CS421: Assignment #3 **Srikanth Maganti**

3.1 Word Similarity

In the given python file, run the similarity(uncomment) function

The words which I have chosen are Chair, Mobile, School, Numbers, Bottle

Similar words for Chair are:

```
similar words for chair
[(u'chairs', 0.7871319055557251), (u'Chair', 0.7488438487052917), (u'chairperson', 0.6301862597465515), (u'chairwoman',
0.6271359920501709), (u'chairman', 0.6161440014839172), (u'Vice_Chair', 0.5646666288375854), (u'Co_Chair', 0.54698097705
84106), (u'chairing', 0.5431761741638184), (u'Chairs', 0.5395591259002686), (u'cochair', 0.5284922122955322)]
```

Most of the words obtained here are the words which are same, If we do stemming for all these words, the obtained word is same as the given word.

Similar words for Mobile are:

```
similar words for mobile
[(u'mobile_phones', 0.7054649591445923), (u'Mobile', 0.6691667437553406), (u'smartphone', 0.6600653529167175), (u'smartp
hones', 0.6404396295547485), (u'handsets', 0.640427827835083), (u'handset', 0.6366176605224609), (u'mobiles', 0.63570129
87136841), (u'Impoverished_NKorea_gets', 0.6308283805847168), (u'Jason_Fulmines_director', 0.6140598654747009), (u'Cyfro
wa_PTC', 0.6105130314826965)]
```

In the obtained words Impoverished_nkorea obtained because of the context on mobile phones and Jason_fulmines_director also obtained because of the words related with mobile occurred a lot in that context.

Similar words for School are:

```
similar words for school
[(u'elementary', 0.7868632078170776), (u'schools', 0.7411909103393555), (u'shool', 0.6692329049110413), (u'elementary_sc
hools', 0.6597154140472412), (u'kindergarten', 0.6529810428619385), (u'eighth_grade', 0.6488089561462402), (u'School', 0
.6477997899055481), (u'teacher', 0.63824063539505), (u'students', 0.6301523447036743), (u'classroom', 0.6281620860099792
)]
```

In the above similar words, all the words were almost subset of the school or the things which we use quiet frequent with the context of school.

Similar words for Numbers are:

```
similar words for numbers
[(u'Numbers', 0.6641744375228882), (u'number', 0.6367050409317017), (u'percentages', 0.6088082194328308), (u'figures', 0
.5667737722396851), (u'statistics', 0.5654458999633789), (u'totals', 0.5642670392990112), (u'stats', 0.5475853681564331)
, (u'Loto_Qu\xe9bec_Banco', 0.5058497786521912), (u'num_bers', 0.5040186643600464), (u'Loto_Qu\xe9bec_Extra', 0.5024887
19425964)]
```

Most of the words obtained here are similar words that are quiet frequently used with the numbers context except “u'Loto_Qu\xe9bec_Banco”. It’s because of textbook name or it is some kind of formula.

Similar words for bottle are:

```
similar words for bottle
[(u'bottles', 0.8106202483177185), (u'jug', 0.6945109367370605), (u'###ml_bottle', 0.6480176448822021), (u'corked_bottle
', 0.638135552406311), (u'##cl_bottle', 0.6309486627578735), (u'carafe', 0.625845193862915), (u'brandy_snifter', 0.61814
96977806091), (u'gallon_jug', 0.6143518090248108), (u'decanter', 0.6073787808418274), (u'###ml_bottles', 0.6036193966865
54)]
```

All of the words obtained here are almost similar or the same stem word with how the bottle can be pronounced.

3.2 Antonyms

In the given python file, run the antonyms(uncomment) function

Chosen words are ascend, open, arrive, export, come

Similar words for ascend are:

```
ascend
[(u'ascended', 0.718950629234314), (u'ascending', 0.7094937562942505), (u'ascends', 0.6623241901397705), (u'climb',
 0.6534912586212158), (u'ascent', 0.6063636541366577), (u'descend', 0.5348040461540222), (u'ascension', 0.529545068
7408447), (u'clamber', 0.5279775857925415), (u'descending', 0.5162655115127563), (u'Ascending', 0.5003010034561157)
]
```

The obtained antonym from ascend is descend

Similar words for open are:

```
open
[(u'opened', 0.596660315990448), (u'closed', 0.5819659233093262), (u'opens', 0.5362014770507812), (u'opening', 0.51
58241391181946), (u'reopen', 0.5051860809326172), (u'closes', 0.46577221155166626), (u'close', 0.46365123987197876)
, (u'9am_5pm_Mon', 0.4396822452545166), (u'reopened', 0.433058500289917), (u'ajar', 0.4187529683113098)]
```

The obtained antonym for open is close

Similar words for arrive are:

```
arrive
[(u'arriving', 0.7587564587593079), (u'arrived', 0.6935393214225769), (u'arrives', 0.680608868598938), (u'depart',
0.6519900560379028), (u'Arriving', 0.5462287068367004), (u'arrival', 0.5358603000640869), (u'arive', 0.523791253566
7419), (u'Arrive', 0.5181905627250671), (u'departs', 0.4872068464756012), (u'disembark', 0.4833933115005493)]
```

The obtained antonym for arrive is depart

Similar words for export are:

```
export
[(u'exports', 0.8484325408935547), (u'exporting', 0.792079746723175), (u'Export', 0.7513514757156372), (u'imports',
 0.7324199080467224), (u'import', 0.7218946218490601), (u'exported', 0.7092705965042114), (u'exporters', 0.69927537
44125366), (u'exportation', 0.6791931390762329), (u'imported', 0.642731428146362 3), (u'Exports', 0.6384708285331726
)]
```

The obtained antonym for export is import

Similar words for come are:

```
come
[(u'coming', 0.7331234216690063), (u'go', 0.6604677438735962), (u'bring', 0.6255675554275513), (u'came', 0.61203360
55755615), (u'get', 0.5817449688911438), (u'brought', 0.5523512363433838), (u'gone', 0.5440618395805359), (u'happen
', 0.5239023566246033), (u'put', 0.5215755701065063), (u'comes', 0.5176242589950562)]
```

The obtained antonym for come is go

The embeddings might have this tendency because antonyms can be used in same sentence with different meaning of the whole sentence like for example.

“I will go” and “I will come” can be used in different contexts but the word can be used as a substitute for the other word. While, we actually calculate cosine similarity “go” and “come” can be used with the same words. So, while calculating cosine both of them mutually forms a high cosine value.

The above same condition applies for all the other antonyms also.

4.1 Existing Analogies

In the attached python file, run the existing_analogies (uncomment) function

Few screenshots of the output:

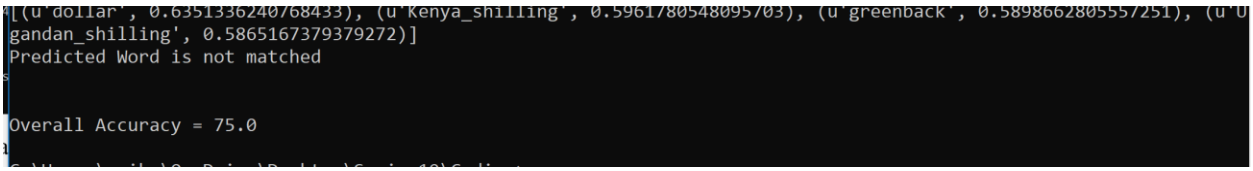
If the word didn’t get matched then it will print as not matched

```
['Tokyo', 'Japan', 'London', 'England']
[(u'Britain', 0.67340087890625), (u'UK', 0.6446690559387207), (u'Britian', 0.5892243385314941), (u'London', 0.58164
62635993958)]
Predicted Word is not matched


['Berlin', 'Germany', 'Madrid', 'Spain']
[(u'Spain', 0.7713854312896729), (u'Madrid', 0.7545335292816162), (u'Barcelona', 0.6232754588127136), (u'Real_Madri
d', 0.6106424331665039)]
```

If the word got matched then it will print as matched and word

```
['boy', 'girl', 'brother', 'sister']
[(u'brother', 0.8473406434059143), (u'sister', 0.8196011781692505), (u'cousin', 0.7446461915969849), (u'daughter',
0.7395669221878052)]


Predicted Word is sister Matched
```

Overall Accuracy is 75%



Interesting observations:

v_a = vector for a

v_b = vector for b

v_c = vector for c

v_d = vector for d

1.In most of the cases v_c is obtaining in the first similarity because when we do subtract v_a and v_b, if v_a and v_b are very near they will get cancelled and the resultant vector is v_c. So, v_c is obtained as first similarity.

2.Sometimes, v_b and v_c are near then the obtained vector may be v_a. so, we need to exclude all these v_a, v_b, v_c from the top 4 similarities.

3.After removing those, the obtained resultant 1st similarity is the v_new that is v_d.

4.when we calculate analogy for the currency, It is not able to predict dinar because of less usage of dinar with respect to Algeria. Instead of dinar, It is showing as currencies of Kenya and Uganda.

5.If v_a-v_b constitutes a very small amount and gets added to v_c. Then the obtained 1st similarity may be the word which v_d and the next similarity may be v_c.

4.2 New Analogies

In the attached python file, run the new_analogies(uncomment) function

New Analogies:

Patna Bihar Hyderabad Telangana:

India Delhi Columbia Bogota:

Indonesia rupiah Czech koruna:

husband wife patrolman patrolwoman:

Edmonton Alberta Toronto Ontario

For Patna Bihar Hyderabad Telangana

```
['Patna', 'Bihar', 'Hyderabad', 'Telangana']
[(u'Hyderabad', 0.742686927318573), (u'Bihar', 0.7246538996696472), (u'Andhra_Pradesh', 0.7225805521011353), (u'Mah
arashtra', 0.7135010361671448)]
Predicted Word is not matched
```

For India Delhi Columbia Bogota

```
['India', 'Delhi', 'Colombia', 'Bogota']
[(u'Colombia', 0.7623927593231201), (u'Bogot\xe1', 0.7047733068466187), (u'Bogota', 0.6952516436576843), (u'Colombi
an', 0.6916632056236267)]
Predicted Word is not matched
```

For Indonesia rupiah Czech koruna

```
['Indonesia', 'rupiah', 'Czech', 'koruna']
[(u'koruna', 0.7259178757667542), (u'zloty', 0.6826668977737427), (u'rupiah', 0.6746503114700317), (u'forint', 0.65
44080972671509)]


Predicted Word is koruna Matched
```

For husband wife patrolman patrolwoman

```
['husband', 'wife', 'Patrolman', 'Patrolwoman']
[(u'Patrolman', 0.9424651265144348), (u'Patrolmen', 0.7858415246009827), (u'Ptl', 0.6932757496833801), (u'patrolman
', 0.6422308683395386)]
Predicted Word is not matched
```

For Edmonton Alberta Toronto Ontario

```
['Edmonton', 'Alberta', 'Toronto', 'Ontario']
[(u'Alberta', 0.7619175910949707), (u'Ontario', 0.7228963375091553), (u'Toronto', 0.696752667427063), (u'Saskatchew
an', 0.6630114912986755)]


Predicted Word is Ontario Matched
```

In the example 1

It need to predict as Telangana but instead of Telangana it predicted as Andhra Pradesh as the 1st stereotype word. It has obtained because the previous capital for Andhra Pradesh is Hyderabad as recently the state got divided and named the new state as Telangana. It may be because the corpus is old one or in the corpus capital of Andhra Pradesh stated as Hyderabad.

In the example 2

Actually there is a word called Bogota in top 4 similar words but the 2nd word is Bogot\xe1 and the 3rd word is Bogota. So, actually the prediction is correct if we have done some stemming.

In the example 4

It clearly not able to predict patrolwoman in the top 4 similar words. Later, I checked it in top 10 similar words but it is missing.

5 Train new embeddings

In the attached python file, run train_new_embeddings(uncomment) function

Chosen words: take, care, able, ride, learn

Sample output screenshots:

Through model-2:

```
take
[('advantage', 0.9974828362464905), ('public', 0.9969087243080139), ('encourage', 0.9777777791023254), ('people', 0.9735
3196144104), ('stop', 0.9718875885009766), ('should', 0.964842677116394), ('strategies', 0.9636461138725281), ('efficien
t', 0.9634613394737244), ('create', 0.9619571566581726), ('devices', 0.9619486927986145)]



care
[('great', 0.9980047345161438), ('radiation', 0.9961645603179932), ('handled', 0.9937071800231934), ('destroys', 0.99316
5910243988), ('cells', 0.9872718453407288), ('material', 0.9820441603660583), ('with', 0.9769070744514465), ('packaging'
, 0.9657315015792847), ('becomes', 0.9635647535324097), ('contaminated', 0.9631073474884033)]



able
[('live', 0.9748848676681519), ('nothing', 0.9734159111976624), ('will', 0.9711107611656189), ('without', 0.967206358909
6069), ('pollute', 0.9444761276245117), ('thinking', 0.9408302903175354), ('savings', 0.9348658919334412), ('baby', 0.93
3980405330658), ('masses', 0.9337320923805237), ('turtles', 0.9315887689590454)]



ride
[('walk', 0.9979011416435242), ('bikes', 0.9977059364318848), ('consolidate', 0.9916572570800781), ('vegetables', 0.9907
454252243042), ('trips', 0.9871940612792969), ('grown', 0.9869441986083984), ('fruits', 0.9846140146255493), ('local', 0
.9815576076507568), ('reusable', 0.9795756936073303), ('items', 0.9734048843383789)]
```

```
learn
[('major', 0.9953861236572266), ('polluters', 0.9845947623252869), ('together', 0.9762715101242065), ('encourage', 0.971
689164638193), ('area', 0.9689171314239502), ('people', 0.9672800302505493), ('protest', 0.9672478437423706), ('stop',
0.965239405632019), ('everyday', 0.964461088180542), ('them', 0.963740885257721)]
```

Through model-1:

```
similar words through model1 take
[(u'taking', 0.7301175594329834), (u'took', 0.6650052070617676), (u'takes', 0.6607239842414856), (u'taken', 0.6450951099
395752), (u'Taking', 0.6425644755363464), (u'give', 0.5908799171447754), (u'Take', 0.5748313069343567), (u'totake', 0.54
44613695144653), (u'go', 0.521253764629364), (u'hold', 0.49019262194633484)]


similar words through model1 care
[(u'Care', 0.6274117827415466), (u'caring', 0.5817017555236816), (u'cared', 0.5646678805351257), (u'maintenance_Aurum_Mi
ne', 0.5575696229934692), (u'hospice_palliative', 0.5240262746810913), (u'Care_Nurse_Practitioners', 0.5190137028694153)
, (u'homecare', 0.5140517354011536), (u'health_care', 0.5037510991096497), (u'palliative_care', 0.5036935806274414), (u'
CareScout_helps', 0.5012749433517456)]


similar words through model1 able
[(u'unable', 0.6783158779144287), (u'ableto', 0.6157487630844116), (u'trying', 0.5668548345565796), (u'can', 0.565153837
2039795), (u'allowed', 0.5598537921905518), (u'managed', 0.5453858375549316), (u'enough', 0.5436093807220459), (u'abilit
y', 0.5332520008087158), (u'could', 0.5249689817428589), (u'going', 0.5218186378479004)]


similar words through model1 ride
[(u'rides', 0.8065767288208008), (u'riding', 0.6786860227584839), (u'rode', 0.6538475155830383), (u'Ride', 0.59314429759
97925), (u'Rides', 0.5494017601013184), (u'Fabio_complimented', 0.5408456921577454), (u'joyride', 0.5404961109161377), (
u'riders', 0.5354757905006409), (u'DALMAC', 0.5303251147270203), (u'hop', 0.5091893672943115)]
```

```
similar words through model1 learn
[(u'teach', 0.669853687286377), (u'Learn', 0.652957558631897), (u'learned', 0.6429628729820251), (u'learning', 0.6365275
382995605), (u'visit_www.imuc.com', 0.6094604134559631), (u'discover', 0.6056274175643921), (u'Vectren_visit_http://www.
vectren.com', 0.5989390015602112), (u'please_visit_www.ubisoftgroup.com', 0.5981004238128662), (u'Company_visit_http://w
ww.csst.com', 0.5960359573364258), (u'IMUC_please', 0.5930354595184326)]
```

Comparison:

Example-1:

Through model-2 none of the words actually fall into similar words for the “take” given word but in model-1 actually there are many tenses obtaining for that word.

Example-2:

Through model-2, actually 1 word got matched that is handled because in few contexts care can be used as handled and the last word contaminate completely comes under different meaning.

Example-3:

Through model-2, “will” is little similar to “able” and there are other words which can be used in the same sentences but through model-1 1st obtained similar word pure antonym for the given word that is “unable” which says model-2 is little better.

Example 4:

Through model-2, “walk” comes little similar to ride and there are few words which are completely irrelevant like vegetables etc but in model-1 most of the words are tenses for the given word.

Example 5:

Through model-2, “encourage” is little similar to learn but a word like “stop” is completely different which is antonym but in model-1 the similar words are almost like perfect.