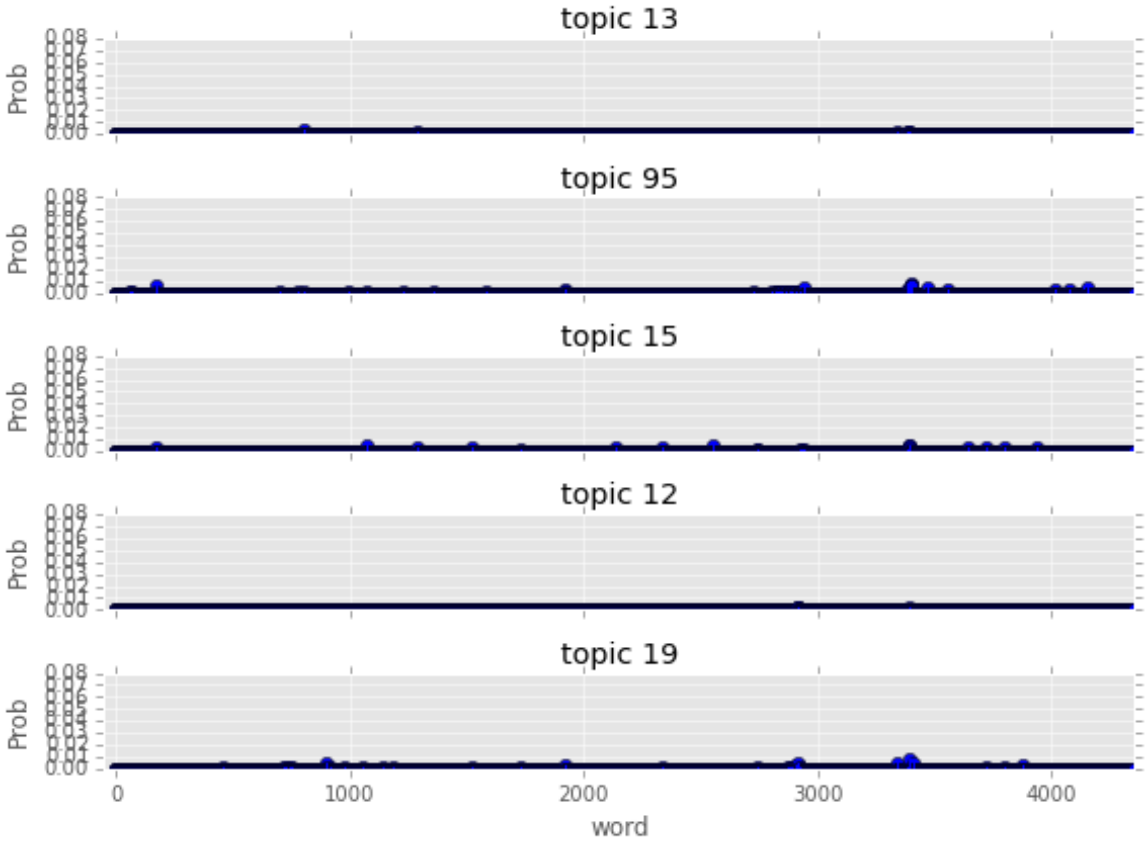
This was done in java using Lucene's getTermVector. The result was saved in json format (frequency of words for each document using the country field) and loaded in python for lda.

The term-document matrix was created in python from the json format. Since the matrix is too big, a sample of the matrix was outputted.

Top 10 topics by counts (topic assignments), and for each topic the top 10 words:

topic	counts	top 10 words
Topic 13	240712	which also most other year on ha peopl it from
Topic 95	154569	from were it have state nation popul which includ between
Topic 15	139965	govern ha had member elect mai be us also council
Topic 12	134973	countri from forc war militari region govern citi state had
Topic 19	104941	year rate ha per us econom level trade industri centuri
Topic 79	86987	countri world ha region largest million develop economi major rank
Topic 5	31336	island from were ship which territori archipelago mi fish sea
Topic 48	28552	countri african presid africa elect nation coloni govern group west
Topic 58	27289	centuri most art music film modern power empir literatur architectur
Topic 67	25985	speci south forest land coast mountain mani north found fish


Document 92
Document 139
Document 186
Document 197
Document 239
Prob
Topic


topic 13
topic 95
topic 15
topic 12
topic 19
Prob
word

References

Differences between BooleanQuery and QueryParser
http://www.gossamer-threads.com/lists/lucene/java-user/144374

Boolean Query
http://www.avajava.com/tutorials/lessons/how-do-i-combine-queries-with-a-boolean-query.html

Phrase Query
http://www.avajava.com/tutorials/lessons/how-do-i-query-for-words-near-each-other-with-a-phrase-query.html

Query Parser
http://lucene.apache.org/core/2_9_4/queryparsersyntax.html#+

Lucent tutorial
http://www.lucenetutorial.com/lucene-in-5-minutes.html
http://oak.cs.ucla.edu/cs144/projects/lucene/

Beautiful Soup
http://www.crummy.com/software/BeautifulSoup/bs3/documentation.html

Fuzzy Query
http://www.tutorialspoint.com/lucene/lucene_fuzzyquery.htm

TermVector
http://stackoverflow.com/questions/14363377/in-lucene-4-indexreader-gettermvectordocid-fieldname-returns-null-for-every
http://stackoverflow.com/questions/12098083/term-vector-frequency-in-lucene-4-0
http://stackoverflow.com/questions/14363377/in-lucene-4-indexreader-gettermvectordocid-fieldname-returns-null-for-every
http://stackoverflow.com/questions/8938960/how-to-get-document-ids-for-document-term-vector-in-lucene
http://stackoverflow.com/questions/11945728/how-to-use-termvector-lucene-4-0