INTERACTIVE RECOMMENDATION SYSTEM WITH WORD EMBEDDINGS USING WORD2VEC, PLOTLY, AND NETWORKX

Objective:

In this project, I explore how to build a model that can understand words in a mathematical way, such that words with similar meanings that share certain character tics belong close to each other in vector space.

Importance:

That's a fancy way of saying that the mathemical representations for words that have some similar meaning are close to each other, and words that don't share a lot of meaning are further apart.

Outcome:

This model can be adapted to be used in a recommendation system, or as a discovery tool for finding new kinds of ingredients. (i.e. we don't have this in stock, maybe you'd like this alternative?).

Structure:

1. preprocess and prepare a text dataset comprising recipes.
2. build an understanding between different kinds of words that make up ingredients that intern make recipes.
3. train and use a Word2Vec model using Gensim.
4. effectively visualize and evaluate a trained model in an interactive graph network using NetworkX and Plotly.
5. Browse **the interactive and annotated 2-D scatter plot**

Project Breakdown

1) Dataset
2) Exploratory Data Analysis and Preprocessing
3) Word2Vec with Gensim
4) Exploring Results
5) Building and Visualizing Interactive Network Graph

1) Dataset:

This is the dataset(https://eightportions.com/datasets/Recipes/#fn:1) we will be using. It is collated by Ryan Lee, sourced from Food Network(https://www.foodnetwork.com/), Epicurious(https://www.epicurious.com/), and Allrecipes(https://www.epicurious.com/).

The columns are created as ‘source’, ‘title’, ‘ingredients’ and ‘instructions.

2) Exploratory Data Analysis and Preprocessing:

First, we removed certain stop words. Because these words might make it more difficult for our model to learn as they are not always informative. In general, it is advisable to test multiple approaches to see if these words may have some value. In our use case, removing them did not hinder performance.

Also, instructions and ingredients are formed in list format and preprocessed by removing the stop words and special characters.

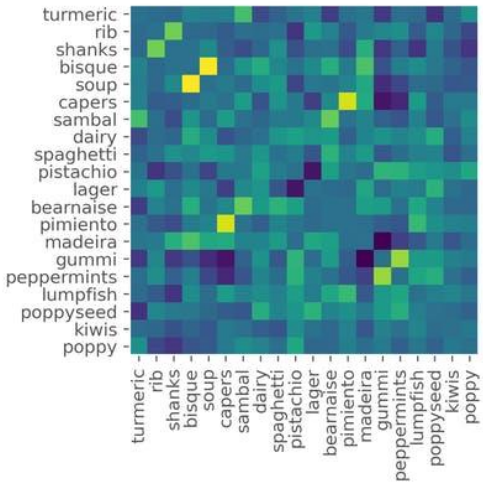
Finally, they are stored after merging to be able to train as data.

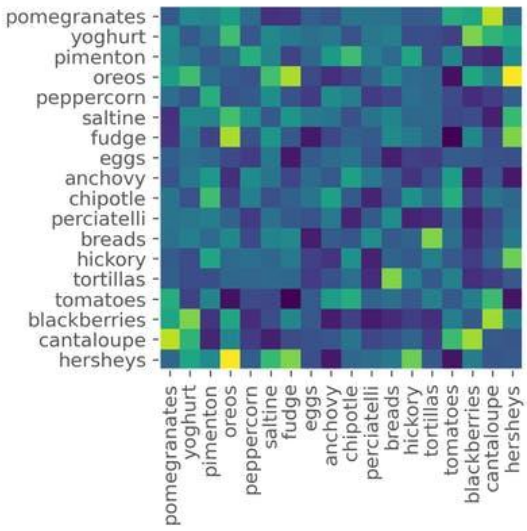
3) Word2Vec with Gensim:

The data is trained by the model of Word2Vec and it is stored as model file.

4) Exploring Results:

Similar words are saved as lists of words and their vectors. The vectors are linked with indices. Their correlation is visualized with heat map and the plot is saved as ‘plot2.png’





Difference Between Continuous Bag Of Words (CBOW) And Skip-Gram (SG) Approaches To Word2Vec:

CBOW uses context to predict a given word while Skip-Gram uses a given word to predict the context.
These models are similar in structure, but have inverted inputs and outputs.

Downsides of Word2Vec in General:

Word2Vec like all NLP models will be biased depending on our training data. For example, our recipes are primarily Western-oriented. Recipes of traditional regions in less-popular areas in the world might not always be represented in this model.

Bias must always be considered when training a model on large amounts of any kind of data!

All distance metrics have to be evaluated every time a new value is introduced. This is expensive and can be slow.

Words not in our Word2Vec model are not handled directly, and will result in an error.

5) Visualizing Interactive Tool using Word Embeddings and Network

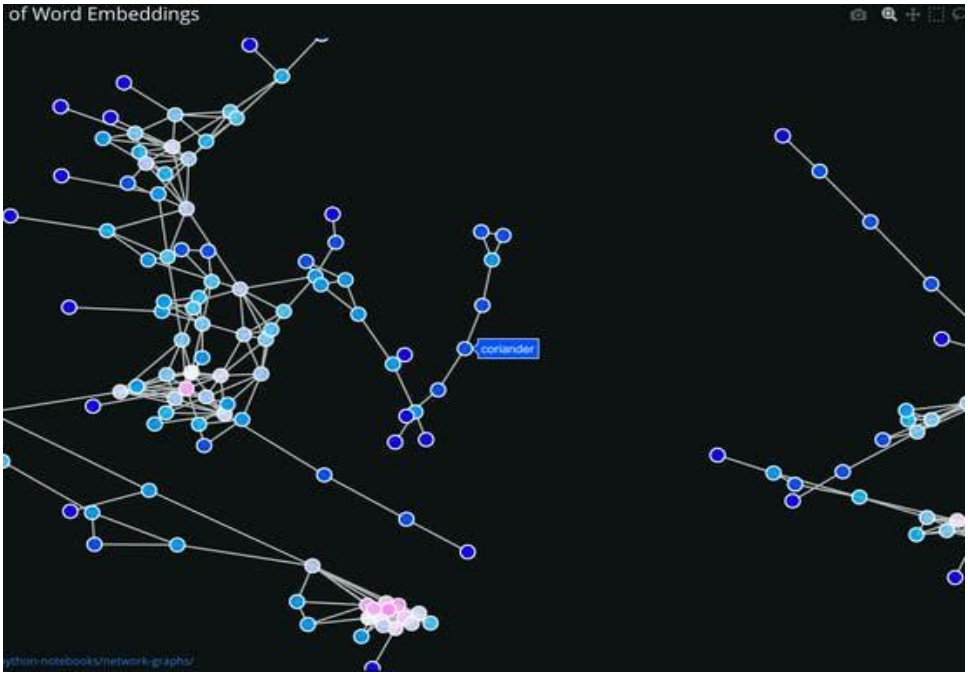
Heatmaps are not effective while visualizing word embeddings because:

--It needs a lot of computer power and it would be very difficult to read hundreds of words on both dimensions in a plot.

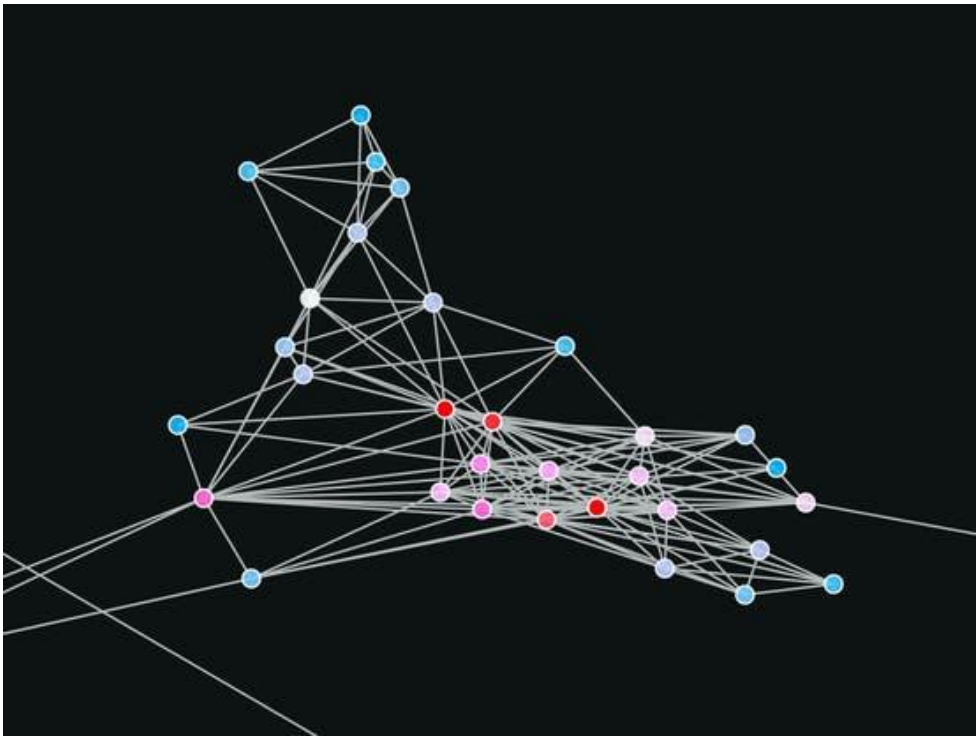
--Half of the computations in the similarity matrix would be wasted.

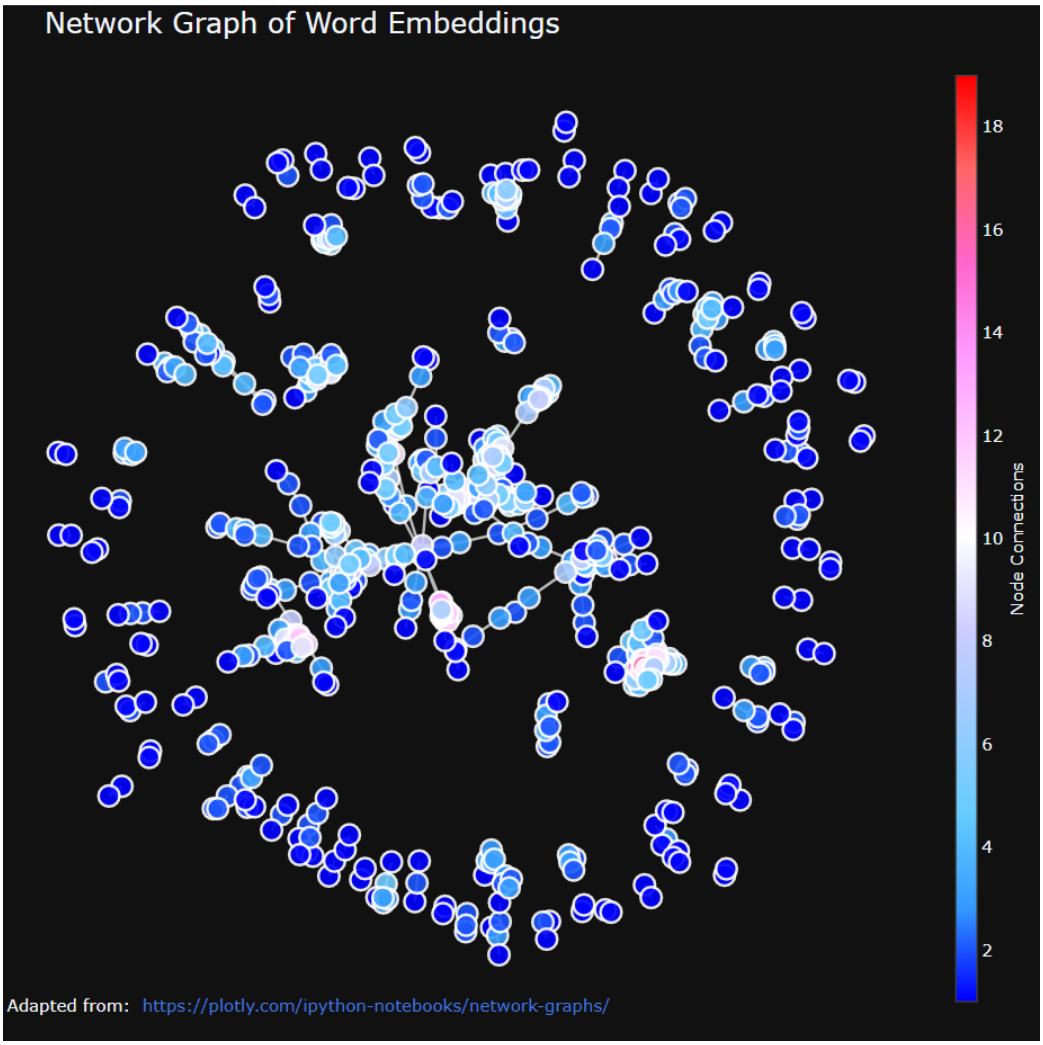
--It is simply not feasible to have a thorough understanding of a model in this way.

Thus, we have created an interactive visualization tool to explore the word embeddings thanks to Networkx and Plotly as seen below.



You can zoom in and out:





I strongly encourage to download 'temp-plot.html' and explore this interactive tool on your browser.

Potential Improvement in the Future:

I think a dimensionality reduction algorithm, such as an Autoencoder or PCA might be useful to deal with adding data to an existing graph-like structure. Furthermore, it might also be clever to add in some artificial data such that less common words won't exactly throw an error, even if the model is less confident about these words.