SmartRecepies: Towards Cooking and Food Shopping Integration via Mobile Recipes Recommender System

Josef Starychfojtu
Department of Software Engineering,
Faculty of Mathematics and Physics, Charles University
josef.starychfojtu@gmail.com

Ladislav Peska
Department of Software Engineering,
Faculty of Mathematics and Physics, Charles University
peska@ksi.mff.cuni.cz

ABSTRACT

Recommender systems are now part of our daily life more than ever and most users are confronted with some form of recommendation on a daily basis. As users of such systems, we don't need to actively seek for new content, but let it be comfortably recommended to us instead. One of the important parts of our lives that is yet to be covered in this way is the domain of cooking. A traditional dilemma of a person, who is currently in the process of shopping for food is "What else should I buy, so that I can cook something new?" In another words, the person either has to look for novel recipes upfront (which does not have to correspond with available ingredients in the shop), or buy ingredients intuitively (which does not have to correspond with recipes).

The main objective of this paper is to bind cooking and shopping activities together via a mobile recipes recommendation application. The application responds on the content of a user's shopping list and strives for calibration of recommended recipes. In an online user study, we also show that calibrated recommendations outperform both diversity enhanced and plain similarity-based recommendations.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

recipes recommendation, mobile applications, calibration

ACM Reference Format:
Josef Starychfojtu and Ladislav Peska. 2020. SmartRecepies: Towards Cooking and Food Shopping Integration via Mobile Recipes Recommender System. In *The 22nd International Conference on Information Integration and Web-based Applications Services (iiWAS '20), November 30-December 2, 2020, Chiang Mai, Thailand.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3428757.3429096

1 INTRODUCTION

Recommender systems (RS) belong to the class of automated content-processing tools, aiming to provide users with unknown, surprising, yet relevant objects without the necessity to explicitly query for them. Recommending services spread across many different domains including e-commerce, music and video streaming services, point-of-interests and trip planning services and many others. Most of the internet users are affected by some form of recommendation on a daily basis.

One of the domains, where recommender systems are so far not utilized much are cooking and food shopping. Similarly as watching a movie or listening to some music, cooking is often considered as an enjoyable activity. This inherently brings a strive for innovations, which can be at least partially satisfied by the recommendation of novel cooking recipes. Although cooking seems as a suitable area for recommender systems deployment, we are not aware of any commercially successful recommendation application in this area. We assume that there are three primary reasons for this: more difficult monetization, localization of cooking endeavors and the need for incorporation with food shopping. As for the monetization, note that users by default do not need any software application in order to cook (as compared e.g. to music or video streaming services). Therefore, it may be difficult to persuade users about the added value of any cooking-related application. Also, due to the influence of local history, traditions and availability of certain ingredients, cooking has always been highly location-sensitive. This limits the impact (or increase the complexity) of any generic service. Finally, cooking activity is often bound to the food shopping (i.e. ingredients gathering) and the lack of coordination between these two activities may decrease the usability of any cooking-support application.

In this work, we focus primarily on the last mentioned obstacle, i.e., connecting the domain of cooking with food shopping. Specifically, we propose a SmartRecepies mobile application that aims on recommending recipes based on the content of user's shopping list. The main obstacle to solve is inherent heterogenity of shopping lists that does not correspond to a single goal (i.e., users do not shop just to cook one recipe). To handle this task, we utilized diversity-based and calibration-based recommendation post-processing. Although both methods benefits in different model situations, the results of performed user-study favors the usage of calibration techniques over both diversity enhancements. Nonetheless, both post-processing techniques improves over plain similarity-based recommendations.

2 RELATED WORK

There are several applications that focus on the cooking activities in general. Let us mention allrecipes.com service that is a repository containing some tens of thousands cooking recipes. However, its GUI does not go beyond a simple binary and-like ingredient search. Other applications, such as supercook.com or myfridgefood.com

iiWAS '20, November 30-December 2, 2020, Chiang Mai, Thailand

ACM ISBN 978-1-4503-8922-8/20/11…$15.00
https://doi.org/10.1145/3428757.3429096

provide similar levels of GUI (i.e., ingredients search), but their search algorithm is rather or-like. None of these applications explicitly support recommending based on user's shopping lists, neither are their search GUI suitable for such tasks.

On the other side, there is a range of research papers focused on the problem recipes recommendation (e.g. Anderson [1] provided a nice survey). Also, studies focusing on recommending healthy recipes [5] or food suitable for particular health conditions [8] are quite common. Recipes recommendation approaches mostly focused on collaborative or hybrid techniques, although some content-based recommenders were proposed as well (e.g. [3]). Note that collaborative filtering is generally impractical for cold-start scenarios (such as newly developed application without many users so far). Aside from that, surveyed papers usually adopted a common personalized RS scenario, i.e. recommendations are provided based on past user's interactions with other recipes. This does not correspond with our use-case much, because connecting food shopping with recipes recomendations has a significant contextual dependence. This dependence could, however, be at least partially expressed by the content of user's shopping lists (i.e., *"I would like to cook from these ingredients"*).

In contrast to the related approaches, our work utilizes content-based recommendations, where the user's shopping list is a primary source of information. Instead of a plain binary search, we aim on ranking recipes based on their suitability w.r.t. current shopping list and aim on an appropriate coverage of the full list of ingredients. Therefore, our approach may partially resemble sequence-based recommendation problem, where the current shopping basket represents the user sequence.

Health-aware recipes recommendation [5, 8] on the other hand is rather orthogonal to our work and it can be incorporated into our model in the future.

3 DATASET AND RECIPES DOMAIN

In order to construct a viable application for recipes recommendation from users shopping lists, there are several important constraints for the underlined data. First, it has to contain a reasonable volume of recipes to cover all user needs. Second, each recipe should contain its ingredients in a machine readable format and these ingredients should be mappable to the shopping list content. Finally, recipes should also contain additional metadata, such as difficulty, cooking time, description, photos etc. so it can be properly evaluated and eventually cooked by the user.

Despite a thorough review of existing datasets, we did not find any suitable one that would fulfill all three conditions. Therefore, we decided to construct a new dataset by scraping the AllRecepies website. AllRecepies contain a sufficient volume of individual recipes and all necessary metadata, so the resulting dataset easily fulfills first and third condition. However, it does not contain ingredients in a machine readable format. In order to get this information, we utilized NYT Cooking ingredients parser [1], which aims on parsing the ingredient description (in free text format) into several fields including ingredient name, quantity, unit and comment. Based on the parser results, we can both utilize ingredients names as suggestions for the shopping list entries and utilize the quantity information for the ingredients importance assessments.

In total, the collected dataset contains 10646 recipes, 4864 unique ingredients and 103754 recipe-ingredient relations. Among the most common ingredients were garlic, salt, onion, pepper, olive oil and butter. For over two thirds of recipe-ingredient mappings, unit and quantity information was detected with in total 55 different unit measures (e.g. *tablespoon*, *cup* or *pieces*). Also, around two thirds of mappings contain some comment, where *ground*, *chopped*, *minced* and *dried* were the most common ones. The resulting dataset contained some inconsistencies, e.g. *leftover pork* and *salt and pepper* were parsed as ingredient names. Although the dataset could be manually cleaned, we kept it as is and instead aimed on utilizing recommending methods that can overcome these inconsistencies.

During the data preparation, we were considering also crawling for even larger dataset, or to collect data from multiple websites. However, we found the content of various recipe's websites rather uniform as for both the coverage and provided information, so collecting data from multiple domain would not be much more useful. The dataset of over 10000 recipes seemed sufficiently large for the evaluation purposes[2] and the initial version of SmartRecepies application. However, we plan to re-visit this point in the future work.

3.1 Recommending Algorithms

As the content of the shopping list can be directly mapped to the recipes ingredients, one can utilize some simple similarity-based recommending algorithms. Probably the simplest variant is to use a **Jaccard similarity** $(I_S \cap I_r)/(I_S \cup I_r)$ between the list of shopping list items I_S and recipes ingredients I_r. However, such approach has several disadvantages.

- First, each ingredient is considered equally important, which may not correspond to the reality.
- Second, considering that shopping lists tends to be relatively short (several dozens of items at most), Jaccard similarity would create too many ties in the recipes ordering. Also, Jaccard similarity tends to prefer recipes with very short list of ingredients, which are often over-simplistic or focus only on a fragment of the dish (e.g., only sauce).
- Third, only the exactly matched ingredients counts, yet there may be some similarity or interchangeability relations among them (e.g. onion vs. red onion).
- Finally, in the dataset, there are clusters of similar recipes that use (almost) the same set of ingredients (e.g., variants of stuffed peppers). If such cluster has the highest intersection with the shopping list, all recipes are usually reported and the rest of the shopping list is ignored. Although each of the individual recipes from the cluster is a relevant response, reporting all members of the cluster lacks sufficient diversity and may annoy the user.

In order to cope with the mentioned problems, we proposed several alternative recommending approaches. First two issues may be solved by the introduction of some form of ingredients weighting scheme. We utilized modified **TF-IDF** weighting scheme for this

[1] https://github.com/nytimes/ingredient-phrase-tagger

[2] With larger datasets, the problem of too uniform recommendations would only get more severe.

task. While the IDF definition can remain unchanged, Term frequency (TF) cannot be used directly, i.e., each ingredient is usually mentioned just once in both recipes and shopping lists. Instead, we opted to utilize the quantity information. We defined several heuristic rules to convert parsed quantity and unit information into an ingredient importance feature and use this instead of TF. Ingredient weighting can be also utilized on the shopping list side if the user specifies item's quantity as well. Cosine similarity is used to compare both the recipe and shopping list vectors.

While TF-IDF weighting may solve the first two challenges, it still considers only the exact ingredients matches. Therefore, we adopted a variant of **Word2Vec** model [6] to establish ingredients similarity. Word2Vec model considers sequences of one-hot-encoded entities (sentences of words in the original version) and provides embedding vectors for each entity based on their contextual similarity (i.e., the embeddings are more similar for entities that are often surrounded with the same entities w.r.t. some window size).

For the training, the list of ingredients names (in the same ordering as in the original recipes description) was considered as a sentence. During the hyperparameter tuning, Word2Vec results were rather stable. The final model uses embedding size 256, window size 10 and 100000 iterations. Upon inspecting the most similar ingredients of the final model, most relations seemed reasonable, such as *Red onion* vs. *Sweet onion* or *Chicken breast* vs. *Chicken breast fillet*.

Given the Word2Vec vectors of individual ingredients v_i, the final vectors corresponding to a recipe v_r and a shopping list v_S are constructed as a weighted average of enlisted ingredients' vectors, i.e., $v_r = \sum_{i \in r} w_{i,r} * v_i$, where $w_{i,r}$ is the same TF-IDF weight as used in the previous algorithm. In this way, the notion of ingredients importance is preserved. Recipes are then recommended according to the cosine similarity of v_S and v_r vectors.[3]

Note that although the first three challenges were addressed by TF-IDF and Word2Vec models, these methods still utilizes the shopping list as a whole. This may lead to too uniform recommendations as described in the fourth challenge. In order to cope with this problem, we considered two solutions: diversity enhancements and calibrated recommendations.

For **diversity enhancements**, we utilized standard results reranking via Maximal Marginal Relevance (MMR) [2], where the next recommended item is selected based on the weighted average of item's relevance and its distance to the closest already selected item. In experiments, we set the weighting hyperparameter $\lambda = 0.5$, i.e., we put equal weight on both relevance of the recipe and its distance from already selected ones. Nonetheless, we plan to experiment a bit with the diversity tuning in the future work.

While the diversity problem is well known, the aim for **calibrated recommendations** was first described quite recently [9]. In short, if multiple diverse user's interests (with varying importance) are detected, the list of recommended items should reflect all these interests with proportional representation. However, common recommending algorithms often samples recommendations only w.r.t. the most significant user's interest. In a follow-up work, the Fuzzy D'Hondt's approach [7] was proposed to solve the variant of calibration problem, where diverse interests are represented via individual recommenders. Another advantage of Fuzzy D'Hondt's algorithm is that it also provides a fair ordering of recommended items - not just their calibrated selection.[4]

The original Fuzzy D'Hondt's algorithm assumes to integrate recommendations originating from multiple base recommenders (with possible overlaps). In contrast, our use-case considers only one recommending algorithm, but diverse user's interests may be represented through individual ingredients and their weights. So, the list of recommendations should be proportional to the ingredients represented in the user's shopping list. To comply with these requirements, we propose a modification to the original Fuzzy D'Hondt's algorithm as follows.

Let S be a current user's shopping list, comprised from the list of (a_i, w_i) pairs, where a_i is an ingredient and w_i its weight given by TF-IDF or any other weighting scheme. Similarly, $r_j \in R$ is a recipe object with the same inner representation. Let $\alpha(S, r)$ be a recommending algorithm (cosine similarity over TF-IDF or Word2Vec embeddings in our case) that evaluates proximity of the shopping list S to the recipe r. Let S_i be a fraction of a shopping list containing only the (a_i, w_i) pair and k_i a counter of ingredients usage. Finally, let $\bar{S}$ be a shopping list, where individual ingredients a_i have modified weights $\bar{w}_i$.

The ingredients-wise Fuzzy D'Hondt's algorithm is initialized with $\bar{S} = S$ and $\forall i : k_i = 1$. Then the algorithm iterates over the following steps until the demanded volume of recommended items is selected:

- Select $r_{max} = \text{argmax}_{\forall r \in R}\, \alpha(\bar{S}, r)$ as a next item
- $\forall a_i \in \bar{S}$ update counter $k_i = k_i + \alpha(S_i, r_{max})$ and then update current ingredient's weight $\bar{w}_i = w_i / k_i$.

Note that the output of diversity and calibration enhancements often overlaps. Calibrated recommendations are inherently more diverse if the shopping list is large enough and similarly, diversified recommendations may incorporate otherwise neglected ingredients and therefore enhance calibration as well. However, there are situations, where both methods would behave differently. One example would be shopping lists that correspond to a single user's interest (i.e., a single cooking goal), where ingredients-based calibration would not increase the diversity much. On the other hand, by increasing diversity, there is no guarantee that all user's interests are accommodated. So, in order to understand how are these enhancements perceived by users, we evaluated them in a small user study.

4 RECOMMENDING METHODS EVALUATION

In order to verify our assumptions about the considered recommending methods, we conducted a user study.[5] Each participant was tasked to evaluate predictions of individual recommending methods in six different pre-selected scenarios. Methods were evaluated simultaneously. Each column of the grid of recommendations corresponds to the results of one recommending algorithm and this information was made transparent to the user. Therefore, the user could compare the methods directly.

[3] We also evaluated free text variants of TF-IDF and Word2Vec recommenders. However, their preliminary results were inferior to the ingredients-based variants, so we did not consider them any further.

[4] I.e., the calibration feature is preserved also for the prefixes of the result list.

[5] The study is available on https://hardcore-agnesi-f5351b.netlify.app.

Method	Mean Rating	Jaccard	TF-IDF	TF-IDF & MMR	TF-IDF & calib.	Word2Vec & MMR	Word2Vec & calib.
Jaccard sim.	-0.006	×	–	–	–	–	–
TF-IDF	0.105	–	×	–	–	–	–
TF-IDF & MMR	0.241	∗ ∗ ∗	∗	×	–	–	–
TF-IDF & calib.	0.321	∗ ∗ ∗	∗	∘	×	∗∗	–
Word2Vec & MMR	0.167	∗	–	–	–	×	–
Word2Vec & calib.	0.340	∗ ∗ ∗	∗ ∗ ∗	∘	–	∗	×

Table 1: Results of the user study. Mean results are depicted on the left, while p-values of *"row better than column"* comparison is on the right: ∘, ∗, ∗∗, ∗ ∗ ∗ stands for p-values < 0.1, < 0.05, < 0.01 and < 0.001 respectively.

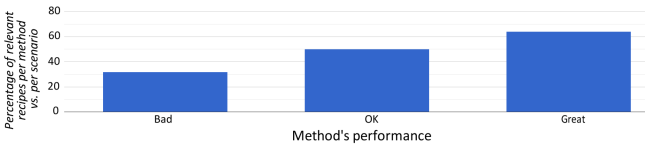

Figure 1: Dependence of method's rating on the ratio of relevant recipes it covered.

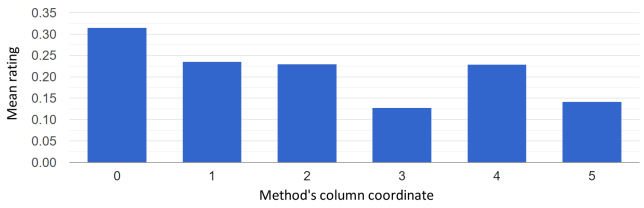

Figure 2: Dependence of method's rating on its position in the grid of recommending methods.

In order to prevent biases, method names were anonymized[6] and their position in the grid was randomized. For each scenario, users may inspect the details of individual recommendations, select some recipes as relevant (this recipe is automatically marked as relevant for all other methods as well) and evaluate the overall method's performance as "Great", "OK" or "Bad". This feedback is numerically transformed into 1/0/-1 ratings[7]. Some statistics (the volume of relevant recipes and the volume of utilized ingredients) were displayed for each method to simplify the evaluation task.

In total 6 recommender variants were evaluated: Jaccard similarity, plain TF-IDF, TF-IDF with diversity and calibration enhancements and Word2Vec with diversity and calibration enhancements. In total 32 users participated in the evaluation from whom we recorded 908 method's ratings covering 192 evaluation screens (users × scenarios). We also recorded additional 1916 recipes inspection or selection events. Results of individual methods are depicted in Table 1. Both plain TF-IDF and Jaccard similarity were clearly inferior to the other methods. Calibration-based recommendation performed slightly better as compared to the diversity enhancements. However, the added value of Word2Vec over TF-IDF was not confirmed. We can also note that apart from plain Jaccard similarity, all methods were evaluated mostly positively.

In additional analysis, we focused on the correspondence of the volume of selected recipes to the overall methods evaluation. As can be seen e.g. on Figure 1, the overall evaluation of methods largely corresponds to the volume of items that were considered as relevant by the user.[8]

We also observed an interesting bias in the methods rating: methods placed in the left columns received higher ratings in average (see Figure 2). Thanks to the randomization of method's positions, this does not affect the overall results. However, this is another confirmation of the search engine bias problem [4]. Note that in the instructions, we specifically asked users to evaluate all methods, so it seems that this may be an involuntary consequence of the human nature.

Finally, we utilized the results of the user survey in the proposed SmartRecepies mobile application that aims on linking the shopping and cooking activities.

5 SMARTRECEPIES APPLICATION

The SmartRecepies application is a client-server application written in F#, with main aim to provide recipes recommendation for partially constructed user's shopping lists and augment them with selected recipes. The application has a classical client-server architecture, where the server side maintains the recipes dataset and provides a REST-like API for recipes recommendation and data augmentation (e.g. creating new ingredients, updating shopping lists, request recommended recipes etc.).

The client side is a mobile application,[9] which implements the business logic and visualize shopping lists contents and recommended recipes. The application was designed according to minimalism design principles - Figure 3 shows several screenshots. Upon authentication, users are allowed to construct their shopping lists, update ingredient's quantities, view suggested recipes and eventually save them for further utilization.

In the current version, SmartRecepies application utilizes Word2Vec recommender with calibration. Nonetheless, the server-side API features all mentioned recommender variants, so the utilized recommender variant may be tuned or explicitly set by the user in future versions. Also, server API can be in principle queried by any client application, so it is plausible to develop additional, e.g., desktop applications easily.

6 CONCLUSIONS AND FUTURE WORK

In general, this paper focused on the challenge of linking shopping and cooking activities together. Specifically, we focused on

[6]Greek letters alpha, beta etc. were used instead of the method names. The mapping was static, so the user could follow the method's performance across different scenarios if he/she wanted to.

[7]If the user did not explicitly rate a method in some scenario, we implicitly assume 0.

[8]Note that additional visualizations corroborated these findings, but we omit them for the sake of clarity and space.

[9]Only available for Android at the moment. The application can be downloaded from https://github.com/starychfojtu/SmartRecipes.Client/blob/master/com.smartrecipes.apk

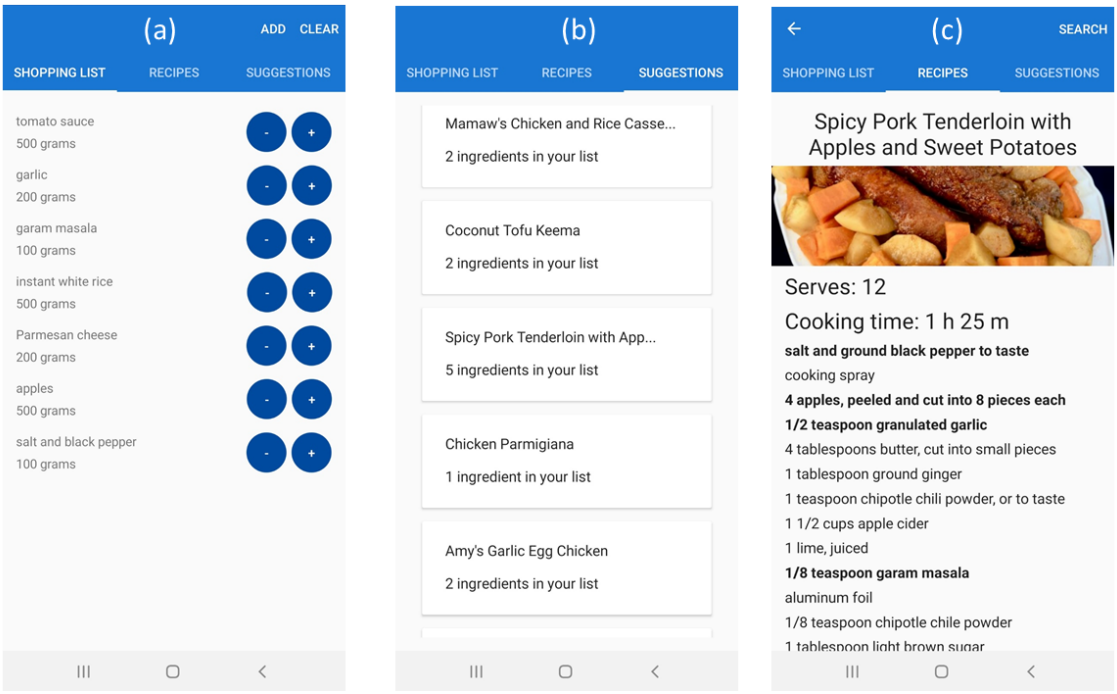

Figure 3: Screenshots from the SmartRecipes application. a) shopping list construction, b) recommended recipes for the current shopping list c) detail of a selected recipe with highlighted ingredients already in the shopping list.

the problem recipes recommendation conditioned by a partially created shopping list of the user. We presented and evaluated several recommending algorithms for this task and demonstrated the importance of recommendations calibration in this domain. Furthermore, we presented a SmartRecepies mobile application that utilizes these findings. The application recommends recipes based on the content of user's shopping list with respect to the proportional representation of shopped ingredients in the resulting recipes.

Our work is in early stages at the moment and there are multiple directions for the future enhancements. First, the application is currently rather minimalistic. For instance, user preferences on both explicit (e.g. liked/disliked ingredients, cooking skills) and latent features could be incorporated into the recommendations. Also, ingredients importance assessment can be extended e.g. by their mean per-recipe quantities or some 3rd party knowledge bases. On more theoretical side, we would also like to focus on merging the advantages of diversity and calibration principles in a single recommender. Also, it could be interesting to learn how to distinguish between various stages of user's interaction process, where the needs for calibration, diversity or maximal similarity may vary. Finally, SmartRecepies is currently a standalone application, which is not too convenient for users. Therefore, one of the main future objectives is to bind it either to some on-line food shopping service, or to some self-scanning devices in classical shops.

ACKNOWLEDGMENTS

SmartRecipes source codes are available from https://github.com/starychfojtu in *SmartRecipes* and *SmartRecipes.Client* repositories. Android application can be obtained from https://github.com/starychfojtu/SmartRecipes.Client/blob/master/com.smartrecipes.apk.

This paper has been supported by Czech Science Foundation project GACR-19-22071Y and by Charles University grant SVV-260588.

REFERENCES

[1] Carl Anderson. 2018. A survey of food recommenders. *CoRR* abs/1809.02862 (2018). arXiv:1809.02862 http://arxiv.org/abs/1809.02862
[2] Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *SIGIR '98*. ACM, New York, NY, USA, 335–336.
[3] T. L. Cheng, U. K. Yusof, and M. N. A. Khalid. 2014. Content-based filtering algorithm for mobile recipe application. In *2014 8th. Malaysian Software Engineering Conference (MySEC)*. 183–188.
[4] T. Joachims and F. Radlinski. 2007. Search Engines that Learn from Implicit Feedback. *Computer* 40, 8 (2007), 34–40.
[5] G. Kovásznai. 2011. Developing an expert system for diet recommendation. In *International Symposium on Applied Computational Intelligence and Informatics (SACI '11)*. IEEE, 505–509.
[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781
[7] Ladislav Peška and Stepan Balcar. 2019. Fuzzy D'Hondt's Algorithm for On-line Recommendations Aggregation. In *ORSUM '19*, Vol. 109. PMLR, 2–11.
[8] M. Phanich, P. Pholkul, and S. Phimoltares. 2010. Food Recommendation System Using Clustering Analysis for Diabetic Patients. In *International Conference on Information Science and Applications*. IEEE, 1–8.
[9] Harald Steck. 2018. Calibrated Recommendations. In *RecSys '18*. ACM, New York, NY, USA, 154–162.