Distributional Thesaurus Part 2

April 28, 2018

1 JoBimText: Creating Distributional Thesaurus

PART 2 : Similarity measures, Pruning, Aggregation and Sorting

1.0.1 Import required libraries

```
In [1]: import sqlite3
```

1.0.2 Open Database `thesaurus.db`

```
In [2]: conn = sqlite3.connect('thesaurus.db')
        c = conn.cursor()
```

1.0.3 Create table with LE-CF and corresponding Similarity measure

We 'll use PMI here

```
In [3]: c.execute("DROP TABLE IF EXISTS sm;")

        # This table's has ids corresponding to lecf's
        c.execute('''CREATE TABLE sm
                     (id INTEGER PRIMARY KEY,
                     pmi FLOAT NOT NULL)''')

        conn.commit()
```

1.0.4 Calculating PMI from tables LE, CF and LE-CF

- `pmi = le-cf.count/(le.count * cf.count)`
- For each row of LE-CF table, compute `PMI` and insert it into `sm`

```
In [4]: c.execute('''
        INSERT INTO sm (pmi, id)
        SELECT pmi, id FROM
        (SELECT lecfc * 1.0/(lec*cfc) as pmi,
        lecfid as id from
            (SELECT lecf.id,
                    lecf.le as leid,
                    lecf.cf as cfid,
                    le.count as lec,
                    le.name as lename,
                    cf.count as cfc,
                    lecf.count as lecfc,
                    lecf.id as lecfid
                from lecf, le, cf
                where leid == le.id and cfid == cf.id)
                )''')
        conn.commit()
```

1.0.5 The next step is Pruning

- Remove from `sm` values with `PMI` less than some limit
- Since the corpus is very small, we'll also set the pruning limit to be very small
- `pruning limit = 1.0/32768`

1.0.6 Create Pruned Similarity Measure `psm` Table:

```
In [48]: c.execute("drop table if exists psm")
         c.execute("drop table if exists simcount")

         c.execute('''
         CREATE TABLE psm (
         id INTEGER PRIMARY KEY NOT NULL,
         pmi FLOAT NOT NULL
         )''')

         c.execute('''
         INSERT INTO psm (pmi, id)
         SELECT pmi, id from sm where pmi > 1.0/(1024)
         ''')

         c.execute('''
         CREATE TABLE simcount (
         le1 INTEGER NOT NULL,
         le2 INTEGER NOT NULL,
         count INT NOT NULL
         )''')

         conn.commit()
```

1.0.7 Creating the Similarity count Table

- This step integrates the aggregate per feature table, it is not possible to have such a table in RDBMS.
- So from the pruned data we directly compute similarity counts with the number of Context-Features each Language-Element shares with every other.

```
In [49]: c.execute('''
         INSERT INTO simcount (le1, le2, count)
         SELECT E.id, F.id, count(*) FROM
         (SELECT A.id, B.id,
                 C.le as le1, C.cf as cf, C.id,
                 D.le as le2, D.cf, D.id
             FROM psm AS A,
                  psm AS B,
                  lecf AS C,
                  lecf AS D
             WHERE A.id == C.id AND
                   B.id == D.id AND
                   C.cf == D.cf AND
                   C.le <> D.le) AS sim, le AS E, le AS F
                                 where SIM.le1 == E.id AND
                                       SIM.le2 == F.id
                                   GROUP BY E.name, F.name
         ''')
         conn.commit()
```

1.0.8 Querying the thesaurus:

- Now we just query the DB for synonyms of the given word.
- The results are sorted according to similarity counts.

```
In [61]: def query_Thesaurus(word):
             for i in c.execute('''
             select a.name, b.name, c.count from
             simcount as c, le as a, le as b
             where a.id == c.le1 and
                     b.id == c.le2 and
                     c.count > 2 and
                     a.name == '{}' order by c.count desc;'''.format(word)).fetchall():
                 print(i)
```

1.1 Some intresting Results:

```
In [62]: query_Thesaurus("many")
```

```
('many', 'Some', 6)
('many', 'Many', 5)
('many', 'other', 5)
('many', 'two', 5)
('many', 'These', 4)
('many', 'different', 4)
('many', 'numerous', 4)
('many', 'Several', 3)
('many', 'any', 3)
('many', 'four', 3)
```

```
('many', 'most', 3)
('many', 'several', 3)
('many', 'these', 3)
('many', 'various', 3)
```

```
In [63]: query_Thesaurus("livestock")
```

```
('livestock', 'sheep', 6)
('livestock', 'were', 4)
('livestock', 'bison', 3)
('livestock', 'him', 3)
('livestock', 'two', 3)
```

```
In [64]: query_Thesaurus("several")
```

```
('several', 'Most', 4)
('several', 'different', 4)
('several', 'three', 4)
('several', 'two', 4)
('several', 'Both', 3)
('several', 'Many', 3)
('several', 'Several', 3)
('several', 'both', 3)
('several', 'five', 3)
('several', 'many', 3)
('several', 'other', 3)
('several', 'some', 3)
```

```
In [65]: query_Thesaurus("mm")
```

```
('mm', 'cm', 7)
('mm', 'inches', 6)
('mm', 'centimeters', 5)
('mm', 'grows', 5)
('mm', 'approximately', 4)
('mm', 'grow', 4)
('mm', 'mm', 4)
('mm', 'range', 4)
('mm', 'carapace', 3)
('mm', 'feet', 3)
('mm', 'measures', 3)
('mm', 'millimeter', 3)
('mm', 'reach', 3)
('mm', 'reached', 3)
('mm', 'reaches', 3)
('mm', 'weighs', 3)
```

```
In [66]: query_Thesaurus("four")

('four', 'three', 5)
('four', 'two', 4)
('four', 'All', 3)
('four', 'many', 3)
```