NLP Assignment 2

1 Negative Sampling

Word2Vec is used to create word embeddings which help us store words in forms of vectors with numerical data which helps identify relationship between words. It can be implemented using either the CBOW or the Skip-Gram methods. However, both the models involve tweaking huge amount of data in the model every time its being trained or tested, this is in the order of O(V) where V is the size of the vocabulary, This will require a lot of computation as the order can go to millions. Negative Sampling is a process which makes this easier by an approximation. As the name describes, we sample out few of the data points and only tweak them so that the computational cost is lesser and the model is faster. Instead of checking every output possible and changing it according to the error, we sample few points from the output which cannot be close to the correct output (Hence the name negative). We do this by picking words with low unigram probability, which becomes better when the word counts are raised to the power 3/4. Overall, this helps in making the training and evaluating process much faster and also helps in improving the model accuracy by avoiding overfitting.

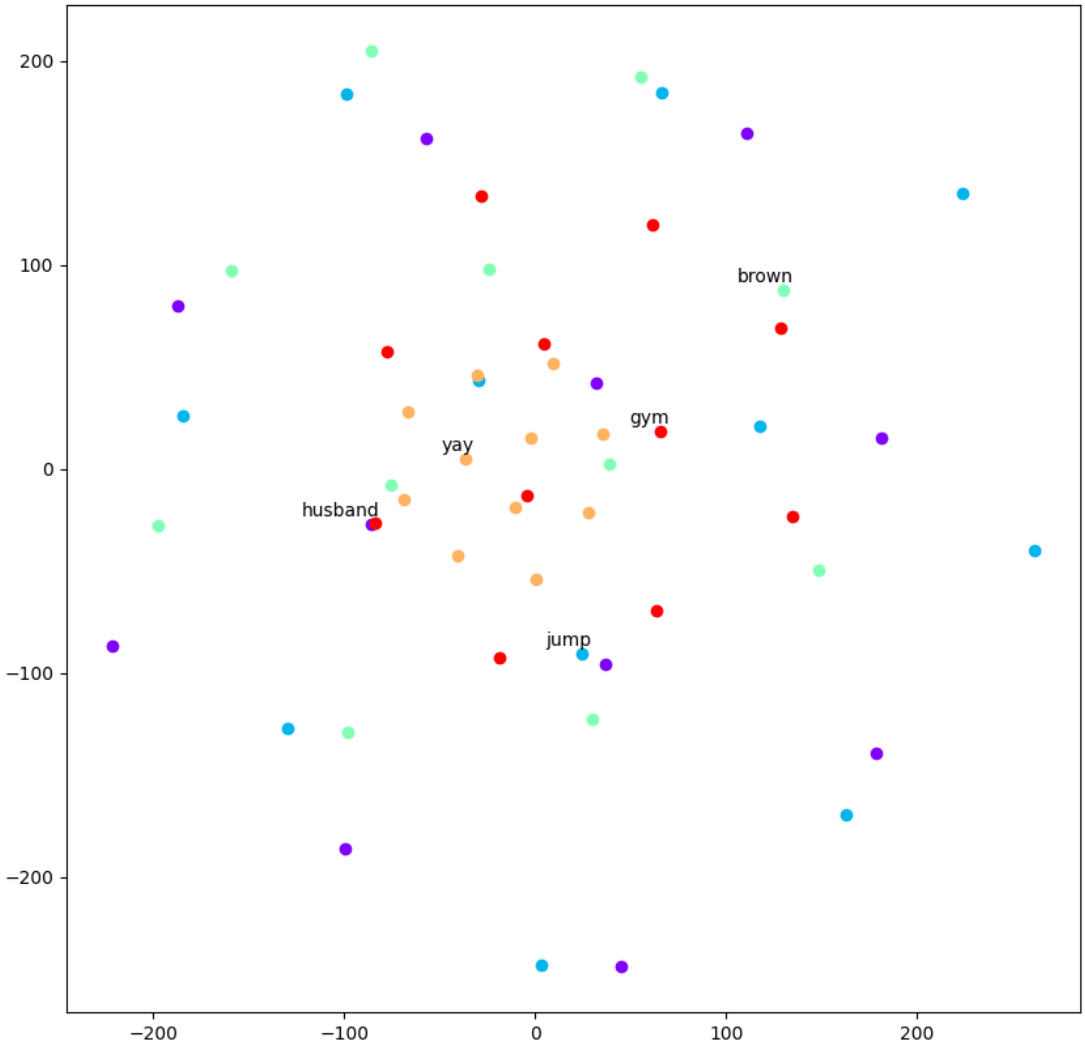
2 Implementation

A word embedding model was implemented by building a Co-occurence matrix followed bty applying SVD, lets call this M1.

Another model was implemented using word2vec algorithm with negative sampling, lets call this M2.

3 Analysis

M1 SVD



Below are the top 10 closest words for five different words, the plot above shows the t-SNE plot of the same

husband
('wife', 0.992027653660948)
('dad', 0.9916529813274371)
('girlfriend', 0.991423945188775)
('son', 0.9878581324728365)
('sister', 0.9874197317683705)
('grandson', 0.9859388228282593)
('daughters', 0.9837203866037998)
('granddaughter', 0.9788691342888013)
('mom', 0.9779894336553238)
('daughter', 0.9766958213299107)
jump
('add', 0.9402843708352886)
('learn', 0.9399077463176139)
('capture', 0.9386934103648702)
('convert', 0.9369529819722059)
('adapt', 0.9341764351287913)
('manage', 0.9322908542148359)

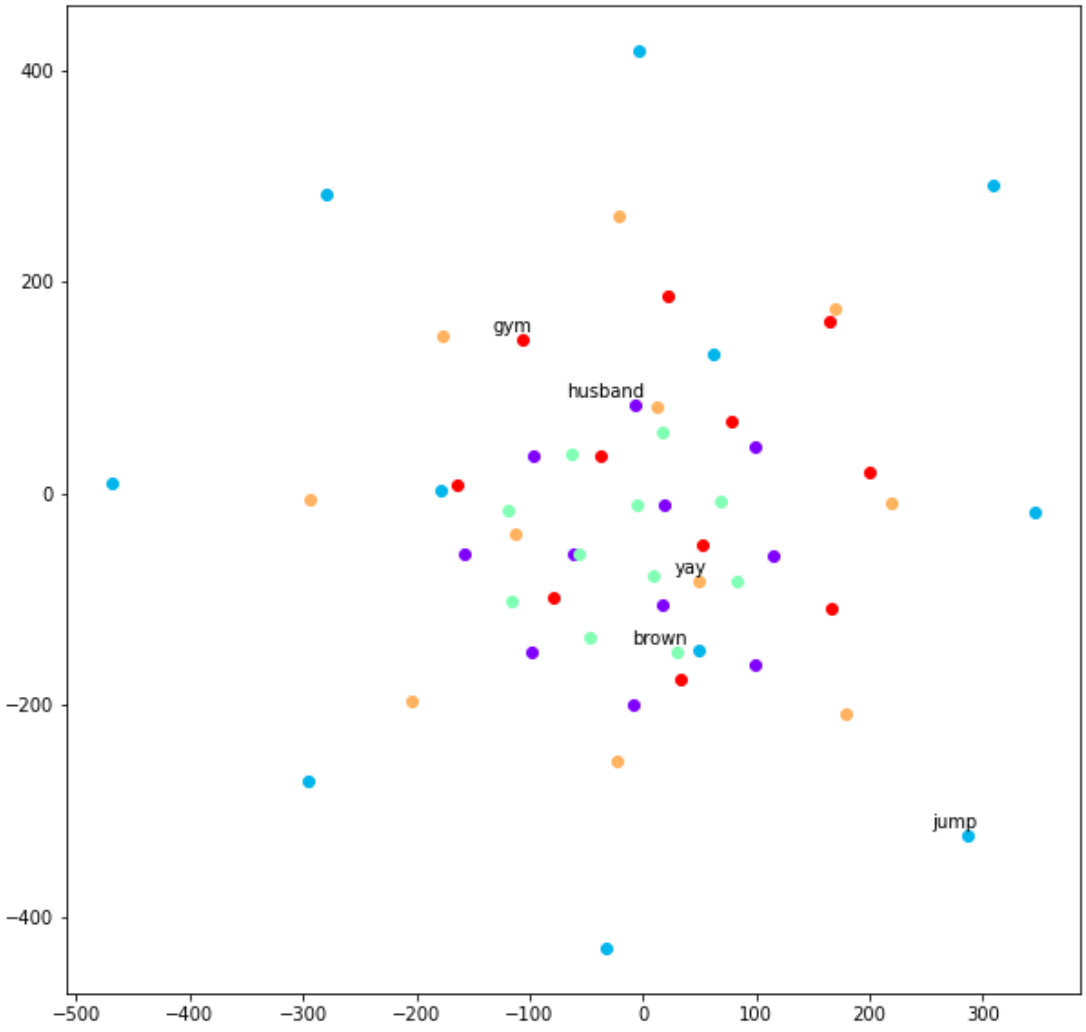
('attempt', 0.9319926556074012)
('prevent', 0.9300430212231718)
('connect', 0.9292614559098973)
('navigate', 0.9285640485697229)
brown
('yellow', 0.8919192470240338)
('white', 0.8905126929716102)
('bush', 0.8735582784830808)
('red', 0.8730165289661088)
('hiss', 0.8723403562852493)
('latch', 0.8567363127849832)
('green', 0.8565830926635449)
('boom', 0.8548303007335698)
('refresh', 0.8542418696714715)
('blue', 0.8492980170052243)
yay
('whew', 0.9532642623067438)
('omg', 0.9394115179346204)
('bah', 0.9357207285845475)
('arrggh', 0.9313735581760757)
('shesh', 0.9313735581760757)
('aghh', 0.9313735581760757)
('humbugg', 0.9313735581760757)
('hhm', 0.9313735581760757)
('jjt', 0.9313735581760757)
('woof', 0.9313735581760757)
gym
('road', 0.9894104677425155)
('wall', 0.9822807294717624)
('box', 0.98056018039237)
('internet', 0.9784257665602768)
('street', 0.9777988645512872)
('market', 0.9776905821803065)
('woods', 0.976790512959066)
('ghost', 0.9765635550065005)
('web', 0.975237227137714)
('floor', 0.9743076984344966)

We see that most of the results make a lot of sense, husband is a word which expresses relationships and so are the words close to it such as wife which is the closest.

The other words also show similar results where we can clearly see some sort of similarity between the words and results like "yay" and its results being expression words and gym being an inanimate word and jump and its results being a action words(verbs). Hence we see that the word embeddings actually provide a lot of information and help classify words.

The graph above also represents words which are similar to each other on circumferences of circles with similar radii, as we can see with the color codes.

M2 Word2Vec



husband
('rome', 0.24986514449119568)
('=pi', 0.21309199929237366)
('fatter', 0.21105843782424927)
('fellows', 0.21100041270256042)
('onstage', 0.21055074036121368)
('challenges', 0.20659497380256653)
('commander', 0.2063833624124527)
('competitors', 0.20503093302249908)
('achieving', 0.1981040984392166)
('frontales', 0.1943659484386444)
jump
('serious&;', 0.23996391892433167)
('farsightedness', 0.23238928616046906)
('prolong', 0.22614991664886475)
(':>)', 0.20843660831451416)
('fangled', 0.206498384475708)

('letter', 0.2057553380727768)
('dissuaded', 0.20305152237415314)
('and-true', 0.20280280709266663)
('erp', 0.20165900886058807)
('beautiful', 0.200969859957695)
brown
('exchanges', 0.22398288547992706)
('george', 0.22261027991771698)
('loaded', 0.2081892043352127)
('opticallogitech', 0.2068214863538742)
('small', 0.20618101954460144)
('cheapexcellent', 0.20614148676395416)
('nevetheless', 0.20595170557498932)
('tempermental', 0.20586207509040833)
('nimhbatteries', 0.20583981275558472)
('thirds', 0.19648779928684235)
yay
('vindigo', 0.24857290089130402)
('faces', 0.2296067774295807)
('adjustability', 0.22461621463298798)
('uptake', 0.2219250500202179)
('marriage', 0.2160802036523819)
('convection', 0.20923718810081482)
('bombproof-', 0.20917439460754395)
('automated', 0.20304429531097412)
('thread', 0.2016470581293106)
('wring', 0.20044545829296112)
gym
('genuinely', 0.23877976834774017)
('nettings', 0.2159147560596466)
('acceptable', 0.20865419507026672)
('greatly', 0.2073860764503479)
('youtube', 0.206322580575943)
('codec', 0.20543263852596283)
('planner', 0.2007838636636734)
('sd', 0.19450969994068146)
('ishort', 0.1940993368625641)
('generating', 0.19232888519763947)

We can see somewhat similar results in M2 compared to M1 but they are also very far apart in terms of making sense and similarity scores, the same goes with the plots.

Results with the word Camera

Below are the top 10 closest words to the word "camera" produced by both the models and another pretrained Gensim model.

SVD

('keyboard', 0.9657517626087294)
('router', 0.9602507615507271)
('lense', 0.9600257873161063)
('unit', 0.9553567584965549)
('radio', 0.9538898496260968)
('device', 0.94558820496555)
('seller', 0.942540846227179)
('remote', 0.9419897368860137)
('case', 0.9418164313925634)
('cable', 0.9404134275311115)

Word2Vec

(':[-]', 0.2255605161190033)
('system--', 0.20657318830490112)
('hangers', 0.20358818769454956)
('control+o', 0.19815048575401306)
('stylii', 0.19654597342014313)
('sull', 0.19393952190876007)
('diving', 0.19371920824050903)
('pixma', 0.19365279376506805)
('oepn', 0.19288615882396698)
('sangean', 0.1889047473669052)

Gensim Pretrained:

which , part , in , of , on, one, ., as, this, its

Comparing the results, we can clearly see that the first model makes a lot of sense when it comes to similarity but the other 2 do not. The Gensim model just presents us with most common words like articles which does not give much information, the word2vec model performss a bit better but with low scores but the svd model performs the best with amazing scores and makes the most sense. Hence we can conclude that the svd model is the most robust among the models.