Multimodal Analysis of User-recipes Interactions

Emilija Georgievska[1][a], Martina Stojanoska[1][b], Sanja Mishovska[1][c] Tome Eftimov[2][d] and Dimitar Trajanov[1][e]

[1]*Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, Skopje, Republic of North Macedonia*
[2]*Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia*

Keywords: Food, Recipes, Interactions, Recommendations.

Abstract: A good diet is essential for good health and nutrition, but also as a way of expressing and feeling good. Culinary and food recommender systems are becoming increasingly popular at a time when people are facing fast-paced lifestyles. In this paper, we are analysing interactions between users and recipes in order to make food recommendations based on their previous behaviour which would result in higher personalization for every single person. This also raises the question of whether people stick to what they know well or are open to new suggestions, or do personal recommendations lead to more homogeneity.

1 INTRODUCTION

As technology advances and succeeds to rise above its peak in plenty of industries, there is still space for a big breakthrough in the food industry. For sure, it is great to have a digital menu on your phone with a single scan of a QR code, but what about food recommendation systems? A recommendation system with the possibility of generating good food suggestions can revolutionize not only the food industry but also people's lives.

However, the food domain is not straightforward or uncomplicated since it revolves around people and their unique characteristics. To be successful, one recommendation system needs to deal with numerous features such as recipes, food ingredients, and their mutual combination, nutritional values, and cooking methods.

Following the factors that should be addressed and the features that should be taken into consideration, it is vastly complex to build a recommendation system that does it all. There are numerous types of food recommendation systems, each with its features and focusing on different problems. For instance, a recommender system that focuses on the person's behaviour and the food industry.

What additionally complicates the development of recommender systems is the concept of filter bubbles. The term filter bubble suggests a situation that means that every user on the Internet exists in its universe of information where we receive information based on our behaviour, knowledge, and activities, neglecting the diversity of information that we should be receiving (Bruns, 2019). Even in environments where there are no recommender systems, a large portion of the users tend to repeat their eating patterns and habits. The presence of recommender systems especially ones that are based on previous user's behaviour can furthermore lead to homogenization (Aridor et al., 2020). The question is while developing recommender systems should we try to burst the bubble or give the user what he already knows and likes (Amrollahi, 2019).

The whole idea behind recommender systems and recommendations themselves lies behind the desire to help people by assisting them in the decision-making process and overcoming the load of available

[a] https://orcid.org/0000-0001-9975-6649
[b] https://orcid.org/0000-0002-7287-436X
[c] https://orcid.org/0000-0002-0880-9205
[d] https://orcid.org/0000-0001-7330-1902
[e] https://orcid.org/0000-0002-3105-6010

information (Trattner and Elsweiler, 2017). As in everyday life, they are also a helping hand in e-commerce and in classic commerce so the businesses can bring their products as close as possible to the needs of their customers (Schafer et al., 1999). Recommender systems in the food industry are becoming more relevant over the years and the demand for them is increasing (Trattner and Elsweiler, 2017).

In this paper, the focus is put on a single person's lifestyle and eating habits. By using multimodal analysis of the users-recipes interactions the recommendation models will be able to predict possible future food choices, and with comparison between the two, we will be able to tell if the results acquired are similar or a complete match.

With the ubiquity of the Internet, individuals tend to share everything in their lives nowadays, and food recipes are not an exception. Currently, there are many cooking websites, few clicks away from us on the Internet, which provide various recipes and food content (e.g., description, meal photos, cooking videos, how-to guides), as well as useful functions for searching and filtering.

The dataset used in this research is data from Food.com, publicly available on Kaggle (Li, 2019).

Following the data, the main goal is to build a system that can efficiently predict which recipe can be of interest to a user by learning about the user's past choices and preferences and also adding diversity to the recommendations This would minimize user's efforts to search through enormous databases of recipes on websites, only to find ones that are not to their liking.

2 RELATED WORK

Recommender systems have a core function to recommend items that the user would actually take into consideration (Ricci et al., 2011), with content filtered recommendations that consider the user's past choices and preferences, or collaboratively filtered recommendations that consider similar users and their choices and interests (Shokeen and Rana, 2020).

Some researchers say that to give good food recommendations, it is necessary to take into consideration the quantity of the ingredients and also the specificity of the ingredients in the recipes that the user browsed or cooked (Ueda et al., 2014). Given this, the algorithms often find similar recipes based on overlapping ingredients, either treating each ingredient equally or by identifying key ingredients.

(Geleijnse et al., 2011) built a version of a personalized recipe recommendation system that suggests recipes to users based on past food choices and nutrition intake.

Though there are several choices, to our knowledge, we chose a hybrid, multimodal approach, which connects the previous history of recipe usages for the user and also the ingredients contained, in order to result in recommendations that are familiar to the user, but also recommendations that incorporate new various ingredients. Therefore, in this paper, we do our first attempt to investigate how good the recommendations can be if they are based on the user's previous experience or as we like to say food choices and preferences and the ingredients in those food choices.

But in order to map the whole user's history into one piece, a specific approach is needed. If interactions are mapped to a bipartite user-recipe interaction graph, a recommendation problem can be transformed into a link prediction problem (Li and Chen, 2009). The constructed bipartite graph can capture important information on the relationship between the users and case recipes (Li and Chen, 2009). However, a weighted network is much more informative than an unweighted one, so a lot of techniques can be applied to determine the link's weights (Zhou et al., 2007).

Even though low-dimensional node embeddings in large graphs have been proven as useful in link prediction problems as this one, a big deal of the approaches require all of the nodes to be present during embeddings training (Hamilton et al., 2017). However, due to the low generalization to unseen nodes of these approaches, they do not seem fit to recommend problems. GraphSAGE and his heterogeneous version HinSAGE, efficiently generate node embeddings for previously unseen data, where instead of training individual embeddings for each node, a function that generates embeddings by looking into the node's local neighborhood is learned (Hamilton et al., 2017).

On the other side of the story, in the last few years, many meta-path-based algorithms are proposed to carry out data mining tasks over HINs, including similarity search, personalized recommendation, and object clustering. In particular, the concept of meta-paths, which connect two nodes through a sequence of relations between node types, is widely used to exploit rich semantics in HINs. Following the example of the metapaths and their meaning, in this particular research the metapath P1: $U \rightarrow R \rightarrow U$, will mean that two users have cooked the same recipe, whereas the metapath P2: $R \rightarrow U \rightarrow R$, will mean that a user has cooked two of the recipes. Metapath2Vec is proven to capture better proximity properties, semantics between different types of nodes and learn better than other the algorithms which abandon different node and edge types (Zhang et al., 2018).

3 DATA-SETS AND FEATURES

The data is crawled from Food.com which is one of the largest online recipe sharing communities, covering 18 years of activity (January 2000 to December 2018).

There are three key parts in the data: recipes, users, and interactions between them. For every recipe there is detailed information such as the ingredients that we need to prepare it, preparation directions, categories added by users, and also the nutrition facts of the given recipe. Also, for every user there is information about the cooking techniques encountered by the user, recipes interacted with and rating to the recipes retrospectively.

The raw data consists of more than 180 thousand recipes and 700 thousand user-recipe interactions (reviews and ratings).

One very mitigating circumstance is that the data mainly comes in 2 different formats, where the first part of the data-sets are more natural with descriptive textual attributes in natural English language and other languages, that we use to understand and analyse the data. The second part of the data-sets are the same data-sets as the first, but pre-processed, without missing values, and encoded textual attributes to numeric attributes. These include separate pre-processed data-sets for training, testing and validating. This makes it very easy to use the pre-processed data-sets directly as an input to algorithms. All of the data-sets consist of comma-separated values.

After analysing each one of the data-sets, there are some interesting facts about the overall data: 231.637 recipes were published by 27.926 users, and most recipes were published by the user with id: 89831 (3118 recipes). The average steps for making one recipe are 9 steps, and there is a recipe with 145 steps. The average ingredients for making a recipe are also 9 ingredients. Recipes that contain more than 25 ingredients are very rare. It is also shown that over the years, users tend to change their lifestyle and interact with healthier recipes.

4 METHODS

Using a multimodal approach, the first aspect that is being considered is the temporal dimension of the data with the help of the StellarGraph Python library and the HinSAGE algorithm which is a heterogenous extension of the GraphSAGE algorithm (CSIRO's Data61, 2018).

The goal here is to predict if the interactions that we know really occurred. For that purpose, the data is divided in two parts using the first half of the data for training and the second half of the data for testing the models.

The second aspect focuses on future recipe recommendations, instead of the ones that already happened. First of all, we obtain recommendations with similar ingredients based on text similarity, and then recommendations based on the graph structure with MetaPath2Vec (Dong et al., 2017).

4.1 Temporal Interaction Prediction

Since the main goal revolves around personal recommendations, it is crucial to modify the data in a way that the information within is captured correctly and efficiently.

The nature of the data implies that there are two distinct entities - users and recipes which are mutually connected by interactions. The interaction can be a rating or a review of a particular recipe which is preceded by trying the same recipe. However, although the users and recipes are connected, it is not naturally implied that the users are mutually connected or the recipes are mutually connected. Because of this structure, a user-recipe network can be created as a complex network which displays a natural bipartite structure.

A bipartite graph is a graph whose vertices can be split in two independent sets, where every edge connects one vertex from the first set and one vertex from the second set (Guillaume and Latapy, 2004).

Following the particular data, a bipartite graph with two types of nodes (users and recipes) and edges that correspond to user-recipe interaction can be constructed. Besides that, a proper weighting method is required so that the original information is captured and retained (Zhou et al., 2007). All of the links have the default weight of 1, which suggests that a link is existent (note that if a link weights 0, it is not present in the graph). Weighing the links by their existence contributes more to the link-prediction problem in graphs than weighing the links (note that a link is an interaction) by the rating.

The first step towards constructing some kind of structure that can be used for predictions was trying out the NetworkX library X (Hagberg et al., 2008). NetworkX is a Python software package that is used to study, create and manipulate large complex networks that may be represented in the form of graphs, like this case. Adjacency matrix was created with users on one side and recipes on the other side, with a 1 on places where interaction took place, and 0 where it did not. This matrix later was fed into a NetworkX graph. One drawback was that the matrix was very sparse, and it required great computational power for processing. Another drawback was that this structure was successful only to the point where predicting algorithms ought to be used. Many state-of-the-art algorithms require a specific kind of graph as an input, so they can thoroughly learn about the graph and the relations it has.

The most suitable structures and state-of-the-art prediction algorithms for building bipartite graphs and link prediction problems are part of the StellarGraph library, which offers an easy way to discover and learn the patterns in the data and guide towards good recommendations.

StellarGraph as a class for graph machine learning offers an easy way for storing graph structure by providing collection of nodes and collection of edges which connect a source node to a target node. In order to achieve and preserve the heterogeneity of the data, a heterogeneous undirected StellarGraph is defined, with users as the first and recipes as the second type of the nodes.

Given a bipartite user-item interaction graph, we can convert the recommendation problem in a link prediction problem (Li and Chen, 2009).

Considering that, with the help of HinSAGE, supervised link prediction is the next step towards building a recommendation system. HinSAGE supports representation learning, node classification/regression and link prediction/regression as the original GraphSAGE but for heterogeneous graphs. Link prediction is one of the most important research topics in the field of graphs and networks. The primary goal of link prediction is to discover pairs of nodes which are likely to form a link in the future. Retrospectively, in this research paper, given a pair of nodes, the goal is to identify whether a link would or would not exist in the future. But how can solving link-prediction problems be of a service to the construction of recommendation systems? Given the recommendations obtained from the next section, the HinSAGE model could predict the probability of a link between a particular user and the recommendation suggested.

The model has the following architecture: two-layer HinSAGE model that takes labeled (user, recipe) node pairs that correspond to a particular user-recipe interaction and output a pair of node embeddings for the user and recipe nodes. Then these embeddings are fed into a link regression layer, which concatenates them and turns them into a link embedding. The newly constructed link embeddings are passed through a link regression layer to obtain the probability of a certain user-recipe node pair existence. The entire model is trained end-to-end by minimizing the root mean square error as the loss function. The model's ability to predict a certain user-recipe interaction lies behind the manipulation of the link's weights and therefore simplifying the problem as a binary classification problem.

As said before, all the existent links in the graph have the default weight of 1. To balance the model, non-existent links were generated and weight of 0 was given to them (a link between a user and recipe has weight of 0 only if there was no interaction between the pair). To treat the problem as a classification problem, one more thing was done. The results from the link regression layer were rounded with a threshold of 0.5. If the result is below 0.5 – there is no significant evidence that shows that there is or will be an interaction between a pair of a user and a recipe. Likewise, if the result is above 0.5 or 0.5 – there is evidence that leads the model to believe that there is or will be an interaction between a pair of a user and a recipe.

Following this principle, all of the known interactions were divided by years, taking the first ten years for training and the rest for testing (following the 70:30 rule). Figure 1 shows that interactions are at their peak in 2008 and thereafter begin to drop, owing to people sharing less information and interacting

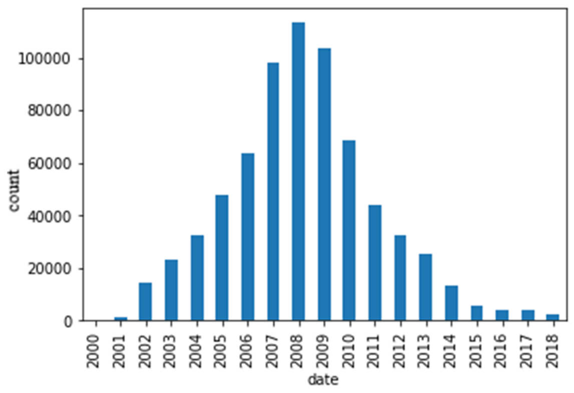

Figure 1: Interactions between users and recipes by years.

with previously submitted recipes over time. This was done to research more about the temporal ability of graphs and how they change over time.

It is obvious that some people's preferences and choices change over time, along with the trend of a healthier life that has been more present over the past few years. All of this is visualized in Figure 2, from which we can see how the consumption of olive oil is increasing year after year.

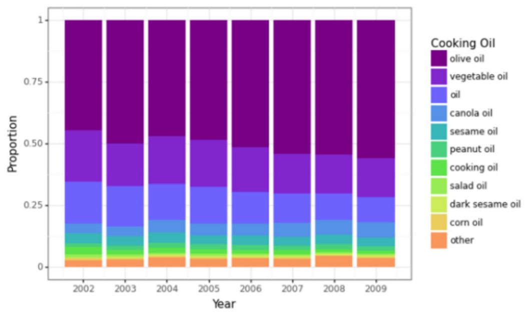

Figure 2: Cooking Oil usage through the years.

By testing with the rest 30% of the data (from 2011 to 2018), the model had an accuracy of 0.828. The accuracy led to a conclusion that the model gives good results based on a user's previous behaviour, given the fact that over time people's taste can change. Given the model which can predict if there is or will be an interaction between a user and a recipe with an accuracy of nearly 0.828 shown on Figure 3, the next part of the research is to find the right algorithm for recipe recommendations. Then using the model, we can test if the corresponding user would take the recipe recommendation in consideration. This can be done by testing if the model predicts an interaction between the pair of user and recipe in consideration.

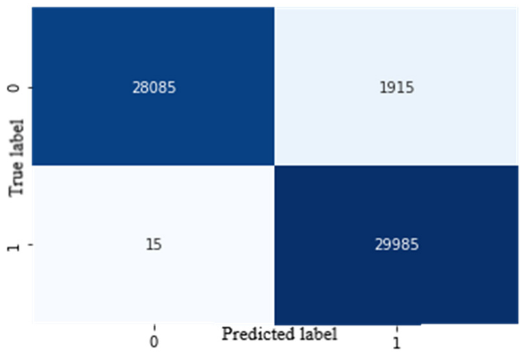

Figure 3: Prediction if a future interaction will happen.

4.2 Future Recipe Recommendation

This aspect is completely different from the first one, and it focuses on the future recipe recommendations.

It is implemented in 2 parts. The first one is done using the ingredients as the primary tool for the recommendations and the second one focusing on the user's previous behaviour. The reason behind trying two different methodologies is to try and analyse the concept of filter bubbles and how to burst them. The goal behind every recommendation is to be of a user's taste but also try to stray of a straightforward personal recommendation.

Additionally, it is expected that the following two approaches give completely different results, so the idea is to create a new, hybrid approach that would take into account the habits of the user, but would also add a new perspective and different recommendations, thus bursting the filter bubble that could happen.

4.2.1 Text Similarity Recommendations

The purpose of generating recommendations based on text similarity leads to recipes that are ingredient-wise to the ones that the user already interacted with. The algorithm for generating recipe embeddings that was taken into consideration was BERT (Devlin et al., 2018). BERT stands for Bidirectional Encoder Representations from Transformers. BERT learns contextual relationships between words in a sentence or text.

The goal here was to get recommendations based on the recipe's name - which almost always includes the ingredients of the recipe. A sentence-transformer BERT was chosen to get the embeddings for the recipes with help of the raw recipes data-set. The raw recipes data-set does not contain pre-processed data, therefore all of the attributes for one recipe are textual and not numeric, which is good for BERT to take them as text and encode them by its rules. The text similarity embeddings actually rely on the recipe's name, which in this case is highly productive because of the way the recipes are named (e.g. arriba baked winter squash Mexican style). As said, the recipe's name includes some of the ingredients (mostly the key ingredients), the cooking technique (e.g. baked) and other adjectives which additionally describe the food recipe (e.g. Mexican food).

Large numbers of the recipes included in the data-set had zero reviews and ratings - never seen recipes, so they are not taken into consideration while building the recommendation system based on text similarity for better accuracy. Also, it is relevant to mention that some of the recipes had rather strange, creative (e.g. smells like Sunday chicken fricassee with meatballs) names that may confuse the text similarity model in the process of generating the embeddings.

The newly generated recipes embeddings by text similarity are then used for cosine similarity calculation. In other words, there is a pairwise comparison or cosine similarity between pairs of recipes - each recipe compares with the rest of the recipes. This computation is done on just one part of the embeddings because of the big computation power that is needed. Cosine similarity, mathematically, measures the cosine of the angle between two vectors in a multi-dimensional space, in this context two recipes transformed into embeddings. When plotted on a multi-dimensional space, where each dimension corresponds to a word in the recipe, the cosine similarity captures the orientation (the angle) of the recipes. The cosine similarity is advantageous because even if the two similar recipes are far apart by the Euclidean distance they could still have a smaller angle between them. If the angle is smaller, the similarity will be higher and otherwise.

After the cosine similarity is calculated, the closest or the most similar recipes to a given one can be found - based on the highest cosine similarity score.

The recommendations from the text similarity model, as expected, were not surprising. BERT using text similarity shows a great focus on the key ingredients of the recipe. If the recipe of interest was a dessert for example, all of the recommendations would also be desserts. Or if a recipe has chicken as its key ingredient, then all of the recommendations would also contain chicken, nevertheless if it is a breakfast or lunch. It can be concluded that text similarity would be a great deal if the goal is to generate recommendations only based on an ingredient of preference, but only to the point where no other factors are taken into consideration.

4.2.2 Node Embedding Recommendations

The idea of generating recommendations based on the whole graph is to find recipes that the user might like based on the whole graph structure.

The algorithm that was chosen for generating recipe embeddings in the process of creating recommendations was Metapath2Vec along with cosine similarity calculations.

As already described, the bipartite graph has a simple structure, consisting of two types of nodes (recipes and users) connected with a link that represents an interaction. There are no connections between two users or two recipes.

Given this structure, with the help of Metapath2Vec and Word2Vec node embeddings were generated. The Metapath2Vec model formalizes meta-path based random walks to construct the heterogeneous neighbourhood of a node and then leverages a heterogeneous skip-gram model to perform node embedding (Dong et al., 2017). The Metapath2Vec algorithm includes 2 steps: First, Use uniform random walks to generate sentences from a graph. A sentence is a list of node IDs. The set of all sentences makes a corpus. The random walk is driven by a metapath that defines the node type order by which the random walker explores the graph.

The random walks have fixed maximum length and are controlled by the list of metapath schemas. Second, the corpus is used to learn embedding vector for each node in the graph. Each node ID is considered a unique word/token in a dictionary that has a size equal to the number of nodes in the graph. The Word2Vec algorithm is used for calculating the embedding vectors.

In this case, the metapath which defines the node type order of the random walk consists of two schemes which change the order of the nodes. To be more precise, each metapath schema must start and end with the same type of node (e.g. start and end with a recipe or a user). In this particular case, there is a scheme for a user to a recipe, ending with a user, and from a recipe to a user, ending with a recipe. Schemes that included a walk between the same type of nodes (e.g., from a user to a user) weren't taken into consideration because of the graph structure which states that there are no interactions between the same type of nodes.

Given the newly constructed node embeddings, cosine similarity was again used as a measure of nodes similarity. Given a particular recipe node, the top 10 similar nodes were considered to be the most similar ones to that node. So in order to test the previous hypothesis, explore more about the difference between the two algorithms and to find the possible link that connects them both, few experiments were run.

The results from Metapath2Vec came back rather different than the ones with the previous algorithm. It was concluded that the most similar recipes to a particular recipe, were more diverse and varied within multiple types of meals (dessert, breakfast, lunch, dinner), opposite to BERT that sticks to a same type of meal, and almost same ingredients, making the recommendation very similar to the past food choices of the user.

5 EXPERIMENTS AND RESULTS

Text similarity recipe recommendations for a lunch come back as lunches with focus on the ingredients of the recipe, whereas the results from the Metapath2Vec vary among different types of meals such as lunch, dessert, or dinner with no focus on the ingredients whatsoever.

To test the previous assumptions, a few experiments were conducted to compare the recommendations from text similarity and node embeddings. For this purpose, the following steps are applied: Choose a random user, choose a recipe that the user has interacted with, make text similarity based recommendations for that recipe, make node embedding recommendations for that recipe and last check if text similarity recommendations and node embedding recommendations overlap.

So, 100 random users were taken and for each of them, a random recipe from the list of their interactions was chosen. The next step was a generation of 10 text similar recommendations and 10 node-embedding recommendations. The recommendations were compared with each other, and they didn't overlap in any of the cases, which means that there is a difference between them.

Text similarity focuses on the words while node embeddings are calculated based on the whole graph structure, aka the user's previous behaviour. However, this does not mean that we can overthrow any of these approaches, but moreover gives an opportunity to create a new, hybrid one that combines the previous two. It is expected that by combining an approach that focuses on a user’s past behaviour and an approach that focuses on the ingredients, the filter bubble that personal recommendations can create, can easily be burst and a new perspective can be shown.

6 CONCLUSIONS

The overall goal of the whole research is to test various approaches for making recipe recommendations, and combine them into one hybrid approach that takes the best of them.

To do so, we analyse user-recipe interactions from two aspects, through future recipe recommendations and through temporal interaction predictions.

In the first aspect after analysing the results from the recipe recommendations, two conclusions were derived. The recommendations based on node embeddings are more diverse. For example, if you are looking for a recommendation for a starter meal, the recommender will offer a main course or a dessert rather than a similar starter meal, whereas text similarity as different approach focuses on the ingredients and will offer similar starter meals or meals that have similar ingredients. The text similarity method would give great recommendations for people who like to stay consistent with their food choices and like to repeat the same food pattern and ingredients. The approach with node embeddings would give great recommendations for people who like to try out new things and follow the food trends. Furthermore, knowing that filter bubbles are not the most desirable situation, the combination of these two approaches would prevent their formation (Aridor et al., 2020).

ACKNOWLEDGEMENTS

This work was supported by the Slovenian Research Agency (research core funding programmes P2-0098); the European Union's Horizon 2020 research and innovation programme under grant agreement 863059 (FNS-Cloud, Food Nutrition Security) and under grant agreement 101005259 (COMFOCUS).

REFERENCES

Ricci, F., Rokach, L., Shapira, B., Kantor, P. B. (Eds.). (2011). Recommender Systems Handbook. *Springer.*

Ueda M, Asanuma S, Miyawaki Y, Nakajima S. (2014). Recipe recommendation method by considering the users preference and ingredient quantity of target recipe. In: *Proceedings of the international multiconference of engineers and computer scientists, vol 1.*

Geleijnse, G., Nachtigall, P., van Kaam, P., Wijgergangs, L. (2011). A personalized recipe advice system to promote healthful choices. In: *Proceedings of the 16th International Conference on Intelligent User Interfaces.*

Li X., Chen H. (2009). Recommendation as link prediction: a graph kernel-based machine learning approach In: *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries.*

Zhou, T., Ren, J., Medo, M., Zhang, Y.-C. (2007). Bipartite network projection and personal recommendation. *Physical Review E 76:046115.*

Hamilton, W., Ying, Z., Leskovec, J. (2017). Inductive representation learning on large graphs. In: *Advances in Neural Information Processing Systems.*

Zhang, D., Yin, J., Zhu, X., Zhang, C. (2018). MetaGraph2Vec: Complex Semantic Path Augmented Heterogeneous Network Embedding. In *PAKDD. Springer.*

Trattner, C., Elsweiler, D. (2017). Food recommender systems: Important contributions, challenges and future research directions.

Guillaume, J. L., Latapy, M. (2004). Bipartite structure of all complex networks. In: *Information processing letters.*

Devlin, J., Chang, M., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Hagberg, A. A., Schult, D. A., Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In: *Proceedings of the 7th Python in Science Conference*

CSIRO's Data61 (2018). StellarGraph Machine Learning Library. *GitHub Repository*. https://github.com/stellargraph/stellargraph

Dong, Y., Chawla, N. V., Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Bruns, A. (2019). Filter bubble. *Internet Policy Review [Internet].*

Aridor, G., Goncalves, D., Sikdar, S. (2020). Deconstructing the filter bubble: User decision-making and recommender systems. In: *Fourteenth ACM Conference on Recommender Systems.*

Amrollahi, A. (2019). Burst the Filter Bubble: Towards an Integrated Tool. In: *30th Australasian Conference on Information Systems.*

Schafer, J.B., Konstan, J. Riedl, J. (1999). Recommender systems in e-commerce. In: *Proceedings of the 1st ACM conference on Electronic Commerce.*

Shokeen, J., Rana, C. (2020). A study on features of social recommender systems. Artificial Intelligence Review 53(2): 965–988

Shuyang Li. (2019, August) Food.com, Recipes and Interactions, Version 2. Retrieved March, 2021 from https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions