NLP with Deep Learning
Mini Project 1

Name: Poojith Reddy Maligireddy
UID: u1405749

1)Below is the output (best learning rate is 0.001 and the lowest development set loss is 5.947) after running main.py:

The lowest development set loss of 5.947161763167578 is with learning rate 0.001.

2)Below is the output after running report_ques.py file:

Similarity score for 'cat' and 'tiger': 0.8942.
Similarity score for 'plane' and 'human': 0.8804.
Similarity score for 'my' and 'mine': 0.8685.
Similarity score for 'happy' and 'human': 0.8689.
Similarity score for 'happy' and 'cat': 0.8767.
Similarity score for 'king' and 'princess': 0.8644.
Similarity score for 'ball' and 'racket': 0.9115.
Similarity score for 'good' and 'ugly': 0.8473.
Similarity score for 'cat' and 'racket': 0.9211.
Similarity score for 'good' and 'bad': 0.8754.

king:queen, man:queen
king:queen, prince:prince
king:man, queen:queen
woman:man, princess:princess
prince:princess, man:man

Below are the answers using above output:

(a) (i) [cat, tiger] or [plane, human] → Answer is [cat, tiger]
(ii) [my, mine] or [happy, human] → Answer is [happy, human]
(iii) [happy, cat] or [king, princess] → Answer is [happy, cat]
(iv) [ball, racket] or [good, ugly] → Answer is [ball, racket]
(v) [cat, racket] or [good, bad] → Answer is [cat, racket]

(b) (i)king:queen, man: ? → Answer is queen
(ii) king:queen, prince: ? → Answer is prince
(iii) king:man, queen: ? → Answer is queen
(iv) woman:man, princess: ? → Answer is princess
(v) prince:princess, man: ? → Answer is man

3) Below are three examples each for word similarity test:

["front", "back"] and ["chair", "racket"]
["open", "close"] and ["game", "ship"]
["rich", "poor"] ["farmer", "ring"]

But, the similarity scores given by my code for above examples as below:

Similarity score for 'front' and 'back': 0.8397.
Similarity score for 'chair' and 'racket': 0.9216.
Similarity score for 'open' and 'close': 0.9135.
Similarity score for 'game' and 'ship': 0.9143.
Similarity score for 'rich' and 'poor': 0.8887.
Similarity score for 'farmer' and 'ring': 0.8849.

Below are three examples each for word analogy test:

happy:joyful, sad:gloomy
fast:running, slow:walking
east:west, north:south

But, the results given by my code for above examples as below:

happy:joyful, sad:joyful
fast:running, slow:running
east:west, north:north

4)Below is the output after running eval_embs.py with necessary command line arguments by using embeddings.txt file which we generated.

Word Similarity Test Pearson Correlation: 0.2585294850515952.
Accuracy on Analogy Test: 0.03466872110939907.

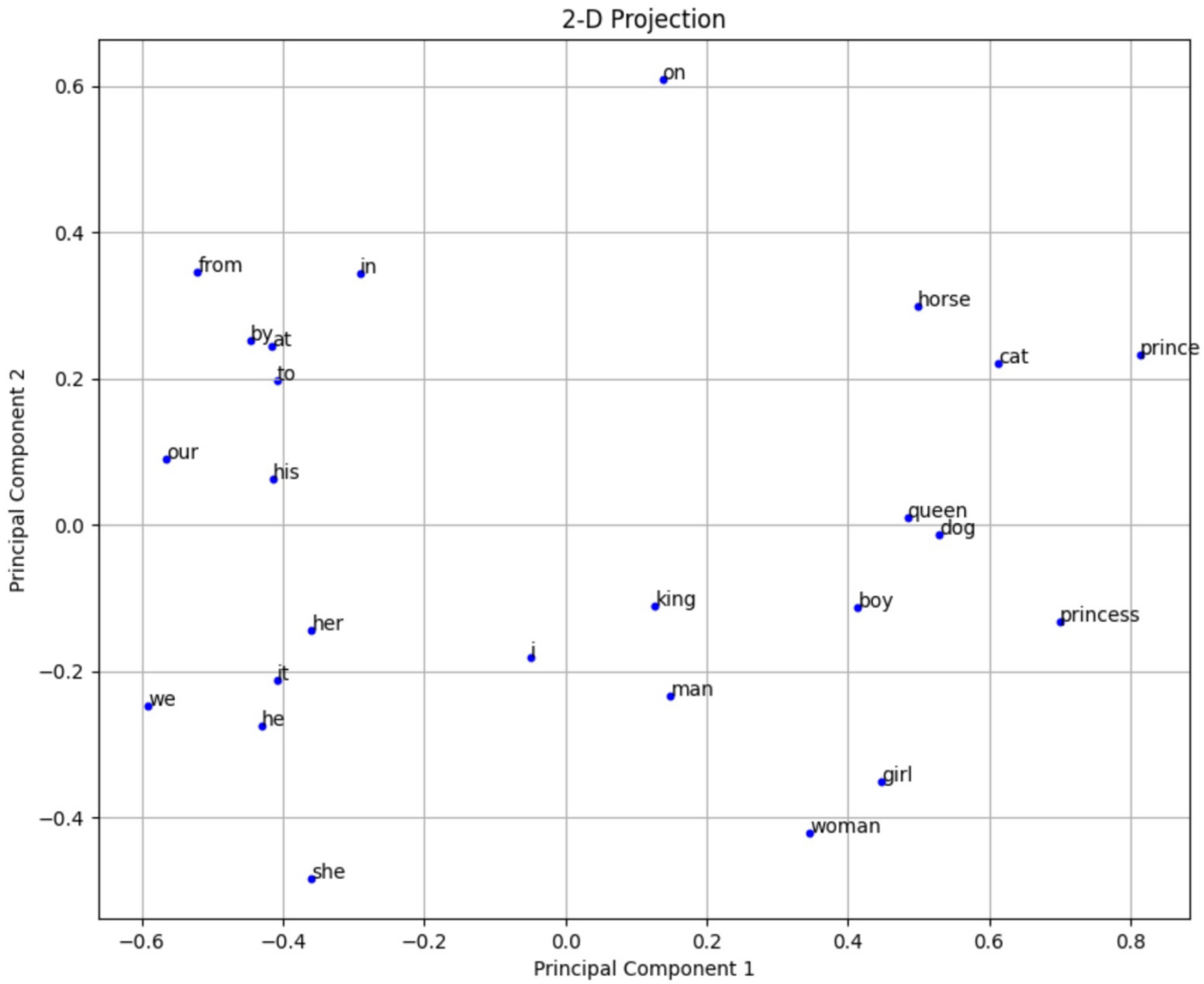
5) In CBOW, the primary objective is to learn word embeddings, where each word is represented by a continuous vector in a higher dimensional space. These embeddings capture semantic relationships between words by considering the local context of words within sentences. The paper "Bag of Tricks for Efficient Text Classification" focuses on text classification tasks and explores various techniques to improve the train and test time efficiency of models.

The "Bag of Tricks" paper introduces the fastText classifier, in which we use a bag of n-grams as additional features to capture some partial information about the local word order instead of Bag of words which is invariant to word order. This is different from CBOW where we typically deal with whole words. It also talks about hierarchical softmax technique to efficiently reduces the computational complexity during training and also it is advantageous at test time when searching for the most likely class.

In addition, In the "Bag of Tricks" paper, the word representations are averaged into a text representation, which is in turn fed to a linear classifier. The text representation is a hidden variable which can be potentially be reused. This architecture is similar to the CBOW model, where the middle word is replaced by a label. In summary, while CBOW focuses on learning word embeddings by predicting words based on their context, the "Bag of Tricks" paper addresses practical techniques for improving the efficiency and performance of text classification models by handling large vocabularies and improving training and testing speed.

Extra Credit:

1)Below is the plot generated after running projections.py by using PCA.



Below are my observations:

In this 2-D projection of word embeddings, each point represents a word. In above plot, we could see prepositions, pronouns, nouns are almost forming separate clusters. Also, we could see prepositions and pronouns are almost on the left half of the plot and nouns are on the right half of the plot. This projection captures some relationships between words but loses other details like relation between nouns and pronouns when projected in lower dimension.

Reasons for why a 2-D Projection May Be Misleading:

PCA is a dimensionality reduction technique which helps to visualize high-dimensional data in lower dimensions. Word embeddings typically have very high dimensions, which allow them to capture rich semantic relationships between words. But, from projecting high-dimensional word embeddings into just two dimensions may result in significant information loss and may not fully capture the semantic relationships between words. The relationships such as analogies and subtle semantic differences may not be well preserved in a 2-D projection. In other words, using higher-dimensional embeddings can capture more information and gives better semantic relationships.