Natural Language Processing - CS6120
Assignment 3 Report

PART I - WORD REPRESENTATION

Introduction

The goal of this assignment is to create deeper insights into the world of word representation and word embeddings. We are using the Simpson data data corpus from the this URL.

1.1 DATA PREPARATION AND PREPROCESSING

The dataset used for this assignment contains two columns: **raw_character_text**: Names of characters from "The Simpsons" , **spoken_words**: Actual dialogues spoken by each character.

Before applying TF-IDF and Word2Vec, we prepared the data through the following steps:
- Dropping NaN and Lowercasing : We drop all the rows with NaN and convert the corpus into lowercase.
- Removal of Punctuation and Stop Words: All the punctuation is removed using regular expression. The stop-words are handled using the NLTK library.

After preprocessing, we group the data by characters (Lisa Simpson, Bart Simpson, Homer Simpson) , resulting in a document for each character containing all their spoken words.

1.2 WORD FREQUENCY ANALYSIS

For the characters "Lisa Simpson," "Bart Simpson," and "Homer Simpson," we filtered out words with fewer than 3 letters and calculated word frequencies for each character's dialogue.

```
2most frequently spoken words by Lisa Simpson, Bart Simpson, Homer Simpson:
dont: 2766 times
well: 2294 times
```

1.3 TF- IDF CALCULATION

We implemented TF-IDF from scratch. The following steps outline the approach:

Step 1 : **Calculating Term Frequency (TF)**
Term Frequency (TF) measures how often a word appears in a document relative to the total number of words in that document. We use Python's *Counter* to compute relative frequency of each word. We then calculate TF for all documents.

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

Step 2 : **Calculating Document Frequency (DF)**
Document Frequency (DF) counts how many documents contain a particular word. We convert each document's list of words into set to make sure there is no duplication & then calculate DF for all documents.

Step 3 : **Calculating Inverse Document Frequency (IDF)**
IDF measures how common or rare a word is across all documents. The words that appear in many documents are less informative. IDF penalises the words that occur in many documents. In our code, *idf_values* contains the IDF score for each word across all documents.

$$IDF = log(\frac{\text{number of the documents in the corpus}}{\text{number of documents in the corpus contain the term}})$$

Step 4 : **Calculating TF-IDF**
This step calculates the TF-IDF scores for every word in each document.

$$TF\text{-}IDF = TF * IDF$$

Results for TF IDF

```
TF-IDF Results for Lisa Simpson:
bart - TF: 0.0132, IDF: 2.9913, TF-IDF: 0.0395
dont - TF: 0.0108, IDF: 1.8655, TF-IDF: 0.0202

TF-IDF Results for Bart Simpson:
dont - TF: 0.0102, IDF: 1.8655, TF-IDF: 0.0189
like - TF: 0.0079, IDF: 1.8134, TF-IDF: 0.0144

TF-IDF Results for Homer Simpson:
marge - TF: 0.0118, IDF: 3.2971, TF-IDF: 0.0390
dont - TF: 0.0101, IDF: 1.8655, TF-IDF: 0.0189
```

1.4 IMPLEMENTING WORD2VEC

We followed the guidelines on the following page and using the corpus above to train a skip-gram model from scratch: **https://www.kaggle.com/code/pierremegret/gensim-word2vec-tutorial/notebook** Spacy is used for text preprocessing - tokenization and lemmatization. The stop-words are removed since they do not contribute to the meaning of the text.

1.4.1 Creating Bigrams using Gensim's Phrases function

Phrases ( ) looks for common pairs of words called *Bigrams*, (like "New York" or "high school") in a bunch of sentences. After training the *Phrases* model, we can use the *Phraser* to transform new sentences.

1.4.2 Word2Vec Model Training

8 Word2Vec models were trained using the preprocessed text. The models were trained for 30 epochs to learn word embeddings using a Learning Rate of 0.03 .

- Model 1: A context window of 5, vector size of 100, and 5 negative samples.
- Model 2: A larger context window of 10, vector size of 300, and 15 negative samples.

1.4.3 Results and Hyper parameters

Window Size : number of words considered around a target word when training the model. Smaller window captures more local context between the words whereas a larger window captures a more broader relationship because it includes more distant words.

Vector Size : Smaller size leads to simpler models which may not capture all essence of word meaning but they are faster and efficient

Negative Samples : they are randomly chosen words from the vocabulary that do not appear in the context

1.4.4 t-SNE Visualization

- In both plots, the word "Homer" is highlighted in **red**. It serves as the focal point for analyzing its relationships with other words.
- Words that are most similar to "Homer" are marked in **blue**. These words should cluster closely around "Homer" because they are similar according to the model.
- Randomly selected words from the vocabulary are represented in **green**. Their distribution helps provide context for the clustering of the blue words around "Homer."

Model 1 graph- tight clusters close to "Homer" . Bird, Duff, Maude and Bob are far from central point indicating model is learning to distinguish them from the words closely related to Homer. Mel is unrelated to Homer.

Model 2 - scattered related words, showing weak clusters. "Heck" and "Snuggle" are closer to Homer. Bird, Duff, Maude and Bob are more spread out indicating they are unrelated to Homer.

Model 3- Words close to Homer are "Shoulda" . "comfort". "rude" , "drunk" which are describing the behavior associated with Homer. Bird , Bob, Dog, Ah are unrelated .

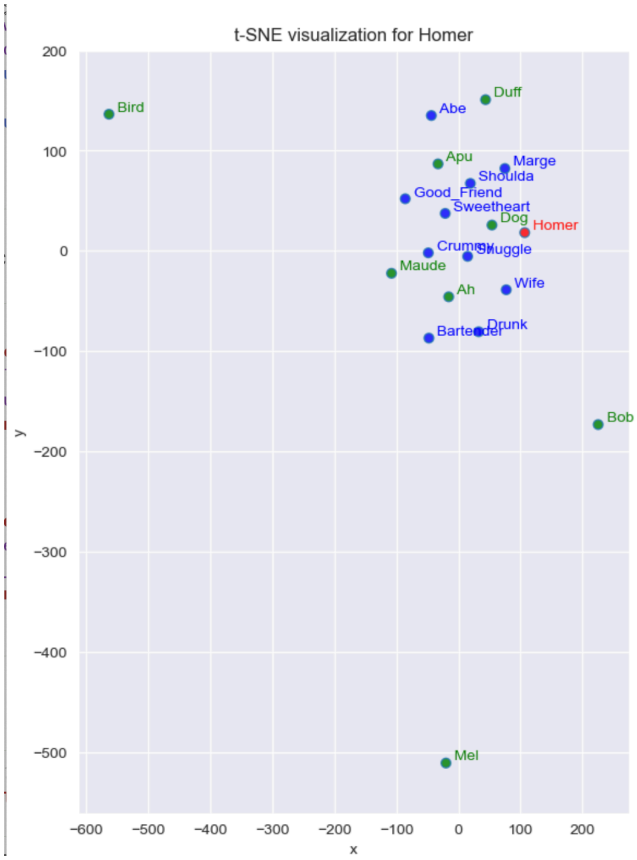
Model 4- Words such as "Awfully", "sweetheart" "marge" are closer to Homer these are behavioral words and this graph shows emotional / behavioural words grouping.

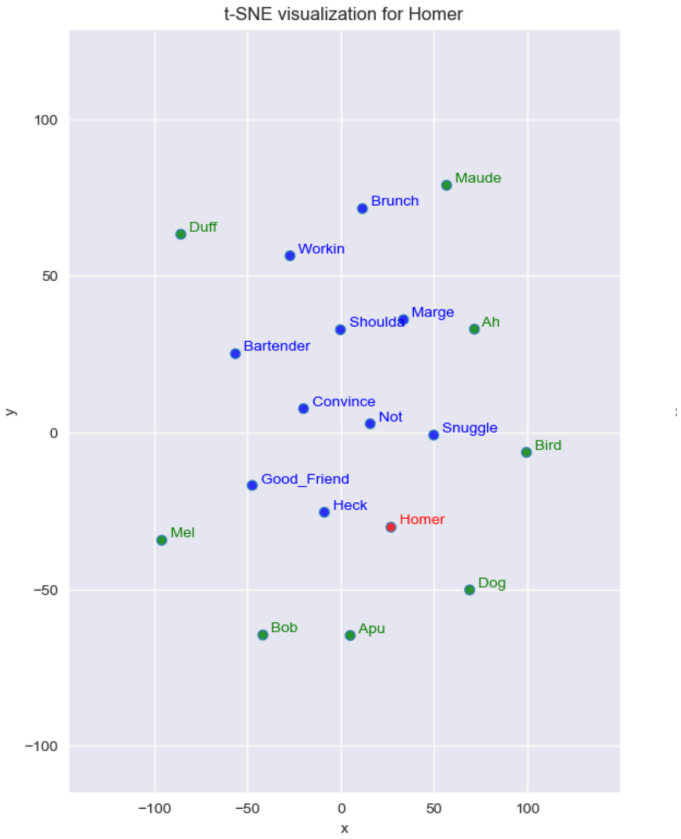
Model 5- "Rude," "Convince," and "Ah." are near to Homer. The spread suggests some clustering around personality traits or words related to Homer's interactions, with groups like "Marge," "Duff," and words like "Brunch" closer to each other.

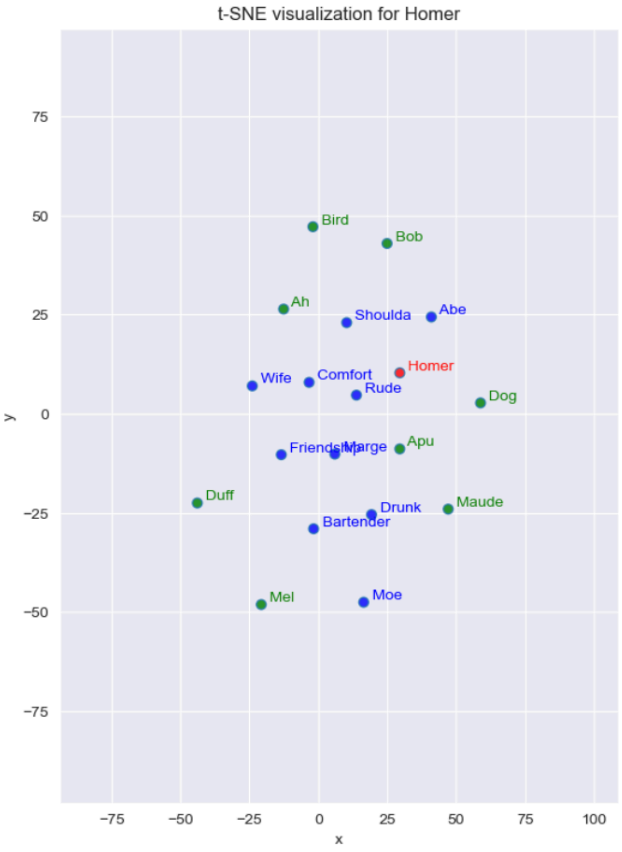
Model 6- The clusters look tighter. "Ashamed," "Sweetheart," and "Pfft" are nearby Homer. Compared to Model 5 plot, "Bob" and "Maude" appear in different locations.

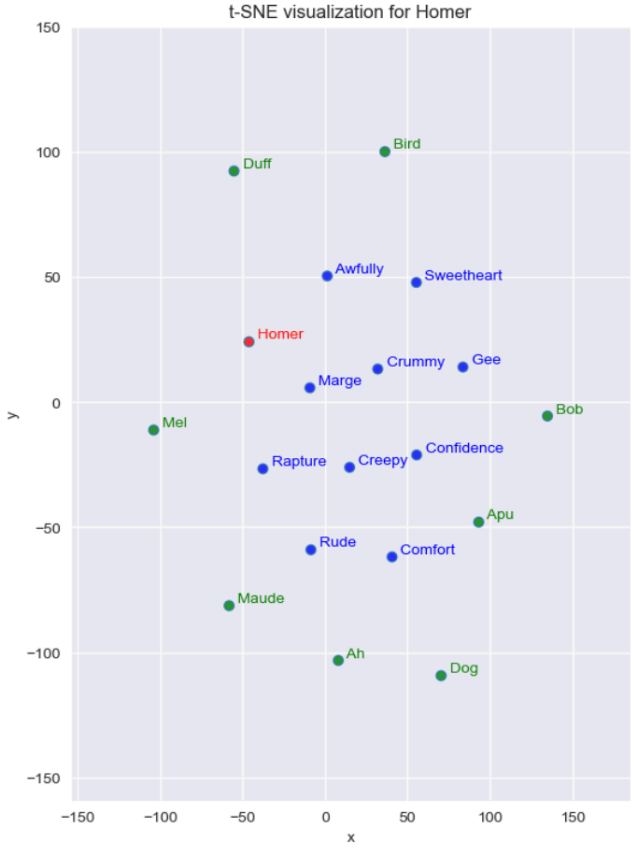
Model 7 - "Homer" is placed farther left, surrounded by words like "Shoulda," "Crummy," and “Snuggle." New words like "Wife" and "Married" show up near characters such as "Apu" and “Drunk” .

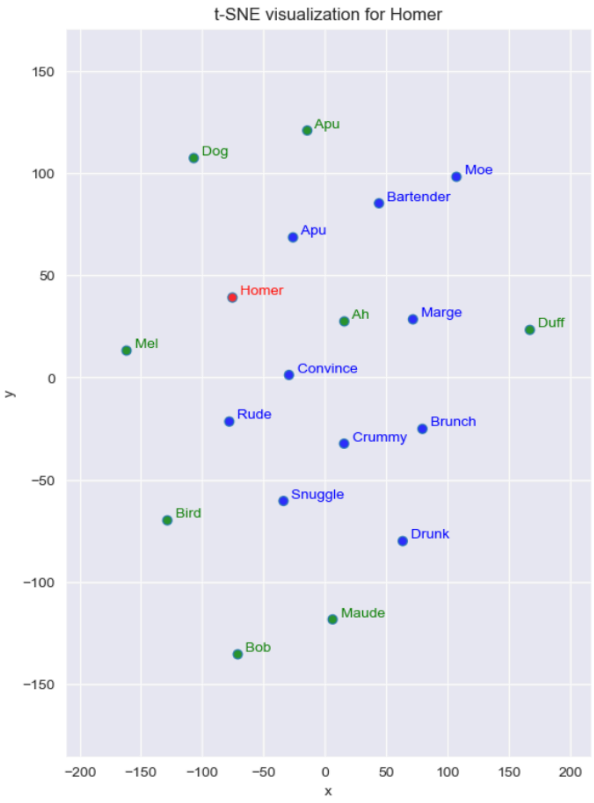
Model 8 - "Brad," "Marge," "Bob," and "Maude," are clustered close to Homer, suggesting more overlap or stronger association. Terms like "Hammock," "Creepy," "Asleep," and "Snuggle" around Homer

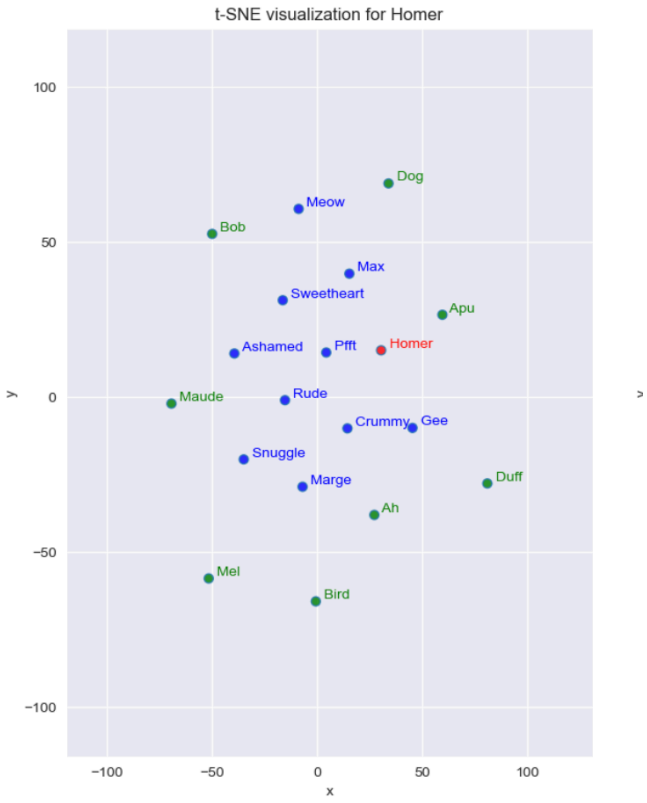


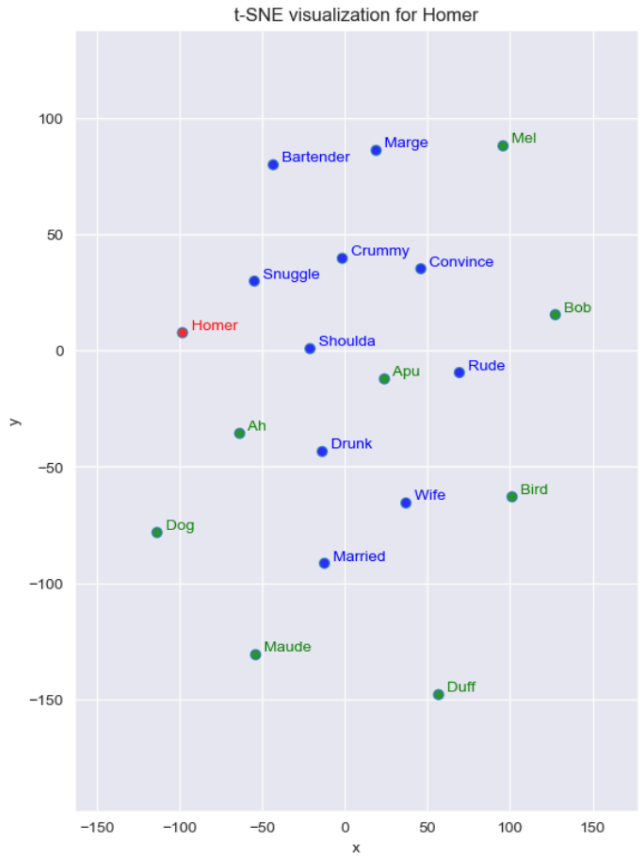


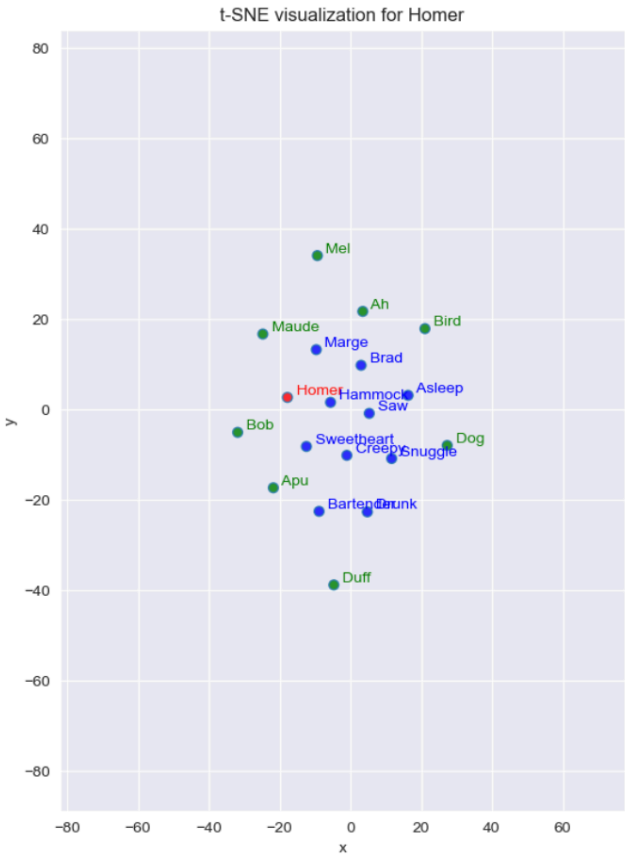





t-SNE visualization for Homer
Apu
Dog
Moe
Bartender
Apu
Homer
Ah
Marge
Duff
Mel
Convince
Rude
Brunch
Crummy
Snuggle
Bird
Drunk
Maude
Bob
x
y


t-SNE visualization for Homer
Dog
Meow
Bob
Max
Sweetheart
Apu
Ashamed
Pfft
Homer
Maude
Rude
Crummy
Gee
Snuggle
Marge
Duff
Ah
Mel
Bird
x
y


t-SNE visualization for Homer
Bartender
Marge
Mel
Crummy
Snuggle
Convince
Bob
Homer
Shoulda
Apu
Rude
Ah
Drunk
Wife
Bird
Dog
Married
Maude
Duff
x
y


t-SNE visualization for Homer
Mel
Ah
Maude
Bird
Marge
Brad
Homer
Hammock
Asleep
Saw
Bob
Sweetheart
Dog
Snuggle
Apu
Duff
x
y