**Portfolio Project Option 1**

Lance I. Evans

Colorado State University Global

CSC510-1: Foundations of Artificial Intelligence

Dr. Jonathan Vanover

12 February 2023

**TV-Show Recommendation System**

The “final” version of this program could be considered an expert system that recommends TV shows based on their content, while utilizing a combination of a rule-based system and a neural network to make recommendations. This program was written based on the work from Jha (2019), while implementing additional features and removing some as well. The goal of the program was to create an AI system that is able to solve the problem of getting recommendations from a TV show you have recently watched and enjoyed. Throughout this review we will be discussing several aspects of this program, to include; the tools, libraries, and APIs utilized; the search methods used; deep learning models; how expert system concepts and knowledge representation is demonstrated; and how symbolic planning is used.

**Libraries**

For this program, there were several libraries used, including Scikit-learn, Natural Language Toolkit (NLTK), TensorFlow, and Pandas. Scikit-learn has many different features, of which we used it to compute the cosine similarity between the vectors of the many different TV shows. The cosine similarity is used as a measure of similarity between two non-zero vectors of an inner product space (Pedregosa et al., 2011). In this case, the feature vectors are created by transforming the tags associated with each TV show into a numerical representation using the CountVectorizer class. The cosine\_similarity function from Scikit-learn’s metrics.pairwise module is then used to compute the cosine similarity between all pairs of TV show vectors. In addition to Scikit-learn, we also used NLTK. NLTK is another library that provides tools to work with human language data (Aarsen et al., 2023). In the program created, we used PorterStemmer from NLTK to perform stemming on the words in the text data, to reduce the dimensionality of the tags and make the analysis more efficient. We also used TensorFlow, which is a library for dataflow and differentiable programming across a range of tasks (Abadi et al., 2015). We used TensorFlow to create a multi-layer neural network that takes the TV show’s features as input and returns a dense vector representation of the TV show, the cosine similarity between the shows is then calculated to determine the similarity between two shows. The shows with the highest similarity scores are then recommended to the user. The neural network is implemented as a Keras Sequential model, and it is trained on the feature vectors of the shows using the “fit” method. We also used the Pandas library, which provides data structures, tools for reading, writing, and processing data (The Pandas Development Team, 2020). We used the Pandas library to do a variety of preprocessing processes of the TV show database. We also used the os library to suppress warning messages that were clogging the output terminal.

**Search Method**

This program uses the cosine similarity method to determine the similarity between two sets of data. The cosine similarity measure is a widely used method for determining the similarity between two documents in a document retrieval system (Eghbali & Tahvildari, 2019). The method measures the cosine of the angle between two vectors, where each vector represents a document. A similarity score of 1 indicates that the two documents are identical, while a similarity score of 0 indicates that the documents are completely different (Krish Naik, 2019).

**Deep Learning**

The program includes a neural network, implemented using the TensorFlow library, for embedding TV shows into a lower-dimensional vector space. The embeddings are then used to calculate the cosine similarity between different shows. The deep learning model used is a Multi-Layer Perception model built using the Keras library in TensorFlow. The model is a fully connected network consisting of 5 dense layers with 128, 64, 32, 16, and 8 neurons respectively. The activation function used for all the layers is ReLU (Rectified Linear Unit). The show data is processed and vectorized, and the vectors are used as input to the model. The vectors are trained on the model using the Adam optimizer and the mean squared error loss function (Heaton et al., 2017). After training the model, it is used to predict the vectors of the shows and cosine similarity is used to calculate the similarity scores between the show to be recommended and the other shows in the dataset. The deep learning model is a critical component of the expert system as it provides a means to measure the similarity between shows, which is then used to make recommendations to the users.

**Expert System Concepts**

Aspects of the program that utilize expert system concepts are the usage of rules and knowledge representation. First the expert system uses a set of rules to make recommendations (Hatzilygeroudis & Prentzas, 2004). These rules are represented by instances of the Rule class, which includes an antecedent (a condition), a consequence (the result of the condition being met), and a weight (the importance of the rule). These rules can be used to represent expert knowledge in a specific domain. The program also uses knowledge representation, which is represented by the show vectors and encapsulates the features of the shows. These vectors are used to make recommendations and to measure similarity between shows. The if-then rules provide a way to capture the relationship between the shows and the recommendations to be made. By representing knowledge in this way, we can make use of mathematical concepts to make recommendations based on the represented knowledge.

**Symbolic Planning**

Symbolic planning is a subfield of Artificial Intelligence that deals with creating automated systems that can generate plans to solve problems (Illanes et al., 2020). It involves representing a problem as a set of symbols or objects and the relationships between them, and then using algorithms to generate a sequence of actions that leads to a goal state (Grounds & Kudenko, 2008). Symbolic planning was used in the program to sequence the steps needed to generate recommendations to the user. The symbolic planning involved defining the steps needed to prepare the data, train the deep learning model, and generate recommendations based on the similarity scores and if-then rules.

**Conclusion**

In conclusion, the program that was created was designed solve a real world problem in a way that I was able to. For this program there were several libraries used, each with their own specific usages, while also needing each other in order for the program to operate. We used the cosine similarity as the search method for this program. The implementation of deep learning models was through a neural network using the TensorFlow library. Expert system concepts that were used for this program were through the rules and knowledge representation. Finally, symbolic planning was used in the program to sequence the steps needed to get recommendations.

**References**

Aarsen, T., Nothman, J., Bird, S., Dimitradis, A., Sepler, D., Milajevs, D., Bond, F., Kurenkov, I., Tan, L., Garrette, D., Ljunglof, P., Korobov, M., Rudnick, A., Loper, E., Klein, E., Cohn, T., Pantone, P., Schneider, N., Justen, A., & Neergaard, M. M. (2023, January 2). NLTK. Retrieved February 12, 2023, from https://www.nltk.org/

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *Preliminary White Paper*. https://doi.org/10.5281/zenodo.4724125

Eghbali, S., & Tahvildari, L. (2019). Fast cosine similarity search in binary space with angular multi-index hashing. *IEEE Transactions on Knowledge and Data Engineering*, *31*(2), 329–342. https://doi.org/10.1109/tkde.2018.2828095

Grounds, M., & Kudenko, D. (2008). Combining reinforcement learning with symbolic planning. *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, 75–86. https://doi.org/10.1007/978-3-540-77949-0\_6

Hatzilygeroudis, I., & Prentzas, J. (2004). Integrating (rules, Neural Networks) and cases for knowledge representation and reasoning in expert systems. *Expert Systems with Applications*, *27*(1), 63–75. https://doi.org/10.1016/j.eswa.2003.12.004

Heaton, J., McElwee, S., Fraley, J., & Cannady, J. (2017). Early stabilizing feature importance for tensorflow deep neural networks. *2017 International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/ijcnn.2017.7966442

Illanes, L., Yan, X., Toro Icarte, R., & McIlraith, S. A. (2020). Symbolic plans as high-level instructions for reinforcement learning. *Proceedings of the International Conference on Automated Planning and Scheduling*, *30*, 540–550. https://doi.org/10.1609/icaps.v30i1.6750

Jha, A. (2022, December 2). *Movie recommendation system: With Streamlit and python-ml*. CopyAssignment. Retrieved February 12, 2023, from https://copyassignment.com/movie-recommendation-system-with-streamlit-and-python-ml/

Krish Naik. (2019, August 19). *Cosine Similarity and Cosine Distance*. YouTube. https://www.youtube.com/watch?v=ieMjGVYw9ag

The Pandas Development Team. (2020, February). *Pandas*. pandas. Retrieved February 12, 2023, from https://pandas.pydata.org/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.