Generating Efficient Product Embeddings and Developing a Semantic Product Search Engine

Term Project for
BIG DATA ANALYSIS (MA60306)

Indian Institute of Technology Kharagpur

by

Anubhab Mandal 20MA20080
Rohan Das 20MA20077
Rangoju Bhuvan 20MA20048
Mangalik Mitra 20MA20070
Samarth Somani 20MA20049
Sandeep Mishra 20MA20071
Kattunga Lakshmana Sai Kumar 20MA20026
Prabhav Patil 20MA20042
Jitendra Padmanabhuni 20MA20039
Shatansh Patnaik 20MA20067
Arup Baral 20MA20010
Atharv Bajaj 20MA20014
Vishwash Kumar 20MA20079

Under the supervision of

Prof. Bibhas Adhikari

Indian Institute of Technology Kharagpur

Spring Semester, 2023

April 11, 2023

Abstract

In this term project, we present a detailed analysis of the problem of **Generating Efficient Product Embeddings and Developing a Semantic Product Search Engine**.

Our project is based on the hypothesis that the **Contextual Embeddings from BERT capture better product representation compared to classical methods** like **Skip-Gram (Word2Vec)** and **GloVe**. For this, we needed product data - lots of product data.

We collected the data from the widely used e-commerce website: Amazon using the **BeautifulSoup4 Library** in Python. We designed scrappers to scrape product information from the website. This scrapped data includes **product name, and metadata like product description, category, rating, review count, and price**.

Now using the scrapped data and the SKip-Gram and GloVe vectors libraries from **Gensim**, we generated the baseline embeddings by mean pooling over the word embeddings to generate the product embeddings. Next, we used the BERT Tokenizer and BERTModel from **Huggingface** library to generate the BERT-based embeddings for all the scrapped products.

After this step, we evaluated the results on our **Contrastive Metric** to prove that BERT embeddings are superior to baseline embeddings.

Finally last but not the least, we performed a **visualization of the word embeddings** through **PCA** and **t-SNE** plots in 3D using the **Tensorflow Embedding Projector** where we can clearly observe the presence of dense clusters corresponding to different categories of the products. Dense clusters along with adequate cluster separation shows the ability of the BERT embeddings to create an embedding space with a strong representation of every product. This Embedding Projector also serves as a **semantic search engine** for products where we can search for products and get relevant suggestions for products.

Contents

Chapter 1

The Problem Formulation

1.1 Introduction

The growth of e-commerce industries has been phenomenal in recent years, with consumers increasingly turning to online platforms for their shopping needs. As a result, businesses are now facing the challenge of providing personalized and relevant product recommendations to their customers. This is where **product embeddings** come in. Product embeddings are a way of **representing products in a high-dimensional space, allowing for similarities between products to be calculated and enabling better recommendations to be made**. With the help of product embeddings, e-commerce businesses can provide their customers with more **targeted and accurate recommendations**, resulting in higher customer satisfaction and increased sales.

1.2 Motivation and Objectives

The idea of embeddings comes from Natural Language Processing, where, scientists have developed ideas like **Word2Vec** which convert different words in the vocabulary into dense vectors in a latent dimension which reduces the compute by **reducing the number of parameters** involved while modeling the input as well as **captures the similarity between different words**. Analogous to this, E-Commerce industries have widely used Product Embedding which can capture the proximity between 2 products efficiently while discriminating them from other products.

However, with the advent of Transformers and Pretrained large language models, it has become increasingly easy to perform various downstream tasks with improved accuracy and better results. We have thought of **harnessing this power and using it to generate Product Embeddings in an easier fashion which are stronger than those in use today**.

1.3 Problem Statement

The problem discussed and attended to in this paper is as follows:

1. To **scrape** product information and meta-data from e-commerce websites like Amazon and Flipkart
2. To **generate efficient behavior-based product embeddings** using Pre-trained Large Language Model - BERT (Bi-directional Encoder Representation of Transformers) by Google
3. To **compare embeddings** generated by BERT with those generated by classical methods like Skip Gram Prod2Vec architecture and GloVe model (baseline embeddings)
4. To **design an effective evaluation metric** for the above comparison
5. To build a **semantic product search engine**
6. To make sure that the **embeddings are dynamic** and new/rare products can be incorporated into the system at any point in time
7. To **visualize the embeddings in a low-dimensional representation** using Principle Component Analysis (PCA) and t-Stochastic Neighbourhood Estimation (t-SNE) plots.

1.4 Literature Review

There have been diverse research in the field of product embeddings with researchers trying one method over another with hopes of improving upon the present state-of-the-art in e-commerce industries.

Our work takes motivation from Grbovic et al. (2015) which was the **first paper to extend the idea of Word2Vec from NLP to the field of product embeddings**. They used **CBOW** (Mikolov et al. (2013a)) to generate low-dimensional product representation. This method was **put into production in 2014**. However, this paper only focuses only on the names and transaction data of the products which leads to capturing only limited data. Vasile et al. (2016) talks about taking this one step further by **incorporating product metadata** into the embeddings to make the embeddings more resilient and unique. Since then, prod2vec has been a popular choice in the industry and has been used widely in cross-domain recommendation systems.

Another inspiration is the Arora et al. (2020) which discusses how **BERT's contextual embeddings can be domain invariant and out-perform simpler embedding methods like Skip Gram and GloVe** in case of language containing **complex structure, ambiguous word usage, and words unseen in training**.

Some recent literatures like Yan et al. (2022) involve **Attention Based Graph Neural Networks** to produce personalized product embeddings for a better recommendation. However, these methods come at the cost of huge expenses in terms of computational power and data requirements.

Chapter 2

Data Sources and Collection

2.1 Data Sources

We used popular e-commerce websites - **Amazon** and **Flipkart** for data scraping and we extracted data for different categories in the domain of **clothing** and **electronics**, which involved Men's and Women's traditional and western wear and various electronic accessories.

2.2 Data Scraping

Libraries used for Data Scraping: BeautifulSoup4, Pandas and Numpy
Website from which data was scraped: Amazon
Categories from which the data was scraped:

1. Electronic items and their accessories
2. Fashion including men's and women's wear
3. Home and Kitchen Appliances
4. Furniture and other home decor products

Brief Description about data scraping using Python: We come across a massive number of e-commerce websites, these websites display various data of different products along with different information on the browser. For this project, we wanted to craft a data frame containing useful entries such as the name of the product, its brand, the price, the rating, the total number of reviews, and a brief description of the product. So for each web page made using dynamic templating, the names of the classes and sections of the corresponding HTML files remain the same. So, we copied the sections we wanted and appended them in a CSV file.

Problems Faced and how we tackled them:

1. **Unstable or slow load speed** Most of the time the code had to be re-run to scrape the data from the web pages. Besides due to a lot of usual traffic on the website, bulk data took a huge amount of time to load usually resulting in a connection timeout error.

Solution: We circulated the code for scraper amongst our team members where each of the members was required to run the loops in succession parallelly on their respective devices and append the data obtained to a single data frame and then combine it in a single CSV file.

2. **Structure of the Web Page** E-Commerce websites like Flipkart and Amazon design different web pages for different categories pertaining to attract different groups of customers, for instance, the overall layout and structure of Men's Wear is different from that of smartphones.

   Solution: We designed different scrapers for different categories and assimilated data accordingly.

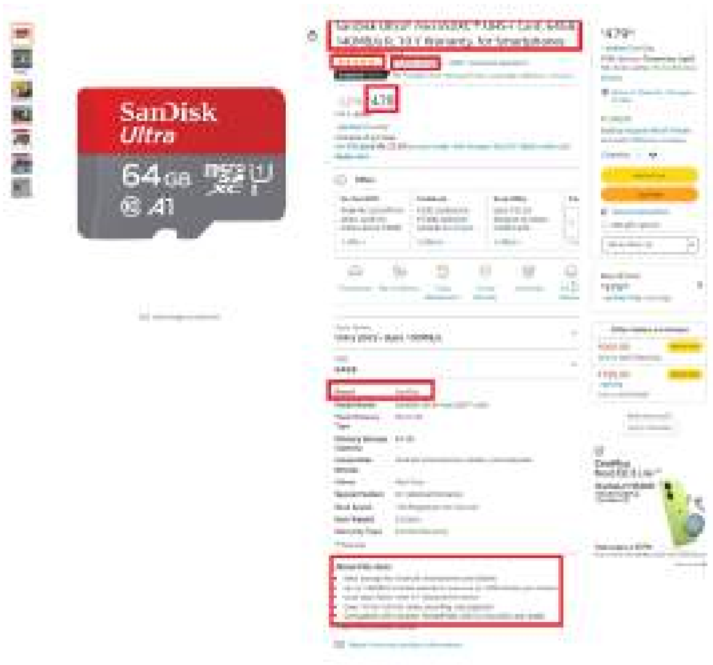


Fig. An instance of the data collected through the scrapper

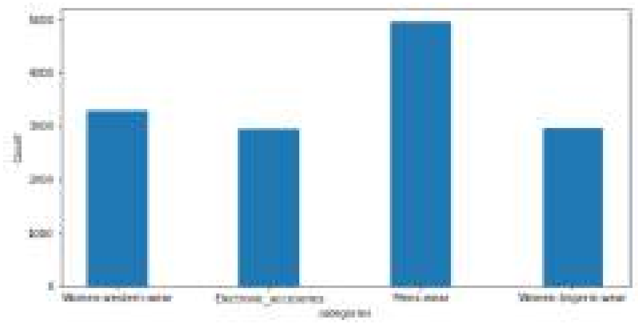
2.3 Data Cleaning

The meta-data we get after scraping has the following fields: title, brand, price, rating, review count, description, and categories, and it is stored in a CSV format. Using the pandas library from Python, the scrapped data is cleaned in the following steps.

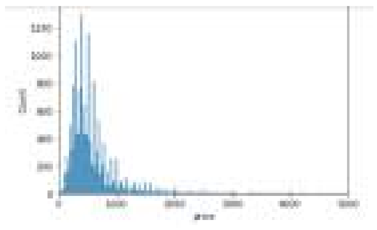
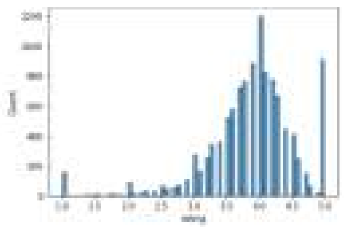
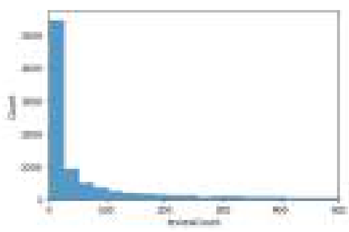
1. Removing Rows with **Null Value in Title column**
2. Removing **corrupted data** and data with **incorrect data types**
3. Removing **Duplicate entries of products**

2.4 Properties of the Data - Diving into the intricacies

The cleaned dataset obtained now has **14152** rows, each classified into one of four categories, as shown in the figure below. The table shows that Mens-wear has the most variety, followed by Women's western wear, Women's lingerie wear, and Electronic accessories.

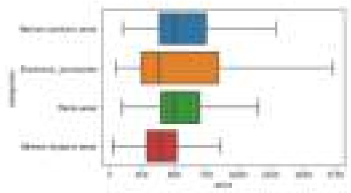
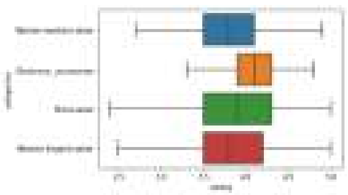
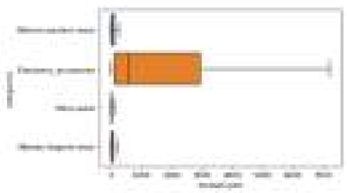


Further classification of the data gives the following histograms,

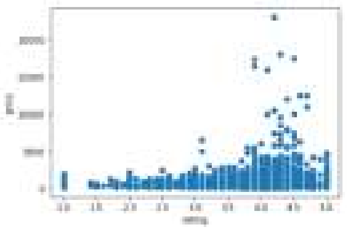
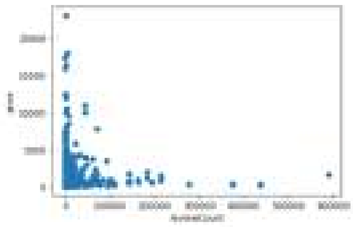




Here we can see that the count of products is a right-skewed graph with a peak at lower prices with very few products with prices above Rs. 2000. Also we can observe that the ratings for products peak at around 4 with very few ratings below 2.5 (shows a left-skewness). Similarly, we can observe that most of the products have review counts ranging from 1-50 with very few products which have review counts above 200 (right-skewed).

The distributions of the quantitative attributes of the data are described in the following box plots after removing the Inter-Quartile Range (IQR) outliers.

The correlations for price vs rating and price vs review count are as follows,

Chapter 3

Methodology and Experiments

3.1 Conventional Methods: Before Big Data Era

The conventional method for grouping similar products involves **manually assigning** or **tagging** products with **categorical** or **descriptive attributes** such as **brand, size, or color**, and subsequently **recommending similar products based on these pre-defined characteristics**. However, this approach has limitations in identifying the true similarity between products. Another widely used approach was suggested in Sarwar et al. (2001), where the authors say that a **KNN based collaborative Filtering** approach worked quite efficiently before the data-driven deep learning methods came into the application. **Hidden Markov Models** can model the probability of a sequence of words using a chain of hidden states. However, it requires **explicit labeling of the training data** and **cannot capture complex relationships between words**.

3.2 Classical Methods: Skip-Gram and Glove Era

Pre-trained word vectors have been used with mean pooling to generate product embeddings efficiently. Specifically, we use two sources of pre-trained word vectors:

1. **Word2vec-google-news-300:** Word2vec efficiently implements the **continuous bag-of-words** and **skip-gram** architectures for computing vector representations of words. This particular pre-trained model contains vectors trained on a subset of the **Google News dataset** (about **100 billion words**). The model contains **300-dimensional vectors** for **3 million words** and **phrases**. The embeddings were obtained using a simple data-driven approach described in Mikolov et al. (2013b)

2. **GloVe-twitter-200:** The pre-trained model in question was released in 2014 by the Computer Science Department at Stanford University. These **200-dimensional representations** were trained using **Global Vectors (GloVe)**. The training dataset consists of **2 billion tweets with about 27 billion words**. It encodes **1,193,515** tokens as unique vectors, with **all tokens outside the vocabulary encoded as the zero-vector**. Tokens are uncased. The training methodology is as described in Pennington et al. (2014).

These pre-trained models are part of the **Gensim-data repository** and were accessed using the open-source Python library, Gensim.

Of all the columns in the CSV file of the scrapped data, only **'title', 'description' and 'categories' contain usable text data**.

3.2.1 Preprocessing

We perform the following steps individually on the usable columns for a product:

1. Removal of punctuations
2. Lowercasing of the text as the models are uncased
3. Tokenization of the text using the nltk word tokenizer
4. Removal of stop words (overused words such as 'is', and 'the')
5. Lemmatization (changing inflected forms of words to the base/dictionary form) of each token using nltk lemmatizer

Finally, we **concatenate the token lists** generated for each usable column for a given product into a token list containing comprehensive information about the product.

3.2.2 Pooling

This step involves making one embedding for a product using the embeddings (from pre-trained models) of the tokens in the token list (generated in the last step) representing the product. Having one-vector representations for sentences and documents in addition to word embeddings is often advantageous for practical systems (Yan et al. (2016), Palangi et al. (2016)).

Here, we adopt **mean pooling**. As apparent from the name, it involves taking the **element-wise average of the token-level embeddings**. Intuitively, it might seem that compression of the token-level representations in this way must lead to information loss. However, according to Aldarmaki and Diab (2018), **mean pooling performs better than context-sensitive methods in several benchmarks**.

Also, during mean pooling, we **skip tokens which do not have representations** in the pre-trained models.

3.3 Modern Methods: The Transformers Era

The Transformer model is a deep learning architecture designed to generate contextual embeddings for natural language processing tasks. Unlike traditional machine learning models that rely on pre-defined features and statistical techniques to classify data, the **Transformer model learns the contextual relationships between words and generates embeddings that capture the semantic meaning of a sentence**.

We used **BERT (Bi-directional Encoder Representation of Transformers)** which is a state-of-the-art transformer-based model for generating the contextual embeddings for our dataset. BERT is a pre-trained deep learning model for natural language processing (NLP) developed by Google in 2018.

The model takes a **sequence of words as input** and before generating **respective contextual embeddings**, BERT uses an algorithm called **subword-tokenization** which further **breaks every word/token into sub tokens separated by '#'**. This ensures in **handling out of vocabulary (OOV) tokens** and thus **preserving better context** compared to classical methods like Skip-Gram and GloVe. This is the reason for our hypothesis.

The CLS token, short for **"classification token"**, is a special token added to the beginning of every input sequence in BERT. When the input sequence is fed into BERT, the **final hidden state corresponding to the CLS token is used as the aggregate representation of the entire input sequence**, which can then be used for downstream tasks.

Preprocessing is done in a similar manner as mentioned in section 3.2.1. For every input sentence in the preprocessed dataset, we take the embedding of the CLS token. The embedding thus obtained is of **768 dimensions** and is regarded to embed the context of the product's features, description, etc and is popularly known as the **contextual embedding**.

3.4 Visualisation: Multi-Dimensional Scaling and Principle Component Analysis

Principal Component Analysis (PCA) is a statistical technique that is commonly used for data reduction and dimensionality reduction. The primary goal of PCA is to identify the most **important features or components** in a dataset and represent them in a lower-dimensional space while retaining as much of the original variance as possible. PCA works by transforming the data into a **new coordinate system** where each axis represents a principal component that explains the maximum amount of variance in the original data. These principal components are computed by performing an **eigendecomposition on the covariance matrix of the data**. By retaining only the top few principal components, the high-dimensional dataset can be effectively reduced to a lower-dimensional space while still capturing most of the variance. PCA is widely used in various applications such as image processing, finance, and bioinformatics.

Multi-dimensional scaling (MDS) is a statistical method that helps researchers identify and analyze the underlying themes or dimensions that explain the similarities or differences between sets of data. It can be used to **analyze any kind of similarity or dissimilarity matrix**. In market research, MDS is often used to plot data, such as the perception of products or brands, in a way that makes it easy to interpret visually. Any data that has meaningful similarities or distances can be displayed using MDS. The classical MDS technique is based on PCA. It displays the similarities and differences between pairs of items on a coordinate matrix, making it simple to interpret the dimensions by plotting them on a two-dimensional matrix.

In our project, we use these methods by providing vector inputs to the TensorFlow embedding projector, which applies the necessary multi-dimensional scaling and plots the input points in a 3D coordinate system.

3.5 Embedding Visualisation and Semantic Product Search Engine: Tensorflow Embedding Projector

We have used the **TensorFlow Embedding Projector** to visualize the embedded products' data points.

The first one of the models is obtained using **Principal Component Analysis**. PCA simply projects the high dimensional data points (in our case 768 dimensions for BERT) linearly into 3 dimensions preferring the **top-3 principal vectors**. As a result, it **preserves the global structure of the data but the local structures might still get lost**. In our model, however, **similar data points still remain close to each other** since they were close in the high dimensional embedding.

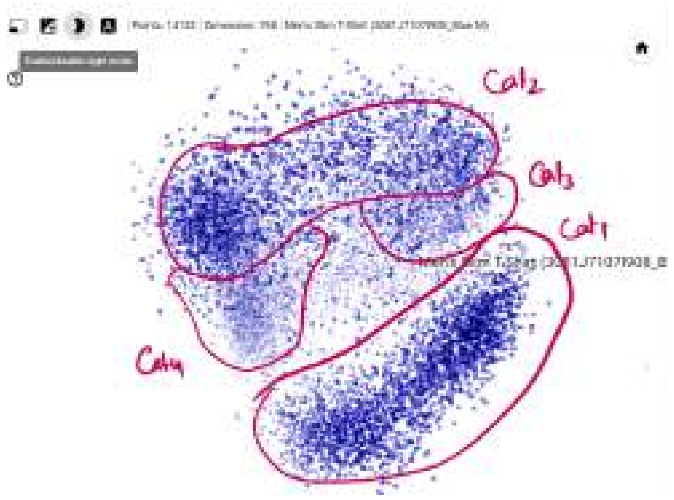


Fig. 3D PCA plot of the product embeddings from BERT clearly showing the clusters of the products by their categories and description.

The other model was obtained using **t-SNE** which is a **non-linear dimensionality reduction technique**. It works by assigning a **Gaussian probability distribution** over pairs of high dimensional data and another similar probability distribution over the points in the 3-dimensional map, and it **minimizes the Kullback–Leibler divergence using gradient descent between the two distributions with respect to the locations of the points in the map**. As a result, it **preserves the neighborhoods of the points more effectively**. Here we obtain some nice clustering of the data points in contrast to PCA and hence this is **visually more suggestive as compared to PCA**, although the final results are **highly parameter dependent**.

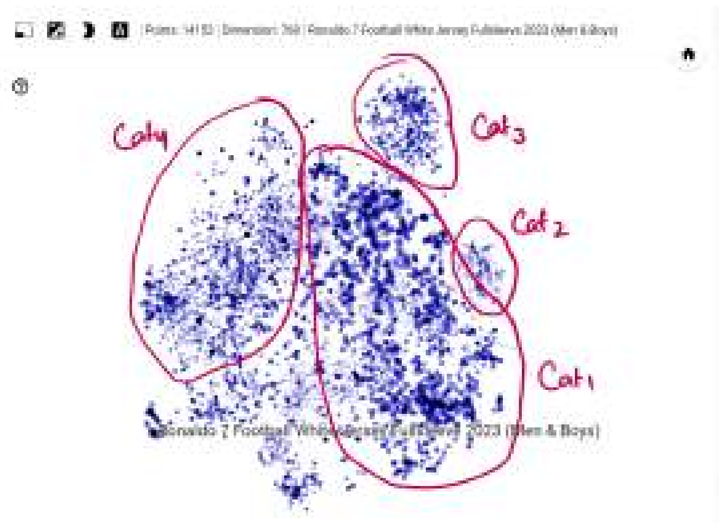


Fig. 3D t-SNE plot of the product embeddings from BERT clearly showing great clustering of the products by categories and description.

There was a need to segregate the JSON file into text files of labels and vectors so as to get the reduced three-dimensional plot of the dataset.

In order to load the data into Tensorboard, we saved the labels datafile and vectors datafile to its directory, along with metadata that allowed for visualization of a specific layer of interest in our model.

Using the TensorBoard Embedding Projector, it was possible to graphically represent high-dimensional embeddings. This is helpful in visualizing, examining, and understanding your embedding layers. This tool is used to **search for specific keywords associated with the embeddings or highlight similar points in space**. Ultimately, its goal was to provide a way to better interpret the embeddings that our model is generating as well as to check if similar ones according to our definition are plotted nearby in the three-dimensional space.

Using this tool we could also build a **Semantic Product Search Engine** as promised whereby one can **visualize the clusters** and s**earch for products which are semantically similar** embedding-wise to the product searched for. This will heavily **improve product recommendations and search on e-commerce websites like Amazon and Flipkart.**

Fig. Instance of Product Search and Recommendation on Tensorflow Embedding Projector

Chapter 4

Results and Discussion

4.1 Evaluation of the Product Embeddings: The Contrastive Metric

After having generated the product embeddings for more than 14k different products on Amazon in 3 different fashions namely - Skip-Gram, GloVe, and BERT Contextual Embeddings, we had to decide on a metric to evaluate the relative performance of the 3 different embeddings.

For each product **P, the anchor**, in the dataset, we choose **N positive samples** (related products from the N most suggested product for it on Amazon) and **K×N negative samples** (random products from the dataset) and **compute their embeddings** by all three different methods.

For a method $i \in BERT, SG, Gl$, we take the product embedding as P_i and the positive samples as $S_{p_i}^j \forall j \in [1, N]$ and the negative samples as $S_{n_i}^j \forall i \in [1, NK]$ and calculate the score of the method as:

$$\textbf{Score (s)} = \frac{1}{N}\sum_{j=1}^{N} P_i^T S_{p_i}^j - \beta \frac{1}{KN}\sum_{j=1}^{KN} P_i^T S_{n_i}^j$$

Here we are taking **cosine similarity** between the embeddings as the **measure of similarity** with a similar product and **K=5** where K is the **ratio between positive to negative samples**. While β **is a hyperparameter** to **scale down the contribution of negative samples**.

This method is **similar to a Triplet Loss setup** where we try to **reduce the distance between the anchor and positive samples while correspondingly increasing the distance from negative samples**. The closer the anchor representation will be to the positive samples, the higher will be the score for the first part, and the further the anchor will be from the negative samples, the lower will be the score for the second part of the equation. As a result, $Pos_{score} - Neg_{score}$ **being higher indicates better representation in the embedding space**.

4.2 Why Language Model-Based Embeddings are Better

Contextual embeddings from BERT (Bidirectional Encoder Representations from Transformers) are considered better than those from skip-gram or GloVe models for several reasons:

1. **Bidirectionality**: BERT is bidirectional, which means it can take into account the context of a word from both the left and right directions. This allows it to capture more complex relationships between words and better understand the meaning of a sentence.
2. **Deep architecture**: BERT is a deep neural network that has been trained on a large corpus of text using a masked language modeling task and a next-sentence prediction task. This deep architecture allows BERT to capture more complex syntactic and semantic relationships between words.
3. **Pre-training**: BERT is pre-trained on a large corpus of text, which allows it to learn general language representations that can be fine-tuned for a variety of downstream tasks. Skip-gram and GloVe models, on the other hand, are trained on a co-occurrence matrix of words and do not have the same pre-training advantage.
4. **Task-specific fine-tuning**: BERT embeddings can be fine-tuned on specific tasks, which allows them to adapt to the specific context and requirements of a particular task. This fine-tuning process can improve the accuracy of downstream tasks such as sentiment analysis, named entity recognition, and question answering.
5. **Out-of-Vocabulary Embeddings**: BERT can generate embeddings corresponding to any word given to it due to its ability of subword tokenization. While for classical embedding generation methods like Skip-Gram and GloVe

To prove this claim experimentally we evaluated our BERT embeddings against the baselines Skip-Gram and GloVe and found **BERT to have out-performed all the baselines**. Also as expected, for most products, **GloVe scores are superior to those of Skip-Gram** (GloVe embeddings are considered better than Skip-Gram in NLP). The evaluation results can be seen in the table below:

Table 4.1: Evaluation Results

WEIGHT OPTIMIZATION						
Product Name	Category	K	β	S_{BERT}	S_{SG}	S_{Gl}
Dennis Lingo Men's Solid Slim Fit Cotton Casual Shirt with Spread Collar and Full Sleeves	Men's Wear	5	1	**16.878**	0.303	2.301
SET WET Deodorant Spray Perfume Cool, Charm and Mischief Avatar for men	Perfumes	5	1	**4.667**	0.194	2.698
ASUS Vivobook Pro 16	Electronics Accessories	5	1	**15.8997**	0.295	1.872
Amazon Brand - Vedaka Medium Poha, 500g	Groceries	5	1	**0.813**	0.204	-0.432

4.3 Our Limitations and Future Scope

One of the most important factors in a data-driven project is the availability of resources in the form of Computational Power. We carried out all our experiments on the free version of **Google Colaboratory** and our **local systems** due to which we could run a limited number of experiments in the stipulated time.

Still, we were able to successfully run all the aimed experiments - from scraping around 15k products from 4 different categories from Amazon to generating product embeddings for each of them in 3 different methods - Skip-Gram, GloVe, and BERT to the evaluation of the results and visualization of the obtained embeddings using Tensorflow Embedding Projector.

There is a vast scope for future work in this area, Some of them are as follows:

1. With the advent of more powerful and larger transformers with a huge number of pre-trained parameters, there is always a way to **finetune the Encoder models with lots of product data** which will definitely result in better embeddings for products.

2. There are 3.5million products on Amazon. We can use efficient industry-level scrappers with permission from Amazon to generate embeddings for all these products. We can use **Distributed Frameworks like Hadoop and Apache Servers** to handle and process this vast amount of data.

3. **Our method can be industrialized** to the profit of different e-commerce industries out there who can integrate our semantic product search method into their websites for better user satisfaction and product recommendation.

4. The concept of converting words to vectors to study the similarities and dissimilarities between them is not only useful for e-commerce websites as discussed above but also in various domains of science and its studies, the possible use of this system can already be seen as required in the future. There is a lot of scope for the system in the future and some of that is as follows:

   (a) **Biological Sequences:** Word embeddings for n-grams in biological sequences (e.g. DNA, RNA, and Proteins) for bioinformatics applications have been proposed by Asgari and Mofrad (2015) Named bio-vectors (BioVec) to refer to biological sequences in general with protein-vectors (ProtVec) for proteins (amino-acid sequences) and gene-vectors (GeneVec) for gene sequences, this representation can be widely used in applications of deep learning in proteomics and genomics. The results presented by Asgari and Mofrad (2015) suggest that BioVectors can characterize biological sequences in terms of biochemical and biophysical interpretations of the underlying patterns.

   (b) **Game Designing:** Word embeddings with applications in game design have been proposed by Rabii and Cook (2021) as a way to discover emergent gameplay using logs of gameplay data. The process requires transcribing actions during the game within a formal language and then using the resulting text to create word embeddings. The results presented by Rabii and Cook (2021) suggest that the resulting vectors can capture expert knowledge about games like chess, which is not explicitly stated in the game's rules.

   (c) **Cyber Security:** Word embeddings can also be used in cybersecurity to detect threats, such as malware and phishing, by analyzing patterns in language and identifying potentially suspicious or malicious communications (Mumtaz et al. (2020)). By applying word embeddings to cybersecurity, organizations can better protect their systems and data from a range of threats.

5. We have only used embeddings in Euclidean Space in this project. However, in the future, we can try to generate embeddings in **non-Euclidean Space**. For example, **Hyperbolic embeddings** which have been seen to **perform superior to their Euclidean counterparts in case of image segmentation tasks** (Atigh et al. (2022)).

Appendix A

Relevant Links and Packages Used

Relevant Links

1. Link to Github Repository
2. Link to Directory on Google Drive

Packages, Libraries, and Frameworks

Numpy, Pandas, Matplotlib, Seaborn, Tensorflow, Huggingface, Pytorch, Gensim, Sci-kit Learn, NLTK, Tensorboard, and Tensorflow Embedding Projector

Pseudocode used for scraping data

```
for page number in range (x,y): do
  Get the dynamic URL
  Get the content of the webpage (in the form of a card) using Beautiful Soup so that the corresponding card details (HTML Attributes) can be extracted from their respective classes.
  for card in Cards(extracted above) do
    Dict['title'] = Extract the title of the product
    Dict['brand'] = Extract the name of the brand
    Dict['rating'] = Extract the rating
    Dict['reviewCnt'] = Extract the number of reviews
    Dict['description'] = Extract the description
  end for
  DF ← Dict {Convert the dictionary Dict to the dataframe}
  Save DF in a CSV format
end for
```

Bibliography

Aldarmaki, H. and Diab, M. (2018). Evaluation of unsupervised compositional representations. *arXiv preprint arXiv:1806.04713*.

Arora, S., May, A., Zhang, J., and Ré, C. (2020). Contextual embeddings: When are they worth it? *arXiv preprint arXiv:2005.09117*.

Asgari, E. and Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287.

Atigh, M. G., Schoep, J., Acar, E., Van Noord, N., and Mettes, P. (2022). Hyperbolic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4453–4462.

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., and Sharp, D. (2015). E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1809–1818.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Mumtaz, S., Rodriguez, C., Benatallah, B., Al-Banna, M., and Zamanirad, S. (2020). Learning word representation for the cyber security vulnerability domain. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., and Ward, R. (2016). Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):694–707.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Rabii, Y. and Cook, M. (2021). Revealing game dynamics via word embeddings of gameplay data. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 17, pages 187–194.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Vasile, F., Smirnova, E., and Conneau, A. (2016). Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM conference on recommender systems*, pages 225–232.

Yan, A., Dong, C., Gao, Y., Fu, J., Zhao, T., Sun, Y., and McAuley, J. (2022). Personalized complementary product recommendation. In *Companion Proceedings of the Web Conference 2022*, pages 146–151.

Yan, Z., Duan, N., Bao, J., Chen, P., Zhou, M., Li, Z., and Zhou, J. (2016). Docchat: An information retrieval approach for chatbot engines using unstructured documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 516–525.