Automated Classification of Media Content (Books and Movies) as Violent or Not

Purpose: Visualization of Violent words Glove Embeddings using t-SNE

Input: "data_for_model.pkl" Output: Graphs plotted inline

Importing Packages

In [7]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import re
import os
import pickle

######################################### For NLTK ########################################

import nltk # NLTK for NLP utils and corpora
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import wordnet


#############################################SCIKIT-LEARN#########################################

# SK-learn libraries for learning.
from sklearn.linear_model import LogisticRegression

# SK-learn libraries for evaluation.
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

# SK-learn libraries for feature extraction from text.
from sklearn.feature_extraction.text import *
# This tells matplotlib not to try opening a new window for each plot.

#############################################UTILS#################################################


import json, shutil, sys, time
from importlib import reload
import collections, itertools
import unittest
from IPython.display import display, HTML# Standard python helper libraries.
from __future__ import print_function
from __future__ import division

# Numerical manipulation libraries.
from scipy import stats, optimize

from IPython.display import Image
```

Reading Data

In [3]:

```
# read the data
DataDir = '..\\DataFiles\\'
non_csm = pd.read_pickle(DataDir + "data_for_model.pkl")
csm = pd.read_pickle(DataDir + "csm_data_for_model.pkl")
print("CSM:",csm.shape)
print("Non CSM:", non_csm.shape)
```

```
CSM: (12317, 40)
Non CSM: (11421, 32)
```

In [4]:

```
print("CSM:\n",csm['violence_ratings'].value_counts())
print("Non-CSM:\n",non_csm['violence_ratings'].value_counts())
```

```
CSM:
 3    3602
0    2690
2    2144
1    1721
4    1407
5     753
Name: violence_ratings, dtype: int64
Non-CSM:
 3    3486
0    2453
2    1836
4    1423
1    1423
5     800
Name: violence_ratings, dtype: int64
```

In [5]:

```
non_csm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11421 entries, 0 to 11420
Data columns (total 32 columns):
adult_content_ratings                    11421 non-null int32
age                                      11421 non-null int64
consumerism_ratings                      11421 non-null int32
drugs_ratings                            11421 non-null int32
entity_type                              11421 non-null object
language                                 7760 non-null object
language_ratings                         11421 non-null int32
positive_messages_ratings                11421 non-null int32
sub_type                                 11415 non-null object
title                                    11421 non-null object
train_summary                            11421 non-null object
violence                                 9601 non-null object
violence_ratings                         11421 non-null int32
adult_content_ratings_bin                11421 non-null int64
consumerism_ratings_bin                  11421 non-null int64
drugs_ratings_bin                        11421 non-null int64
language_ratings_bin                     11421 non-null int64
positive_messages_ratings_bin            11421 non-null int64
violence_ratings_bin                     11421 non-null int64
clean_summary                            11421 non-null object
clean_stop_words_free_summary            11421 non-null object
clean_stop_words_free_lemma_summary      11421 non-null object
clean_summary_tokens                     11421 non-null object
clean_summary_tokens_len                 11421 non-null int64
clean_stop_words_free_tokens             11421 non-null object
clean_stop_words_free_tokens_len         11421 non-null int64
clean_stop_words_free_lemma_tokens       11421 non-null object
clean_stop_words_free_lemma_tokens_len   11421 non-null int64
sentences                                11421 non-null object
train                                    11421 non-null float64
clean_limited_vocab                      11421 non-null object
clean_limited_vocab_tokens               11421 non-null object
dtypes: float64(1), int32(6), int64(10), object(15)
memory usage: 2.5+ MB
```

Logistic Regression for Violent or Non-Violent Summary Classification using Count Vectorizer

In [6]:

```
train_data = non_csm[non_csm['train']==1]['clean_stop_words_free_summary']
train_Y = non_csm[non_csm['train']==1]['violence_ratings_bin']
dev_data  = non_csm[non_csm['train']==0]['clean_stop_words_free_summary']
dev_Y = non_csm[non_csm['train']==0]['violence_ratings_bin']

#### Count Vectorizer

print("Count Vectorizer:")

vectorizer = CountVectorizer()
train_X= vectorizer.fit_transform(train_data)
dev_X = vectorizer.transform(dev_data)

#Logistic Regression Model
lg = LogisticRegression()
lg.fit(train_X, train_Y)
lg_pred = lg.predict(dev_X)
lg_accuracy = lg.score(dev_X, dev_Y)
print ('\nLogistic Regression for All:\n-------------------------------\nUsing default C value = 1
and L2 regularization:')
print ('Logistic Regression F1-Score: %3.2f' %metrics.f1_score(dev_Y,lg_pred,average =
'weighted'))
print ('Logistic Regression Accuracy: %3.2f' %lg_accuracy)
print ("Classification Report:\n",classification_report(dev_Y, lg_pred))
print("Confusion Matrix for test_data:\n",
          confusion_matrix(dev_Y, lg_pred, labels =[0, 1]))
```

```
Count Vectorizer:

Logistic Regression for All:
-------------------------------
Using default C value = 1 and L2 regularization:
Logistic Regression F1-Score: 0.78
Logistic Regression Accuracy: 0.78
Classification Report:
              precision    recall  f1-score   support

          0       0.69      0.65      0.67       388
          1       0.83      0.85      0.84       755

avg / total       0.78      0.78      0.78      1143

Confusion Matrix for test_data:
 [[253 135]
 [114 641]]
```

Get list of words which have largest weights for each label - Violent and Non-Violent

In [18]:

```
#Finding n features with largest weights for each label
n = 300
features = vectorizer.get_feature_names()
matrix, feature_names, features_index = np.zeros(n), [] , []
weights = sorted(lg.coef_[0],reverse=True)[:n]
for weight in weights:
    for k in range(train_X.shape[1]):
        if lg.coef_[0][k]==weight:
            features_index.append(k)
            feature_names.append(str(features[k]))
for (k,j) in zip(range(n),features_index):
    matrix[k] = lg.coef_[0][j]
for i in range(n): print(matrix[i],feature_names[i])
```

```
1.4462878217584636 killed
1.1312692351573383 danger
1.1053441165969133 murder
1.0503798104260644 violent
1.0455372888370296 killer
0.9679844530499901 war
0.8820512827421234 intense
```

0.8793446878218609 mysterious
0.8351826744628348 joins
0.8339112518500725 caleb
0.8297855096108785 malala
0.8262253053076529 escape
0.8226126064105379 gaiman
0.8211589814014777 dark
0.8063682513768 15 deeper
0.8032145228200696 tough
0.7993065549880035 abusive
0.7968010484205896 grimm
0.7947001473980196 drugs
0.7815910900312982 native
0.7777712424191983 kidnapping
0.7732118132544027 shocking
0.7730837836104623 sinister
0.7686595939620048 accident
0.7641776922695405 strange
0.7547284952752482 kidnapped
0.7463266398048742 was
0.7417390907606228 miri
0.7353433194888566 agent
0.7283682792043094 kills
0.7197903404082026 intimate
0.7119820575888195 images
0.7090507126133068 fire
0.7060074074889366 worth
0.6909145610456069 violence
0.6883667904606757 vampire
0.6881854451005108 kill
0.6865033855225889 monsters
0.6859847215520258 horror
0.6839792327956418 bomb
0.6829462095868396 saved
0.6829337977269043 survival
0.6828809671425617 crocodile
0.6814884622724909 tragic
0.6793193811977423 unless
0.6788350558178533 malcolm
0.670787057711532 battle
0.6669126763158469 scary
0.6644562836590916 write
0.6561804937069569 slave
0.6541424846705804 plot
0.6493063832970648 shannon
0.649261743708752 older
0.6418015254975131 trust
0.6381170108344371 groundbreaking
0.6374633357158723 prize
0.6368662087155178 copies
0.6354525891385481 fairyland
0.6318136658036855 you
0.6285412154608168 detective
0.6276863394709545 blackie
0.6270849735274205 poor
0.6260431805788538 harsh
0.6185931168073643 catches
0.6183823959196644 ricky
0.6131755606493468 places
0.6085534276741884 horton
0.6082975526251201 champion
0.6075368628092376 threat
0.6074847707922773 die
0.6065962882754173 starts
0.6018107972685722 survivors
0.6012736984572145 forces
0.6006599586095587 christian
0.6000726140871243 dust
0.5997105352503147 cop
0.5987587361014772 perilous
0.5987302794186357 fate
0.5983115346528898 erupts
0.5979802531877569 review
0.5977960160835724 working
0.5963895074333296 haunting
0.5860728612951327 justice
0.5852362286213691 fight

0.5827863669620953 serve
0.5825680428730816 player
0.581859276290698 frightening
0.5813742749747152 responsible
0.580904603252082 evil
0.5800064733584186 worldwide
0.579283178442994 humanity
0.5785574331575486 egyptian
0.5785515508849723 further
0.5784102731440357 shooting
0.577628364485245 become
0.5769832267530746 technician
0.5757595167383155 maps
0.5742634804975191 guys
0.5722971072204942 teen
0.5709082140061549 took
0.5706628648193693 terrifying
0.5680369303046692 vicky
0.5677027490241771 laws
0.5672199136547175 sammy
0.5616488480576453 texas
0.5603864368520763 separated
0.5594430592574096 bullies
0.5594122242389162 fails
0.5583888742555805 shiloh
0.5550359202736784 hatches
0.5539690564783963 declan
0.5535987059640196 dawn
0.5532666201566305 arranged
0.552379194661376 raised
0.5516170679669187 skinny
0.5504335108622888 copper
0.5502621513124085 swan
0.549664264407652 rats
0.5494765950415 83 pirates
0.5487512611410401 suicide
0.548255921953247 upon
0.5481049093891527 pregnant
0.5474670314804893 wrenching
0.5469589918011062 pratchett
0.5443085339292965 bugs
0.5437321159639713 legendary
0.5429510750142491 saving
0.5426551609625206 steinmark
0.5424350400814385 warriors
0.5416568607575794 destructive
0.5414196211601758 open
0.541382155092489 chase
0.541057086025467 unspeakable
0.5401449983953651 utterly
0.5395038778941958 passed
0.5372942826416274 murdered
0.5366181344817357 hates
0.535330918686698 heartland
0.5349455168433556 series
0.5344791033812446 swimmy
0.5339948634893622 contains
0.5338905354759632 played
0.533285232614718 creatures
0.5329654688461377 raymie
0.5328011635518651 torn
0.5319791468476209 samuel
0.5313778853367523 freedom
0.5312950384014755 sparks
0.5302505839469923 abuse
0.5301781427058924 permission
0.5287585957649094 mommy
0.5276900463750223 birdee
0.5271593111117816 thriller
0.5267708067820729 army
0.5241116271197582 creek
0.5226971265827183 transported
0.5217780403760938 homeless
0.52116379212602 destiny
0.5198012593436899 undercover
0.5196224673775947 west
0.519232352359825 boxing

0.5189254877240383 male
0.5175847475350934 debt
0.5168571747086798 gun
0.5153258985248605 control
0.5149717689577716 stepmother
0.5129367986298771 totally
0.5124352550320501 simone
0.5111169514239484 rumor
0.5109899452157125 arrogant
0.5085926602996577 indian
0.5070649568695035 times
0.5068155090744386 anger
0.5058192031962141 fourteenth
0.5056253164855857 dead
0.504519671257717 videos
0.5027168771601902 lure
0.502443958546046 darkly
0.5011309731970026 telling
0.5008433814865212 lone
0.49775185695636215 fantasy
0.4959959279974336 tressa
0.49555992696307877 message
0.495428404621259 events
0.4952488807099357 mighty
0.4933502611928308 celebrities
0.4931623548037489 television
0.4922633787256277 novel
0.49139180655619624 introduce
0.49135688096393193 howie
0.49089386404766316 tacos
0.49085036520373726 despair
0.49016213850121365 trudie
0.4899588676845987 francie
0.4897894663772174 turning
0.4896082887150958 trilogy
0.4895522062702846 benedict
0.48895394379346335 rap
0.488457611623904 bullied
0.48656947739863415 railroad
0.486362732746543 globe
0.4861772418058236 moses
0.4855240107470635 apollo
0.4855086995945211 amina
0.48547208837461264 lucille
0.4851749203194239 tory
0.484064846819499 illus
0.48339789797916427 nutcracker
0.47992179587750544 recover
0.47848423116248967 fourteen
0.477967829523644 mentor
0.4772436514211226 wolves
0.47685432264242483 cia
0.47590108445761986 troubled
0.4758549751984997 stranded
0.4748849128203801 lil
0.4748284561326966 angie
0.4747583204053732 repair
0.47447589600593143 constantly
0.4741716384830598 hiding
0.474077994724658 courage
0.4740251786472145 eighteen
0.4737042030658316 await
0.4734071254702098 luis
0.47336311433830874 templeton
0.47225504743469543 clueless
0.4719012891671557 paterson
0.47185353385659945 miracle
0.47090231691104856 ozge
0.4697657853520119 purpose
0.46836428407992625 warrior
0.46697771530374305 apprentice
0.46621794060674354 unpredictable
0.4660309099140953 hurt
0.4658387669570742 journalist
0.4646861467693516 sabine
0.4636528485784158 attacked
0.46281242964953107 hilo

```
0.4627301651532422 winnerthe
0.4619114577749526 emotions
0.4614068152969436 sentenced
0.460924459812461 unbeatable
0.4589930183632173 addie
0.4587529798335094 patterson
0.458249566500883 struggling
0.4580722390927536 account
0.4578269387547097 endings
0.45752705506239294 bliss
0.45670848164419214 centuries
0.45497478584383827 juvenile
0.4541814657621 35 impy
0.4539056688647084 twisted
0.45362050314954283 scarlet
0.45352021505597573 man
0.45298102707504523 overdose
0.45266034408492334 frankenstein
0.4520302113804873 bully
0.45182021509347253 changed
0.45177338701133174 buried
0.4515082849063873 hearted
0.44992660048898486 shark
0.44988402022975654 nerd
0.449709293627374 fateful
0.4496399759303152 crow
0.44954707938371224 they
0.4491315695917892 viola
0.4482678802972037 slavery
0.44808328994839197 harper
0.447868301679335 plan
0.4478608423501238 wars
0.44737362072411 54 pilgrim
0.44555277490705614 rebels
0.4454582176990436 belongs
0.44500789784815925 yasuyo
0.44485390459021523 phoebe
0.44475813902057476 cross
0.44424183785252064 ala
0.4440211829533496 gallagher
0.44396767945300986 evidence
0.4439309503820668 newbery
0.44373303855123875 raw
0.4433127780237192 maura
0.4422015214153998 vivid
0.44166085787410847 divergent
0.4413203106203919 bank
0.4412235296748218 year
0.4410075155529046 marlin
0.44069819888459544 probably
0.4404463880379591 absolutely
0.44025894453021 46 tatiana
0.43964315163176876 trade
0.43943751021389943 conor
0.43924081062367454 black
0.4389153106014067 strikes
0.43787954666468715 logan
0.4375739076216241 hood
0.43750472071013297 wake
0.43629531883356537 chronicle
0.4355797371159545 companions
0.43536105678112114 similar
```

In [19]:

```
n = 300
features = vectorizer.get_feature_names()
matrix, feature_names_least, features_index = np.zeros(n), [] , []
weights = sorted(lg.coef_[0],reverse=False)[:n]
for weight in weights:
    for k in range(train_X.shape[1]):
        if lg.coef_[0][k]==weight:
            features_index.append(k)
            feature_names_least.append(str(features[k]))
for (k,j) in zip(range(n),features_index):
    matrix[k] = lg.coef_[0][j]
```

```
for i in range(n): print(matrix[i],feature_names_least[i])
```

```
-0.8980607763195061 jewels
-0.8294585841554443 illustrator
-0.8029364691624318 fourth
-0.7853622602270677 amazing
-0.7691031256412175 hot
-0.7624895859255705 attracted
-0.7341655417528307 senior
-0.7273856904311607 caldecott
-0.7236618943821836 newly
-0.7233026537343329 silent
-0.7225456251731133 una
-0.7166374304612435 frustrated
-0.7165855684706534 vast
-0.7158886988056573 connection
-0.7059247533334025 ready
-0.7007858890425164 loves
-0.6901536543598139 graduation
-0.6792673480152133 gifted
-0.6637495733275247 cares
-0.6580976109434618 household
-0.6527978301652106 mackenzie
-0.6495856016658461 description
-0.6492330491140019 heartwarming
-0.6479001211695091 warm
-0.6427961518242173 dancers
-0.6322650266302879 track
-0.6262848072994289 boyd
-0.6220070212397502 gem
-0.6178919726573515 icon
-0.6134433010032279 ability
-0.611894732006636 problem
-0.6086425204437835 goldmember
-0.6075985006606965 friendly
-0.6070659180823714 details
-0.6057571393632092 margo
-0.6044664232199136 importance
-0.6042047594480402 inevitable
-0.6028980575247217 rules
-0.5994312000847936 fun
-0.5989584136686371 bedtime
-0.5962692040688433 backdrop
-0.5961710183149609 nasa
-0.5957180411083007 vibrant
-0.5938269017955179 depression
-0.5934608231385272 holiday
-0.5932563138423199 charming
-0.5908290907504132 lovely
-0.5899640409860135 garden
-0.5895989828556628 convinces
-0.5892592231068923 scientist
-0.5891444543310311 marks
-0.5877323676941992 trouble
-0.587156766440465 boris
-0.5863108393694301 goat
-0.5861955668335403 fitting
-0.5835206047278101 picture
-0.5820681267175125 grail
-0.5818191170293604 apple
-0.580900986537337 idea
-0.5800437723466328 photographs
-0.579606492370442 surprise
-0.5790607720392903 sure
-0.5777734698021444 beloved
-0.576914576287048 mysteries
-0.5741449124304926 spectacular
-0.5725000276146766 vacation
-0.5682804911430005 happily
-0.5668464355771932 compelling
-0.5666860149139727 meet
-0.5656380917671998 seeking
-0.5644954125634798 theodore
-0.5606917656877782 voices
-0.5601452991343465 waits
-0.5576928038185154 square
-0.5537626529545975 devices
```

-0.5524556311143874 wordless
-0.5516954449721357 imagines
-0.5513306300969444 floor
-0.550936790301155 closely
-0.5508463338416972 sad
-0.5493541581226901 dig
-0.5492143588928344 computer
-0.5484536495367633 possibly
-0.5475334251512664 carefully
-0.5475296504800207 allowed
-0.5463504546035288 lenny
-0.5459264186472134 hurry
-0.5456061246897339 reality
-0.5452824431113459 hears
-0.5450594281336333 getting
-0.5434672357485395 impress
-0.5433861634297098 actress
-0.54322723783097 nerdy
-0.5428276741285564 rio
-0.5426318135712117 dantdm
-0.5377715567841587 squid
-0.5373777110706626 starting
-0.5363915506267324 winnie
-0.5361237805492037 independent
-0.5356876308534732 phillips
-0.5344933635087961 counting
-0.5338276497871137 lazlo
-0.5329413431030321 that
-0.5323071529424949 flying
-0.5289623454259335 duane
-0.5287844460253761 cake
-0.528167166723432 equality
-0.5279825335405636 pretorius
-0.5269817270700122 cookies
-0.5257787070102108 responsibilities
-0.5230815409566316 traditions
-0.5229653004149646 damian
-0.522343066604318 goal
-0.5210470403599109 income
-0.5193527569565007 right
-0.5192809166022385 strip
-0.5181469807053384 eat
-0.5180301339817662 marco
-0.5173857673874056 chinese
-0.5163413696250244 gaz
-0.5158637430412917 evening
-0.514806298988404 realization
-0.5146184191222972 achieve
-0.5140367548031205 why
-0.5130359020749538 dessen
-0.5123999187750451 uchenna
-0.5087880572111672 whitacre
-0.5072981217965573 lana
-0.5072041927767946 newborn
-0.5054945197079691 wood
-0.5046661527906589 oren
-0.5031893641468489 ideas
-0.5030886206458292 gottie
-0.5030052328308102 dastardly
-0.502427094790691 weeks
-0.5014188114976682 auggie
-0.5010997003348034 feelings
-0.49846881547137073 somewhat
-0.4981446672848823 mate
-0.4979794916277167 cute
-0.497302223015943 tia
-0.4970660197754406 earn
-0.4962618792138169 mention
-0.4960648200692566 mandela
-0.4956446307071395 colored
-0.4954809638437233 dol
-0.4945522565027538 capital
-0.4942378755716579 janet
-0.49423566369983424 tournament
-0.4941385020643897 visit
-0.49334332261732844 gerald

-0.4929402689404402 adorable
-0.49035719121374594 goodbye
-0.4894343556681608 paris
-0.4889903051250883 tree
-0.4883721704176521 finley
-0.4882139626110296 inspector
-0.48817444831585277 segments
-0.4871302221929962 misses
-0.48665887347617287 praised
-0.48660171897354737 entire
-0.4861383946066274 devonny
-0.48593635026009846 musical
-0.4857608496684896 beans
-0.4853608913877032 ploo
-0.48461473548378164 louise
-0.48398972581653527 solution
-0.48397215213977196 machine
-0.4829936170232579 broadway
-0.4824160514560664 enchanting
-0.48239863556349427 ones
-0.4822171752025461 porter
-0.4797413908381868 ottaviano
-0.47953380173388505 goats
-0.47923399473756995 suddenly
-0.4790918559583661 leukemia
-0.4770358390121699 foxxy
-0.47666698436841065 cheese
-0.4756979159866358 spare
-0.47567677269349123 came
-0.47553652863035345 around
-0.4752723639709815 warmth
-0.47433538040657724 crush
-0.4740197288849554 mira
-0.4735256815797672 kindness
-0.47344568786790986 gardening
-0.4733922031447659 friedman
-0.4728186259101826 least
-0.4721054133183649 pacific
-0.47174033165507284 haakon
-0.4715496313604133 mahalia
-0.47131306408788337 psychological
-0.47051922874646096 tests
-0.4704662462156089 afford
-0.4701198587879434 gift
-0.47001427188190825 wendy
-0.4698370326162707 promised
-0.46979478420815346 mitch
-0.4681322342122301 rival
-0.4668488561431814 remin
-0.46665477494200136 gatlin
-0.466479536720669 brady
-0.4663491937565764 mess
-0.46605075810949353 jenny
-0.4654163473940627 determination
-0.4648707903225992 perfect
-0.4648685254941819 perfectly
-0.4644974645549594 marjorie
-0.4644915465972188 tractor
-0.46423876531179586 clue
-0.4636467257134675 accessible
-0.46306174770808584 nigel
-0.46158914191362654 goodnight
-0.46026508620464557 started
-0.4600728202266227 build
-0.4570361158340071 researched
-0.4569211679657506 adventures
-0.45670855207175387 screen
-0.45472208490893556 dolezal
-0.45453366249198723 cream
-0.4542689621153378 amount
-0.4542504076614602 playboy
-0.4535028198447767 docks
-0.45350098777919595 shopping
-0.4533862805581568 introduces
-0.4528495020015639 ranging
-0.4523223499456994 haunt
-0.45219317797600966 trayaurus

```
-0.45196011964881433 rural
-0.4511123560143704 popov
-0.45017546461985575 xan
-0.4497764807447588 months
-0.44827397071825276 listening
-0.44815943111455697 boone
-0.44775650032385444 verse
-0.4474369289451695 process
-0.4469290587294187 guest
-0.4463894096031445 wedding
-0.44632065959589345 misguided
-0.4456987590164349 fuzzy
-0.4456483216919341 study
-0.44515958831340374 swamp
-0.4449996232947108 dot
-0.44441088714651994 garnet
-0.4443433112647687 oliver
-0.4440127352671796 freckles
-0.4439318659616079 suggests
-0.44263877803586615 erika
-0.44242144746481804 do
-0.44232329474880017 joyce
-0.4416115913449614 favorite
-0.4412280014429372 ebook
-0.44094399510678045 willenholly
-0.44041880725860744 fulfill
-0.44001659450559444 allow
-0.4397462137874757 follow
-0.43869105159313226 instructor
-0.43828220472010615 easily
-0.4380204450256549 imagined
-0.4378091573557554 minnesota
-0.4374859740260812 lmntrix
-0.4368839632186472 puts
-0.43678550310233705 lessmore
-0.4362105300676816 touching
-0.4361504387509546 bonds
-0.4348026794257007 arcady
-0.4342793243174882 earthquake
-0.4339917900843033 tuesday
-0.4333703462679076 items
-0.43172155276502616 seas
-0.431624839650741 sleeping
-0.4314701134720608 slightly
-0.43134098253064623 citizens
-0.43111295177544456 hillsong
-0.4309876244355975 guardian
-0.4304507662465463 poetry
-0.43002476838645237 funny
-0.4300227038739611 sled
-0.4297274450459516 sounds
-0.42969640745525645 antoine
-0.42960692938380796 goofy
-0.4293388439535619 neal
-0.42887579922982755 magical
-0.4286934803009825 harder
-0.4286477599067614 adult
-0.4285673444961771 founded
-0.4280354217822874 loco
-0.4279033851435065 adams
-0.4272903672560171 interactive
-0.4272810060328779 pamela
-0.4270681695400561 treasure
-0.42683490419889175 decides
-0.42665567123507153 tomorrow
-0.42633526648932346 failure
-0.4261403325406868 valentine
-0.4257428262049542 vermont
-0.42550509412500426 outstanding
-0.42481137657426193 fell
-0.42447433218221353 songs
-0.42422068506912836 manhattan
```

Visualizing Words Glove Embeddings using t-SNE

Note: Purpose of the plot is to show how closely embedded negative or violent words are to each other in the Glove Embeddings space in a 2-dimensional plot

In [10]:

```
"""
Visualize word embeddings, using tsne.
First computes cosine distance of the 100 closests words, and then shows a clustering graph
of the first 11 closest words (the first one is always the word)
IT REQUIRES GLOVE MODEL.txt
line 31: glove_file = '../TBIR/glove.840B.300d.txt' MODIFY with the appropiate path
To Use it, you can just type: python word_embedding_vis.py <list of words space separated>
                              e.g: python word_embedding_vis.py cake word embedding music
"""
"""
check some glove words
"""
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.manifold import TSNE
from sys import stdout
import numpy as np
from matplotlib import pyplot
import sys


def build_glove_dictionary():
    """
        builds a dictionary based on the glove model.
        http://nlp.stanford.edu/projects/glove/
        dictionary will have the form of key = token, value = numpy array with the pretrained
values

        REALLY IMPORTANT the glove dataset. with the big one finds nearly everything....
        smallest one...quite baaaaaad...
    """
    print ('building glove dictionary...')
    glove_file = DataDir + '\\glove.6B\\glove.6B.300d.txt'
    glove_dict = {}
    with open(glove_file, encoding="utf8") as fd_glove:
        j=0
        for i, input in enumerate(fd_glove):
            input_split = input.split(" ")
```

```
            #print input_split
            key = input_split[0] #get key
            del input_split[0]  # remove key
            j+=1
            stdout.write("\rloading glove dictionary: %d" % j)
            stdout.flush()
            values = []
            for value in input_split:
                values.append(float(value))
            np_values = np.asarray(values)
            glove_dict[key] = np_values
            #else:
                #print key
    print ("")
    print ('dictionary build with length', len(glove_dict))

    return glove_dict

glove_dict = build_glove_dictionary()
```

```
building glove dictionary...
loading glove dictionary: 400000
dictionary build with length 400000
```

In [13]:

```
def build_glove_matrix(glove_dictionary):
    """
        return word2idx and matrix
    """
    idx2word = {}
    glove_matrix = []
    i=0
    for key, value in glove_dictionary.items():
        idx2word[i] = key
        glove_matrix.append(value)
        i+=1
    return np.asarray(glove_matrix), idx2word


def check_similarity(glove_matrix, word):
    return cosine_similarity(word.reshape(1, -1), glove_matrix)
```

```
def build_matrix_to_tsne(glove_dict, tokens):

    matrix = []

    for token in tokens:

        if token in glove_dict:

            matrix.append(glove_dict[token])

    return matrix
```

In [21]:

```
words = []
violent_words = []
non_violent_words = []
for t in non_csm.iloc[2]['clean_stop_words_free_tokens']:
    if t in feature_names:
        if t not in violent_words:
            violent_words.append(t)
    if t in feature_names_least:
        if t not in non_violent_words:
            non_violent_words.append(t)

#if len(sys.argv)<2:

#    print ('Words not specified')

#    words = ["plant", "factory", "machine", "houseplant", "cake"]

#else:

    #for i in range(1, len(sys.argv)):

#        words.append(sys.argv[i])

print ('Non-Violent Words that will be used', non_violent_words)
print ('Violent Words that will be used', violent_words)
```

```
Non-Violent Words that will be used ['getting', 'floor', 'problem', 'decides', 'tree', 'study', 'i
dea', 'items']
Violent Words that will be used ['detective', 'killed', 'dead', 'attacked', 'strange', 'eighteen',
'year', 'older', 'vampire', 'escape', 'black', 'arrogant', 'cross', 'unless', 'kill', 'murder', 'g
uys', 'monsters', 'battle', 'trust', 'copies']
```

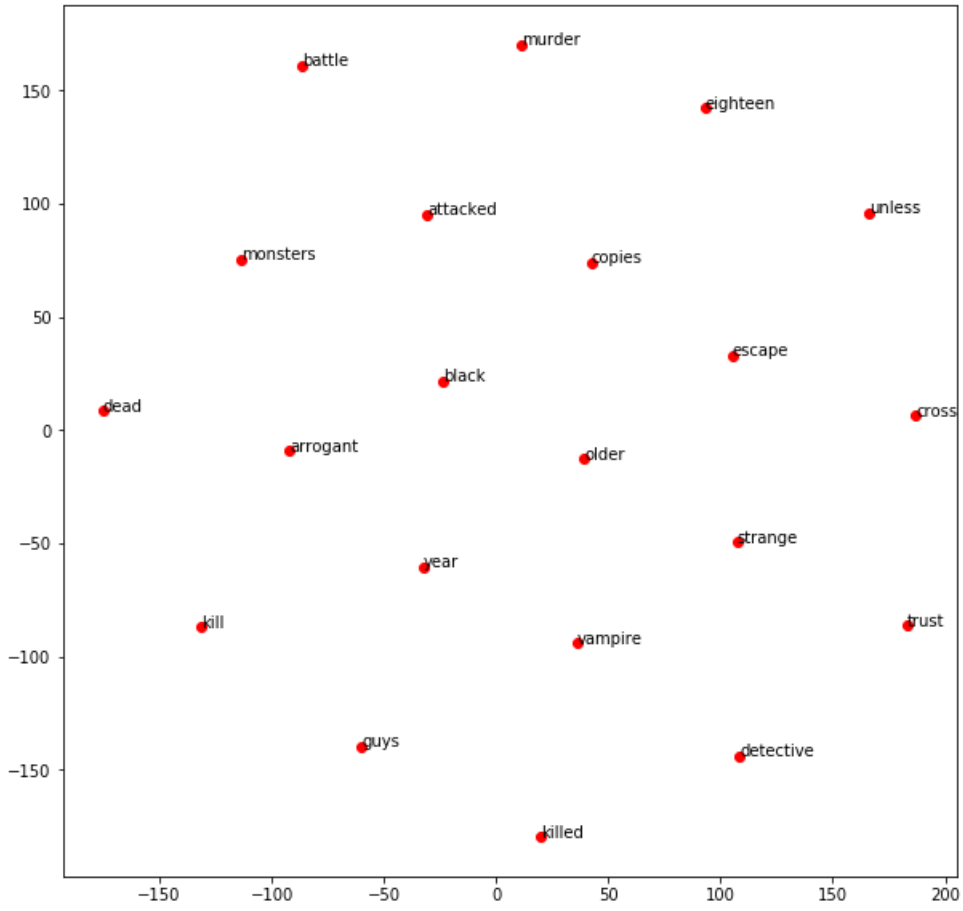
In [22]:

```
words = violent_words
glove_matrix, idx2word = build_glove_matrix(glove_dict)

model = TSNE(n_components=2, random_state=0)



to_plot = []

labels = []

not_found = 0

len_words = len(words)

for word in words:

    try:

        cosine_matrix = check_similarity(glove_matrix, glove_dict[word])

        ind = cosine_matrix[0].argsort()[-1:][::-1]

        closest = ind.tolist()

        tokens = [idx2word[idx] for idx in closest]
```

```
        to_reduce = build_matrix_to_tsne(glove_dict, tokens)
        #print to_reduce.shape
        labels += [token for token in tokens]
        to_plot += [x_y for x_y in to_reduce]
    except:
        len_words-=1
        print ('Word not found', word)


print (len_words)
#print to_plot.shape
#print to_plot
X_hdim = np.array(to_plot)
#print X_hdim
print (X_hdim.shape)
X = model.fit_transform(X_hdim)
X_x = np.zeros((len_words*1, 2))
labels_x = []
print (X.shape)
k=0
ranges = [x*1 for x in range (0, len_words)]
print (ranges)
for i in ranges:
    for j in range(1, 2):
        #print (i+j-1, k)
        X_x[k] = X[i+j-1]
        k+=1
        labels[i+j-1]
        labels_x.append(labels[i+j-1])
print (labels_x)
print (X_x.shape)
plt.figure(figsize=(10, 10))
plt.scatter(X_x[:,0],X_x[:,1], c='red')
for i, label in enumerate(labels_x):
    pyplot.annotate(label, (X_x[i,0],X_x[i,1]))


pyplot.show()
```

```
21
(21, 300)
(21, 2)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
['detective', 'killed', 'dead', 'attacked', 'strange', 'eighteen', 'year', 'older', 'vampire', 'es
```

```
cape', 'black', 'arrogant', 'cross', 'unless', 'kill', 'murder', 'guys', 'monsters', 'battle', 'tr
ust', 'copies']
(21, 2)
```



Comparing Violent and Non-Violent Words in the Glove Embeddings Space

In [ ]:

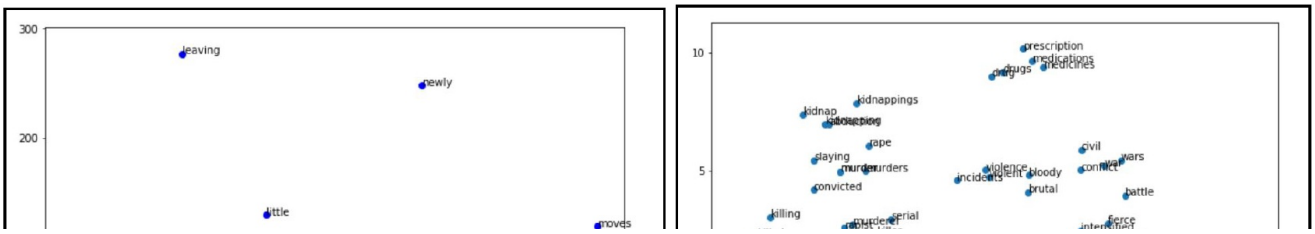
```
words = non_violent_words
glove_matrix, idx2word = build_glove_matrix(glove_dict)

model = TSNE(n_components=2, random_state=0)


to_plot = []

labels = []

not_found = 0

len_words = len(words)

for word in words:

    try:

        cosine_matrix = check_similarity(glove_matrix, glove_dict[word])

        ind = cosine_matrix[0].argsort()[-1:][::-1]

        closest = ind.tolist()

        tokens = [idx2word[idx] for idx in closest]

        to_reduce = build_matrix_to_tsne(glove_dict, tokens)

        #print to_reduce.shape

        labels += [token for token in tokens]

        to_plot += [x,y for x,y in to_reduce]
```

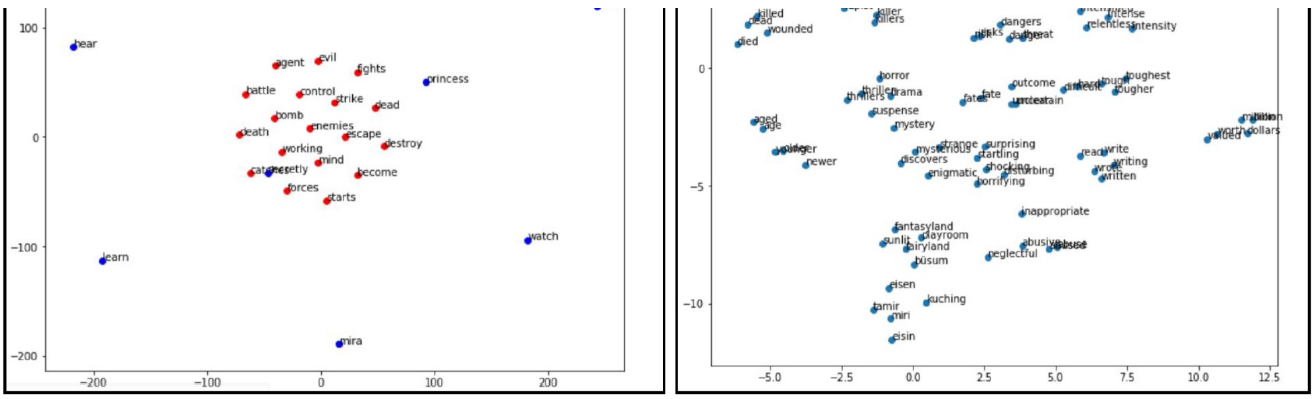
```
        to_plot += [x_y for x_y in to_reduce]
    except:
        len_words-=1
        print ('Word not found', word)

print (len_words)
#print to_plot.shape
#print to_plot
X_hdim = np.array(to_plot)
#print X_hdim
print (X_hdim.shape)
X = model.fit_transform(X_hdim)
X_x = np.zeros((len_words*1, 2))
labels_x = []
print (X.shape)
k=0
ranges = [x*1 for x in range (0, len_words)]
print (ranges)
for i in ranges:
    for j in range(1, 2):
        #print (i+j-1, k)
        X_x[k] = X[i+j-1]
        k+=1
        labels[i+j-1]
        labels_x.append(labels[i+j-1])
print (labels_x)
print (X_x.shape)
plt.scatter(X_x[:,0],X_x[:,1], c='blue')
for i, label in enumerate(labels_x):
    pyplot.annotate(label, (X_x[i,0],X_x[i,1]))
pyplot.show()
```

In [3]:

```
Image(filename="Comparing Violent and Non-Violent Words.png")
```

Out[3]:




hear
agent
evil
fights
princess
battle
control
strike
dead
bomb
enemies
escape
destroy
death
working
mind
become
forces
starts
learn
watch
mira
killed
dead
wounded
died
killer
killers
dangers
relentless
intensity
horror
outcome
toughest
tougher
drama
fate
fates
suspense
mystery
aged
age
newer
strange
surprising
mysterious
startling
discovers
shocking
enigmatic
horrifying
read
write
writing
wrote
written
worth
dollars
inappropriate
fantasyland
dayroom
sunlit
fairyland
neglectful
büsum
eisen
kuching
tamir
miri
eisin