Analogy Prediction Task:

- **Problem 1:**

  Run the analogy test. Pick any **two** sets of pretrained embeddings, implement the analogy prediction method described in Equation 2, and compare their accuracies on the eight analogy tasks listed above. Make sure to mention the details of your selection in writing.

  Solution:

  I have used GloVe and word2Vec pretrained word embeddings in this assignment.

  The following table is the comparison accuracies for the 8 categories (capital-world, currency, city-in-state, family, gram1-adjective-to-adverb, gram2-opposite, gram3-comparative, and gram6-nationality-adjective) between the two chosen pretrained word embeddings.

Category Name	GloVe	Word2Vec
capital-world	0.88	0.79
currency	0.14	0.35
city-in-state	0.30	0.70
family	0.81	0.84
gram1-adjective-to-adverb	0.24	0.28
gram2-opposite	0.20	0.42
gram3-comparative	0.79	0.90
gram6-nationality-adjective	0.87	0.89

- **Problem 2:**

  One known problem with word embeddings is that antonyms (words with meanings considered to be opposites) often have similar embeddings. You can verify this by searching for the top 10 most similar words to a few verbs like increase or enter that have clear antonyms (e.g., decrease and exit, respectively) using the cosine similarity. Discuss why embeddings might have this tendency.

  Solution:

This is because word vector models are generally based on the context of the words, i.e..in a vector space, the word and its antonym , can be quite closer to each other ,as they may have similar contexts in many cases.

For example, when we take increase and decrease, we can use them in the same context as below:

Shares of the Stock market are increased

Shares of the Stock market are decreased – Hence, increase and decrease can be quite closer in a vector model because of their similar context , even though they are antonyms.

- **Problem 3:**

  Below are some analogies that are not part the Mikolovs analogy data set:

  Tested with GloVe and word2Vec models. The results are as follows

  Example 1:

  Glove:

  Given: print (model. most_similar (positive= ['salt', 'cup'], negative=['pepper']))
  Expected: tea /coffee
  Predicted: Most similar is **'olympic'**
  [('olympic', 0.7172178030014038), ('championships', 0.7037688493728638), ('finals', 0.6883701086044312), ('olymsspics', 0.6845704317092896), ('champions', 0.6837102770805359), ('championship', 0.6559773683547974), ('qualifying', 0.6468051671981812), ('world', 0.6393243074417114), ('tournament', 0.6392757892608643), ('event', 0.6345065236091614)]

  Word2Vec:

  Given: print (model. most_similar(positive=['salt','cup'], negative=['pepper']))
  Expected: tea /coffee
  Predicted: Most similar is **'cups'**
  [('cups', 0.5172240734100342), ('salts', 0.4128767251968384), ('frozen_limeade', 0.4061890244483948), ('ice_cubes', 0.3975926041603084), ('cupful', 0.394680917263031), ('resealable_containers', 0.3931456208229065), ('cupfuls', 0.39183327555656433), ('chalice', 0.3886953890323639), ('vacuum_flask', 0.38159316778182983), ('unboiled_water', 0.37690556049346924)]

  Example 2:

  gloVe:

  Given: print (model. most_similar(positive=['love','life'],negative=['hate']))
  Expected: death
  Predicted: Most similar is **'her'**
  [('her', 0.7073758840560913), ('mother', 0.6949683427810669), ('father', 0.6775326132774353), ('she', 0.6731945276260376), ('beauty', 0.6516457796096802), ('dream', 0.6474184393882751), ('grace', 0.6455720663070679), ('great',

0.6450966000556946), ('man', 0.6448743343353271), ('romantic', 0.6362401843070984)]

Word2Vec:

Given: print (model. most_similar(positive=['love','life'],negative=['hate']))
Expected: death
Predicted: Most similar is **'lives'**
[('lives', 0.49568578600883484), ('chiseled_burnished_refined', 0.4618302881717682), ('joys_sorrows', 0.45654046535491943), ('dreams', 0.4481465816497803), ('simpler_pleasures', 0.4440445899963379), ('vocation', 0.43837153911590576), ('happiness', 0.43809962272644043), ('adventuresome_spirit', 0.4316348731517792), ('lifelong_avocation', 0.42875775694847107), ('grandparenthood', 0.4285208582878113)]

Example 3:

gloVe:

Given: print(model.most_similar(positive=['computer','human'],negative=['cpu']))
Expected: brain
Predicted: Most similar is **'rights'**
[('her', 0.7073758840560913), ('mother', 0.6949683427810669), ('father', 0.6775326132774353), ('she', 0.6731945276260376), ('beauty', 0.6516457796096802), ('dream', 0.6474184393882751), ('grace', 0.6455720663070679), ('great', 0.6450966000556946), ('man', 0.6448743343353271), ('romantic', 0.6362401843070984)]

Word2Vec:

Given: print(model. most_similar(positive=['computer','human'],negative=['cpu']))
Expected: brain
Predicted: Most similar is **'employee_Laura_Althouse'**
[('employee_Laura_Althouse', 0.46909165382385254), ('impertinent_flamboyant_endearingly', 0.44120728969573975), ('computerized', 0.4092547595500946), ('Human', 0.4065849184989929), ('embryo_clones', 0.4008949100971222), ('human_beings', 0.39421117305755615), ('bullet_shell_casing', 0.3941192328929901), ('computers', 0.3939198851585388), ('specific_receptor_ligand', 0.3908461928367615), ('humans', 0.3846876621246338)]

Example 4:

gloVe:

Given: print(model.most_similar(positive=['he','she'],negative=['him']))

Expected: her
Predicted: Most similar is **'never'**
[('never', 0.7211065888404846), ('has', 0.7182034254074097), ('also', 0.7164497375488281), ('was', 0.6943008899688721), ('had', 0.6846156120300293), ('.', 0.6806784868240356), ('finally', 0.6729162931442261), ('now', 0.6683130264282227), ('then', 0.6667650938034058), (',', 0.6648724675178528)]

Word2Vec:

Given: print(model.most_similar(positive=['he','she'],negative=['him']))
Expected: her
Predicted: Most similar is **'She'**
[('She', 0.5993854999542236), ('hers', 0.49200111627578735), ('her', 0.46830421686172485), ('never', 0.4296281039714813), ('E_verybody', 0.42840784788131714), ('nobody', 0.4214271903038025), ("she'sa", 0.4203073978424072), ('Singer_Colbie_Caillat', 0.4185062348842621), ('Rivadineira', 0.41582587361335754), ('helmet_Carusone', 0.4154716730117798)]

Example 5:

gloVe:

Given: print(model.most_similar(positive=['smarter','faster'],negative=['smartest']))
Expected: fastest
Predicted: Most similar is **'quicker'**
[('quicker', 0.7282894849777222), ('slower', 0.7108127474784851), ('considerably', 0.6262897849082947), ('stronger', 0.605583906173706), ('smoother', 0.6009765267372131), ('cheaper', 0.5992809534072876), ('simpler', 0.5986998677253723), ('significantly', 0.5984729528427124), ('easier', 0.592199981212616), ('improved', 0.5868023633956909)]

Word2Vec:

Given: print(model.most_similar(positive=['smarter','faster'],negative=['smartest']))
Expected: fastest
Predicted: Most similar is **'quicker'**
[('quicker', 0.747742772102356), ('slower', 0.6253479719161987), ('better', 0.5720376372337341), ('stronger', 0.5620588660240173), ('speedier', 0.5460758805274963), ('swifter', 0.5453687310218811), ('Faster', 0.5452728271484375), ('smoother', 0.5344820022583008), ('Get_GhostTheory_updates', 0.5217344760894775), ('simpler', 0.5133289098739624)]

Example 6:

gloVe:

Given: print(model.most_similar(positive=['example','word'],negative=['examples']))

Expected: words
Predicted: Most similar is **'meaning'**
[('meaning', 0.7650400400161743), ('phrase', 0.7583110928535461), ('name', 0.7272217869758606), ('means', 0.7257716655731201), ('simply', 0.7160940170288086), ('instance', 0.6973064541816711), ('comes', 0.6926121711730957), ('call', 0.6922633051872253), ('does', 0.6922317743301392), ('same', 0.6896651983261108)]

Word2Vec:

Given: print(model.most_similar(positive=['example','word'],negative=['examples']))
Expected: words
Predicted: Most similar is **'phrase'**
[('phrase', 0.5237422585487366), ('verb', 0.49598753452301025), ('synonym', 0.46047794818878174), ('instance', 0.4505717158317566), ('adjective', 0.43867892026901245), ('acronym', 0.4164850413799286), ('noun', 0.40013664960861206), ('descriptor', 0.39837026596069336), ('colloquialism', 0.3982667326927185), ('cuss_word', 0.3902365267276764)]