COMP5349 Assignment 1 Report

THE UNIVERSITY OF SYDNEY

Sid: 510089546

Zhuoxi kuang

1 INTRODUCTION

The project aims to extract keywords from the dataset based on spark API using two different methods. The dataset contains 510 legal contracts and 13,000+ clauses that belong to three different categories, which are“Governing law”, “Change of Control” and “Anti-assignment”.

2 Method1

The first method is to regard each clause in the dataset as a document and each category as a corpus.

Stage 1: Convert the original RDD into the RDD where each element is a tuple like (filename1, clause1) using flatMap, map, and filter operations because a file may contain more than one clause. As shown below, firstly use filter operation to remove nan values(for the other two corpora, it is None values), then flatmap them to the tuple format using flatMap operation, next using map transformation to remove extra spaces and generate the new RDD.

```
#  filter nan values and  spaces
method1_gov_rdd1 = gov_rdd.filter(lambda x: x[1] != 'nan').flatMap(method1_extract).map(lambda x: (x[0],x[1].strip()))
method1_gov_rdd1.take(5)
```

```
[('CybergyHoldingsInc_20140520_10-Q_EX-10.27_8605784_EX-10.27_Affiliate Agreement.pdf1',
  'this agreement is accepted by company in the state of nevada and shall be governed by and construed in accordance with
 ('EuromediaHoldingsCorp_20070215_10SB12G_EX-10.B(01)_525118_EX-10.B(01)_Content License Agreement.pdf1',
  'this agreement shall be governed by laws of the province of ontario and the federal laws of canada applicable therein.'
 ('EuromediaHoldingsCorp_20070215_10SB12G_EX-10.B(01)_525118_EX-10.B(01)_Content License Agreement.pdf2',
  'this agreement is subject to all laws, regulations, license conditions and decisions of the canadian radio-television a
 ('FulucaiProductionsLtd_20131223_10-Q_EX-10.9_8368347_EX-10.9_Content License Agreement.pdf1',
  'all questions with respect to the construction of this agreement, and the rights and liabilities of the parties hereto,
 ('GopageCorp_20140221_10-K_EX-10.1_8432966_EX-10.1_Content License Agreement.pdf1',
  'this agreement shall be governed by and construed in accordance with the internal laws of the state of nevada without g
```

Stage 2: Candidate phrases identification.

Firstly, removing punctuation and stopwords using map transformation. Secondly, remove empty values, and extra spaces and convert them into list format for each candidate phrase of each document using map operation as well. In this way, as shown in the second graph below, each element of the RDD is the tuple whose first position is the filename+index, and the second position is the list of list candidate phrases.

```
method1_gov_rdd_list = method1_gov_rdd2.map(remove_space_list)
method1_gov_rdd_list.take(1)
```

```
[('CybergyHoldingsInc_20140520_10-Q_EX-10.27_8605784_EX-10.27_Affiliate Agreement.pdf1',
  [['agreement'],
   ['accepted'],
   ['company'],
   ['state'],
   ['nevada'],
   ['shall'],
   ['governed'],
   ['construed'],
   ['accordance'],
   ['laws', 'thereof'],
   ['laws', 'shall', 'prevail'],
   ['event'],
   ['conflict']])]
```

Stage 3: Word Score Calculation

First of all, map the above RDD to the following format ((filename, single word), candidate_phrase, 1, length of candidate_phrase) using the flatMap operation. More specifically, as shown below, the first tuple in the entire tuple refers to one word of each document(clause), the second is the list of the word's candidate phrase, the third number is 1 which represents the frequency of every single word for the candidate phrase, the fourth number is the length of candidate phrases since deg(w) = fre(w) + co-occurrence. In detail, fre(w) is equal to 1, and co-occurrence equals the length(candidate_phrase)-1. In this way, deg(w) = length of candidate phrase.

```
method1_gov_rdd2 = method1_gov_rdd_list.flatMap(ls_fre_deg)
method1_gov_rdd2.take(3)
```

```
[(('CybergyHoldingsInc_20140520_10-Q_EX-10.27_8605784_EX-10.27_Affiliate Agreement.pdf1',
   'agreement'),
  ['agreement'],
  1,
  1),
 (('CybergyHoldingsInc_20140520_10-Q_EX-10.27_8605784_EX-10.27_Affiliate Agreement.pdf1',
   'accepted'),
  ['accepted'],
  1,
  1),
 (('CybergyHoldingsInc_20140520_10-Q_EX-10.27_8605784_EX-10.27_Affiliate Agreement.pdf1',
   'company'),
  ['company'],
  1,
  1)]
```

The second step is to calculate the frequency for each word. Firstly selecting the first and third element using map transformation, then using the reduceByKey operation to group by the same word of each document and aggregate the frequency.

The third step is to calculate the degree for each word. Firstly selecting the first and fourth element using map transformation, then using the reduceByKey operation to group by the same word of each document and aggregate the degree.

After that, calculate the ratio(score) of fre(w) and deg(w) for each word by joining the frequency RDD and degree RDD using join and map operations.

Stage 4: Calculate the candidate phrase score.

Using groupByKey and map operations to group the same document(clause), as shown below, each element of RDD is a tuple. The first is the document name(filename+ index) and the second is the tuple containing the candidate phrase and its score for each document.

```
[22] # using groupByKey() to group the candidate phrases with the same filename
     method1_gov_phrase_score = method1_gov_rdd6.groupByKey().map(lambda x: (x[0], list(x[1])))
     method1_gov_phrase_score.take(1)
```

```
[('MusclepharmCorp_20170208_10-KA_EX-10.38_9893581_EX-10.38_Co-Branding Agreement.pdf1',
  [(('executed',), 1.0),
   (('state',), 2.0),
   (('agreement',), 1.0),
   (('performance', 'shall'), 4.0),
   (('laws',), 1.0),
   (('construed',), 1.0),
   (('enforced',), 1.0),
   (('los', 'angeles', 'county'), 9.0),
   (('california',), 2.0),
   (('interpretation', 'validity'), 4.0),
   (('delivered',), 1.0),
   (('accordance',), 1.0)])]
```

Stage 5: Keyword Identification

The first step is to map operation is used to sort the candidate phrase based on the score and add two numbers. As shown below, adding 1(means edf) for the top four candidates for each document and other candidate is 0. For the last number 1, which represents rdf.

The second step is to flatmap the candidate phrase to the tuple like the format(candidate_phrase, keyword_number) using flatMap operation, then using reduceByKey operation to group by the same candidate phrase and aggregate edf, and the final is to use filter operation to remove the number equal to 0 because 0 represents that the candidate is not the keyword.

```
method1_gov_rdd8 = method1_gov_phrase_score.map(method1_candidate_phrase)
method1_gov_rdd8.take(1)
# (candidate_phrase, ratio_score, keyword_score, phrase_score)
```

```
[('MusclepharmCorp_20170208_10-KA_EX-10.38_9893581_EX-10.38_Co-Branding Agreement.pdf1',
  [(('los', 'angeles', 'county'), 9.0, 1, 1),
   (('performance', 'shall'), 4.0, 1, 1),
   (('interpretation', 'validity'), 4.0, 1, 1),
   (('state',), 2.0, 1, 1),
   (('california',), 2.0, 0, 1),
   (('executed',), 1.0, 0, 1),
   (('agreement',), 1.0, 0, 1),
   (('laws',), 1.0, 0, 1),
   (('construed',), 1.0, 0, 1),
   (('enforced',), 1.0, 0, 1),
   (('delivered',), 1.0, 0, 1),
   (('accordance',), 1.0, 0, 1)])]
```

```
# Filter the second element of the tuple being 0, because 0 means it is not the keyword
method1_gov_keyword_edf = method1_gov_rdd8.flatMap(extract_edf).reduceByKey(lambda a,b: a+b).filter(lambda x: x[1] != 0).
method1_gov_keyword_edf.take(10)
```

```
[(('agreement', 'shall'), 216),
 (('laws',), 126),
 (('governed',), 54),
 (('state',), 53),
 (('accordance',), 40),
 (('construed',), 36),
 (('agreement',), 33),
 (('law', 'principles'), 26),
 (('new', 'york'), 25),
 (('laws', 'principles'), 23)]
```

Stage 6: The edf, rdf, and ess calculation for each candidate phrase

The first thing is to calculate rdf and edf. Edf has been calculated in the last stage. Rdf is calculated using map and reduceByKey operations.

The second thing is to join rdf RDD and edf RRD using the join operation, then calculate the essentiality based on the calculation algorithm(ess = edf/rdf * edf) using map operation.

The last step is to remove the number of words for the candidate phrase greater than 4 using filter transformation, then using map operation to convert the candidate phrase of the tuple format, next sort these candidates according to the essentiality using sortBy operation.

Finally, there are results for three corpora and the number in the tuple represents (edf, rdf, ess).

Governing law

```
[('agreement shall', (216, 250, 186.624)),
 ('laws', (126, 387, 41.02325581395349)),
 ('new york', (25, 28, 22.321428571428573)),
 ('new york without regard', (20, 20, 20.0)),
 ('law principles', (26, 36, 18.77777777777778)),
 ('laws principles', (23, 31, 17.06451612903226)),
 ('law rules', (16, 19, 13.473684210526315)),
 ('law provisions', (16, 19, 13.473684210526315)),
 ('substantive laws', (16, 21, 12.19047619047619)),
 ('new york applicable', (12, 12, 12.0)),
 ('california without regard', (12, 12, 12.0)),
 ('delaware without regard', (12, 12, 12.0)),
 ('laws provisions', (13, 17, 9.941176470588236)),
 ('performed entirely within', (9, 9, 9.0)),
 ('parties hereto shall', (9, 9, 9.0)),
 ('new york without reference', (9, 9, 9.0)),
 ('state', (53, 329, 8.537993920972644)),
 ('governed', (54, 358, 8.145251396648044)),
 ('laws principles thereof', (7, 7, 7.0)),
 ('validity construction', (7, 7, 7.0))]
```

Change of control

```
[('prior written consent', (11, 16, 7.5625)),
 ('ownership transfer', (4, 4, 4.0)),
 ('agreement upon written notice', (3, 3, 3.0)),
 ('terminated upon', (3, 3, 3.0)),
 ('control shall', (3, 3, 3.0)),
 ('excitehome named competitor', (3, 3, 3.0)),
 ('licensor may terminate', (3, 3, 3.0)),
 ('would result', (3, 4, 2.25)),
 ('agreement shall terminate', (3, 4, 2.25)),
 ('made third party beneficiaries', (2, 2, 2.0)),
 ('party may terminate', (2, 2, 2.0)),
 ('providing written notice', (2, 2, 2.0)),
 ('control means', (2, 2, 2.0)),
 ('control shall occur', (2, 2, 2.0)),
 ('section 16av due', (2, 2, 2.0)),
 ('shall provide written notice', (2, 2, 2.0)),
 ('authority granted hereunder shall', (2, 2, 2.0)),
 ('agreement shall immediately cease', (2, 2, 2.0)),
 ('sixty 60 days', (2, 2, 2.0)),
 ('providing prior written notice', (2, 2, 2.0))]
```

Anti-assignment

```
[('prior written consent', (206, 242, 175.35537190082644)),
 ('neither party may assign', (43, 44, 42.02272727272727)),
 ('either party without', (30, 40, 22.5)),
 ('either party may assign', (22, 23, 21.043478260869566)),
 ('agreement without', (40, 80, 20.0)),
 ('unreasonably withheld conditioned', (15, 15, 15.0)),
 ('attempted assignment', (20, 27, 14.814814814814815)),
 ('agreement may', (26, 49, 13.795918367346939)),
 ('third party without', (16, 19, 13.473684210526315)),
 ('void', (36, 99, 13.090909090909092)),
 ('neither party shall assign', (12, 12, 12.0)),
 ('express written consent', (12, 12, 12.0)),
 ('obligations hereunder without', (16, 25, 10.24)),
 ('prior written approval', (10, 10, 10.0)),
 ('party may assign', (13, 17, 9.941176470588236)),
 ('unreasonably withheld', (23, 54, 9.796296296296296)),
 ('agreement shall', (20, 44, 9.090909090909092)),
 ('neither party shall', (10, 11, 9.090909090909092)),
 ('express prior written consent', (9, 9, 9.0)),
 ('consent shall', (19, 42, 8.595238095238095))]
```

3 Method2

The second method is to treat each category as a document and extract the top 20 keywords from documents based on frequency and degree. This algorithm is simpler than the method1. The following contains three stages.

Stage 1: Candidate Phrase Identification

In the first step, removing (page n) and converting to lowercase using map operation, then using filer operation to remove nan values, as well as using map transformation to remove extra spaces.

```
# filter nan and delete space
gov_rdd1 = gov_rdd.map(preprocessing).filter(lambda x: x != 'nan').map(lambda x: x.strip())
gov_rdd1.take(5)
```

```
['this agreement is accepted by company in the state of nevada and shall be governed by and constr
 'this agreement shall be governed by laws of the province of ontario and the federal laws of cana
 'all questions with respect to the construction of this agreement, and the rights and liabilities
 'this agreement shall be governed by and construed in accordance with the internal laws of the st
 'this agreement shall be governed by and construed in all respects in accordance with the laws of
```

The second step is to remove stopwords and punctuation, then generate the candidate phrases using map operation.

```
gov_list_candidate = gov_rdd1.flatMap(ExtractContentWord).filter(lambda x: x != ' ').map(lambda x: x.strip()).filter(lambda x: x!= '')
gov_list_candidate.take(10)
```

```
['agreement',
 'accepted',
 'company',
 'state',
 'nevada',
 'shall',
 'governed',
 'construed',
 'accordance',
 'laws thereof']
```

Stage 2: Word Score Calculation

The first thing is to generate candidate phrases of list format using map operation, then using flatMap transformation to convert them into the following format (single word, candidate_phrase, fre(w), deg(w)), this idea is the same as the method1, the third number of each tuple is 1 which means the frequency of each word for the one candidate phrase and the fourth number is the length of candidate phrases since deg(w) = fre(w) + co-occurrence. In detail, fre(w) is equal to 1, and co-occurrence equals the length(candidate_phrase)-1. In this way, deg(w) = length of candidate phrase.

```
# (word, candidate_phrase, fre(w), deg(w))
gov_list_score = gov_list_candidate.map(Tolist).flatMap(ExtractEachWord)
gov_list_score.take(10)
```

```
[('agreement', ['agreement'], 1, 1),
 ('accepted', ['accepted'], 1, 1),
 ('company', ['company'], 1, 1),
 ('state', ['state'], 1, 1),
 ('nevada', ['nevada'], 1, 1),
 ('shall', ['shall'], 1, 1),
 ('governed', ['governed'], 1, 1),
 ('construed', ['construed'], 1, 1),
 ('accordance', ['accordance'], 1, 1),
 ('laws', ['laws', 'thereof'], 1, 2)]
```

The next is to calculate the frequency for each word. Map transformation is used to generate tuples like(single word, frequency), then using reduceByKey to group the same word and aggregate the frequency. There are some empty values according to the collect() action, so using filter operation to remove empty values.

```
gov_fre = gov_list_score.map(lambda x: (x[0], x[2])).reduceByKey(lambda a,b: a+b).filter(lambda x: x[0] != '').sortBy(lambda x: x[1], ascending = False)
gov_fre.take(10)
```

```
[('laws', 622),
 ('agreement', 478),
 ('state', 423),
 ('shall', 397),
```

The third is to calculate the degree of each word which is almost the same as calculating frequency. Map, reduceByKey, filter, and sortBy are also used in the code.

```
gov_deg =  gov_list_score.map(lambda x:(x[0], x[3])).reduceByKey(lambda a,b: a+b).filter(lambda x: x[0] != '').sortBy(lambda x: x[1], ascending = False)
gov_deg.take(10)
```

```
[('shall', 904),
 ('laws', 831),
 ('agreement', 806),
 ('without', 704),
```

After the above steps, as shown below, use the join operation to combine frequency and degree, then use sortBy to sort them in descending order.

```
gov_ratio = gov_fre.join(gov_deg).map(ratio).sortBy(lambda x: x[1], ascending = False)
gov_ratio.take(10)
```

```
[('thereunder', 12.5),
 ('whole', 11.0),
 ('provider', 10.0),
 ('necessarily', 10.0),
```

Stage 3: Candidate Keyword Score Calculation

After calculating the single word, it is important to calculate the score of candidate phrases.

In the first step, as shown below use the join operation to generate scores of candidate phrases.

The following step is to convert the list of candidate phrases into string format using map operation, then using the reduceByKey operation to group the phrase and aggregate the score. Finally, sortBy transformation is applied to sort the candidate in descending order and extract the top 20 keywords.

```
gov_join = gov_distinct.join(gov_ratio).map(lambda x: (x[1][0], x[1][1]))
gov_join.take(10)
```

```
[(['accepted'], 1.0),
 (['state'], 1.1134751773049645),
 (['new', 'york', 'state'], 1.1134751773049645),
 (['united', 'state'], 1.1134751773049645),
```

```
def listTostr(record):
  record = (' '.join(record[0]), record[1])
  return record

gov_result = gov_join.map(listTostr).reduceByKey(lambda a,b: a+b).sortBy(lambda x: x[1], ascending = False)
gov_result.take(20)
```

```
[('sections 1051 et seq', 18.4),
 ('lj 1050 et seq', 16.4),
 ('1 covering competition following', 15.75),
 ('laws principles thereunder', 15.677392172081161),
 ('met independently without reference', 14.203888888888889),
 ('either party herein initiate', 13.75),
 ('without giving effect to', 13.181520467836258),
 ('1980 united nations convention', 12.855821371610846),
 ('performed entirely within pennsylvania', 12.686609686609687),
 ('maryland without giving effect', 12.431520467836258),
 ('1051 et seq', 12.399999999999999),
 ('new york without reference', 12.269493491167902),
 ('intellectual property right applies', 12.238095238095237),
 ('new york without recourse', 12.194493491167902),
 ('pennsylvania without giving effect', 12.181520467836258),
 ('new york without regard', 12.02355332022773),
 ('california without giving effect', 11.998421876286962),
 ('agreement shall become valid', 11.963270554261564),
 ('massachusetts without giving effect', 11.895806182121971),
 ('parties hereto expressly attorns', 11.867417840375587)]
```

Finally, there are results for three documents

Governing law

```
[('sections 1051 et seq', 18.4),
 ('lj 1050 et seq', 16.4),
 ('1 covering competition following', 15.75),
 ('laws principles thereunder', 15.677392172081161),
 ('met independently without reference', 14.203888888888889),
 ('either party herein initiate', 13.75),
 ('without giving effect to', 13.181520467836258),
 ('1980 united nations convention', 12.855821371610846),
 ('performed entirely within pennsylvania', 12.686609686609687),
 ('maryland without giving effect', 12.431520467836258),
 ('1051 et seq', 12.399999999999999),
 ('new york without reference', 12.269493491167902),
 ('intellectual property right applies', 12.238095238095237),
 ('new york without recourse', 12.194493491167902),
 ('pennsylvania without giving effect', 12.181520467836258),
 ('new york without regard', 12.02355332022773),
 ('california without giving effect', 11.998421876286962),
 ('agreement shall become valid', 11.963270554261564),
 ('massachusetts without giving effect', 11.895806182121971),
 ('parties hereto expressly attorns', 11.867417840375587)]
```

Change of control

```
[('home named competitor', 14.920634920634921),
 ('five thousand dollars', 14.857142857142858),
 ('vs key leadership position', 14.5),
 ('providing ebix written notice', 14.015575974899148),
 ('the òroyalty offset periodó', 14.0),
 ('upon sending written notice', 13.948909308232482),
 ('reasonable detail based upon', 13.783333333333333),
 ('without thereby becoming liable', 13.671186440677966),
 ('party representing fifty percent', 13.438871473354233),
 ('authority granted hereunder shall', 13.394047619047619),
 ('providing prior written notice', 13.172438719997187),
 ('upon written prior notice', 13.10577205333052),
 ('admit additional general partners', 13.05),
 ('party shall remain liable', 13.032594417077176),
 ('janssenõs confidential information hereunder', 13.019047619047619),
 ('shall provide written notice', 12.97390930823248),
 ('golf instruction related products', 12.971428571428572),
 ('section 14 ð òterminationó', 12.889830508474576),
 ('by providing written notice', 12.848909308232482),
 ('advertising agency representing tda', 12.8)]
```

Anti-assignment

```
[('distributor becomes insolvent', 20.1),
 ('transporterõs ferc gas tariff', 15.5),
 ('cause forty niners sc', 15.022222222222222),
 ('assigning party holds', 14.988086704215737),
 ('express prior written authorization', 14.94640818980881),
 ('s prior written approval', 14.875798244198865),
 ('restrictions set forth herein', 14.870471014492754),
 ('indirect loss thus caused', 14.444444444444445),
 ('s prior written consent', 14.434506977326205),
 ('prior express written approval', 14.355499098899719),
 ('express prior written approval', 14.355499098899719),
 ('forty niners sc without', 14.301052631578948),
 ('nfla prior written consent', 14.170618088437315),
 ('party shall remain liable', 14.122704146698347),
 ('third party either becomes', 14.055865477166922),
 ('reynolds group holdings limited', 13.958333333333332),
 ('express prior written consent', 13.914207832027056),
 ('prior express written consent', 13.914207832027056),
 ('expressly set forth herein', 13.90893255295429),
 ('assignor shall remain liable', 13.79176029962547)]
```