Embedding Based Recommendations P14

Ian Smart

1 Introduction

Everyone has a collection or genre of books that they love and cherish, but sometimes those can become monotonous. This leaves people looking for a new book that is just similar enough to what they already know, but also unique enough to free them of the monotonous feeling. This is a complex problem, which is then further compounded with the fact that there are so many books available online now, making finding a book that fits that category a daunting challenge.

In order to help alleviate this situation we have attempted to create a recommender system which will recommend new books to users based on how they rated books that they already like. This program will then take those recommendations and compare them against the ratings of other users. It will then be able to output to the user a list of similar books.

For the basis of this reccomender system we will use a collaborative filtering process to reduce the number of potential recommendations based on similar user ratings and embeddings to store the data in higher dimensional space, making finding groupings of similar books easier. It will work by first preparing the data, grouping the data by user, creating an embedding model, training a model based on the grouped data and then querying the model in order to test the results. There are several real-life applications for this program, such as a recommendation tool for libraries, book stores like Amazon, and publishers.

2 Background

2.1 Collaborative Filters and Embeddings

In order to explain how the program will work, we must first cover the basics of Collaborative Filtering, Embeddings and Euclidean Distance. Collaborative Filtering is a complex system in which a large data set is filtered into a smaller, more relevant data set through the use of collaboration among multiple agents, viewpoints, and data sources. The goal of this system is to create a subset of data points, which all share a large degree of similarity. The system uses collaboration in different ways based on the purpose of filtering and the type of data available. For instance, the system is often used to create small subsets of objects that a user is predicted to like, based on how similar users rated that object. In this scenario, collaboration will be the use the reviews of multiple agents to influence how an object's predicted score is calculated. However, for the purposes of this paper, since we are using other books instead of user reviews, we will be using collaborative filtering to create small data sets based on the ingredients they have in common. In order to do that, the system will implement collaboration by comparing the individual data points and using the similarities to produce book groupings.

Embeddings are a very interesting topic, that take a bit of work to conceptually visualize. The exact definition an embedding is an Nth dimensional representation of an object, where each dimension represents a different feature of that object. To break that down a little bit, embeddings work by turning each object into a point in space, with feature of that object then acting as coordinate. This results in clusters of objects that have similar features. It is this clustering of objects that we are most interested in, because those will act as the foundation for the recommendation system. It should also be noted that when the program turns an object into an embedding, it is impossible to know if the embedding is correct until the program re-translates the object. That is because the embeddings are stored in form which can only be understood by the program translating it.

The last thing we need to cover before discussing the method, is Euclidean Distance. Euclidean Distance is the measurement which we will use to calculate the distance between embeddings. It works by finding the distance a straight line would have to take between two points in Euclidean space. While there are a variety of other possible distance metrics, this one is the most applicable given the complex number of dimensions that embeddings are created in. The formula for Euclidean Distance is: ((p1-q1)$^2 + (p2 - q2)^2 + ... + (pn - qn)^2) \cdot 5)$.

2.2 Recommended Further reading

The background information covered here is but a small sampling of the true depths of these fields. If you are interested in learning more about these topics we would like to recommend the following papers:

Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. 2016. Query-based Music Recommendations via Preference Embedding. In Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16). ACM, New York, NY, USA, 79-82. DOI: https://doi.org/10.1145/2959100.2959169

Asnat Greenstein-Messica, Lior Rokach, and Michael Friedman. 2017. Session-Based Recommendations Using Item Embedding. In Proceedings of the 22nd International Conference on Intelligent User Interfaces (IUI '17). ACM, New York, NY, USA, 629-633. DOI: https://doi.org/10.1145/3025171.3025197

Florian Schroff, Dmitry Kalenichenko, James Philbin; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823

2.3 Similar Works

While there are a few papers out there that have similar ideas to this one, all of them differ in some significant aspect. For instance: Cataldo Musto's work on Content Reminder Systems[1], uses the contents of wikipidea pages in order to gather similarities of books instead of using user ratings. Similarly, Fuzheng Zhang's work on collaborative embeddings, is along the same lines but is used for reccomending movies instead of books.[2]

3 Method

There are 6 main steps to the program: data extraction, data conversion, establish, training and then querying recommendations. During the data extraction phase, the program will gather the data from the data set and convert it into a more manageable data type. In the data conversion phase, the program will then take the newly extracted data and convert it into a use able form by filling in missing data and whatever else is necessary. Then, in the establishment phase, the program attempts to create embeddings by creating a neural network with an embedding layer. The program then enters the training phase, in which the embeddings are trained based on user ratings. Before finally using embeddings to find recommendations in the querying phase.

4 Experimental Setup

4.1 Hypothesis

There are several key questions that we are trying to solve with this experiment. The first one is whether an embedding based reccomeder system can actually create plausible recommendations. This is the main driving force behind this experiment and will serve as the primary problem that we are trying to solve. The second question is whether meta data on books can be turned into embeddings.

4.2 General Setup

For the implementation of this program we decided to write primarily in python 3.74, using a windows development environment. The Anaconda Jupyter Labs IDE was used to code it and all of the libraries were updated to the latest version before running.

4.3 Libraries

For this project we decided to use a variety of different libraries. For the data extraction side of things we used the Json library[3] to convert the data set, the Pandas library[4] to store the data and Numpy[5] to onehot encode the ids. The remaining sections relied on the previous libraries, plus Keras[6] and Tensor Flow[7] creating the neural network and the embeddings. We also de-

[1]For more information on the paper, please visit: `https://www.researchgate.net/profile/Giovanni_Semeraro/publication/309021705_Learning_Word_Embeddings_from_Wikipedia_for_Content-Based_Recommender_Systems/links/59ea02d34585151983c7e093/Learning-Word-Embeddings-from-Wikipedia-for-Content-Based-Recommender-Systems.pdf`

[2]For more information on the paper, please visit:`https://www.kdd.org/kdd2016/papers/files/adf0066-zhangA.pdf`

[3]For more information on the Python Json library, please visit `https://docs.python.org/3/library/json.html`

[4]For more information on the Pandas library, please visit: `https://pandas.pydata.org/`

[5]For more information on the numpy library, please visit: `https://numpy.org/`

[6]See `https://keras.io/` for more information about the Keras Library.

[7]Please see `https://www.tensorflow.org/` for more information about the Tensor Flow library.

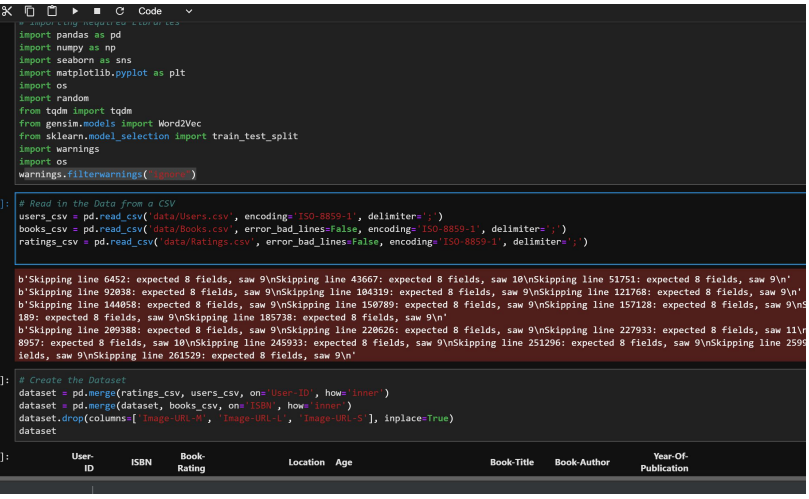
[8]Please see `https://radimrehurek.com/gensim/models/keyedvectors.html#gensim.models.keyedvectors.Word2VecKeyedVectors.init_sims` for more information about the GenSim library.

cided to use the GenSim library in order to create the embedding model.[8]

4.4 Data set

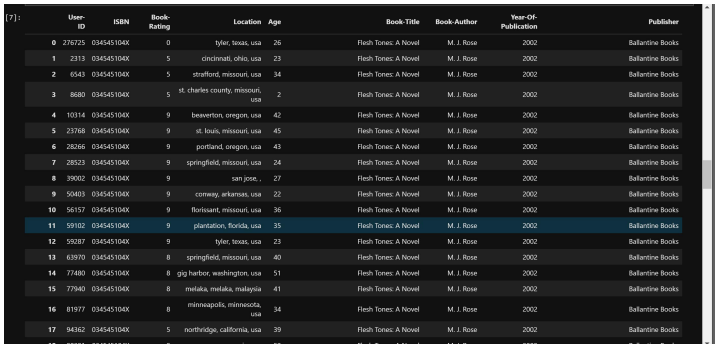
We had a hard time figuring out which data set to use for this program. The majority of the data sets either contained the wrong types of data for the specific recommendations we had in mind or didn't have enough information to provide proper recommendations. We ended up settling on data provided by the Book-Crossing Dataset. It is a collection of demographics about a wide variety of books gathered by the Institut for Informatik Freiburg. It contains information such as title, author, year of publication, individual user ratings, ISBN and Location.

It was the only data set which contained both categories of data we needed: continuous/discrete data that could be used to fine tune the relationships between data points and purely categorical data to aid in the embedding/prediction process. For reference, please see figure 1.



4.5 Data Extraction/Conversion

In order to handle the data we decided to first transform the data set from it's native csv form into something a bit more manageable, a pandas data frame. [9]. We then transformed in several different ways to make it more usable. We reduced the number of dimensions by discarding useless features. Replaced missing or non-existent data by calculating the Inter-Quartile Range then randomly selecting a value in that range. Finally, we removed extraneous data and in order to make some data more discrete, for example: changing the "Location" column from the specific city to just the country. The resulting data can be seen in Figure 2. We then grouped the ratings based on the user and passed the data to the embedding model.



	User-ID	ISBN	Book-Rating	Location	Age	Book-Title	Book-Author	Year-Of-Publication	Publisher
0	276725	034545104X	0	tyler, texas, usa	26	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
1	2313	034545104X	5	cincinnati, ohio, usa	23	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
2	6543	034545104X	5	strafford, missouri, usa	34	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
3	8680	034545104X	5	st. charles county, missouri, usa	2	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
4	10314	034545104X	9	beaverton, oregon, usa	42	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
5	23768	034545104X	9	st. louis, missouri, usa	45	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
6	28266	034545104X	9	portland, oregon, usa	43	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
7	28523	034545104X	9	springfield, missouri, usa	24	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
8	39002	034545104X	9	san jose, ,	27	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
9	50403	034545104X	9	conway, arkansas, usa	22	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
10	56157	034545104X	9	florissant, missouri, usa	36	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
11	59102	034545104X	9	plantation, florida, usa	35	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
12	59287	034545104X	9	tyler, texas, usa	23	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
13	63970	034545104X	8	springfield, missouri, usa	40	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
14	77480	034545104X	8	gig harbor, washington, usa	51	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
15	77940	034545104X	8	melaka, melaka, malaysia	41	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
16	81977	034545104X	8	minneapolis, minnesota, usa	34	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
17	94362	034545104X	5	northridge, california, usa	39	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books

4.6 Embedding Establishment and Training

In order to create embeddings, we had to create a nueral network. We were originally planning to use the Keras and TensorFlow libraries to create a basic model and fed it our reshaped data. We would then configured the model by adding 2 embedding layers, 2 merge layers and an output layer. We would then set the optimizer function to adam and the loss function to cross entropy. However, after several failed attempts, we decided to switch to using the GenSim Word2Vector model instead.[10] This provided a better basis for which to work from. We then passed the model the grouped list of ratings and trained it for 10 epocs. We then queried it to see the results.

5 Results

One of the main problems with grading the results of a reccomender system, is that there aren't any great ways to test to them. This is because their results are highly variable, aren't easily quantifiable and can vastly vary between tests. The best way is to simply see if the results make sense. The results of our experiment can be seen in the figure below:

```
similar_isbns = model.similar_by_vector('0743467523')
similar_books = []
for isbn in similar_isbns:
    pair = (all_books_dict[isbn[0]][0], isbn[1])
    similar_books.append(pair)
print("Books similar to Dreamcatcher by Stephen King")
similar_books
```

```
Books similar to Dreamcatcher by Stephen King
[('Sleepwalk', 0.7612367868423462),
 ('Wizard and Glass (The Dark Tower, Book 4)', 0.7372331023216248),
 ('The Formula', 0.7296830415725708),
 ('Outrage: The Five Reasons Why O. J. Simpson Got Away With Murder',
  0.7212274074554443),
 ('The Regulators', 0.7150816321372986),
 ('Night Shift', 0.7086629867553711),
 ('Masquerade', 0.7068302631378174),
 ('The Servants of Twilight', 0.7037485837936401),
 ('The Blooding', 0.7024586200714111),
 ('The Courtship of Princess Leia (Star Wars)', 0.7002884149551392)]
```

[9]For more information on the Pandas DataFrame object, see `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html`

[10]Please see `https://radimrehurek.com/gensim/models/keyedvectors.html#gensim.models.keyedvectors.Word2VecKeyedVectors.init_sims` for more information about the GenSim library.

These results are all fairly similar to the original, as most of the provided books are in the same genre (horror) and have the same genre. However, the existence of "The Courtship of Princess Leia (Star Wars)" in the results show that the experiment could still be improved. The results also proved of our hypothesis were successful. The results of the recommendations show that an embedding based recommender system is indeed possible and that the results of which are believable. That also proves that book meta data can be used to create embeddings. Overall, we can say that this experiment was a success.

6 Conclusion

Overall the project was a success. The data produced by the recommendation system was reasonably appropriate and usable. However, as good as the results were they could be better. The If the model was tweaked a bit more and the data was better process, the results could be improved. These tweaks would probably include an increase in the number of epochs, a larger training network and possible outlier removal. Over the course of this experiment, we learned several important lessons. Firstly, we that embeddings are incredibly interesting, complex objects that are going to have a major impact on the future of machine learning and the future. Secondly, we learned a lot about staying strong despite set backs. We hit a lot of road blocks throughout this experiment, but managed to stick through with it until the end. Thirdly, we were forced to learned patience as many of the steps involved waiting for multiple hours to complete.

7 Github

In order to see our github, please visit: `https://github.ncsu.edu/irsmart/p14-embeddings`

8 Citations

Asnat Greenstein-Messica, Lior Rokach, and Michael Friedman. 2017. Session-Based Recommendations Using Item Embedding. In Proceedings of the 22nd International Conference on Intelligent User Interfaces (IUI '17). ACM, New York, NY, USA, 629-633. DOI: https://doi.org/10.1145/3025171.3025197

Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. 2016. Query-based Music Recommendations via Preference Embedding. In Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16). ACM, New York, NY, USA, 79-82. DOI: https://doi.org/10.1145/2959100.2959169

Florian Schroff, Dmitry Kalenichenko, James Philbin; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823

Jozef L. Kokini, The physical basis of liquid food texture and texture-taste interactions, Journal of Food Engineering, Volume 6, Issue 1, 1987, Pages 51-81,ISSN 0260-8774, https://doi.org/10.1016/0260-8774(87)90021-5.

Koehrsen, Will. "Building a Recommendation System Using Neural Network Embeddings." Medium, Towards Data Science, 6 Oct. 2018, https://towardsdatascience.com/building-a-recommendation-system-using-neural-network-embeddings-1ef92e5c80c9.

"Mathematical Expressions." Overleaf, Online LaTeX Editor, https://www.overleaf.com/learn/latex/Mathematical$_e xpressions$.

"NumPy¶." NumPy, https://numpy.org/.

Robinson, Allan, and Robinson. "How to Calculate Euclidean Distance." Sciencing, https://sciencing.com/how-to-calculate-euclidean-distance-12751761.html.