Ensembled Approach for Web Search Result Diversification Using Neural Networks

Shreya Sriram[1], Madhuri Mahalingam[1], Sarah Aymen Naseer[1], Shajith Hameed[1], Rahul Rajagopalan[1], Sai Shashaank R[1], Lekshmi Kalinathan[1] and Prabavathy Balasundaram[1]

[1]*Sri Sivasubramaniya Nadar College of Engineering, Chennai, Tamil Nadu, India*

Abstract

Result diversification provides a broader view of a topic, while maximizing the chances of retrieving relevant information. It avoids the bias in results, thus improving the user experience. This area finds a lot of applications in web searches and recommendation systems. The existing literature on this domain has achieved good accuracy on smaller datasets and using single models. An ensemble approach, using three neural network models, has been proposed to improve the existing predictions using a bigger dataset.

Keywords

Result diversification, Ensembling methods, Grid Search, Voting Regressor

1. Introduction

Diversification of results is one of the most important trends in the areas of web searches, recommendation systems and structured databases. With the development of image resources in searches, retrieving diverse and relevant results for a query has become a challenging task. This is due to the requirement that the retrieved images should satisfy various semantic intents of the queries, based on the visual attributes and features of an image. Context information, such as captions, descriptions, and tags, provides opportunities for image retrieval systems to improve their result diversification. [8]

The objective of result diversification is to provide user satisfaction, improve productivity, reduce bias or homogeneity in results and to be able to cater to alternate interpretations of a query. Diversification is relevant in image retrieval since it avoids retrieval of superficial results, it provides multifaceted results for a query. eg: If a user searches for pictures of cars, the system needs to retrieve images of different kinds of cars, taken at different locations and time of the day from various angles. Result diversification also guesses the intent of the query and obtains satisfactory results, for ambiguous and incomplete queries. [9]

There are existing methods that use late fusion systems as shown in Table 1. The approach

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy

shreya19106@cse.ssn.edu.in (S. Sriram); madhuri19057@cse.ssn.edu.in (M. Mahalingam); sarah19100@cse.ssn.edu.in (S. A. Naseer); shajith2010537@ssn.edu.in (S. Hameed); rahul2010222@ssn.edu.in (R. Rajagopalan); saishashaank2010084@ssn.edu.in (S. S. R); lekshmik@ssn.edu.in (L. Kalinathan); prabavathyb@ssn.edu.in (P. Balasundaram)

Table 1

Existing Work

Existing Work	Proposed Methodology	Performance
Domain Independent System Fusion [3]	Cross Space Fusion Layers,Attention Layers are used with Dense Layers	F1 score: 0.2823
Differential Evolution-Based Fusion for Results Diversification of Web Search [6]	Differential evolution-based fusion (DE)	Normalised Discounted Cumulative Gain (NDCG): 0.5476
Fusion-Based Methods for Result Diversification on the web [7]	Linear Fusion	F1 score: 0.3987

taken in [3] uses Deep Neural Networks architectures as the primary ensembling learner with various network configurations that use dense and attention layers. Cross- Space-Fusion layer has been used here to manipulate the newly created spatial information. When compared with the current state of the art and traditional ensembling approaches, the proposed model showed significant improvements, by a margin of at least 38.58%.

Search result diversification of text documents is necessary when a user issues an ambiguous query to the search engine. Such ambiguous queries require a diversified resultant list that includes documents that are relevant to as many different types of subtopics as possible. A group of fusion-based result diversification methods with novel methods of weight assignment for linear combination is proposed in [7]. This aims to improve performance that considers both relevance and diversity. The study [6] uses data fusion for result diversification and it investigates how to use differential evolution to learn weights for the linear combination method.

The latest developments in this field, using deep neural networks as the primary ensembling method has shown major improvements over the traditional ensembling methods by increasing the performance of the individual inducers [3, 4, 5]. The existing methods work more efficiently only under certain conditions therefore, it is of vital importance to come up with new computer vision and deep learning methods that can enable achieving diverse and appropriate search results for all kinds of data. In this context ImageCLEF [1], a benchmarking activity on the cross-language annotation and retrieval of images in the Conference and Labs of the Evaluation Forum (CLEF) has proposed the ImageCLEFfusion task [2].

2. Task and Dataset Description

Result diversification fusion task [2] aims to maximise the chances of retrieving relevant answers that correspond to the query. In the context of image retrieval, an inducer is generally responsible for retrieving a set of relevant images for the given query id. Output of the inducer consists of the relevant images, their similarity scores and ranks. However, a single inducer is disadvantageous for application in certain areas due to low precision and lack of performance. Hence, this task involves the use of ensembling to overcome this scenario. Ensembling is a technique which aggregates the predictions of several inducers.

The training data is fed into three or more models and their predictions are then combined to obtain a final prediction using a fusion algorithm (ensembling). The ensembled system is expected to yield a better performance compared to the highest performing individual inducer.

The data for this task is obtained from the Retrieving Diverse Social Images Task dataset [Ionescu2020]. The outputs of 56 inducers, representing a total of 123 queries (topics) are stored in separate text files. Each entry or row in these files is of the format as given below in the Table 2.

Table 2

Attributes of Inducer file

Fields	Representation
query_id	unique id of the query
inter	ignored value
photo_id	unique photo id
rank	photo rank
sim	similarity score
run_name	name of inducer

3. Methodologies Used

Different networks namely, Multilayer Perceptron, Ridge Regressor using Grid Search and Keras Regressor using Sequential model were studied to learn the patterns of the outputs of the inducers.

3.1. Multilayer Perceptron Regressor

Neural networks are mathematical structures that are formed with a neuron as the fundamental element. Artificial neurons are arranged in layers and coupled to build neural networks. Multi Layer Perceptron (MLP) network is the composition of neurons as shown in Figure 1. It is a feed forward neural network made up of successive layers that communicate and exchange information via synaptic connections represented by an adaptive weight.

The structure of a multilayer network includes multiple layers of perceptrons. The input layer has number of perceptions same as the number of data attributes, an output layer with one perceptron in the case of regression, and all other layers are considered to be hidden. The information flows unidirectionally, from input layer to output layer, through the hidden layers. The hidden layers are the computation engine of the MLP. The weight adjustment training is done via backpropagation. In this method, an error is calculated when the network output is compared to the expected output. The error is then propagated back through the network, one layer at a time, and the weights are modified based on their contribution to the error. Backpropagation is a method of repeatedly adjusting the weights in order to minimize the difference between the actual and desired output. In the regression scenario, activation

function will not be applied for the output of the dense layer. Hence, this output will serve as the predicted one.

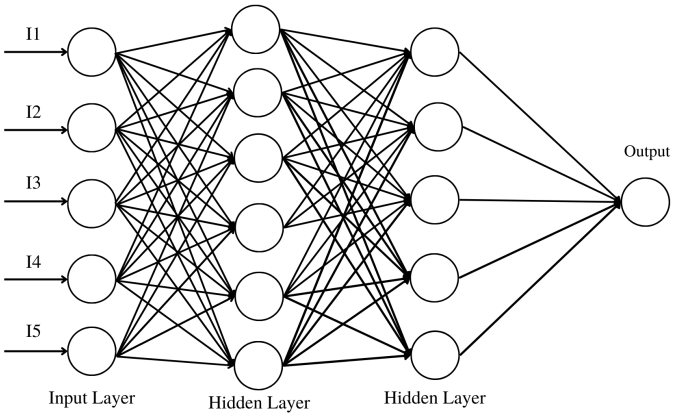

Figure 1: Multi layer Perceptron model

3.2. Ridge Regressor using Grid Search method

The Ridge regression model is a linear regression model upon which Grid Search is applied to find the hyperparameters. Grid search is a parameter tuning method in which a model is built and evaluated for each set of algorithm parameters specified in a grid. The method calculates the performance of all combinations of the specified hyperparameters and their values and gives the best option. Hyperparameters are the values that are manually set before training. If these are set appropriately then the performance of the model can be improved. To minimize the overfitting with the ordinary Grid search, stratified cross-validation is applied where samples are divided into K-folds at random. An iterative approach is used to divide the training data into k parts. In each iteration, one division is kept for testing, and the remaining k-1 partitions are used to train the model. In the next iteration, the next partition is the test data and the remaining k-1 is the train data. The model performance is recorded and average of results is provided. The advantage of this method is that it gives less biased results compared to test - train split. The GridSearchCV model from Scikit Learn[1] is used to get the parameters. Grid Search is easy to implement and is reliable.

3.3. Keras Regressor using Sequential Model

Regression is implemented using the KerasRegressor [2] class in Keras, which is applied over a Sequential model . A sequential model is a linear stack of layers, where each of the layers is a

[1]https://scikit-learn.org/stable/
[2]https://keras.io/

neural network layer with exactly one input vector of n-dimensions and an output vector of n-dimensions. These vectors of n-dimensions are also called tensors or n-dimension matrices. The sequential model consists of an input, multiple hidden and output dense layers as shown in Figure 2. A dense layer is a regular deeply connected neural network layer that on an input returns or outputs the activated sum of the dot product of the inputs with the kernels or weights and the bias.

$$output = activation(dot(input, weight) + bias),$$

The dense layer comprises of neurons or input nodes that are activated based on an activation function. An activation function decides if a neuron or node should be activated or not. The activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input at each layer. These functions are used to enhance the performance of deep learning models. The simplest activation function is referred to as the linear activation, where no transform is applied at all. However, nonlinear activation functions are preferred as they allow the nodes to learn more complex structures in the data. The widely used non-linear activation functions include ReLu, softmax and sigmoid. The bias used in the calculation of the output is a constant value or vector that is added to the weighted sum. It helps in shifting the result of the activation function towards the positive or negative side , in other words it's used to offset the result obtained. This addition of bias introduces flexibility and better generalization to the neural model.

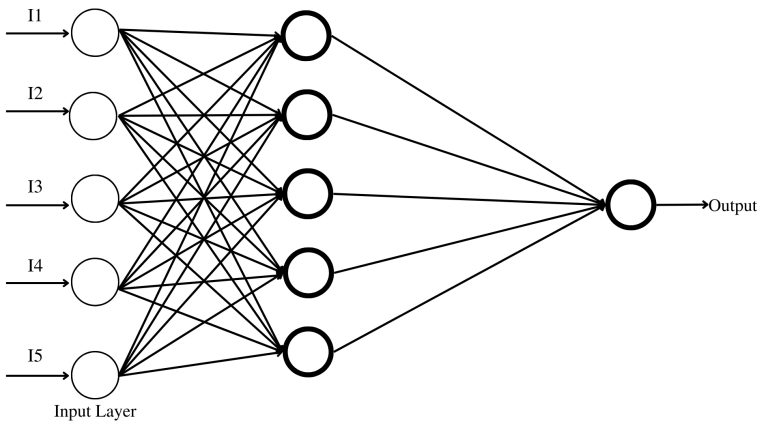

Figure 2: Sequential Neural Model

3.4. Voting Regressor

Given the similarity scores of the *m* inducers corresponding to *n* queries, it is essential to apprehend the evaluation of similarity scores by the inducers. In order to maximize the similarity scores, voting regressor has been adapted as shown in Figure 3. A *Voting Regressor* is an ensemble meta-estimator which fits several base regressors on the whole dataset.

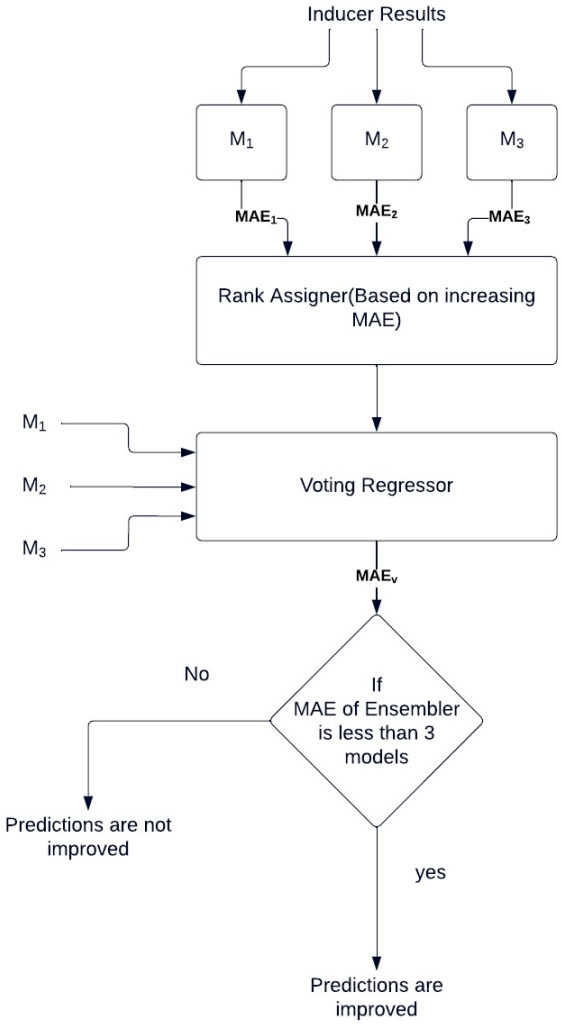

Figure 3: Voting Regressor

It consists of three different predictors namely, MLP Regressor (M_1), Ridge Regressor using Grid Search (M_2) and Keras Regressor using Sequential model (M_3) for the evaluation of the results. Performance of theses models are provided in terms of Mean Absolute Errors (MAE). The regressors are ranked based on their MAE with the help of *Rank Assigner.* These ranks and the models are provided as input for the Voting Regressor[10][11] which will consider the weight of the occurrences of predicted values before averaging. Further, the performance of the voting regressor can again be measured with MAE. The predictions of the voting regressor will be considered to be improved, if the MAE of it is lesser than the other predictors.

4. Implementation

The given dataset was split into 90% for training and 10% for validation data. The similarity score column is extracted from the dataset and is normalized to ensure similar data distribution, to achieve faster convergence. This serves as our input and output. Three predictor models M_1, M_2 and M_3 were built to study how similarity scores are assigned.

The model M_1 was implemented using sklearn.neural_network, which consists of an input layer of 5 neurons, 2 hidden layers of 6 & 5 neurons each and an output layer of 1 neuron. The random state is set to 5 to avoid random split of data at each iteration. A constant learning rate of 0.01 is initialized to control the step size in updating the weights.

The model M_2 was created to solve the regression using the hyperparameters tuned by Grid Search. A search space with all possible hyperparameters is defined by Grid Search. Cross validation of the data is performed 3 times by splitting the data into 10 folds to obtain multiple iterations of training and testing on the data. The best model is chosen and the performance of the cross validated is evaluated by setting the scoring parameter of GridSearchCV as *neg_mean_absolute_error*.

The model M_3 was created using an input layer of 5 neurons, 2 dense layers of output dimensions 5 and 1, stacked over each other. The optimizer function is set as *Stochastic Gradient Descent* (SGD) with a learning rate of 0.0008.

The data is sampled into batches of size 5, such that every set of 5 inputs is used to predict the next input's similarity score, for every predictor. The above predictors are trained with 90% of the training set and validated with the remaining training set. The predicted values are compared with the actual values of the inputs and the mean absolute deviation is calculated as error. Ranks have been assigned for the models M_1, M_2 and M_3 based on the error values.

The Voting Regressor is constructed with the ranks obtained and the 3 predictor models. The regressor is trained and tested on the entire training and validation data respectively. In the training phase, the output of the *voting regressor* for every batch is the prediction for the next input. This is compared with the actual value and error is calculated as the difference between these values. Further, it is tested on the test data provided and the original similarity score of the data is replaced with the improved score predicted by the model.

5. Results and Analysis

Validation dataset is used to test the models and voting regressor is used to predict the similarity scores. These predicted and the actual similarity score values were plotted for each one of them as shown in Figures 4,5,6,7. It is seen from figures 4 & 5, that the model M_2 predicts better when compared to the model M_1. Further, it is also seen from Figure 7, that the Voting Regressor predicted the similarity score better when compared to the model M_2, as it utilized the weighted occurrences of predicted values before averaging. Table 3 shows the MAE and rank values for the base and voting regressor models.

The voting regressor model has been tested with the CLEF test data and metrics - F1 measure and cluster recall are used to compare and analyze the performance of the results thus obtained. Cluster recall is a metric that assesses how many different clusters from the cluster labels are

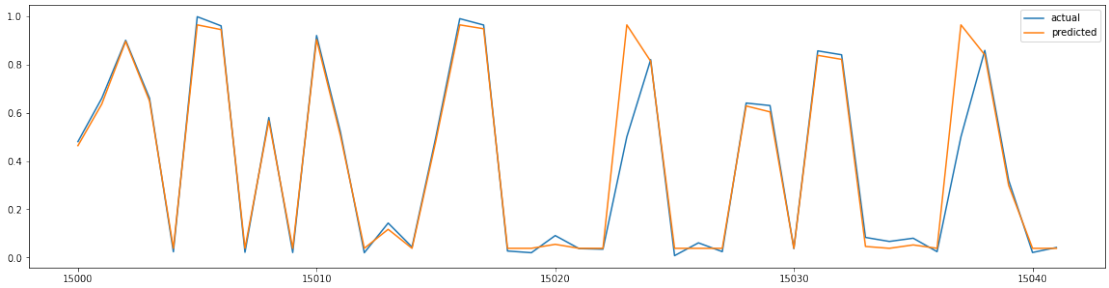

Figure 4: Actual and Predicted Values of Similarity Score using MLP Regressor

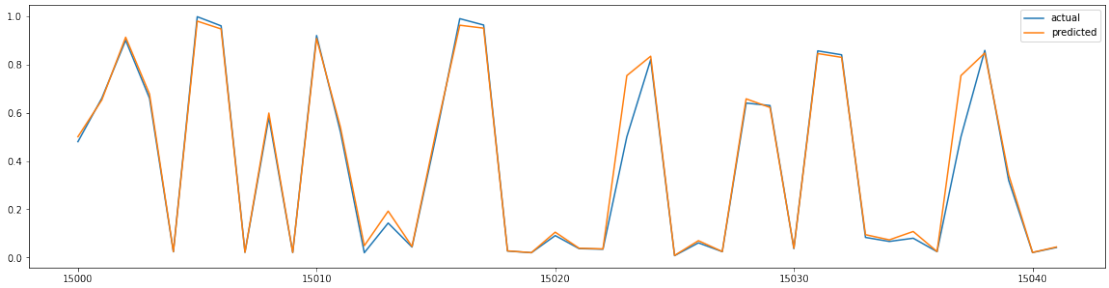

Figure 5: Actual and Predicted Values of Similarity Score using Ridge Model using Grid Search

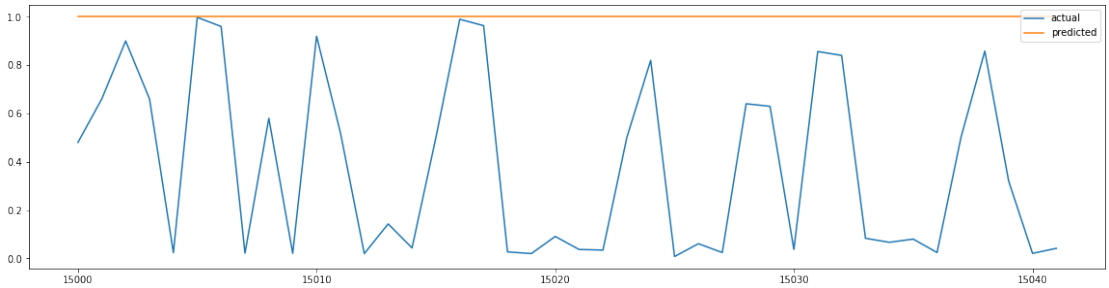

Figure 6: Actual and Predicted Values of Similarity Score using Keras Regressor using Sequential model

represented. A cluster recall of 0.4384 has been observed among the top 20 results. F1 measure is the harmonic mean of cluster recall and precision, where precision measures the number of relevant images among the top 20 results. An F1 measure of 0.5604 has been obtained for the top 20 results.

The *voting regressor* is used to predict the updated similarity score values for the testing data which contains 175,591 entries. 10 different variations of the voting regressor were built by varying the parameters, iteration size. Table 4 illustrates the F1 scores and CR scores evaluated

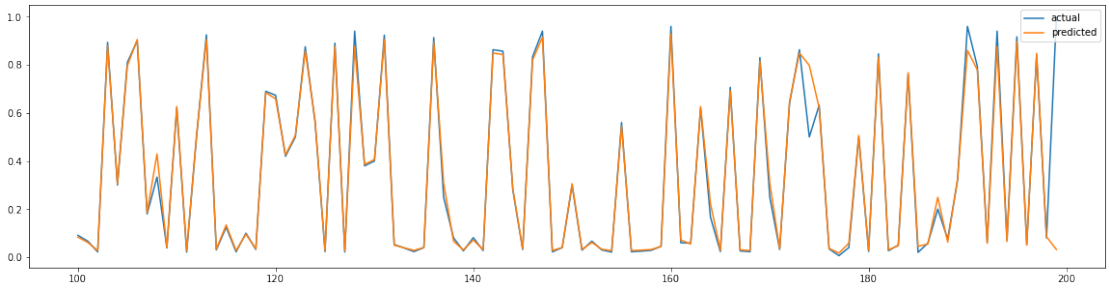

Figure 7: Actual and Predicted Values of Similarity Score using Voting Regressor

Table 3

Results of the base and voting regressor models

Models	MAE	Rank
M_1 (Keras Regressor using Sequential Model)	0.626	1
M_2(MLP Regressor)	0.041	3
M_3 (Ridge Regressor using Grid Search)	0.032	2
Voting Regressor	0.030	-

Table 4

F1 score and Cluster Recall rates of 10 best runs

Run No	F1@20 score	CR@20 score
1	0.4316	0.3167
2	0.5095	0.4053
3	0.5398	0.4276
4	0.4929	0.3906
5	0.5563	0.4332
6	0.4963	0.3743
7	0.5533	0.4341
8	0.5604	0.4373
9	0.5547	0.4384
10	0.5568	0.4362

for 10 best file submissions.

6. Conclusion

In order to improve the predictions of the results of the inducers, the proposed ensemble model was implemented using three neural networks as base regressors. The model was trained on

data from 56 different inducers, containing 167,139 training values and tested on data from 55 inducers, containing 175,591 testing values. The base regressors obtained MAE values of 0.626, 0.041 and 0.032 each. The ensemble method obtained an improved MAE score of 0.030. Among the ten best submissions, the best F1 score and CR score are 0.5604 and 0.4384 respectively.

References

[1] Bogdan Ionescu, Henning Müller, Renaud Péteri, Johannes Rückert, Asma Ben Abacha, Alba García Seco de Herrera, Christoph M. Friedrich, Louise Bloch, Raphael Brüngel, Ahmad Idrissi-Yaghir, Henning Schäfer, Serge Kozlovski, Yashin Dicente Cid, Vassili Kovalev, Liviu-Daniel Ștefan, Mihai Gabriel Constantin, Mihai Dogariu, Adrian Popescu, Jérôme Deshayes-Chossart, Hugo Schindler, Jon Chamberlain, Antonio Campello, Adrian Clark, Overview of the ImageCLEF 2022: Multimedia Retrieval in Medical, Social Media and Nature Applications, in Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the 13th International Conference of the CLEF Association (CLEF 2022), Springer Lecture Notes in Computer Science LNCS, Bologna, Italy, September 5-8, 2022.

[2] Liviu-Daniel Ștefan, Mihai Gabriel Constantin, Mihai Dogariu and Bogdan Ionescu. Overview of the ImageCLEFfusion 2022 Task: Ensembling Methods for Media Interestingness Prediction and Result Diversification, in CLEF2022 Working Notes. CEUR Workshop Proceedings (CEUR-WS.org), Bologna, Italy, September 5-8, 2022.

[3] Constantin, Mihai Gabriel, Liviu-Daniel Ştefan, and Bogdan Ionescu. "DeepFusion: Deep ensembles for domain independent system fusion." International Conference on Multimedia Modeling. Springer, Cham, 2021.

[4] Constantin, M.G., S¸tefan, L.D., Ionescu, B., Duong, N.Q., Demarty, C.H., Sj¨oberg, M.: Visual interestingness prediction: A benchmark framework and literature review. International Journal of Computer Vision pp. 1–25 (2021).

[5] S¸tefan, L.D., Constantin, M.G., Ionescu, B.: System fusion with deep ensembles. In: Proceedings of the 2020 International Conference on Multimedia Retrieval (ICMR 2020). pp. 256–260. Association for Computing Machinery (ACM) (2020).

[6] Xu, Chunlin, Chunlan Huang, and Shengli Wu. "Differential evolution-based fusion for results diversification of web search." International Conference on Web-Age Information Management. Springer, Cham, (2016)

[7] Wu, Shengli, et al. "Fusion-based methods for result diversification in web search." Information Fusion 45 (2019): 16-26.

[8] Zheng, K., Wang, H., Qi, Z. et al. A survey of query result diversification. Knowl Inf Syst 51, 1–36 (2017).

[9] Drosou, Marina, and Evaggelia Pitoura. "Search result diversification." ACM SIGMOD Record 39.1 (2010): 41-47.

[10] Maxwell, D., Azzopardi, L. Moshfeghi, Y. The impact of result diversification on search behavior and performance. Inf Retrieval J 22, 422–446 (2019).

[11] Sagi, Omer, and Lior Rokach. "Ensemble learning: A survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8.4 (2018): e1249.