COMPSCI 4NL3 Homework 2: Corpus Analysis

Alan Zhou

February 2025

1. **Embeddings**

   Embeddings trained:

   1. Skip-gram Model
   2. CBOW (Continuous Bag of Words) Model

   Skip-gram Model:

   - Library: `Word2Vec` from Gensim
   - Hyperparameters:
     - `vector_size=100` (Embedding dimension of 100)
     - `window=5` (Context window size of 5 words)
     - `sg=1` (Skip-gram architecture)
     - `min_count=5` (Ignores words with frequency less than 5)
     - `workers=4` (Uses 4 CPU cores for training)
   - Learnings: Skip-gram is often better for capturing rare words and relationships in smaller datasets due to its focus on predicting context words.

   CBOW (Continuous Bag of Words) Model:

   - Library: `Word2Vec` from Gensim
   - Hyperparameters:
     - `vector_size=200` (Embedding dimension of 200)
     - `window=10` (Larger context window size of 10 words)
     - `sg=0` (CBOW architecture)
     - `min_count=5` (Ignores words with frequency less than 5)
     - `workers=4` (Uses 4 CPU cores for training)
   - Learnings: CBOW seems to be faster and performs better on frequent words due to its averaging of context vectors.

   The five queries that I chose are as follows:

   (a) `"basketball - ball + basket"`

   (b) `"piano"`

   (c) `"bottle - water + sink"`

   (d) `"apple"`

   (e) `"car + road"`

Results:

```
Query: basketball - ball + basket
Skip-gram Results:
[('tennis', 0.6301530599594116), ('nba', 0.6290186047554016), ('wnba', 0.6182593703269958), ('draftees', 0.6087247729301453), ('celtics', 0.5963279604911804), ('volleyball', 0.5891280770301819), ('nets', 0.5693768858909607), ('clippers', 0.568810582160949 7), ('trailblazers', 0.5650078058242798), ('76ers', 0.5606467127799988)]
CBoW Results:
[('corn', 0.4233836531639099), ('cheesecake', 0.41933536529541016), ('carrot', 0.41884684562683105), ('squash', 0.4143407642841339), ('apricot', 0.41349685192108154), ('wnba', 0.4133473038673401), ('spice', 0.4114230275154114), ('anise', 0.41119593381881714), ('strawberry', 0.4111737012863159), ('ornamental', 0.4108926057815552)]
--------------------------------------------------
Pretrained Model 1 Results:
[('badminton', 0.8803533911705017), ('futsal', 0.843710720539093), ('rugby', 0.8186061382293701), ('alumni', 0.8047816157341003), ('netball', 0.7972772717475891), ('derby', 0.7896385788917542), ('tournament', 0.7812114953994751), ('varsity', 0.780616819858551), ('gp', 0.7690523266792297), ('softball', 0.7619743943214417)]
--------------------------------------------------
Pretrained Model 2 Results:
[('badminton', 0.7616748213768005), ('futsal', 0.7421615719795227), ('soccer', 0.719953715801239), ('softball', 0.7063926458358765), ('varsity', 0.7058970928192139), ('volleyball', 0.7039079666137695), ('rugby', 0.7006819844245911), ('football', 0.6885684728622437), ('tennis', 0.6829694509506226), ('alumni', 0.6766970157623291)]
```

```
Query: piano
Skip-gram Results:
[('violin', 0.8489059209823608), ('cello', 0.8310992121696472), ('harpsichord', 0.806052029132843), ('orchestral', 0.8035516738891602), ('concerto', 0.8021896481513977), ('clarinet', 0.7992780208587646), ('violins', 0.7922540307044983), ('concertos', 0.7920403480529785), ('sonatas', 0.7917948961257935), ('accordion', 0.7873266935348511)]
CBoW Results:
[('cello', 0.7972230911254883), ('violin', 0.7946211099624634), ('concerto', 0.7783765196800232), ('harpsichord', 0.7649140357971191), ('clarinet', 0.7645699381828308), ('pianos', 0.7583957314491272), ('orchestral', 0.7560009956359863), ('sonatas', 0.7349601984024048), ('concertos', 0.7339935302734375), ('sonata', 0.7247752547264099)]
--------------------------------------------------
Pretrained Model 1 Results:
[('guitar', 0.8560521006584167), ('playlist', 0.8479223251342773), ('cover', 0.8451604247093201), ('gangnam', 0.8332557678222656), ('dragon', 0.8329268097877502), ('saga', 0.8301511406898499), ('violin', 0.8296527862548828), ('metal', 0.8292565941810608), ('version', 0.8258488178253174), ('acoustic', 0.8204707503318787)]
--------------------------------------------------
Pretrained Model 2 Results:
[('guitar', 0.8645602464675903), ('violin', 0.8496300578117371), ('drum', 0.7750863432884216), ('bass', 0.7730393409729004), ('ukulele', 0.7599596977233887), ('vocal', 0.7515949606895447), ('cello', 0.7394928336143494), ('flute', 0.7386735677719116), ('saxophone', 0.7354364991188049), ('acoustic', 0.7280368208885193)]
```

```
Query: bottle - water + sink
Skip-gram Results:
[('tossed', 0.6591199636459351), ('ramrod', 0.6422083377838135), ('slipping', 0.6417225003242493), ('tosses', 0.640838086605072), ('wipe', 0.6390410661697388), ('screwdriver', 0.6384618282318115), ('mattress', 0.6382548213005066), ('blasting', 0.634975254535675), ('unexploded', 0.633830726146698), ('toothbrush', 0.6328871846199036)]
CBoW Results:
[('bag', 0.6303448677062988), ('bubble', 0.6009581685066223), ('stuck', 0.5846257209777832), ('cans', 0.5808385610580444), ('accidentally', 0.5754303932189941), ('wrap', 0.5748594403266907), ('hook', 0.5718513131141663), ('kisses', 0.5674431324005127), ('ignited', 0.565552830696106), ('crack', 0.5655403137207031)]
--------------------------------------------------
Pretrained Model 1 Results:
[('broom', 0.9117616415023804), ('fold', 0.9005420207977295), ('brush', 0.8974438309669495), ('comb', 0.8955177068710327), ('wax', 0.8934656977653503), ('rope', 0.8920356631278992), ('towel', 0.8857064247131348), ('stain', 0.8786723017692566), ('trunk', 0.8769235610961914), ('tuck', 0.8738833069801331)]
--------------------------------------------------
Pretrained Model 2 Results:
[('nickel', 0.7280511260032654), ('shelf', 0.721893846988678), ('shade', 0.7202038764953613), ('ceiling', 0.719837486743927), ('knob', 0.7159472107887268), ('towel', 0.7101408839225769), ('straw', 0.6966026425361633), ('tub', 0.6868497729301453), ('candle', 0.6834232211112976), ('flask', 0.6824853420257568)]
```

```
Query: apple
Skip-gram Results:
[("apple's", 0.7625170350074768), ('macs', 0.761378824710846), ('macintoshes', 0.7469151616096497), ('iphone', 0.744951725
0061035), ('ipad', 0.7447630167007446), ('ipados', 0.7422172427177429), ("inc.'s", 0.7314006686210632), ('powerbook', 0.72
81565070152283), ('ios', 0.7253535985946655), ('smartphone', 0.7188824415206909)]
CBoW Results:
[('iphone', 0.7245023846626282), ("apple's", 0.6781405806541443), ('ipad', 0.6597625613212585), ('ipod', 0.656492292881012
), ('ios', 0.6563741564750671), ('macintosh', 0.6397703289985657), ('smartphone', 0.6224755644798279), ('linux', 0.5985157
489776611), ('microsoft', 0.5954751372337341), ('app', 0.594367265701294)]
--------------------------------------------------
Pretrained Model 1 Results:
[('windows', 0.8948712944984436), ('microsoft', 0.8858076333999634), ('google', 0.8823867440223694), ('galaxy', 0.88063913
58375549), ('flash', 0.8793812394142151), ('android', 0.8782057762145996), ('nokia', 0.8770236372947693), ('samsung', 0.86
97316646575928), ('chrome', 0.8691699504852295), ('ipad', 0.8670315742492676)]
--------------------------------------------------
Pretrained Model 2 Results:
[('microsoft', 0.8666837811470032), ('google', 0.8337891697883606), ('samsung', 0.8189260363578796), ('nokia', 0.813397049
9038696), ('ipad', 0.806841254234314), ('iphone', 0.806834876537323), ('galaxy', 0.8041617274284363), ('blackberry', 0.795
3041791915894), ('android', 0.7887364625930786), ('smartphone', 0.788325846195221)]
```

```
Query: car + road
Skip-gram Results:
[('pickup', 0.448901891708374), ('roadster', 0.44671204686164856), ('camaro', 0.44426578283309937), ('chevrolet', 0.411176
860332489), ('suv', 0.40761202573776245), ('truck', 0.4063415229320526), ('lamborghini', 0.4037291705608368), ('punched',
0.39072537422180176), ('jalpa', 0.3824504017829895), ('briatore', 0.3806528151035309)]
CBoW Results:
[('roadster', 0.44162485003471375), ('camaro', 0.431618332862854), ('lamborghini', 0.428401380777359), ('santos-dumont', 0
.41214752197265625), ('engine', 0.4066927433013916), ('citroen', 0.40443840622901917), ('convertible', 0.4021846055984497)
, ('bmw', 0.39587101340293884), ('mercedes', 0.3930000066757202), ('citroën', 0.39264267683029175)]
--------------------------------------------------
Pretrained Model 1 Results:
[('écouteur', 0.8014513850212097), ('appels', 0.7671477198600769), ('smileys', 0.743070662021637), ('inconnu', 0.734136700
630188), ('tatouages', 0.7309343814849854), ('icone', 0.7269665598869324), ('statuts', 0.7230100035667419), ('prénom', 0.7
229117751121521), ('statut', 0.7151960730552673), ('envoies', 0.7150381207466125)]
--------------------------------------------------
Pretrained Model 2 Results:
[("l'iphone", 0.6552570462226868), ('batterie', 0.6096706390380859), ("l'ipad", 0.6080988645553589), ('garder', 0.60802680
25398254), ('acheté', 0.6014876365661621), ("l'ambition", 0.6006343364715576), ('changé', 0.5945946574211121), ('appareil'
, 0.5875667333602905), ('copain', 0.5761011838912964), ('tablette', 0.5735448002815247)]
```

Comments about results:

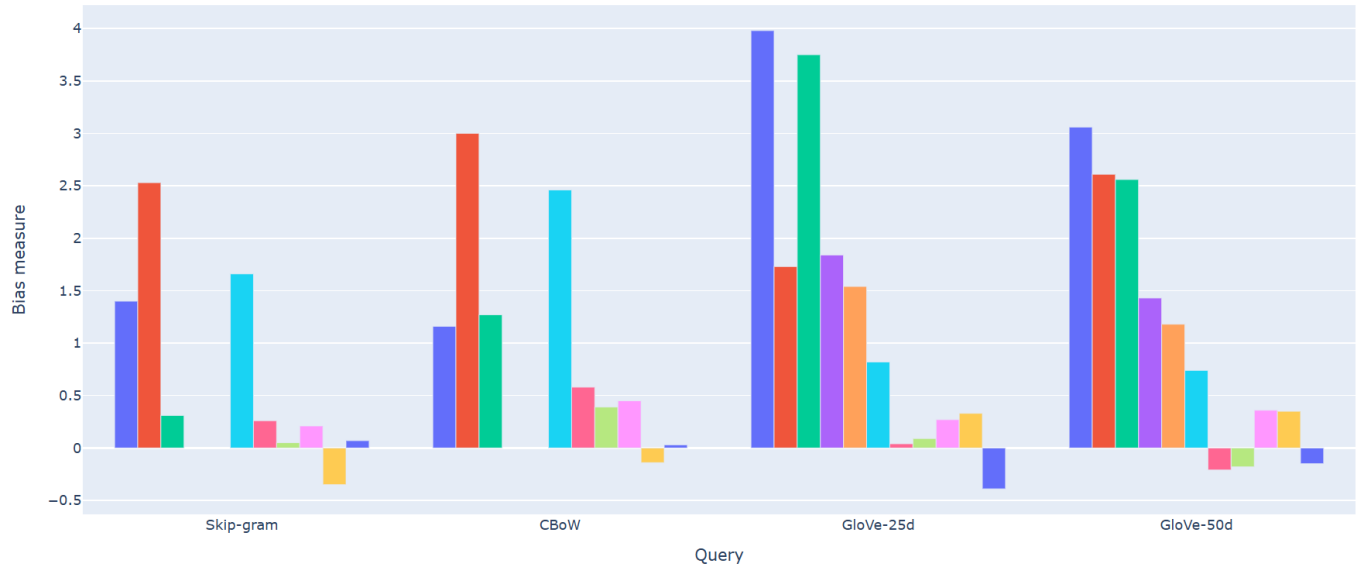
With respect to my queries, I found the results of the singular word `"apple"` to be quite interesting. The **skip-gram's top results** were: braeburn, blackberry, iphone, inc.'s, and imac, while the **CBOW's top results** were: iphone, apple's, ipod, ios, and macintosh. This is noteworthy because it appears that the skip-gram model handles the different possible meanings of `"apple"` better than the CBOW model. The skip-gram mentions `braeburn` (a variety of apple that I did not even know about until searching it), as well as words related to the tech company. On the other hand, CBOW focuses more on the tech company, as nothing in the top is related to fruit.

In addition, both **pre-trained models** returned words including: windows, microsoft, google, android reflecting their training on modern corpora. This is all interesting because although Apple dominates the tech market, especially in North America, I still feel shocked that there are so few words outputted that are related to the fruit (let alone it being one of the most common in the world).

2. **Bias**

   - Original: WEAT for gender bias
   - Extension: Added a new query for age bias using custom word sets (these custom sets were generated using ChatGPT):
     - Target sets: `young_people_names` and `old_people_names`
     - Attribute sets: `positive_attributes` and `negative_attributes`

   Results:




```
Testing for Gender Bias:

Test: man : king :: woman : ?
Skip-gram Results:
[('queen', 0.6886000633239746), ('monarch', 0.6129544973373413), ('consort', 0.5985515117645264), ('ugyen', 0.5946809649467468), ("vi's", 0.5920376181602478)]
CBoW Results:
[('queen', 0.5800480842590332), ('consort', 0.5582652688026428), ('throne', 0.5281981229782104), ('regent', 0.5064131617546082), ('princess', 0.49291056394577026)]
Pretrained Model 1 Results:
[('meets', 0.8841924071311951), ('prince', 0.832163393497467), ('queen', 0.8257461190223694), ('’s', 0.8174097537994385), ('crow', 0.813499391078949)]
Pretrained Model 2 Results:
[('prince', 0.759779691696167), ('stone', 0.7595877051353455), ('queen', 0.754626989364624), ('meets', 0.7404001355171204), ('royal', 0.7376410365104675)]
```

```
Test: man : engineer :: woman : ?
Skip-gram Results:
[('aeronautical', 0.6761454939842224), ('engineering', 0.6187294721603394), ('cardiologist', 0.6126991510391235), ('academic', 0.6025742292404175), ('epidemiologist', 0.5891098380088806)]
CBoW Results:
[('aeronautical', 0.5338456034660339), ('engineering', 0.5073961019515991), ('balloonist', 0.49392542243003845), ('technician', 0.491915225982666), ('educator', 0.48091983795166016)]
Pretrained Model 1 Results:
[('representative', 0.8719702959060669), ('vacancy', 0.8590602278709412), ('logistics', 0.8582232594490051), ('consulting', 0.8555269837379456), ('realty', 0.8547706007957458)]
Pretrained Model 2 Results:
[('technician', 0.7694835662841797), ('engineering', 0.7573692202568054), ('consulting', 0.7528563141822815), ('representative', 0.7497202157974243), ('architect', 0.7484485507011414)]
```

```
Test: man : doctor :: woman : ?
Skip-gram Results:
[('medical', 0.6500079035758972), ('surgeon', 0.6472622156143188), ("doctor's", 0.6325979828834534), ('juris', 0.6170447468757629), ('osteopathic', 0.6161447167396545)]
CBoW Results:
[('nurse', 0.5959671139717102), ('surgeon', 0.5270876884460449), ("doctor's", 0.5112552642822266), ('doctors', 0.4894491136074066), ('medical', 0.4700901210308075)]
Pretrained Model 1 Results:
[('finds', 0.8187651038169861), ('child', 0.8062381744384766), ('father', 0.8005850911140442), ('grandmother', 0.7968820333480835), ('bartender', 0.7911259531974792)]
Pretrained Model 2 Results:
[('child', 0.7199426889419556), ('grandmother', 0.6972634196281433), ('birth', 0.6851900815963745), ('mother', 0.6771568059921265), ('children', 0.6721060276031494)]
```

Comments about results:

Gender Bias in Analogies:

Strengths:

- All models correctly identify `"queen"` as the female equivalent of `"king"`, demonstrating their ability to handle simple gender-based analogies.
- Skip-gram and CBOW perform well on domain-specific analogies (e.g., `"man : engineer :: woman : ?"`) but produce noisy results.

Weaknesses:

- Models struggle with compound or less frequent terms (e.g., `"computer_programmer"`).
- Pretrained models often produce noisy or less coherent results, possibly due to biases in their training data.

Vocabulary Limitations:

- The inability of all models to handle `"computer_programmer"` highlights a limitation in their vocabularies, especially for compound or niche terms.
- This suggests that custom-trained models may need larger and more diverse training corpora to improve their performance.

Bias in Pretrained Models

- Pretrained models often return noisy or unrelated terms (e.g., *finds*, *child*, *bartender* for `"man : doctor :: woman : ?"`).
- This may reflect biases in their training data, where certain professions or roles are stereotypically associated with specific genders.

Performance Differences

- **Skip-gram vs. CBOW**: Skip-gram generally performs better on analogical reasoning tasks, while CBOW produces slightly noisier results.
- **Pretrained vs. Custom Models**: Pretrained models generalize better but often produce less coherent results, while custom models show potential for domain-specific tasks but require more data and tuning.

Consequences of Using These Embeddings

Bias Amplification

- If these embeddings are used as features in a machine learning model, they could perpetuate and amplify existing biases (e.g., associating `"nurse"` with women and `"engineer"` with men).
- This could lead to unfair predictions in sensitive applications like hiring or loan approvals.

Fairness Concerns

- Models trained on biased embeddings may make unfair or discriminatory predictions, particularly in applications involving gender, race, or other sensitive attributes.
- For example, a resume screening model might unfairly favor male candidates for engineering roles.

3. **Classification**

Results:

```
Comparison of Models:
BoW Model - Accuracy: 0.85, F1 Score: 0.85
Word Embedding Model - Accuracy: 0.475, F1 Score: 0.3059322033898305
```

Model Performance

- **BoW Model**:

  - **Accuracy**: 0.85
  - **F1 Score**: 0.85
- **Word Embedding Model**:
  - **Accuracy**: 0.475
  - **F1 Score**: 0.3059

Task: Binary classification of text documents into *win* or *lose* categories.

Comparison of Models

- The **BoW Model** significantly outperforms the **Word Embedding Model** in both accuracy and F1 score.
- The BoW Model achieves an accuracy of 85% and an F1 score of 85%, indicating strong performance.
- The Word Embedding Model, on the other hand, performs poorly, with an accuracy of 47.5% and an F1 score of 30.59%.

Analysis Why BoW Model Performed Better

- The BoW Model (likely a Bag-of-Words or similar traditional model) appears to be well-suited for this task, possibly because the task relies heavily on word frequency or specific keywords.
- Traditional models like BoW are often effective for tasks where word presence or frequency is a strong indicator of class labels.

Why Word Embedding Model Performed Poorly

- The Word Embedding Model may struggle due to the nature of the task or the dataset.
- Word embeddings capture semantic relationships, which may not be as important for this classification task.
- The poor performance could also be due to insufficient training data, improper tuning, or the embeddings not capturing task-specific features effectively.

4. **Reflection**

- **Training Embeddings**: I learned how to train custom word embeddings using Gensim and the impact of hyperparameters like `vector_size`, `window`, and `sg` on model performance.
- **Bias in Embeddings**: I gained a deeper understanding of how biases in training data can propagate into embeddings and downstream applications.
- **Classification with Embeddings**: I learned how to use word embeddings as features in a machine learning model and compare their performance to traditional methods like BoW.

Challenges and Solutions

- **Challenge**: Out-of-vocabulary words in custom-trained models.
  - **Solution**: Used pretrained embeddings as a fallback and explored techniques like subword embeddings.
- **Challenge**: Bias detection and mitigation.
  - **Solution**: Extended the WEAT framework to include age bias and explored debiasing techniques.